Computer Science and Technology

NBS Special Publication 500-95

Proceedings of the Computer Performance Evaluation Users Group 18th Meeting

CPEUG82

"Improving Organizational Productivity"

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Chemical Physics — Analytical Chemistry — Materials Science

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Manufacturing Engineering — Building Technology — Fire Research — Chemical Engineering[2]

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

Computer Science and Technology

NBS Special Publication 500-95

Proceedings of the Computer Performance Evaluation Users Group (CPEUG) 18th Meeting

Washington, DC
October 25-28, 1982

Proceedings Editor
Carol B. Wilson

Conference Host
Naval Data Automation Command
Department of the Navy

Sponsored by
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued October 1982

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publication.

Library of Congress Catalog Card Number: 82-600622

National Bureau of Standards Special Publication 500-95
Natl. Bur. Stand. (U.S.), Spec. Publ. 500-95, 414 pages (Oct. 1982)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1982

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402
Price $11.00
(Add 25 percent for other than U.S. mailing)

CPEUG 82

"Improving Organizational Productivity"

Foreword

CPEUG 82 directly confronts the principal pervasive challenge facing information system management during the 1980's: IMPROVING ORGANIZATIONAL PRODUCTIVITY through improvements in the effectiveness, efficiency, and range of automated information services. Productivity improvement in all corporate activities, of necessity, will be the continuing goal of commercial and governmental management for many years. For organizations whose success is closely tied to large volumes of accurate and timely information, improving automated information services is central to achieving improvements in productivity throughout the organization. The Federal Government is one such information-intensive organization, and improved automated information systems are critical to meeting the goal of providing vital public services for fewer tax dollars.

To help improve total productivity throughout an organization, decisions about information services and resources must be made within the context of their impact on the total organization. Information system managers must continually search for ways to help increase the productivity of users while simultaneously improving the cost-effectiveness of their own information processing activities. To be successful, information management must direct its attention primarily to the long-term, strategic, high-payoff areas of information processing: personnel, information data base, data communication network, and software. Information managers must also ensure that their activities in these areas are totally integrated with functional management plans and activities throughout the organization.

Since its founding in 1971 by the US Air Force, CPEUG has focused on the most important areas of automated information system management. As the technology and economics of information systems evolved, so did CPEUG. This year's conference continues the historical emphasis on techniques for increasing the operational efficiency of information systems, while increasing the coverage of other important issues of the day - for example, data bases, local area networks, strategic planning, software, office automation, and chargeback. CPEUG 82 offers a carefully balanced program that addresses the important topics needing attention during the 1980's. Welcome to CPEUG 82.

This conference and these proceedings are the results of the diligent work of an excellent team of dedicated professionals on this year's conference committee. To them all (their names appear later in this document), I offer my sincerest appreciation for a job well done. Ms. Sylvia Mabie and Ms. Donna Granahan have earned special thanks for their continuing excellent administrative support.

Thomas F. Wyrick
CPEUG 82 Conference Chairperson
October 1982

CPEUG 82

"Improving Organizational Productivity"

Preface

The theme of CPEUG 82, IMPROVING ORGANIZATIONAL PRODUCTIVITY, focuses on the major business challenge of the 1980's. While severe budgetary constraints persist, the objective of improving the effectiveness, efficiency, and range of automated information services puts a heavy burden on information managers and performance analysts. The CPEUG 82 program reflects increased sophistication in all areas of systems performance, including information management, hardware, software, communications, and personnel.

The CPEUG 82 program has been designed to meet many needs with its three track approach. The three tracks focus on the issues of performance in the system life cycle, advances in traditional CPE areas, and learning about CPE. In addition, on Tuesday, an experimental session from the Computer Measurement Group (CMG) will be given. This year, several sessions focus on specific types of systems, as well as on local area network performance and remote terminal emulation design and development. Also, the expansion of traditional areas of performance improvement will help information managers evaluate new technologies and improve ADP planning and life cycle processes. CPEUG continues to be a forum for providing open discussion on areas relating to the Federal Government ADP procurement process. Planned discussions focusing on design and acquisition strategies bring to the forefront the issues of needs analysis and procurement in a dynamic technological environment.

The CPEUG 82 program was the work of many people. Dr. Mani Chandy, Vice Chairperson for academia; Terry Potter, Vice Chairperson for industry; and Barry Wallack, Vice Chairperson for Government, were this year's vital links to broaden program participation. The Conference Committee, session chairpersons, authors, tutors, and referees all deserve recognition for their time, patience, and participation. The invaluable support of Joyce Stellar merits special thanks.

James Sprung
CPEUG 82 Chairperson
October 1982

CPEUG82

"Improving Organizational Productivity"

Abstract

These Proceedings record the papers that were presented at the Eighteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 82) held October 25-28, 1982, in Washington, DC. With the theme, "Improving Organizational Productivity," CPEUG 82 reflects the critical role of information services in the productivity and survival of today's organization. To meet this challenge, the scope of CPE must be expanded to address performance issues in all aspects of information systems (hardware, software, facilities, communications, personnel, policies, and procedures) throughout the system life cycle. The program was divided into three parallel sessions and included technical papers on previously unpublished works, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

Key words: benchmarking; capacity planning; chargeback systems; computer performance management systems; queuing models; resource measurement facilities; simulation; supercomputers; workload characterization.

"Improving Organizational Productivity"

CPEUG Advisory Board

Carl R. Palmer, Chairman
U.S. General Accounting Office
Washington, DC

Allen L. Hankinson, Executive Secretary
National Bureau of Standards
Washington, DC

Dennis M. Gilbert
National Bureau of Standards
Washington, DC

James E. Weatherby
Federal Computer Performance Evaluation
and Simulation Center
Washington, DC

Thomas F. Wyrick
Federal Computer Performance Evaluation
and Simulation Center
Washington, DC

"Improving Organizational Productivity"

Conference Committee

CONFERENCE CHAIRPERSON	Thomas F. Wyrick Federal Computer Performance Evaluation & Simulation Center (FEDSIM) (703) 274-7910
PROGRAM CHAIRPERSON	James G. Sprung The MITRE Corporation (703) 827-6446
PROGRAM VICE-CHAIRPERSON FOR FEDERAL GOVERNMENT	Barry Wallack Defense Communications Agency/CCTC (202) 695-0856
PROGRAM VICE-CHAIRPERSON FOR INDUSTRY	Terry Potter Digital Equipment Corporation (617) 568-6061
PROGRAM VICE-CHAIRPERSON FOR ACADEMIA	K. Mani Chandy University of Texas (512) 471-4353
PUBLICATION CHAIRPERSON	Peter J. Calomeris Westinghouse Electric Corporation (301) 765-4644
PUBLICITY AND AWARDS CHAIRPERSON	Dennis R. Shaw U.S. General Accounting Office (202) 275-6187
REGISTRATION CHAIRPERSON	Alfred J. Perez U.S. Air Force Data Systems Design Center (205) 279-4051
PROCEEDINGS EDITOR	Carol B. Wilson Fiscal Associates, Inc. (703) 642-1390
NATIONAL ARRANGEMENTS CHAIRPERSON	Felicia Carpenter Navy Supply Systems Command (202) 697-9584
FINANCE CHAIRPERSON	Barbara N. Anderson FEDSIM (703) 274-7910
VENDOR PROGRAM CHAIRPERSON	L. Arnold Johnson Federal Compiler Testing Center General Services Administration (703) 756-6153
LOCAL ARRANGEMENTS AND PRE-REGISTRATION CHAIRPERSON	David E. Koranek Naval Data Automation Command/51D (201) 433-3499

CPEUG82

"Improving Organizational Productivity"

Referees

Barbara Anderson

George Baird

Bernard Domanski

Major Charles Gausche

Bill Hawe

L. Arnold Johnson

John C. Kelly

David Lindsay

Ken Moore

Tim Oliver

James G. Sprung

Kathy Rebibo

CPEUG82

"Improving Organizational Productivity"

TABLE OF CONTENTS

FOREWORD iii

PREFACE iv

ABSTRACT v

CPEUG ADVISORY BOARD vi

CONFERENCE COMMITTEE vii

CPEUG 82 REFEREES viii

TRACK A: LIFE CYCLE MANAGEMENT

STRATEGIC PLANNING FOR ADP: INFORMATION RESOURCES MANAGEMENT

SESSION OVERVIEW
J. Howard Bryant
U.S. Patent and Trademark Office 3

INFORMATION SYSTEMS MANAGEMENT
James J. Spinelli
VITRON Management Consulting, Inc. 5

LONG-RANGE ADP PLANNING: A FEDERAL AGENCY PLANNING MODEL
Paul E. Matthews
The MITRE Corporation 11

PRODUCTIVITY THROUGH INTEGRATED INFORMATION RESOURCE MANAGEMENT
Malcolm Campbell
Missouri State Government 19

FINANCIAL MANAGEMENT CONSIDERATIONS

PRICING STRATEGIES IN PROCUREMENTS CONDUCTED UNDER THE BASIC AGREEMENT
Thomas G. Morrison
MCAUTO Systems Group, Inc. 27

SMALL COMPUTER POLICY AND STRATEGY

PANEL OVERVIEW
Dennis Gilbert
National Bureau of Standards 37

INFORMATION SYSTEMS NEEDS ANALYSIS

FULFILLING BUSINESS NEEDS WITH AN ON-LINE SYSTEM
David R. Vincent
Institute for Software Engineering 41

SOFTWARE TESTING - A LOST ART

SESSION OVERVIEW
George Baird
Federal Software Testing Center 49

CONCEPTUAL PROPOSAL FOR A COBOL ANALYZER SOFTWARE TOOL
L. Arnold Johnson & William R. Milligan
Federal Software Testing Center 51

CAPACITY MANAGEMENT - FROM CONCEPT TO IMPLEMENTATION

SESSION OVERVIEW
Major Charles Gausche
Pentagon AFDSC 63

DEVELOPMENT OF A STANDARD PERFORMANCE MANAGEMENT STRATEGY FOR THE US NAVY
S.B. Olson
Navy Regional Data Automation Center, Pensacola 65

DEVELOPMENT OF A METHODOLOGY FOR THE ANALYSIS OF SYSTEM PERFORMANCE INDICATORS
Paul Chandler
Wilson Hill Associates 75

COMPUTER SYSTEM DATA NEEDED FOR CAPACITY PLANNING
Dr. John T. Peterson
BGS Systems, Inc. 81

A TOOL FOR EDP MANAGEMENT: OMB CIRCULAR A-123

SESSION OVERVIEW
Ted Gonter
General Accounting Office 87

DATA PROCESSING AND A-123
Sheila Brand
Department of Defense Computer Security Center 89

TRACK B: TECHNOLOGY AND APPLICATION ADVANCES

WWMCCS NETWORK PERFORMANCE ANALYSIS

PERFORM - WWMCCS INTERCOMPUTER NETWORK (WIN) PERFORMANCE OPTIMIZATION RESEARCH MODEL
K. Chung, O.A. Mowafi & K.A. Sohraby
Computer Sciences Corporation 97

A SIMULATION STUDY OF A LOCAL AREA NETWORK FOR A COMMAND AND CONTROL CENTER
Kathy K. Rebibo
The MITRE Corporation 107

WORKLOAD CHARACTERIZATION

WORKLOAD CHARACTERIZATION USING IMAGE ACCOUNTING
Rajendra K. Jain & Rollins Turner
Digital Equipment Corporation 111

METHODOLOGY FOR CHARACTERIZING A SCIENTIFIC WORKLOAD
Ingrid Y. Bucher & Joanne L. Martin
Los Alamos National Laboratory 121

CASE HISTORY: BUSINESS DRIVER METHODOLOGY IN A MANUFACTURING LOGISTICS APPLICATION
F.J. Machung
International Business Machines 127

MODELING TECHNIQUES

SESSION OVERVIEW
Kenneth C. Sevcik
University of Toronto 137

DESIGN OF A SOFTWARE TOOL FOR EVALUATION OF COMPUTER AND COMMUNICATION SYSTEMS
Ashok K. Agrawala, Satish K. Tripathi & Ashok K. Thareja
University of Maryland 139

A PERFORMANCE BOUND FOR MULTIPROGRAMMED VIRTUAL MEMORY SYSTEMS
Rollins Turner
Digital Equipment Corporation 155

AN EFFICIENT CAPACITY ASSIGNMENT ALGORITHM FOR COMPUTER COMMUNICATION NETWORKS WITH A TREE TOPOLOGY
Chaim Ziegler
Brooklyn College

Roberta Klibaner
The College of Staten Island 173

COMPONENTS OF SOFTWARE PACKAGES FOR THE SOLUTION OF QUEUEING NETWORK MODELS

G.S. Graham, E.D. Lazowska & K.C. Sevcik
Quantitative System Performance 183

PERFORMANCE MONITORING TECHNIQUES

DESIGN OF EMBEDDED COMPUTER MONITORING SYSTEM
Albundio Alvarez
Naval Ocean Systems Center 191

PROGRAM INSTRUMENTATION TECHNIQUES
Raymond C. Houghton, Jr.
National Bureau of Standards 195

UNIX PERFORMANCE ANALYSIS

PERFORMANCE PREDICTION IN A UNIX ENVIRONMENT
Lawrence W. Dowdy, Lindsey E. Stephens & Alfredo Perez-Davila
Vanderbilt University 205

UNIVAC PERFORMANCE ANALYSIS

SESSION OVERVIEW
John C. Kelly
Datametrics Systems Corporation 215

A STUDY OF DISK I/O ON A UNIVAC SYSTEM IN THE SHUTTLE MISSION SIMULATOR COMPUTER COMPLEX
Ankur R. Hajare
The MITRE Corporation 217

THE APPLICATION OF ANALYTIC AND SIMULATION MODELS TO SIZE A LARGE COMPUTER SYSTEM
Richard W. Tibbs
Martin Marietta Corporation

John C. Kelly
Datametrics Systems Corporation 231

A UNIVAC WORKLOAD CHARACTERIZATION SYSTEM
Walter N. Bays & Dawn L. Voegeli
The MITRE Corporation 259

IBM PERFORMANCE ANALYSIS

SESSION OVERVIEW
Tim Oliver
National Institutes of Health 277

RMF EQUATIONS: OBTAINING JOB CLASS LEVEL RESULTS FROM RMF
Bob Irwin
IKON 279

SERVICE LEVEL MANAGEMENT THROUGH WORKLOAD SCHEDULING
David G. Halbig
U.S. Senate Computer Center 297

EVENT DRIVEN MEASUREMENTS OF MVS THAT IMPROVE CONFIGURATION TUNING AND MODELING
Glen F. Chatfield
Duquesne Systems, Inc 313

A NEW APPROACH TO VM PERFORMANCE ANALYSIS
Bill Tetzlaff & Thomas Beretvas
IBM, Research Division 321

A VM/SP PERFORMANCE MANAGEMENT INFORMATION SYSTEM
John Story
Texas Instruments Inc. 331

PERFORMANCE OF LOCAL AREA NETWORKS

SESSION OVERVIEW
Bill Hawe
Digital Equipment Corporation 363

A COMMON FRAMEWORK FOR STUDYING THE PERFORMANCE OF CHANNEL ACCESS PROTOCOLS
K.K. Ramakrishnan & Satish K. Tripathi
University of Maryland 365

PREDICTING ETHERNET CAPACITY - A CASE STUDY
Madhav Marathe & Bill Hawe
Digital Equipment Corporation 375

EVALUATING LOCAL NETWORK PERFORMANCE
Jonas Herskovitz
Hughes Aircraft Company 389

BENCHMARKING AND REMOTE TERMINAL EMULATION

SESSION OVERVIEW
Dr. Bernard Domanski
College of Staten Island
The College of Staten Island 399

327X EMULATOR PACKAGE FOR SYSTEM RESPONSE TIME EVALUATION
Mary Christ
International Business Machines 401

THE DESIGN AND APPLICATION OF A REMOTE TERMINAL EMULATOR
Michael Proppe
Computer Sciences Corporation

Barry Wallack
Command and Control Technical Center 409

DESIGN OF AN EXTERNAL TEST DRIVER FOR PERFORMANCE EVALUATION
Agu R. Ets & John H. McCabe
Analytics, Inc. 415

TRACK C: UNDERSTANDING EXPERIENCES
Tutorial and Case Study Abstracts

TUTORIAL ON CHARGING SYSTEMS IN THE FEDERAL GOVERNMENT
Dean Halstead
FEDSIM/NA 425

REALLY IMPROVING SOFTWARE MANAGEMENT
Thomas B. Cross
Cross Information Company 427

APPLICATION OF SOFTWARE PERFORMANCE ENGINEERING TECHNIQUES
Dr. Connie U. Smith
Duke University 433

TUTORIAL ON WORKLOAD FORECASTING
Helen Letmanyi
National Bureau of Standards 435

ANALYZING QUEUEING NETWORK MODELS OF COMPUTER SYSTEMS: A TUTORIAL ON THE STATE OF THE ART
Edward D. Lazowska
University of Washington

Kenneth C. Sevcik
University of Toronto 437

CONTINGENCY PLANNING
Susan K. Reed
National Bureau of Standards 439

BENCHMARK CONSTRUCTION AND VALIDATION USING SYNTHETIC SOFTWARE
Bruce D. Grant
Systems Architects, Inc. 443

CPEUG82

"Improving Organizational Productivity"

Strategic Planning for ADP

SESSION OVERVIEW

STRATEGIC PLANNING FOR ADP: INFORMATION RESOURCE MANAGEMENT

J. Howard Bryant

U.S. Patent and Trademark Office
Washington, D.C. 20231

This session focuses on the concept of information as a critical organizational resource and the methods of planning for and controlling it by the organization to gain organizational productivity. The session will be opened by a discussion of the concept and the implications it has for information system developers. This will be followed by the description of a model for information resource strategic planning. A final paper will discuss techniques for controlling the operation of an integrated information resources management function and gaining organizational productivity.

INFORMATION SYSTEMS MANAGEMENT

James J. Spinelli, Vice President

VITRON Management Consulting, Inc.
48 West 48th St. Suite 1501
New York, N.Y. 10036

A few years ago, John Diebold introduced his ideas on a concept he calls, "Information Resource Management (IRM)." The major premise is that information is a vital, valuable corporate resource. As Diebold himself states, "The organizations that will excel in the 1980s will be those that manage information as a major resource." But, we must carry this concept a few steps further, that is, information is the direct "product" of the total organizational business system. Information is a function of the totality of all organizational resources. IRM is the function that represents the business system encompassed by the organization's own existence that is used to produce and disseminate the information resource.

Key words: Information Resource Management; Information Systems Management; Concepts; Strategies; Methodologies; Techniques; Management-Tool.

1. Introduction

Information is a function of the totality of all corporate resources. Within each organization it is the function that represents the core of all business enterprises.

We must look at information as the product of a business enterprise that is at the core of a hierarchy of systems. Information is a "product" that needs a great deal of time spent on educating organizations as to how it is really a direct function of the aggregate of all resources. Information is the single most significant bond that binds the organization together.

When we speak about a business enterprise, we must realize that the enterprise is the information it possesses, interprets and communicates. The effectiveness of management in business is a direct function of the information possessed, interpreted, and communicated by all of the organization's managers.

There is a logical process that proceeds upward: (a) The ebb and flow of information, its production and dissemination, is a subsystem, subordinate to the organization, which too is a system; (b) The organization is a subsystem, subordinate to the industry to which it is part; (c) The industry is a subsystem, subordinate to the economic system of the country to which it serves and responds, etc. Within this context, we may now proceed to address the issue of "Information Systems Management," as the totality of all resources, integrated for purposes of formulating and managing the strategic goals of the organization.

2. Direction

Rapid technological changes are having a dramatic impact on the way organizations conduct business today. Combined with advances in technology are increasing regulations, new and more intense competition, and an unstable financial climate.

These issues bear witness to a business environment that affords opportunities and challenges to those who properly plan for the allocation and utilization of all corporate resources.

Foremost among these resources are information systems and computer technology. These resources are claiming a great share of managerial commitment and expertise.

Studies using the Fortune 1000 indicate that senior management of these firms are essentially dissatisfied with the general unresponsiveness of their information systems to the ever-changing business climate. There is much fragmentation as well. The primary reason given is that the computer has not been adequately integrated into the core of the mainstream of business.

I contend that the reason for this, which management has up to now refused to face, it that management itself must bear the bulk of the re-

sponsibility because it does not have the proper knowledge, concepts, understanding and methods to administer, apply, monitor and coordinate the necessary integration.

To achieve this integration, management has to address key critical success factors that support the attainment of organizational goals and objectives. The dynamic role of information systems and computer technology, effectively applied, provides the firm with competitive products and/or services, as well as the information required to manage effectively. Today, information systems help us achieve goals that have been heretofore previously unattainable.

This challenge can only be met with a strategy -- a corporate information-systems strategy -- that allows management to manage today and plan for tomorrow. Management, in order to develop a proper and adequate strategy that remains attuned with products, services and markets, must continually develop, examine and improve its information systems.

The way an organization manages its information systems [resources] is a measure of its concern for the quality of its products and services and of its desire for both short- and long-term satisfaction. Those institutions with a positive philosophy and reputation for successful information systems have a distinctively competitive advantage.

3. Services' New Environment and Outlook

Everyone who is connected with service industries, for example, banking, brokerage, insurance, and the like, knows of the profound changes that are now altering the entire complexion of service industries in this country. The changes now underway are more basic and far-reaching than anything which has happened in these industries since the Great Depression.

Today, the name of the game is "competition." Barriers which have insulated service institutions from competition from each other and from other kinds of businesses are being whittled away. Firms are exposed to more geographic competition, competition from each other, competition from other firms which are permitted to perform more kinds of services, and competition from businesses outside the traditional service industries. Any firm which cannot succeed in meeting these new kinds of competition and in protecting a nitch of the market for itself, will not be able to survive in the new competitive environment.

All this requires new approaches and more cost-effective methods of operations. Organizations which succeed will be the ones which successfully rise to the challenge.

This challenge directs an organization to:

- Know which customers it serves, what services it sells them, and what returns it makes from each business;
- Know what value it is adding to the customer, what special economic role it is playing, and what need(s) it is fulfilling;
- Know what the critical elements of its operations are, and what mix of people, skills and resources give it a competitive edge -- know where it is strong and where it is vulnerable;
- Most importantly, develop a concept of the operations it performs -- this is a collection of activities (strategies) based on the assessment of customers, products, services, economics of operation, the competition, the organization itself, and the changing environment in which it operates.

People are selling their ability to provide service. One aspect of the complexity in delivering service is the maintenance of consistency of purpose and quality assurances when the service itself is constantly changing. Very few service firms manage to do this well.

Service marketing is quickly adjusting to an environment where the whole customer relationship is up for grabs from several, somewhat converging sources, and firms must go to the customer!

All of these issues can be addressed, but only through a comprehensive, integrated information systems management climate. With the proper attitide, managers can shift their energies more toward responsible contact with people -- the customers -- and away from the more routine, clerical activities.

For example, as financial services institutions pay more for the funds they receive, they must also earn more to be able to cover their costs. The higher earnings must come from a combination of (a) higher interest rates on loans and other investments, (b) charges for services performed for customers, and (c) reduced costs resulting from increased productivity and effectiveness.

The American Management Association suggests that this "...increased effectiveness...is a direct function of computer technology, properly applied, coupled with an effective information base that provides management with an uncompromising profile of where the organization is, where it wants to be, and where, in fact, it is going...."

No guru here! No crystal ball! Rather, a comprehensive, integrated information systems management approach for measuring, tracking and controlling progress.

It is clear that service institutions, indeed all businesses, are moving into a new era. In this environment management is going to have to be outstanding.

4. Information in the Practice of Management

The computer era of the 1960s and 1970s is giving way to the information era of the 1980s. The traditional emphasis on hardware and software is shifting toward a focus on information systems management. No longer are we concerned with the "by-the-pound" approach of computer output. Rather, the responsibility is changing to that of control over resources.

In the past, the typical corporate EDP function was the manufacture of paperwork. Computers processed the transactions of operating the organization. However, effective information systems require more advanced application of computer-based technology. They require, for example, interactive systems, database management systems, high-level user-oriented "languages," computer-based models, minicomputers [dedicated computers], personal computers [microcomputers], and telecommunications networking. These things are complicated enough for the technologist. Imagine how foreign these things are to the end user?! But, even so, effective information systems require greater user-management involvement than at any other time in commercial computing history. The challenge for all management is to define responsibilities more broadly to include the full scope of all facets of information systems.

Information systems management involves the integration of diverse disciplines and technologies and information-handling resources. Effective information systems management is a desirable goal because it leads to better support of the business activities and information needs of the organization. The critical interrelationships among all managers may be the single most critical success factor regarding the effectiveness of information resources in the business enterprise. Strategies are needed to effect and increase meaningful and continuing communication among all managers.

Information systems management supports organizations by allowing management to be constantly aware of the changing environment. It is adapted to the preparedness that permits accurate, effective, two-way flow of communication necessary for successful operations today. This will produce flexible management and organizational structures that permit responsible and successful responsiveness to change.

5. The Information-Technology Revolution

Technological innovation has begun to show managers the difference between management tools and management toys.

Since the introduction of the first commercial computer, some 32 years ago, computer usage has not reached its potential. The communications barrier between the technologist and the user-of-technology, a barrier that has basically existed since the advent of computing, has been a detriment to the progress of business.

Having technology and using it are two different things. While the concept of management information systems (MIS) has been around for a number of years, its real-life success stories are rare, indeed. The most critical reason for this is because of the gap between "data processors" and the people who use the systems.

However, herein also lies the fallacy: people don't "use" information-processing systems, they are a vital part of them! Yet, our so-called "users" have little comprehension of this esoteric technology. But, worse, data processors have only a rudimentary understanding of the businesses they are being paid to support and service. As such, too many managers refuse to risk their only product -- the decision -- in a game of what they call, "computer chance."

Yet, there are organizations that do succeed. Do they have better technology? No! Better technologists? Not necessarily. More adaptive management? Probably. Better grasp of concepts and communications? Absolutely!

But, success stories require more than simply grasp of concepts and communications. They need a coalition -- a merging of all resources, products, services and goals that compose any given enterprise. The user needs an understanding of what computers can do; a willingness, even an eagerness, to use the computer; and the knowledge to influence and make decisions pertaining to the information systems each is a part of. The technologist requires an in-depth understanding of corporate goals; a current knowledge of information-systems alternatives; an empathy with, and not antipathy for, users and their problems; and a willingness to shed any "empire-building" attitudes that have thus far retarded the progress of information systems management. The danger is that if all else stays the same, the proliferating growth of computer technology will only cause any gaps to become ever wider.

It is absolutely necessary for all managers to manage in ways that are markedly different from the techniques of today. All managers must be involved in the process because each is a vital part of it. All managers must determine needs, examine alternatives, undertake evaluations, and make decisions -- all because each manager is being reared in a data/information revolution. Most are not yet ready for it.

6. The Changing Role of the Information Systems Manager

There is a mission here. This mission involves management coming to terms with the fact that computers affect an organization's bottom line. As such, managers must unify their efforts and adapt to the "revolution" in order to reap its inestimable benefits.

The new role and responsibilities of this new "breed" of manager, the information-systems manager, may be defined as follows:

1. Enable the organization to understand, apply and control internal forces, and properly utilize external forces, that shape a firm's computing and business environments;

2. Apply proven management principles, particularly strategic planning, to information-systems functions;

3. Provide the organization with planning and development concepts and tools that effectively enable the firm to develop and implement a corporate computing/business strategy, and to provide a logical framework in which it can understand newest usage trends in information systems management and computer technology;

4. Provide the organization with a forum to exchange ideas and discuss opportunities with information systems specialists;

5. Comprehend the critical nature of information systems functions to overall organizational success.

The business impact of the new role and responsibilities of the information systems manager may be outlined as follows:

1. Tools and Technology: The management and utilization of computer technology and all other reasources; service levels, productivity measurements and improvements, and quality control; molding of resources to fit divergent business operations.

2. Organization and Communications: Developing organizational approaches that contribute to achieving goals; new organizational approaches that facilitate change.

3. People: Broadening the experience base so that change along with computers can be accepted and implemented.

7. Implications of Information Systems Management

The basic implications here are that:

- Timely, relevant, accurate information, effectively produced and disseminated, is a vital, valuable corporate resource. As such, it must be managed just like any other corporate resource;

- In the aggregate, information systems represent the totality of the organization itself, in that one vital corporate resource -- information -- is provided to the other vital corporate resource -- people.

Despite the extraordinary progress made in information management practices during the last ten years, a dichotomy still exists between producers of information services and the users of information. Most of the literature and programs in the field are concerned with technical details and thus fail to make the connection between information resources and the overall goals of the organization.

This gap must be bridged. The major goal is to deal with information systems management from the general management perspective: to show users and information specialists how to work effectively together to achieve organizational objectives.

Today, we have both a problem and an opportunity. Simply stated, it is the need and the desire on the part of management to mold information systems with computer technology into the core of the mainstream of business.

This represents an ever-widening divergence between the user and the computer, between systems management and ever-changing business requirements.... Over the years the user has viewed the computer as an enigma, something too complex to understand, never mind control. As such, the user has removed himself from the necessary efforts to fully integrate this information-processing and information-disseminating machine into the overall structure of his organization and his operations.

But, just like the corporate treasurers and comptrollers of the 1960s and 1970s, who became more knowledgeable and sophisticated in money-market techniques and investment vehicles, users are demanding more and more from their investment in their information machines. The proliferation of relatively cheap hardware is forcing users to come to terms with what has been, up to now, an esoteric tool. But, desire alone is not nearly enough.

We have a problem because over the last 32 years, since the first commercial computer was introduced, users have:

- Abdicated responsibilities,

- Distrusted data processors and computers,

- Lacked concepts and knowledge of computer utilization.

Users have been indifferent to the new concepts and requirements, and support a "business-as-usual" attitude. The computer has not yet reached its potential, but users must be educated to understand their role and responsibilities to effectively communicate with information specialists, and to learn new concepts and methods so that they may understand and apply new techniques that go beyond the status quo, that go beyond the norm.

We have an opportunity because MIS managers are the agents for change, are the architects of their organization's information capabilities. As

Richard Nolan states, "The migration from data processing to information management in the 1980s creates new roles for managers. Those who successfully identify and manage these new roles will rise in the organization."

There is, thusly, a requirement for a balance between the user's needs and objectives and MIS's role and responsibility to service and support those needs and objectives. These objectives involve a proactive, not a reactive, role in searching out opportunities to apply computer technology, where and when appropriate, to the solution of business problems.

We are all too familiar with the horror stories...of systems that did not meet expectations, that took too long to develop, that cost too much; of technicians who do not understand the businesses they serve; of users who have no understanding or appreciation of the complexities involved in meeting their information systems requirements.

Isn't it now time to put a stop to this nonsense and chaos? Isn't it now time for all of the promised cost-savings and labor-savings to be finally realized? Isn't it now time for results to equal expectations? Isn't it now time for integrated communication and cooperation -- that which is marshalled through a common understanding of the roles and responsibilities, concepts, strategies and techniques to achieve goals?

For years the computer has been little more than a number-cruncher, residing in the back-office. It is now, finally, coming out into the front-office. Indeed, we are learning that computers support organizing, staffing, controlling, planning and decision-making; indeed, the computer is truly a complete management tool!

The objective throughout is to achieve harmony -- to illustrate that information systems management represents the effective integration and management of all corporate resources to achieve a common purpose. Ultimately, this means that we recognize our opportunities for change, for creativity and innovation, in order to effectively address and resolve our business problems and achieve organizational unity and success.

It is a long, arduous task. Education and training may be the key.

8. Conclusions

Today, we hear much about the idea that business is in the midst of an "information explosion." Managers need information so that they may adequately perform their functions as a manager as well as take reasonable and prudent business risks.

Information is necessary to expand one's awareness and understanding of a real or perceived event or situation in order to achieve a specific objective -- that is, information is necessary to increase knowledge. The expansion of knowledge is necessary to reduce both uncertainty and risk in the management process. Effective information has a direct bearing on the level of reasonableness and prudence of business risks.

Information and computer technology are vital corporate resources. Organizations that can effectively manage these resources and integrate them into the core of the mainstream of business have a substantial competitive edge and they will be better equipped to achieve planned objectives.

The ability to develop insight into expectations and problems provides managers with the knowledge of where problems lie, their nature and severity, and feasible alternatives for their resolution and ultimate elimination.

LONG-RANGE ADP PLANNING:

A FEDERAL AGENCY PLANNING MODEL

Paul E. Matthews

The MITRE Corporation
McLean, Virginia 22102

This paper describes a long-range ADP planning process developed for a large Federal agency with assistance from the MITRE Corporation. The general acceptance of long-range ADP planning took place in the 1960s, with the introduction of planning, programming, and budgeting. Current planning issues include implementing paperwork reduction under the Paperwork Act of 1980 and strengthening internal controls to prevent fraud, waste, and abuse, under OMB Circular A-123. Because of the volume and complexity of laws, regulations, and policies which impact ADP management, proper integration of Federal regulation into the planning process is a continuing concern for agency ADP planners.

Topics which a Federal agency planning model must address are the functions it performs, the types of organizations required to perform these functions, the tools needed to develop an ADP plan and oversee its execution, and steps to implement the new planning processes. This paper describes the process in terms of: Planning organizations (top management involvement, organization of the function, and integration with operations); planning functions (preparation, execution, and maintenance); and steps to implement the planning model (establishment of formal agency planning; organization of responsibilities and relationships, implementation of a formal life cycle process, approval of interim plans, implementation of planning support databases, and initial plan execution).

Key words: Long-range planning; ADP planning; life cycle management; systems planning and control; Federal ADP procurement.

1. Introduction

With the publication in 1960 of Hitch and McKean's The Economics of Defense in the Nuclear Age, long-range systems planning became an accepted discipline. The application of quantitative techniques to management problems, as exemplified by planning, programming and budgeting systems, has advanced to routine use today in performance monitoring, workload forecasting, and optimization techniques in network analysis. Planning and managing computer systems require a similar formal analysis and discipline.

Long-range ADP planning is an appropriate function for the use of a structured discipline incorporating both quantitative and nonquantitative techniques. Areas especially amenable to quantitative analyses are the resources being managed in the ADP plan: equipment, computer programs, data, telecommunications, and facilities. Nonquantitative factors which can be addressed by means of a structured model include manual procedures, conversion strategies, and personnel retraining.

2. Current Issues in Planning

Based on The Brooks Act, Federal regulation has caused government agencies to develop computer systems which are obsolescent in the planning stage, become frozen in time, and today, represent opportunities for technological improvement. A current and continuing issue in ADP planning is the pressing need for ADP equipment and software replacement.

Changes in the Federal ADP regulatory environment impact long-range ADP planning requirements. The Paperwork Act of 1980 imposed a set of evolving requirements on the planning process. The most recent major impact occurred with the publication of OMB Circular A-123 in October 1981. These are but two recent policy changes which exemplify the continuing need on the part of Federal ADP managers to update the processes and procedures for ADP planning.

This is a continuing need; both ADP plans and ADP technology are continually changing. The Federal regulatory environment changes so frequently that the long-range ADP planning process, itself, is a moving target.

3. Planning Functions

There are three major functions associated with the planning process:

- Plan Preparation
- Plan Execution
- Plan Maintenance

Plan preparation consists of the development of an agency-wide, long-range ADP plan. Plan execution covers the interfaces of the strategic planning process with the day-to-day operational plans for individual system developments and acquisitions. Plan maintenance addresses the organizations required for planning, the functions they perform, and the support services required to maintain the long-range ADP plan.

3.1 Plan Preparation

A key ingredient in the recommended planning model is integration with the Federal budget process. To increase the probability of success, the planning and budgeting processes must be integrated into an organic whole. This integration ensures that when the plan has been developed, the budget for plan implementation will be known, since planning and budgeting take place simultaneously.

Accordingly, a planning calendar correlates the planning and budget cycles for the agency. Major milestones in plan development are the following:

- Developing the agency information strategy
- Preparing long-range ADP plan submission guidance
- Preparing agency component ADP objectives
- Analyzing requirements and constraints
- Assigning priorities to ADP objectives

Development of an agency information strategy is based on budget guidance available from OMB, policy guidance from higher monitoring authorities (including OMB, the Congress, the General Accounting Office and the General Services Administration), strategic planning, and feedback from prior planning activities. Produced by a management advisory council, the agency information strategy is the statement of overall ADP goals, which guide agency managers in determining ADP requirements.

When the overall ADP goals are defined, the agency ADP planners issue long-range ADP planning submission guidance as a formal planning call which parallels the budget call.

In response to the submission guidance, agency components then propose long-range ADP objectives for inclusion in the plan. Current systems, planned modifications and anticipated developments must all be included in the agency-wide long-range ADP plan. Once an ADP objective has been prepared, its resource requirements and constraints are analyzed. Preliminary life cycle costs are calculated, and alternative expenditure requirements projected over the system life. These are tied to the agency budget submission.

Each ADP objective, therefore, becomes a preliminary life cycle strategy (LCS) for a specific system development or acquisition. The LCS is the critical interface between strategic planning and operational planning. At the operational level, the LCS plans and monitors the system project, from proposal to operations and maintenance. The LCS furnishes analytical background for resource allocations and assignment of priority to the ADP development or acquisition. The long-range ADP plan for the agency contains all approved LCSs, consolidated, reviewed, and published by the agency ADP planners. Figure 1, Planning/Budget Cycle, displays the planning calendar and shows major tasks in the plan preparation function, as well as the appropriate decision level.

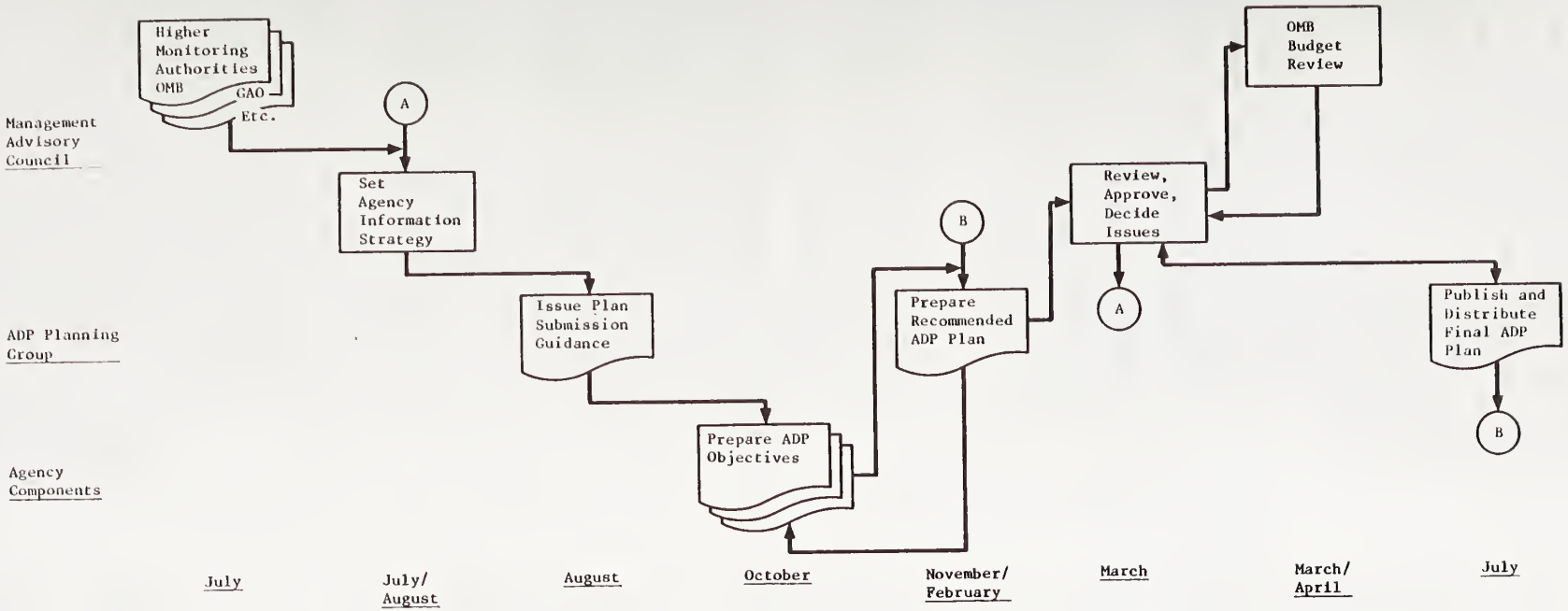


Figure 1

PLANNING/BUDGET CYCLE

3.2 Plan Execution

The planning process must be organic to system acquisition and development to ensure implementation of ADP plans. As a separate activity, the long-range ADP planning process could not keep track of policy changes and changing user needs; therefore, planning mechanisms would have to be rebuilt from planning effort to planning effort, with a resulting loss of institutional planning memory. Accordingly, the recommended planning model requires that any change, addition, or deletion of an approved ADP system acquistion or development in the long-range ADP plan be made by an update to the appropriate LCS. This feature ties the planning function directly to operational management.

Major tasks in the plan execution function are preparing and/or reviewing the LCS, management decision making, and reviewing the long-range ADP plan on a periodic schedule. Preparing or reviewing the LCS ensures the effectiveness of the long-range ADP plan. During plan execution, any new or changed ADP requirements must be specified whenever a new function is assigned to the agency, a technological opportunity is recognized, or a policy change is adopted affecting workloads and/or capacity. Entry of a new ADP system project into the plan (even outside the annual planning call) requires that an LCS be prepared and approved.

3.3 Plan Maintenance

Plan maintenance requires a planning support organization with two primary functions: user liaison and technical support. The user liaison function provides representation for agency component organizations and consulting support to them. From the perspective of the agency long-range ADP plan, the user liaison function expresses and/or reviews the mission needs of agency components.

The technical support function provides expertise in the following areas:

- Hardware
- Software
- Operations research
- Database and data administration
- Telecommunications

Figure 2, Automated Planning Support System, displays the functions and databases recommended to support long-range ADP planning.

The functional roles of the planning support organization include acquiring and maintaining all information for effective planning for the agency ADP management program. Three major elements of plan maintenance are: data collection and retrieval, project management, and quality assurance. The basis for these elements is a set of six logical planning support databases which provide baseline information. Each database contains the information required to support planning and the operations of each requested system development/acquisition. The databases serve as a library of information addressing technical management of the ADP program.

The databases are the following:

- Mission Needs Inventory (MNI) -- This database describes the agency mission, mission need/program, ADP capability, need assessment, and ADP objectives.

- Applications System Inventory (ASI) -- The ASI contains data describing past, current, and projected requirements of the various agency applications systems. It also collects and provides application system cost data.

- Computing System Inventory (CSI) -- The CSI serves two purposes: It collects and stores past, current and planned ADP capabilities, and it collects and stores ADP computing system cost data.

- Project Management System (PMS) -- The PMS tracks the movement of the LCSs through ADP processing. Thus, the agency ADP planners can quickly determine the status of any LCS. This database is the active repository of the agency long-range ADP plan. (Figure 3, Project Management System Information Contents, illustrates the types of information maintained for LCS tracking).

- Staff Skills Inventory (SSI) -- The SSI describes capabilities of agency ADP personnel, and it describes their work schedules.

- Cost Performed Index (CPI) -- The CPI collects and stores ADP and telecommunications cost and performance data for ADP planning, based on estimated life cycle costs and performance characteristics.

These databases serve as the focal point for all planning data for the agency ADP program. Installation and utilization of these planning support databases also provides a facility to respond to requests for information from outside monitoring authorities, including GSA, OMB, GAO and the Congress. The role of the planning support databases in formal project management and oversight of the development/acquisition

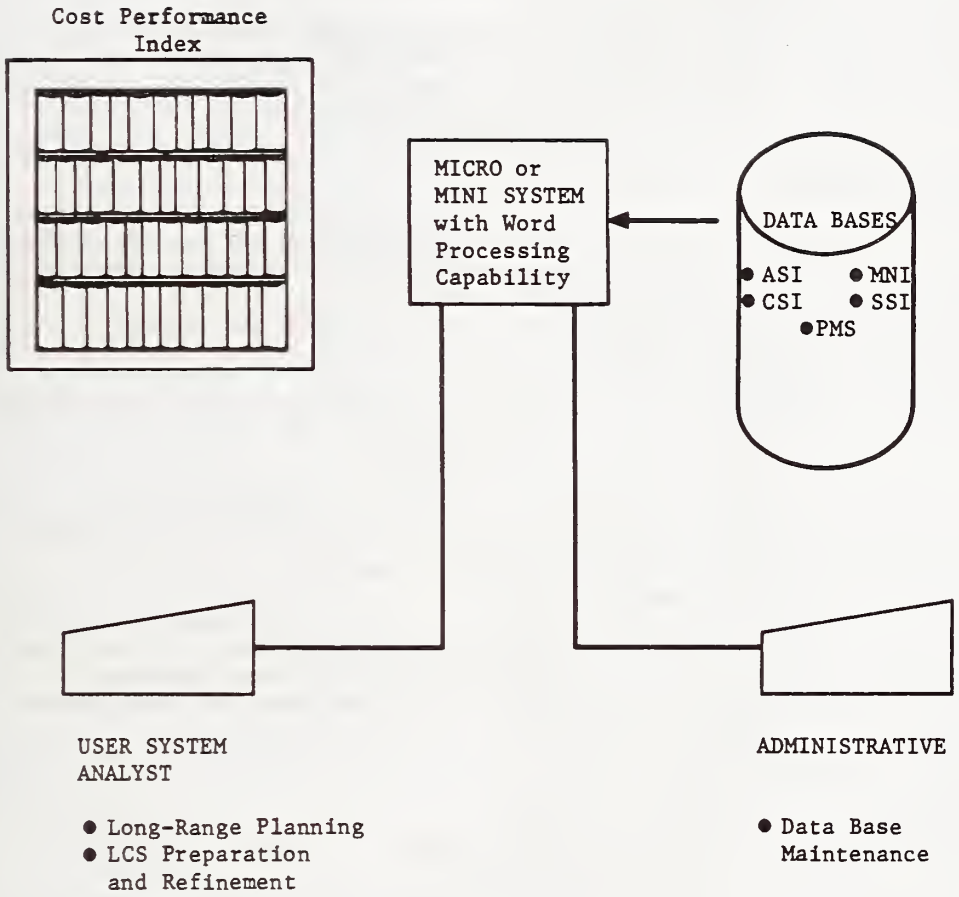


FIGURE 2

AUTOMATED PLANNING SUPPORT SYSTEM

(1) Initial Life Cycle Strategy

- Type of LCS and action (maintenance, enhancement, system development)
- Initial priority within the principal agency component
- Brief description of the request or need
- Desired completion time and operational constraints
- Processing status
- Initial estimate of time to complete the action
- Initial quantitative estimates of total benefit of completing the action if available
- Qualitative statement of benefit
- Planned or unplanned submittal

(2) Approved or Initiated LCS

- Current priority of this action among all pending agency actions, all pending component actions and all pending software maintenance actions
- Estimated start date (includng source and reason)
- Actual start date
- Estimated completion date
- Estimate of the number of staff days required to complete action
- Actual number of staff days used
- Estimate of total cost to complete the action
- Actual total cost
- Type of request (software modification, software addition, hardware addition, hardware upgrade)
- A-109 decision status
- Current work phases (i.e., phase identification -- analysis and design, development and implementation, or operation and maintenance), estimated start, projected completion date, actual start/completion, approval dates and level, tasks within phases, etc.
- For each task:
  - Estimated start date
  - Actual start date
  - Estimated completion date
  - Actual completion date
  - Intermediate milestones
  - Types of staff resources and level of effort required
  - Estimate of total cost to complete the task
  - Type and list of performer(s) (in-house, government laboratory, not-for-profit, private, commercial, etc.)
  - Expected and actual type of procurement and contract
  - Contract officer
  - Task manager
- Expected outlays subdivided into:
  - Hardware/software purchases
  - Site construction and modification
  - Government personnel costs
  - Contract Costs
- Actual outlays, as described for expected outlays

FIGURE 3

PROJECT MANAGEMENT SYSTEM INFORMATION CONTENTS

process is critical. Specifically, the PMS provides a tracking facility of all current ADP projects; completeness and consistency of the LCS database records can be monitored to provide a quality assurance mechanism over the agency ADP plan.

4. Implementation of Long-Range ADP Planning

Since ADP resources exist to serve agency program goals, ADP planning depends upon formal agency-level programmatic planning. This planning is necessary to ensure that the agency long-range ADP plan consistently serves user mission needs. Therefore, the first steps to installing a long-range ADP planning process are to establish formal agency planning and define the relationships between programmatic planning and ADP planning.

The implementation of a long-range ADP planning process is a phased, building block approach. As a first step, an executive, decisionmaking body is chartered to review and approve major planning and policy issues -- the management advisory council. Subsequent steps in the implementation plan are the following:

- Charter the planning functions
- Integrate system life cycle management
- Approve existing or interim plans
- Establish the planning support databases

4.1 Charter the Planning Functions

The specific functions to be assigned to the planning group, the management advisory council, and the agency user components must be defined in terms of operational workflows, areas of functional responsibility, and decisionmaking authority. A top management charter must be issued, designating these planning organizations, delegating authority, and starting the planning processes.

4.2 Integrate System Life Cycle Management

A formal structured system life cycle process is necessary to provide predictability and measurable products to the system building process; accordingly, one which builds on the LCS and supports the monitoring and oversight role must be adopted. Integration of the life cycle process requires that essential management and technical personnel be trained in planning and systems management techniques.

4.3 Approve Existing or Interim Plans

Since implementation of a new long-range ADP planning process is necessarily a phased activity, requiring at least six months to a year, current or existing plans can not simply be discarded. Instead, management commitment to the new planning effort requires that existing plans be carefully but quickly reexamined and approved, with any necessary changes. The existance of an agency ADP plan, although an interim one, lends management support to the planning activity.

4.4 Establish the Planning Support Database

This step consists of two tasks: (1) a survey of major information holdings throughout the agency which could affect the long-range ADP plan, and (2) building the requisite planning support databases. Information about data of the kinds described for the six databases must be inventoried and cataloged.

The most central tool of the recommended process, the project management system, should be initiated first, beginning as a manual project notebook to record life cycle strategies for each system acquisition/development. The system can then be phased in, as the database and associated procedures are implemented. The staff skills inventory should be built manually, based on specific ADP-related training and experience. It may be automated later, as funds permit. The cost performance index can be initially established by using subscription services to technical publications; bibliographic database reference tools can then be added, as resources permit, and the CPI can become an ADP technology forecasting center.

5. Conclusion

Long-range ADP planning in the Federal government is complicated by a myriad of laws, regulations and policies, each of which adds additional tasks. Most tasks are time consuming, and many requirements are conflicting in nature. A comprehensive and formal "plan to plan" is therefore necessary to give structure to the process and provide a predictable outcome. This paper has described major features of such a comprehensive planning process. These are the following:

- The planning model is based upon an organizational building block -- the planning support group.

- The process employs a top management group, the management advisory council, specifically chartered to function as a working, decision-making body, not a steering committee.

- The planning model is integrated with the Federal budget process.

- The model drives an individual system development project by means of a specific plan, the life cycle strategy.

- The process is based on a set of planning support databases which provide technical information for planning and ADP operations management. These are: mission needs inventory, application systems inventory, computing systems inventory, staff skills inventory, cost performance index, and a project management system.

A planning-driven life cycle management process was developed to fit this ADP planning process. The life cycle process incorporated both traditional system development and acquisition processes and the tasks and activities required by Federal policy. (It is cited in the references to this paper).

The agency for which this planning model was developed is currently preparing a long-range ADP plan and expects to implement key concepts of the planning model. Implementing the planning model in other agencies will provide experience to evaluate its effectiveness. The future direction MITRE plans for this comprehensive long-range ADP planning model is to monitor its implementation, with particular emphasis on those areas where it may need to be tailored to individual agency requirements.

REFERENCES

1. Anthony, Robert N., Planning and Control Systems: A Framework for Analysis, Cambridge: Harvard University Press, 1965.

2. Biggs, Clarks L., Managing the Systems Development Process, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1980.

3. Canning, Richard G., "Are We Doing Things Right?," EDP Analyzer, V. 13, November 7, July 1975.

4. Dietrich, Fred, "Discussion of Procurement Under A-109," presentation to The Federal ADP Procurement Conference, Washington, D.C., February 4-6, 1980.

5. Fried, Louis, "Long-Range Planning for DP Management," Data Processing Management, Auerbach Publishers, Inc., 1979.

6. General Services Administration, Federal Property Management Regulations (41 CFR 101-35, 36, 37).

7. General Services Administration, Federal Procurement Regulations (41 CFR 1-4. 11).

8. Head, Robert V., Strategic Planning for Information Systems, Wellesley, Massachusetts: Q.E.D. Information Sciences, Inc., 1979.

9. Heiker, Vincent E., "The Black Art of Systems Planning," Ken Orr & Associates, Inc's Users' Conference Proceedings, 1981.

10. Hitch, Charles J. and Roland N. McKean, The Economics of Defense in the Nuclear Age, Cambridge: Harvard University Press, 1960.

11. Lasdon, M., "Long-Range Planning -- Curse or Blessing?" Computer Decision, February 1981, p. 102.

12. Matthews, Paul, "A Planning-Driven System Life Cycle Model," Computing and Government: Interactions and Achievements, Twenty-First Annual Technical Symposium, Gaithersburg, Maryland: National Bureau of Standards, 1982.

13. Office of Management and Budget, "Preparation and Submission of Budget Estimates," Circular No. A-11.

14. "The Federal ADP Procurement Maze," Government Executive, April 1977, pp. 49-55.

PRODUCTIVITY THRU INTEGRATED INFORMATION RESOURCE MANAGEMENT

Malcolm Campbell

State of Missouri
Division of EDP Coordination
P.O. Box 809, Capitol Bldg.
Jefferson City, MO 65102

Applications of technologies occur in the organization in fluid, ambiguous environments with pressures, costs, and risks. A coherent approach for productivity is needed to apply to the mix of technological, organizational and economic issues.

Stratagic Information Resource Management involves addressing the situation by integrating the components of information, people, technology and capital within a volatile environment. The methods for working this out can be examined by applying the approach to selected emphases that we notice in the conversations of Data Processing people today.

The framework is as follows:

1. Tensions arise in the application of Data Processing to the organization.
2. We apply resources and viewpoints to the tensions.
3. We can do this on the basis of expressed principles.
4. Organizational learning takes place.
5. There are new tensions.
6. When applied appropriately, the components of this cycle contribute to productivity.

This presentation attempts to apply a rationale for the above process, considering specific current concerns of computer-based applications.

Introduction

You possibly have observed situations in which computer technology, when used to address organizational tensions in one area, provokes tensions in another. For example, relieving the application backlog by increasing user involvement may lead to a user documentation control problem. Such tensions continue to emerge in new places as technology changes.

Many of these tensions, at first, appear to be isolated from each other in the organization. The approach of this presentation is to suggest tools for integrating the processes associated with the tensions in pursuit of improved productivity in IRM for the organization.

The outline we will use will be "IRM Elements" and we will apply the categories of "Format For Each Element" to several parts of the "IRM Elements".

The principles we shall use are defined in the explanations of "Terminology".

The sections we have just referred to appear on the following page, in this order:

1. IRM Elements
2. Format For Each Element
3. Terminology

1. IRM Elements

1.1 Frontiers of Information Effectiveness

- 1.1.1 End User Systems Involvement
- 1.1.2 Management Structures as impacted by changing technology
- 1.1.3 Question-driven EDP
- 1.1.4 The Information Center
- 1.1.5 Decision Support Systems

1.2 Control Structures for Cost/Performance Realities

- 1.2.1 Methodologies and disciplines
- 1.2.2 Purposeful networking
- 1.2.3 Distributed Data Processing chaos avoidance
- 1.2.4 Office Automation as EDP
- 1.2.5 Micros where useful

1.3 Defining Functional Data

- 1.3.1 Conceptual data base modelling
- 1.3.2 Organization-wide data structures
- 1.3.3 Data for events and for human action

1.4 Growing Demands

- 1.4.1 Trimming the application backlog
- 1.4.2 Information accessibility for upper management

1.5 A Good Environmental Fit

- 1.5.1 Readiness as a stage in info-systems evolution

1.6 Employee Roles

- 1.6.1 The organization chart
- 1.6.2 Job definitions
- 1.6.3 Employee integrity, humanization and productivity

2. Format For Each Element

2.1 Initiating Tensions

2.2 Occurrences in the Real World

2.3 Resulting Tensions

2.4 Goal-Oriented Constructs

- 2.4.1 Goals
- 2.4.2 Technology
- 2.4.3 People
- 2.4.4 Information
- 2.4.5 Capital

2.5 Integration

3. Terminology

3.1 "Tension"

A condition in the organization that management will not allow to continue as it is without some attempt to modify its effects so as to enhance organizational productivity

3.2 "Organizational Learning"

The changing in an organization that enables it to move toward its goals under new conditions and with new principles

3.3 "Dialectic"

The dynamics, in the organization, of tensions being guided toward goal achievement, resulting in new tensions

3.4 "Integration"

Processes in the organization that work toward:

- 3.4.1 Relevance
- 3.4.2 Attachment of common categories
- 3.4.3 Incorporation of conflict
- 3.4.4 Holism - synthesis thru higher order feed-back, the whole greater than the sum of its parts

3.5 "Constructs"

The reference subsystems applied like models to the dialectic process. These subsystems come out of the organization culture (like "participatory management") or out of technology developments (like "structured design methodology") or out of applied behavioral sciences (like "organization development" or "transactional analysis".)

Constructs aren't always extant reference subsystems; they may be perspectives and images

3.6 "IRM"

Information Resource Management in its commonly accepted sense, with following emphasis:

Strategies for utilizing information as an organization resource and developing information tools in a functional sense

3.7 A Major Theme of This Presentation

Productivity is achieved thru cycles of organizational learning as integration grows out of application of goal-oriented constructs to the dialectic process

4. End User System Involvement

4.1 Initiating Tensions

4.1.1 User is intimidated by the technical literacy gap
4.1.2 User insists upon fruits of technology advances

4.2 Occurrences in the Real World

Hardware and software tools at prices the end user can pay

4.3 Resulting Tensions

4.3.1 Difficulty in relating the new tools to his priorities and the organization context
4.3.2 Uncertainty as to the havoc that can be played by uncontrolled automation

4.4 Goal-Oriented Constructs

4.4.1 Goals

Happy, involved users; answers for training, turnover, control and standardization

4.4.2 Technology

Application aids that are functionally defined and documented

4.4.3 People

HRD that is functional; education addresses information dynamics--not specialized technology; conflict management

4.4.4 Information - Conceptual data bases

4.4.5 Capital

Relate dollar expenditures to user's payback, such as matrix organization, zero-based budgeting, functional cost allocation

4.5 Integration

4.5.1 Higher order intervention for application development

4.5.2 Component sub-groups; heterogeneous teams; walk thru's

5. Management Structures as Impacted by Changing Technology

5.1 Initiating Tensions

5.1.1 Problems of 1960-1975

Acceptance
Training
Conversion
Turnover
Documentation
Control

5.2 Occurrences in the Real World

5.2.1 Proliferation
5.2.2 Improved Cost/Performance
5.2.3 D.P. moving away from being a job-oriented division
5.2.4 Permissiveness, self-advancement, questioning
5.2.5 Data availability
5.2.6 Less demand for traditional middle manager's work

5.3 Resulting Tensions

Management responsibility for:

5.3.1 Data base modelling and administration
5.3.2 User involvement in application development
5.3.3 The Information Center
5.3.4 Office automation
5.3.5 Employee motivation and direction

5.4 Goal-Oriented Constructs

5.4.1 Goals

- Productivity of the infrastructure schema
- Organizational learning
- Effective management according to objectives

5.4.2 Technology

Information Center (an organization structuring concept)

People structues having network and relational components as dist. from hierarchical (e.g. because of electronic communication)

5.4.3 People

Consensus, participation, relevance, currency, teams, task force

5.4.4 Information

IRM - what the manager is managing; Conceptual data base supporting goal-oriented management

5.4.5 Capital

Recognition of people costs as high-cost areas

5.4.6 Integration

5.4.6.1 Root Decisions

- Information loci
- Communication, job description
- Priorities

5.4.6.2 Conflict management

- Span of control
- Data authority
- Procurement

5.4.6.3 Common categories for data, technology, people, capital

- modularity
- involvement
- extension
- definability

5.4.6.4 Relevance demonstration

- manage data for the sake of people
- manage people for the sake of data
- manage technology for the sake of people
- manage people for the sake of technology

6. The Information Center

6.1 Initiating Tensions

6.1.1 Need for user involvement in system development

6.1.2 New tools offered to the user spawned needs for:

6.1.2.1 technical assistance
6.1.2.2 large central site overhead
6.1.2.3 controls, documentation, standardization
6.1.2.4 continuity thru turnover
6.1.2.5 clarity of systems definition
6.1.2.6 system cohesion, validity, credibility, efficiency, effectiveness
6.1.2.7 all the problems of the 60's

6.2 Occurrences in the Real World

6.2.1 Centralization/decentralization issues as a question of organizational structure, knowledge distribution, and application systems (no longer primarily a hardware focus)

6.2.2 Expansion of system software and its usage/maintenance requirements

6.2.3 Large data base development for the total organization

6.2.4 Application development disciplines

6.3 Resulting Tensions

6.3.1 Need to avoid redundancy of systems/data; need for coordination of scattered, independent projects; need to maintain systems integration

6.3.2 Organizational retrofit; such as relationship of the EDP managers to the Information Center

6.3.3 User friendly tools are not friendly enough

6.4 Goal-Oriented Constructs

6.4.1 Goals

User access, organizational integration, clear understanding of responsibility, effective application of resources

6.4.2 Technology

6.4.2.1 Access

- Data in residence, machine can address it
- Language by which user can address the data
- Application Development avenues that are: generalized, question-driven and need-defined
- data becomes information thru interpretive use of DSS, graphics, office automation and user languages

6.4.3 People

EDP employees as enablers rather than doers ("Information Center" is essentially a people concept)

6.4.4 Information

Reservoir of raw data for access, seen as a total information pool

6.4.5 Capital

6.4.5.1 Investment seen in user tools, training; enabler training
6.4.5.2 Costs of actual development will go way down
6.4.5.3 Cost of preparation and overhead will go down

6.4.6 Integration

6.4.6.1 Holism thru higher order feedback

- Decision to change the character of the EDP function
- Essence of information services is found in what happens

6.4.6.2 Conflict incorporation

- The disparities come out of the closet
- Whether a user gets what he pays for is an open matter
- Ownership of data is confronted
- What is friendly is openly questioned

6.4.6.3 Common categories

- Decentralized components are still a part of the total organization
- The EDP person may find his place at the Center or at the staff of a user

6.4.6.4 Relevance

- Organizational goals more visible relative to departmental goals

7. Decision Support Systems

7.1 Initiating Tensions

7.1.1 Existing computer output vs. potential computer output vs. management needs

7.1.2 Traditional computer output too voluminous for the manager

7.1.3 Computer has a fantastic capacity for correlating variables, forecasting, and highlighting unusual conditions

7.1.4 Managers' needs;

- "What-if" options
- Deviation from standards needing attention
- Condensation of transaction conditions for quick evaluation

7.2 Occurrences in the Real World

7.2.1 Packaged "what if" modes (like financial planning)

7.2.2 Decision theory

7.2.2.1 How managers make decisions
7.2.2.2 Information/logic relationships

7.2.3 Management science

7.2.3.1 Quantitative methods
7.2.3.1 Operations research

7.2.4 Management access to computerized systems

7.2.5 Schemas for data base interface

7.3 Resulting Tensions

7.3.1 Need for adaptive determination of what information is of interest and what it means (its significance)

7.3.2 The essence of management decision-making often mitigates against standardization, such as what is important is not always the same or is not predictable

7.3.3 The manager doesn't manage like a systems analyst. Management use of information is often ad hoc, non-structured, self-generated

7.3.4 A manager who wants help does not want to mechanize his prerogatives

7.3.5 There is a very loose use of the term "Decision Support Systems", especially by vendors

7.4 Goal-Oriented Constructs

7.4.1 Goals

Extraction of information which is interesting to the manager

Making available what that information signifies

7.4.2 Technology

Heuristic systems, responsive to discovery and uniqueness (generalization and accessing the data base)

7.4.3 People

The manager's role as applying strategy and values, rather than hunting for information, calculating, and drawing statistical deductions

Scientific management as based on substantive forecasts rather than guesswork

Management objectives as organization goals rather than game playing for position

7.4.4 Information

System design reflecting meaningfulness in the mind of the decision-maker

Environment for unstructured meaning built into the system by the data base designer (in doing the model)

Extraction of meaning from data by use of algorithms

7.4.5 Capital

Cost trade-off of computer system against manager's time and the values of good decisions to the organization

7.4.6 Integration

7.4.6.1 Relevance

Organization questions like "What do we do next to accomplish our mission?", "What do we do next for growth?"

These questions become related to system design and also become related to management practice

7.4.6.2 Common Categories

Management science and data base administration share in common project activity

Systems realism and management pragmatism share in common project activity

7.4.6.3 Conflict Incorporation

"What DSS is" gets discussed

The manager looks at his prerogatives and what he does best that the mechanized system won't do

The manager looks at intuition and accepts it

7.4.6.4 Holism thru Higher Order Feed-back

Interventions to:

Change management roles

Develop new kinds of systems

Use the computer as a different kind of tool

Make the organization more authentic, operating on justifiable processes more than upon whim, pressure and prejudice

8. Conclusion

8.1 The EDP environment is fluid but it doesn't have to be random. It is many-voiced but does not have to be counter-productive. It fosters conflict, but does not have to be destructive.

8.2 We find ourselves seeming to seek stability. What we actually want is growth through processing our areas of tension.

8.3 There are factors which threaten to be fragmenting. What makes those factors integrating is higher level intervention using globally related constructs.

8.4 What makes for growth is goal-oriented coping with tension.

8.5 EDP volatility will not go away. But, because we have sound goals and possible plans, we can address the volatility with tools that make sense.

CPEUG82

"Improving Organizational Productivity"

Financial Management Considerations

PRICING STRATEGIES IN PROCUREMENTS CONDUCTED UNDER THE BASIC AGREEMENT

Thomas G. Morrison

MCAUTO Systems Group, Inc.
2 Research Place
Rockville, Maryland 20850

ABSTRACT

The General Services Administration's Basic Agreement procurement program was implemented for procuring large scale commercial remote computing services. The intent was to provide a flexible pricing environment where major processing requirements could be satisfied with highly discounted customized pricing. In practice, the pricing flexibility has resulted in non-productive vendor gamesmanship. Successful vendors have learned to create pricing strategies which result in evaluated system life costs being a fraction of the actually experienced system life costs. Techniques are available which can create substantial disparities between the evaluated system life costs and the actually anticipated system life costs. By carefully evaluating workload projections and structuring a pricing scheme to fit the evaluation deficiencies a vendor can exploit the deficiencies and dramatically reduce its evaluated cost. Action must be taken by the government community to stop these pricing practices. Mandating that vendors utilize standard commercial pricing practices would help in resolving the problem.

Key words: Basic Agreement Solicitations; Evaluation of System Life Costs; Teleprocessing Services Procurements; Unbalanced Pricing; Workload Forecasting.

1. Introduction

Basic Agreement procurements were introduced by The General Services Administration (GSA) to provide lower cost services for major processing requirements. In practice, Basic Agreement procurements have not resulted in lower costs but have resulted in non-productive vendor gamesmanship. The successful vendors have creatively priced their proposals with low evaluated costs knowing the actual costs will be substantially higher.

Credits, decreasing volume discounts, bundled unit discounts, etc. have been successful schemes for winning government business. The gaming vendors have been obtaining multi-million dollar contracts with very high profit margins. They have taken some risk but the rewards have been substantial.

GSA has recognized the problem and issued an amendment to the Basic Agreement in September 1981, which precluded the pricing schemes mentioned above.(1) This is a first step but it is only a beginning and does not solve the problem. Some government users seem to feel the GSA amendment on unbalanced pricing has eliminated the problem. It has not.

The problem of unbalanced bidding is an escalating one. The relatively simple means of gaming utilized in past procurements have already given way to more sophisticated unbalancing techniques. Most vendors are painfully aware of these practices and know to remain competitive they must find ways to exploit the evaluation methods or simply drop out of the market.

This paper will explore and expose techniques which have been and/or could be utilized by vendors in unbalancing proposals. The intent is to educate and sensitize the government community to the problem. The Basic Agreement procurement environment in its present status is intolerable and action must be taken swiftly by the government community to rectify the problem.

2. Exploiting the Evaluation Criteria

Pricing schemes which unbalance proposals

exploit particular characteristics of the current evaluation methods. These characteristics include:

- average utilization - which often fails to account for actual distribution
- workload projections - which are over or underestimated
- inaccurate mix of forecast resource consumption
- the cyclical nature of actual utilization
- the utilization of unevaluated resources

By identifying any of these deficiencies in any element of the evaluated workload, a vendor can exploit the evaluation criteria.

The situation becomes significantly more complex as you consider all the levels within the evaluated workload where these deficiencies may be identified. I have identified several different levels where pricing schemes can be introduced. Each level will be discussed more fully but summarized they are as follows:

I) Aggregate use patterns

II) Inaccurate evaluated use of processing, storage, connect, etc.

III) Inaccurate evaluated weighting of benchmarked programs.

IV) Inaccurate evaluated usage of machine resources.

V) Highly sensitive machine resource consumption.

VI) Modification of the resource accumulator.

VII) Dedicated or bulk units.

By combining the five exploitable characteristics with the seven levels where evaluation deficiencies can exist, you have 35 combinations that must be accurately accounted for in the benchmark and the forecasted workload. If covering 35 contingencies doesn't sound like a major problem, remember within each level there are multiple elements. For example, Level II contains processing charges, storage charges, communication charges, printing charges, courier charges, etc. If you're still not convinced there is a major problem in preparing your forecasts, consider that the elements just listed will also have multiple elements. Communications, for example, will probably include three different speeds each priced separately and probably be accessed through two or three different networks each priced separately. Undoubtedly, this appears to be a numbers game. This will become even more evident as we review some specific techniques.

3. Examples of Unbalancing Techniques at Each of the Seven Levels

To understand the mechanisms of unbalanced pricing it is necessary to review specific examples in detail. As will be demonstrated in the following examples, unbalancing each succeeding level will become progressively more sophisticated, more difficult to implement, and more difficult to detect. As agencies restrict certain types of pricing, proposed pricing schemes will move more deeply into the pricing structures. This process has already begun and is escalating the problem of unbalanced pricing.

3.1 Level I: Aggregate Usage Patterns

The first level, aggregate usage patterns, has already been discussed extensively.(2) By reviewing invoices, vendors attempt to find unevaluated anomalies. Similarly, vendors look for overall patterns of project usage and growth potential. This has been the level where some vendors have been so successful with decreasing volume discounts, conditional credits, etc. Although some of these techniques are forbidden under Amendment No. 4 to the Basic Agreement, the impact of unevaluated elements can still be dramatic.

Figure 1 shows the significance of improperly evaluated discounts or credits, or of not evaluating a portion of the actual workload. The X axis is the percent discount applied against standard list prices. The lower Y axis is the discount level actually achieved or the evaluated discount level. The upper Y axis is the ratio of evaluated cost to actual cost. The lines represent the percent of

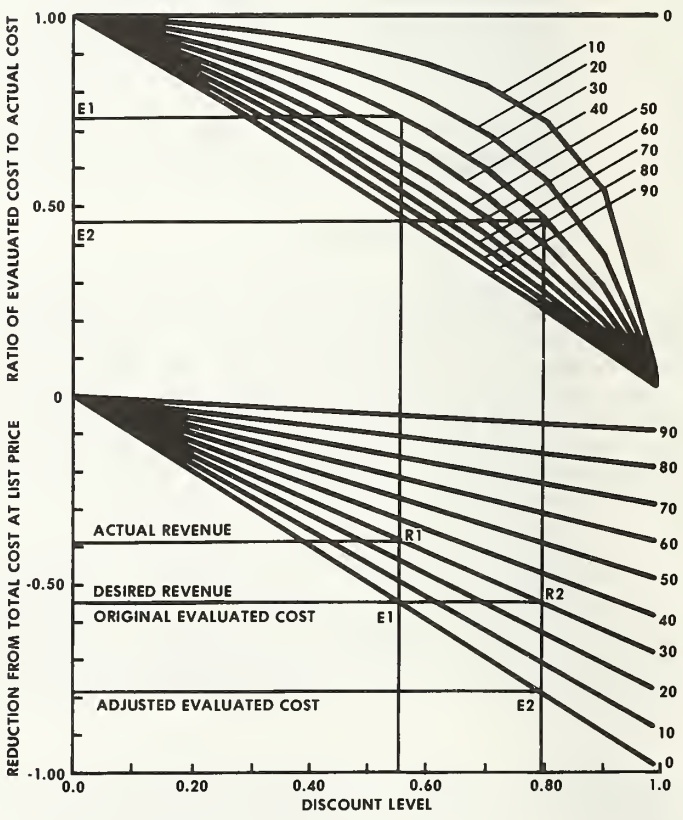


Figure 1. The Impact of Improperly Evaluated Discounts

the total cost which escapes the evaluated discount at list price. In constructing the graphs, all unevaluated processing was charged at list price.

The plotted example assumes a cost proposal with a 55% discount for all evaluated processing. Initially the 30% non-evaluated processing has not been considered. The evaluated cost with these assumptions is E1. The actual revenue produced will be substantialy higher at R1. To adjust the actual revenue to the desired revenue move down the 30% line to R2 and the new discount level. The new evaluated cost will be E2 and the actual revenue will equal the desired revenue. By making this adjustment the evaluated cost will be approximately 45% lower than it was before the adjustment. The ratio of evaluated cost to actual cost on the upper curves will move from E1 approximately 72% to E2 approximately 45%.

By reviewing other combinations of discounts it should be readily apparent that no segment of the actual workload can be overlooked. As discount levels increase the sensitivity to unevaluated workload elements is increased.

3.2 Level II: Usage of Processing, Communications Storage, etc.

The second level involves inaccurate forecasting by the government of resource utilization. Inaccuracies at this level can provide a significant opportunity for evaluated price reduction. Suppose the evaluation is based on a mix of resources like this:

Evaluated	
$ 300,000	Connect
300,000	Storage
400,000	Processing
$ 1,000,000	

But the actual is anticipated to look more like this:

Actual	
$ 100,000	Connect
500,000	Storage
400,000	Processing
$ 1,000,000	

By applying a large discount to the overestimated usage and a small discount or no discount to the underestimated usage, a vendor could reduce its evaluated cost significantly.

The following Table shows the results of applying a 75% discount to connect charges, no discount to storage and a 25% discount to processing.

	Evaluated	Actual
Connect	$ 75,000	$ 25,000
Storage	300,000	500,000
Processing	300,000	300,000
TOTAL	$ 675,000	$ 825,000

The actual cost in this case will be 22% higher than the evaluated cost.

3.3 Level III: Benchmarked Programs

The third level of opportunity involves inaccurate weighting of benchmark runs. Again, I'll use an example.

The following table shows the results of six benchmark runs and the weighted and actually anticipated usage patterns.

Benchmark	CRU	SEC	I/O	Evaluated	Actual
1	20	12	8	20%	20%
2	10	9	1	10%	15%
3	30	24	6	30%	40%
4	10	5	5	10%	10%
5	15	1.5	13.5	15%	10%
6	15	3	12	15%	5%
TOTAL	100	54.5	45.5	---	---

Runs 2 and 3 are underestimated and runs 5 and 6 are overestimated. By modifying the weighting of the resources in a billing algorithm, vendors can again appreciably reduce evaluated costs. The following table shows the results.

	CRU Equivalents		
Algorithm	Evaluated	Actual	% Increase
1.0(SEC)+1.0(I/O)=CRU	100.0	100.0	0%
1.1(SEC)+.9(I/O)=CRU	100.9	102.9	2.0%
1.5(SEC)+.5(I/O)=CRU	104.5	114.5	9.6%
1.8(SEC)+.2(I/O)=CRU	107.2	123.2	14.9%
2.0(SEC)+ 0(I/O)=CRU	109.0	129.0	18.3%

The analysis is over-simplified, but the results are significant particularly if you consider the relatively small errors that were made in forecasting. When some runs utilize surcharged software, similar results could be accomplished by modifying the surcharges.

3.4 Level IV: Machine Resources

The fourth level is similar to the third but here the mix to be analyzed is the consumption of machine resources, i.e., cycles, I/O, etc. Using the same six benchmark runs, I've created the hypothetical benchmark found in Table 1. Notice that the proportion of CPU resources has increased between the modified benchmark mix and the actual mix. By carefully setting up benchmark runs, vendors can slant the resources consumed toward a particular resource. Modifying the algorithm factors or multipliers can then reduce the evaluated cost but maintain revenues in actual use. By accounting for this increase in the alogrithm (.2(CPU)+1.8(I/O)= CRU) the evaluated CRU's would be 45.8 while the actual CRU would be 92.8. This change results in a reduction in the evaluated cost of 50% for processing.

Table 1. Impact of Benchmark and Algorithm Modification

BENCHMARK RUN	UNMODIFIED BENCHMARK								MODIFIED BENCHMARK				
	ACTUAL WORKLOAD		STD ALGORITHM[1]			MODIFIED ALGORITHM[2]			MODIFIED BENCHMARK		MODIFIED ALGORITHM[2]		
	RESOURCE MIX		CPU	I/O	CRU	CPU	I/O	CRU	RESOURCE MIX		CPU	I/O	CRU
	CPU	I/O							CPU	I/O			
1	60%	40%	12	8	20	2.4	14.4	16.8	90%	10%	3.6	3.6	7.2
2	90%	10%	9	1	10	1.8	1.8	3.6	99%	1%	2.0	0.2	2.2
3	80%	20%	24	6	30	4.8	10.8	15.6	95%	5%	5.7	2.7	8.4
4	50%	50%	5	5	10	1.0	9.0	10.0	80%	20%	1.6	3.6	5.2
5	10%	90%	1.5	13.5	15	0.3	24.3	24.6	70%	30%	2.1	8.1	10.2
6	20%	80%	3	12	15	0.6	21.6	22.2	60%	40%	1.8	10.8	12.6
TOTAL	—	—	54.5	45.5	100	10.9	81.9	92.8	—	—	16.8	29.0	45.8

1 — STANDARD ALGORITHM CPU + I/O = CRU

2 — MODIFIED ALGORITHM (.2) CPU + (1.8) I/O = CRU

The extent to which vendors can modify the benchmark runs will depend upon the particular proposal. As with alternate, unbalanced cost proposals, vendors make changes, notify the government, and see which ones the government will accept. If they reject a change, then they will simply have to rerun the benchmark. In other words, there is basically no risk.

In many cases, benchmark modification will not even be necessary because benchmarks are by definition a subset of the actual processing. A vendor simply needs to identify what resources a full-scale run will consume and structure the algorithm accordingly.

3.5 Level V: Highly Sensitive Machine Resources

The fifth level involves highly sensitive machine resource utilization. A vendor could increase revenue without increasing the evaluated cost by establishing a base level of consumption for the benchmark and then accounting for high level peaking in actual use. Increasing the weighting of the sensitive resource in the algorithm would understate CRU's in the benchmark. The evaluation wouldn't account for the peaking from the benchmark level which would occur in operation. Similarly, a non-linear algorithm could be constructed which would dramatically increase processing costs for any variation from the benchmark.

3.6 Resource Accumulators

Even more dramatic are the results of making modifications to the resource accumulator. The resource accumulator is software that monitors the actual computer resources used and reports them to the billing algorithm.

Vendors have tremendous flexibility in the manner in which resources are accumulated. For example, a vendor might not report the first two CPU seconds resulting in a lower cost for small jobs but virtually no reduction for production type runs. Graphically, the results of this modification appear in Figure 2.

Point A is the cost for the benchmark on the incumbent's system. Point B is the cost for a normal production run. In establishing the multipliers for the benchmark, the relationship between Point A and Point B is utilized. The benchmarked vendor reports a cost at Point C. By applying the evaluation multipliers the vendor is evaluated at Point D. However, the actual cost for the production runs will not be Point D but approximately four times higher at Point E.

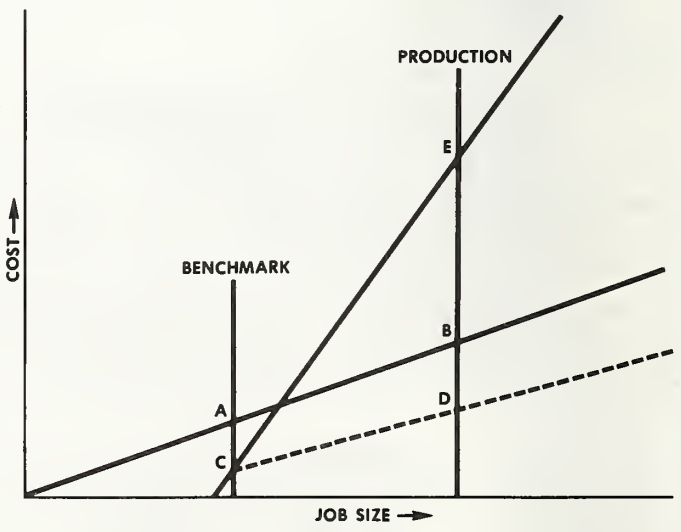


Figure 2. Modification of The Resource Accumulator

Other modifications could produce similar or more pronounced results. For example, resource accumulation could be minimized at only the benchmark level or a step function could be employed to dramatically increase costs with a slight increase in job size over the benchmark. The only limit on gaming with the resource accumulator is a vendor's creativity.

3.7 Dedicated Units

The acceptance of "Dedicated" or "Bulk" resource units introduces a new area of potential vendor abuse. Because, a "Dedicated" unit works like a dedicated machine, you introduce a unit with the characteristic of producing differing quantities of CRU's per dedicated unit. The difference between a dedicated unit and a dedicated machine is that the dedicated unit can be expanded instantaneously unlike the dedicated machine. This difference can be exploited for evaluation purposes.

Suppose the new unit is defined as being capable of producing 100 CRU's per minute. In one hour you could generate up to 6000 CRU's, provided you processed exactly 100 CRU's per minute. On a dedicated machine this is possible, with a "Dedicated Unit" it is not. The reason is that with the dedicated machine as you load the machine you cannot exceed its capacity, so as your workload peaks your turnaround time degrades. With a "Dedicated Unit" as your workload peaks you trigger additional units that increase your cost without degrading your turnaround time.

To analyze the impact of these units, I constructed a model which randomly assigns usage through the day with a morning and an afternoon peak. In the table below, sessions of random lengths between 15 and 45 minutes were assigned CRU's based upon an average number of CRU's per connect minute for the total workload. The model evaluates the peak minute within each hour and assigns sufficient "Dedicated Units" to cover for that hour. The evaluated processing cost is based on the number of dedicated units required to cover the average workload.

Connect Hours		CRU's	
100.517		15650.4	
DAY	HOUR	Peak number of users	CRU Peak
1	1	14	198
1	2	15	167
1	3	18	229
1	4	17	212
1	5	13	154
1	6	14	159
1	7	24	208
1	8	21	260
1	9	14	229

EVALUATED COST	EVALUATED UNIT PRICE
$2,610	$0.17
ACTUAL COST	ACTUAL UNIT PRICE
$18,160	$1.16

The actual cost with this "Dedicated Unit" will be nearly 7 times higher than the evaluated cost.

Another "Dedicated" approach might use memory requirements instead of time as the constraining element. This type of pricing would work in the same manner as the time constrained "Dedicated Unit" just described. Within a given block of memory equal to one "Dedicated Unit" you could generate different quantities of CRU's. If your workload memory requirements expanded you would be charged for more units instantaneously.

Similarly, communication ports, I/O, or other elements of processing could be used as the constraining element. Structuring an evaluation which would adequately account for and anticipate this volatile characteristic of a "Dedicated Unit" would be virtually impossible.

Clearly, not all dedicated units would work in the manner described. However, even with a dedicated unit a gaming vendor can gain a significant advantage in its evaluated cost.

4. Some Further Refinements

Another factor to be considered in this kind of analysis is the impact of changing overall workload projections. By creating a disparity between the evaluated and actual costs a vendor has dramatically destabilized the workload projections. This discrepancy can also be exploited for cumulative effects.

Our Level II (inaccurate forecast) analysis looked like this when we finished:

	Evaluated	Actual
Connect	$ 75,000	$ 25,000
Storage	300,000	500,000
Processing	300,000	300,000
TOTAL	$ 675,000	$ 825,000

Because of the work we did in Level IV (benchmark modification) a new analysis must be conducted for Level II. The imbalance now is partly constructed and partly inaccurate forecasting.

	Before Discount	
	Evaluated	Actual
Connect	$ 300,000	$ 100,000
Storage	300,000	500,000
Processing	200,000	400,000
TOTAL	$ 800,000	$1,000,000

Applying the following discount schedule:

Connect	-100%
Storage	+20%
Processing	+50%

	After Discount	
	Evaluated	Actual
Connect	$ 0	$ 0
Storage	360,000	600,000
Processing	300,000	600,000
	$ 660,000	$1,200,000

If in this case a vendor's target revenue was $825,000, other costs could be reduced by $375,000/year. However, a vendor can't reduce processing and storage costs by that amount because for every $1 reduction in evaluated cost, actual revenue is reduced by $2. Let's suppose this case has a 5-year system life cost with an estimated conversion cost of $1,875,000. The vendor could cover the cost of conversion with no increase in evaluated cost.

The results of our analysis follows:

	Straight Discount	Creative Discount + Algorithm Application	
	Evaluated=Actual	Evaluated	Actual
Year 1	$ 825,000	$ 660,000	$1,200,000
Year 2	825,000	660,000	1,200,000
Year 3	825,000	660,000	1,200,000
Year 4	825,000	660,000	1,200,000
Year 5	825,000	660,000	1,200,000
	$4,125,000	$3,300,000	$6,000,000
Conversion	1,875,000	0	0
	$6,000,000	$3,300,000	$6,000,000

Applying the same technique to the results of our benchmark with the modified resource accumulator also demonstrates the significance of structuring discounts to exploit workload inaccuracies.

	Before Discount	
	Evaluated	Actual
Connect	$ 300,000	$ 100,000
Storage	300,000	500,000
Processing	135,000	540,000
TOTAL	$ 635,000	$1,140,000

Remember the target revenue was $825,000/year with $1,875,000 for conversion.

Applying the following pricing modifications:

Connect	-80%
Processing	+100%

	Evaluated	Actual
Connect	$ 60,000	$ 20,000
Storage	60,000	100,000
Processing	270,000	1,080,000
TOTAL	$ 390,000	$1,200,000

The 5-year evaluated system life cost would be $1,950,000 while the actual 5-year cost would be $6,000,000.

Having reviewed one case specifically, let's turn to the general case. Figure 3 demonstrates the significance of the preceding method. The vertical axis is the evaluated cost. In all cases the actual revenue will be $1,000,000. The plotted lines represent the percent of the systems life cost due to processing. This percentage is calculated after applying the basic overall discount levels anticipated, i.e., a 50% across-the-board discount. The horizontal axis shows the benchmark disparity. Because processing is underestimated I have increased the price of processing by 100% -- returning it to list price. The plotted example assumes that 60% of the system's life cost is due to processing and the benchmark disparity is 50%. A vendor's total evaluated cost then becomes $400,000.

5. Conclusion and Recommendations

Creative pricing is a major problem for those conducting procurements under the Basic Agreement.

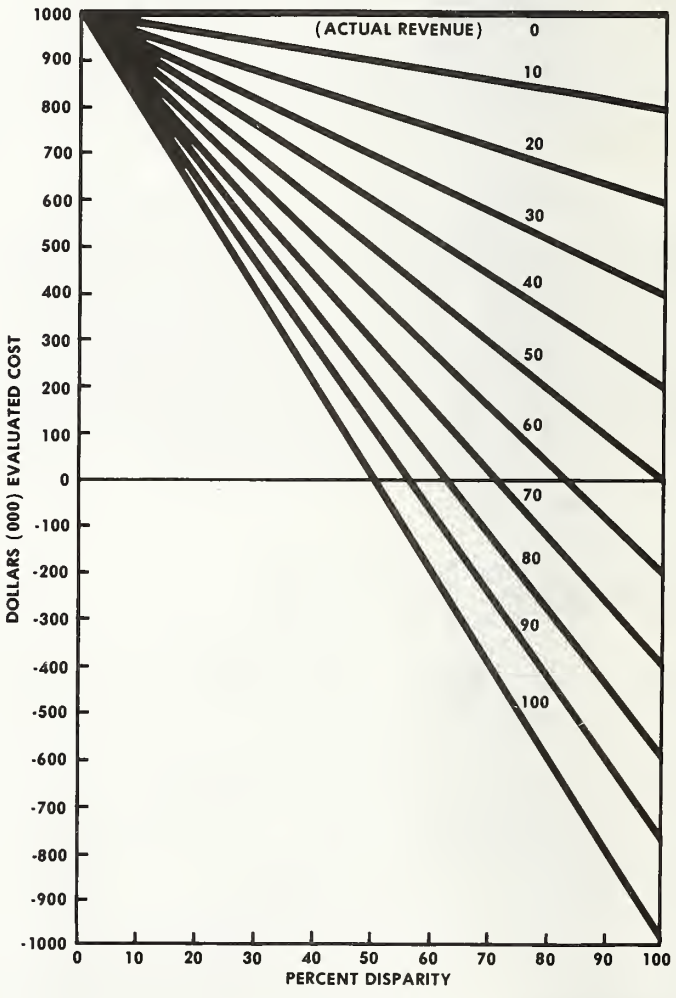


Figure 3. The Impact of The Exploitation of Benchmark Disparities

A perfect benchmark, workload projection and evaluation method will result in evaluated cost equaling actual expenditures. However, the chances of conducting such a procurement is improbable, if not impossible, in most cases. The difficulty in accurately anticipating workloads coupled with the highly formalized low bid system of procurements under the Basic Agreement opens the way for vendor gamesmanship.

The problem of unbalanced bidding will not go away without changes in the Basic Agreement procurement process. These changes must be comprehensive in nature. Any attempt to control the problem by trying to prevent specific types of pricing after they appear in a proposal will fail. The examples presented in this paper demonstrate the breadth of possibilities. They are by no means a complete list. Creativity and technological change will continue to result in new techniques.

In general, the Basic Agreement procurement process should be modified to prevent the exploitation of weaknesses in the evaluation criteria of specific proposals. Cost evaluation criteria and workload projections simply cannot be written which will consistently prevent the problem of unbalanced bidding. The problem can be rectified only by restricting the vendor's pricing flexibility. Deficiencies in the evaluation method of a specific proposal cannot be exploited without pricing flexibility. Although errors will continue to be made the consequence of those errors will be minimized not maximized.

Requiring vendors to utilize commercial pricing structures and practices would make exploitation extremely difficult. Commercial prices are directly associated with a substantial base of existing business. Therefore, vendors could not arbitrarily modify these prices to fit deficiencies in a particular solicitation without economic consequences. Vendors would remain free to set prices but once set the vendor would be committed. Discounts could be offered only if applied uniformly across all processing services for a particular proposal. The Multiple Award Schedule Contract program (MASC) has successfully utilized this approach. A review of past MASC awards reveals the pricing problems associated with Basic Agreement awards have been avoided.

Competitively, procured remote computing services offer a highly desirable and cost effective means of providing computer resources to the government community. With appropriate changes in the Basic Agreement procurement process the problem of unbalanced bidding can be resolved. Competitive energy would then be re-channeled into more productive endeavors.

References

1 Amendment No. 4 of Solicitation No. GSC-CDPSS-A-00002-N-9-20-79, General Services Administration, September 16, 1981.

2 Special Report on Unbalanced Pricing, General Services Administration, TSP Monthly Report, Special, 1980.

"Improving Organizational Productivity"

Small Computer Policy and Strategy

PANEL OVERVIEW

SMALL COMPUTER POLICY AND STRATEGY

Dennis M. Gilbert

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

Over the past few years small computers have begun to proliferate and are rapidly invading many aspects of our work, home, and plan environments. Spurred on by advances in semiconductor and related technologies, computing power that was strictly the province of mainframes and minis, is now, through the use of small computers, being brought closer to the ultimate end users.

There is an excitement in the air! There is a sense we are at a frontier that is changing very rapidly. New opportunities are being created and the promises made for the new technology are very appealing; new jobs, new environments, more productivity, less drudgery, more creativity.

And yet, amid the swirl of activity, there is a growing awareness on the part of Federal agencies (and others) that our tools for dealing with the new technology are far from adequate - as is our ability to fully comprehend what is happening or appreciate the implications. The situation is creating new challenges in which we are being compelled to reexamine some of our basic concepts about data processing. There is a renewed need to better understand the relationship between management and control and to appropriately distinguish between them. There is a corresponding need to distinguish between the distribution of resources and the distribution of information and find new alternatives for achieving the appropriate degree of each.

Part of the new challenge for organizations is to develop policies which promote effective use of the technology and increase productivity, and which at the same time support environments which encourage innovation and creativity rather than stifle them. Ironically, the technology which is creating these challenges may also provide us with some of the tools for dealing with them.

In this panel session we will examine a variety of issues related to these new developments. Among the topics we will explore are the following:

- The policies and strategies organizations are adapting to acquire and manage microcomputer systems.
- The information available to users of micros and workstations and the vehicles available for exchanging information - information utilities, bibliographic retrieval services, electronic bulletin boards, electronic mail, and specialized mailing lists.

- The tools, techniques, and applications available for using microcomputers as powerful, independent workstations and as pieces in a shared-information, shared-resource network.

- The appropriate distribution of mainframes, minis, and micros in a mature information system.

- The approaches which increase or decrease an organization's ability to respond and evolve in a rapidly changing environment.

- The de facto microcomputer standards which appear to be emerging.

- Future directions for the use of micros in the work environment.

The format of the session will be a few brief presentations by panel members to highlight the significant issues followed by audience reactions and participation.

"Improving Organizational Productivity"

Information Systems Needs Analysis

FULFILLING BUSINESS NEEDS WITH AN ON-LINE SYSTEM

David R. Vincent

Institute for Software Engineering
510 Oakmead Parkway
Sunnyvale, CA 94086

Once a long, long time ago a tribe of our earliest ancestors subsisted on coconuts. At first the coconuts were smashed against the walls of the caves in which they lived, but as this was too messy they advanced to using a common large boulder in the center of the community.

When the first hominid member of this tribe picked up a stone to smash open a coconut, it wasn't because he wanted to invent a new technology, but rather because he was hungry and wanted to get at the edible part of the fruit. This new technological development had a great impact on the tribe because it offered mobility and freedom from always having to be near the big boulder when feeding. For this reason, this particular hominid became a very prominent member of the tribe.

As time progressed, this hominid became the specialist in smashing coconuts for the entire society and consequently recruited a couple of trainees thereby creating the CCE (Coconut Cracking Elite). Everybody was happy with the CCE, and all were well fed and content until the technology took its next logical step. Some of the trainees found new ways to open coconuts such as dropping them off a cliff, or even better, using a sharpened rock to cleave the coconut exactly into two pieces with no loss of meat. This caused great consternation in the CCE because the original developer of the MCCT (Mobile Coconut Cracking Technique) preferred his method over all others. This led to great debates and disagreement at the CCE. The result was that the CCE began to work on the problem of deciding if a sharp breaking edge was better than a dull one and which method resulted in the least loss of coconut meat. During this prolonged debate, the actual cracking of coconuts dropped drastically, even to the point that some of the tribe were beginning to starve.

The tribe grew very angry and asked the tribal chief to give them back the responsibility of cracking the cocunuts themselves. Many of the tribe had already started moving back to the big boulder in anticipation of this approval from the chief.

Sure enough, so much pressure was applied on the chief that he disenfranchised the CCE and let all tribal members crack their own coconuts. The members of the CCE were so astounded at this turn of events that they left the tribe having come to the conclusion that this tribe was too ignorant for technological progress and sought out another tribe on which to ply their expertise.

Why did the CCE fail? They simply lost sight of the objective for their establishment in the first place. Rather than working toward the objective of supplying sufficient coconut meat to the tribe, they were caught up on the technology of cracking coconuts which was not necessarily related to meeting the needs of the tribe.

And so it has been down the ages that technology is discovered to meet the needs of a society or an enterprise. And so, also, there is the phenomena occurring again and again that as technology becomes more complex, groups are formed to specialize in resolving real or perceived problems. As these groups develop, they tend to get more and more isolated and elitist, even to the point where some of the perceived problems and resultant solutions may have no impact on the quality of life (and some cases can seriously degrade the quality of life). These solutions have ranged from the wheel which revolutionized transportation to the building of pyramids to assist the "afterlife" of some pharoah.

Our society is raft with such examples, but the one which concerns us here is that which exists in the on-line data processing environment. More specifically this paper will deal with IMS in an MVS environment.

Like the first coconut cracker, the enterprise with a computer purchased it for some very tangible and easily identifiable purposes such as payroll, accounts, invoicing and the replacement of repetitive clerical functions. Some institutions purchased a computer to solve complex equations, otherwise not manually possible. At that time the entire DP staff consisted of a system analyst and an operator who controlled all the work going in the computer. Communication with users was simple and direct. Objectives were easily set and achieved.

Then came on-line systems. Control of the arrival rate of the work passed from the operator to the users. Users' perception of response time changed from hours or days to seconds. The technical staff immediately started to grow with the difficult mission to optimize system performance and to maximize resources available to do on-line work. Today, many organizations' unhappy users abound because of their insatiable demands for more and better service. Many users are demanding their own computer so that they don't have to depend on some remote group that doesn't seem to understand their problems.

Such an environment need not exist. But to prevent this situation, the IMS/MVS system programmers, performance analysts and capacity planners must work as a team solving the real problem. That is, determining the needs of the business for various service levels, and then delivering them as needed. The methodological flow for on-line capacity management may be summarized in the following figure:

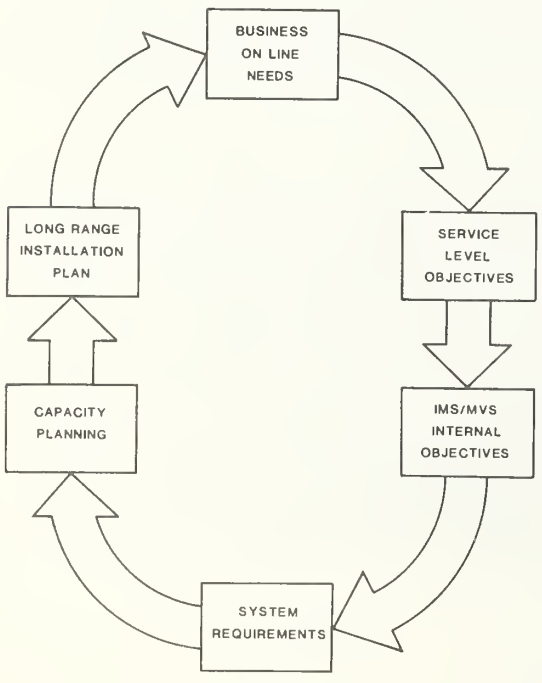


FIGURE 1

A representation of the level of technical detail required for each of the phases of the overall methodology, however, would be as described in Figure 2 with the widest area of the diamond requiring the most technical detail.

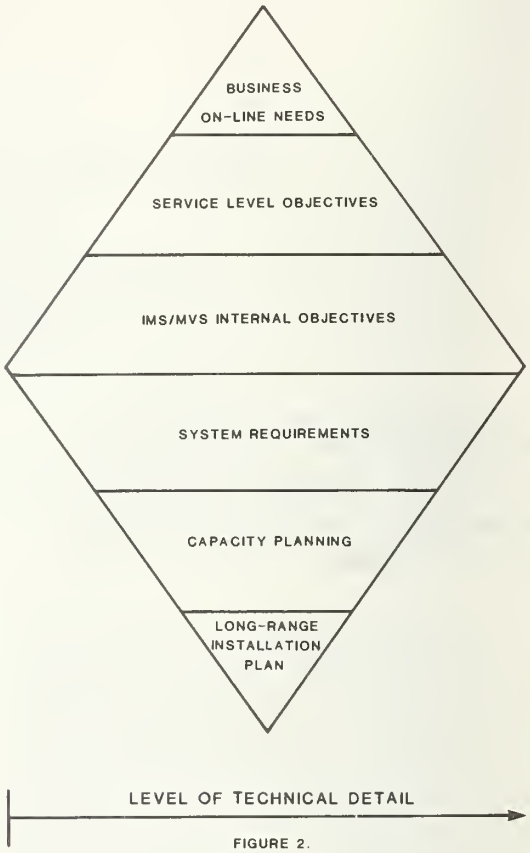


FIGURE 2.

1. Business On-line Needs

This is the most important element in the entire process of fulfilling business needs. The technical people responsible for the performance of the on-line system must identify their mission as satisfying the business needs of their users. For example, if these technicians are providing service to a bank, then they are bankers. If they are located in an insurance company, then they are involved in the operations of the basic insurance industry. They must learn to think about the business much the same as the user community.

Business needs for on-line services can usually be readily identified by following the flow of information through a business. Another technique is to develop and prioritize a list of the Critical Sucess Factors of the basic industry that the enterprise serves as well as that of the organization itself.[1]

For example in the insurance industry there are two basic transactions flows which are critical to the sucess of the business:

1. The issuance of policies and subsequent renewals, additions, and changes.

2. Claims against policies and the resultant procesing and investigation.

There are many more types of work flow, but for the purposes of illustration, I will choose the claims processing department. The technical staff can get a good idea of the flow and rhythm of work to be done by talking to the management of the claims department. This is not always as straightforward as it may seem. There was a case where a claims department was demanding faster response time. The current level was about 5 seconds, so the IMS department obliged with a response time of about 2 seconds. The manager of the claims department noticed after a period of time that the amount paid out per claim was going up. It was hypothesized by the claims manager and the IMS technical staff that the new 2 second response time, while allowing more claims to be processed, resulted in less scrutiny of the items being claimed. They agreed to raise the response time back to 5 seconds and, sure enough, the costs claim dropped.[2] This case points out the need for constant dialogue between the technical staff and the users to provide the right kind of service.

2. Service Level Objectives

A service level objective is the result of a service level agreement negotiated with the end-users. If they are not negotiated, they are probably de facto, i.e., agreements based on what the users consider normal response time to be. Here again, the technical staff must take an initiative to obtain a real agreement from two groups of people:

1. Applications programmers

2. End-users

The applications programmers must design their programs for performance. Design and transaction standards should be strictly enforced here to obtain the possibility that a transaction will be designed that can, under normal conditions, offer the right kind of service to the end-user. Many IMS groups use the concept of the Standard Work Unit as described in Bill Inmon's work done at Amdahl.[3] For example, such standards may enforce a transaction class that cannot exceed 25 D/I calls not to exceed 2 I/O's per DL/I call. Other classes may permit more, but would be clearly designed to run in separate message processing regions with substantially degraded service.

Examples of end-user service level agreements may be described as:

- Accounts Receivable: 10 transactions per second maximum $\leq$3 second response time, 90% of the time.
- Accounts Payable: 5 transactions per second maximum $\leq$8 second response time, 90% of the time (note receivables have greater business priority).
- Payroll/Addemp: 2 transactions per second maximum $\leq$12 second response time, 90% of the time.
- Payroll/Chgemp: 8 transactions per second maximum $\leq$1 second response time, 95% of the time.

These agreements are very important and become the basis for any future discussions with the end-users. The results of service level achievement should be reported to top management on a regular basis. By consistant service level achievement, the technical staff builds credibility and avoids many of the negative aspects of interacting with end-users.

3. IMS/MVS Internal Objectives

Having achieved the task of establishing design standards for IMS transactions (ala standard work units) and service level agreements with the end-users, the next step is to translate these into IMS/MVS internal performance parameters. These are comprised of key indicators and their related guidelines. An example is:

INDICATORS	GUIDELINES
Average Number of DB I/O's	$\leq$0.6 per DL/I call
Average DB I/O Service Time	$\leq$60 msec per I/O
Maximum I/O x***[*	

These internal objectives must be set for both IMS and MVS because they interact to such a great extent that a change on one side will probably effect the other side. For example, paging can bring an IMS system flow to a trickle and this can be caused by either MVS or IMS system changes. Internal objectives should include indicators and guidelines for each of the MVS areas affecting IMS including:

- CPU
- Storage including working net size (Storage isolation or fencing)

- I/O (possibly using isolated devices and/or strings for IMS) as well as indicators and guidelines for each IMS area including:
- OS Communications
- IMS Communicaitons
- Input queue (size is important as it is a tradeoff with paging)
- Scheduling (one of the greatest payback areas)
- Applications (the greatest deterrent to good performance)
- Synch point
- Output queue

Appendix A lists some software products available to monitor the above MVS and IMS indicators to ensure service level achievement. In addition to the software monitor, line delay time may be obtained by line monitoring to calculate actual end-user perceived response time.

4. System Requirements

IMS is a CPU-bound system. If there isn't enough CPU available during critical IMS hours, service will be degraded. The second most critical item is I/O. If channel unitilizations are high in the IMS or MVS I/O applications or system, service will be degraded. And finally, if IMS doesn't have enough storage, paging (and maybe MVS swapping) will go up and response will be degraded.

In defining system requirements, the technical staff must measure the resource requirements to process IMS workloads at key periods. They must also ensure that IMS work is given priority in systems where it must coexist with batch or TSO workloads. Many IMS installations have dedicated systems to ensure adequate system resources for IMS.

5. Capacity Planning

At this step in the methodology, service level agreements should have been made and hopefully are currently being met. But how long with this situation last? Capacity Planning involves three phases:

1. Workload assessment
2. System assessment
3. Management presentation

Assessing the workload means monitoring the actual work coming from end-users as well as anticipating their future needs. One way of doing this is to talk to them regarding their expectations of projected activity. This can be verified by also talking to the corporate planning group that sets the scenario for the annual budgeting exercise. The accuracy of any future service level achievement depends mostly on the correctness of forecasting the amount of work that can be done. For this reason, it is important to monitor the actual versus planned work being done by end-user to ascertain whether service levels have been degraded by unplanned volume or mix of workload from the end-user or from system deficiencies.[4]

Once current and projected workloads have been worked out, the current system must be assessed to determine when it will be exhausted and why. Exhaust point analysis for CPU, Storage and I/O will determine short term equipment needs to maintain service levels.

The second part of system assessment is longer range and deals with alternative hardware and software solutions. This will be covered further in the long-range installation plan. The last phase of capacity planning is management presentation. This must be done in language that management outside data processing can understand i.e.,

- What is being processed (claims, policies, etc.).
- Service level requirements (agreed with end-users).
- DP costs to meet those service levels.

The key to success in this phase is to quantify in financial terms the cost to the organization (user impact cost) of the added investment and the cost if that investment is not made. The end-user and the finance department outside data processing can usually provide the necessary supoport to the technical staff if the problem is properly presented i.e. in terms they can understand.

6. Long-range Installation Plan

Once the capacity planning process has been completed, further analysis is needed including:

- future anticipated technology
- contingency plans for disaster and recovery
- future corporate plans for expansion
- economic and personnel projections and expectations
- office of the future

The long-range installation plan is dynamic and changes each time significant new data is known. In order to be known, the technical staff must be in contact with the highest levels of corporate strategic planning. They must also be aware of the contingency plans which also may be dynamic as critical new applications are added. Changes in technology and level of staff will also be dynamic as IBM and other vendors continually unveil their wares with ever increasing power and function.

In summary, people who chose the technical DP arena as a career were looking for a challange. But in the past that challenge has been mostly internal and isolated from extended interaction outside the data center. The new challenge facing DP technicians with the advent of on-line systems is to not only retain their technical expertise, but to be willing and able to ply this expertise to meet the goals of the organization.

Footnotes

[1] A Primer on Critical Success Factors,, John F. Rockart, Christine V. Bullen, Center for Information Systems Research, Sloan School of Management, Massachussetts Institute of Technology, Cambridge, Mass. (June 1981)

[2] Capacity Management Forum, Institute for Software Engineering, Educational Services Division.

[3] Design Review Methodology for a Data Base Environment, W.H. Inmon and L.J. Friedman, Amdahl Corporation, Prentice-Hall, Englewood Cliffs, N.J. 07632

[4] "Performanance Standards for Capacity Planning", David R. Vincent, Computerworld Extra, (September 1981)

APPENDIX A

Software Monitoring Tools

ON-LINE MONITORS	
RMF II	(IBM)
IMSRTM	(IBM)
CONTROL/IMS REALTIME	(Boole & Babbage)
CMF REALTIME	(Boole & Babbase)
RESOLVE	(Boole & Babbage)
OMEGAMON	(Candle Corporation)
OFFLINE MONITORS	
RMF I	(IBM)
IMS PROFILE	(IBM)
IMS VSAP	(IBM)
DFS PIRPO	(IBM)
DFS ERA 40	(IBM)
DFS ERA 50	(IBM)
DASDSS	(IBM)
GTF PARS	(IBM)
IMS PARS	(IBM)
IMS ASAP II	(IBM)
IMS HRS	(IBM)
DFSUTR	(IBM)
DFSILTAO	(IBM)
DFSISTXO	(IBM)
CONTROL/IMS	(Boole & Babbage)
CMF	(Boole & Babbage)
TSA/PPE	(Boole & Babbage)
MANAGE:IMS	(Capex)

Source: IMS Systems Management Workshop, Institute for Software Engineering.

"Improving Organizational Productivity"

Software Design, Development and Testing

SESSION OVERVIEW

SOFTWARE TESTING - A LOST ART

George N. Baird

Federal Software Testing Center
Falls Church, VA 22041

The majority of the personnel resources involved in ADP are tied up in the maintenance of software - approximately 80%. This is due to the quality of existing software in the Government. Much of todays critical software was designed years ago for an environment that no longer exists. Most maintenance could be better described as survival.

The ability to test software adequately can help to solve this dilemma. If today's older software had been produced in such a way that it had been more completely tested, today's maintenance would be much easier. For example there is generally inadequate test data available to verify that any given program is working properly. How, then, can we be sure a program that has undergone maintenance is working correctly?

Even today, with our better ways to design software, a good programmer will feel he/she has completely tested a program when in reality as little as 25-30% of the code has been executed. We must upgrade the way in which we apply the testing of software. The cost of finding bugs greatly increases during the life cycle of software. A bug found in the design stage may cost $1 to correct, the same bug could cost $2 to correct during unit testing, $15 to correct during integration testing. Once the program is in production, that same bug could cost thousands of dollars to isolate and correct.

Therefore an emphasis on the more complete testing of software and the analysis of the source code as part of the quality control process can place the Government in a position of not having to spend the majority of its resources to merely keep existing systems going.

CONCEPTUAL PROPOSAL FOR A COBOL ANALYZER SOFTWARE TOOL

L. Arnold Johnson

William R. Milligan

Federal Software Testing Center
General Services Administration
Two Skyline Place, Suite 1100
5203 Leesburg Pike
Falls Church, Va. 22041

The COBOL languare, and automated software testing tools, have been studied in order to design a host independent Automated Verification System for COBOL. The proposed functions and conceptual design of the system are summarized in this paper. To provide a perspective for the capabilities of the proposed system, this paper contains a critique of the COBOL language, a description of methods of software testing, and a characterization of errors in COBOL.

1. Introduction

Throughout history mankind has advanced in technology as well as other areas primarily through the utilization of tools. This comparison holds true today. We are constantly searching for new tools to assist us in our highly technically oriented society. Our use of computers grows exponentially each year. The two major components of computer systems, hardware and software, are not growing at the same rate. The term "State of the Art" in computer hardware is rapidly changing with new equipment having expanded capabilities appearing on a daily basis. Hardware is an engineering oriented field that is quantitative and easily measured. Software, on the other hand, is more of an art and is measured qualitatively. Standard units of qualitative measurement are not always available making performance evaluation rather difficult. The languages used to program computer systems were developed in some cases 20 or more years ago. Languages have been upgraded with the new features but nothing like the evolution of hardware. Therefore, growth in productivity using computer languages must rely on additional aids or tools to assist programmers. A COBOL Analyzer is such an aid. Its purpose is to employ standard tested and proven criteria in the evaluation of a COBOL program. The results of this evaluation will act as feedback to the programmer directing him/her to areas of the program which require modification which in turn can reduce execution time as well as identify changes which could benefit the program.

The problem of inefficient COBOL programs exists both in government and industry and has been identified for some time. The COBOL Analyzer as presented in this report will not solve all of the problems currently existing in COBOL Code but it will check structure, scan text for errors, analyze variable utilization, compile run time statistics, identify calling program sequences, check unused code and evaluate many other software functions currently not being checked.

Much of this report is taken directly from the 'COBOL Automated Verification System' study phase document developed for the Department of the Air Force by General Research Corporation of Santa Barbara, California. We found this document an excellent starting point for this report. Special appreciation is given to Mr. Lawrence M. Lombardo of Griffiss Air Force Base, Rome, New York for supplying us with much of the information which allowed this report to be produced.

1.1 Background

During the 1950's as manufacturers entered the computer business, each one developed its own computer programming language for its own machine. Program portability was non-existent and it became increasingly difficult for programmers to be versatile on the rapidly expanding hardware and software universe. The Federal Government, the largest user of computers, became concerned about the need for a "common" programming language for business applications of data processing.

In 1959 the original specifications for the COBOL language were drawn up by a group of computer users and manufacturers. The first documentation was distributed in April 1960. The early 1960's brought several revisions to COBOL, each one making the language less "common" to different computers in use at that time.

Again, the manufacturers assembled and developed a new, "standard" COBOL called American National Standard COBOL (ANS COBOL). The new language gained widespread acceptance in the U.S. business sector and throughout the world. The further development and definition of COBOL is the function of the CODASYL (Conference on Data System Laguage) COBOL Programming Language Committee.

The standard of the language in the U.S. (an extensive subset of the full CODASYL COBOL definition) is the American National Standard COBOL, X3.23-1974, as approved by the American National Standards Institute (ANSI). This has replaced the previous ANS COBOL X3.23.1968.

ANS COBOL-1968 was the first effort by the CODASYL-ANSI group to define COBOL as a programming language to be a standard throughout the world on all those computer systems which chose to conform to its guidelines.

After a few years of working with these guidelines, the group found that the main skeleton of the language was homogeneous among most of the participating computer users. However, due to lack of specificity in some guidelines and the differences in the hardware and design of the computer systems, some sections of the language were markedly different from one machine to another. Gathering all this information, the CODASYL group assembled again and produced a new set of more specific guidelines and released ANS COBOL-1974.

The ANS-74 version of COBOL was created to (1) delete sections that hindered efficient coding or standardization, (2) add sections to enhance COBOL's capabilities, and (3) resolve differences or ambiguities created by the different compiler manufacturers' versions of the language. The third item was accomplished by being more specific in the working of those sections of the Standard, and by requiring the compiler manufacturers to adhere more closely to the specifications.

A great improvement over ANS COBOL-1968 had been realized. As the computer manufacturers finished their versions of COBOL according to ANS COBOL-1974 specifications, and the different versions were compared, it was found that there were far fewer differences this time, although they were not identical because of the inherent differences mentioned above. Each manufacturer respected the guidelines and any serious deviations were noted as extensions to the ANS.

1.2 COBOL Dialect Differences

A truly portable COBOL Analyzer must be capable of recognizing not only the two standards, 1968 and 1974, but also the dialects of different computers. Design and operational differences exist between computers, and although each satisfactorily compiles a program which meets the requirements set by the Standards committee, it will not compile a program which uses another manufacturer's enhancements to the standard. Users at each installation make use of these enhancements.

COBOL was created to solve the special data processing problems of the business world. The language was not designed to solve complex mathmatical or scientific problems or to facilitate number-crunching computer analysis, but rather to expedite the handling of everyday business affairs with great speed and accuracy. COBOL was created to process accounting, payroll, inventory, tax, and data base maintenance programs in a manner which allowed efficient use of large data files of information. Most business programmers are not highly-trained scientists, so the syntax or working of the language was designed to be as similar to everyday English as possible. Importance was placed not so much on features such as mathematical functions and speed of calculations as on efficient input and output of large files stored on magnetic tape or disk.

The Office of Software Development (OSD) within the General Services Administration (GSA) has as its primary task the reduction of costs spent on software in the Government. With COBOL being the most used computer programming language, it is vital that OSD provide both tools and support to ensure cost effective utilization throughout the government. The COBOL Analyzer as explained in this document will provide common criteria with which to evaluate COBOL programs before, during and after program execution. This will alleviate some of the immediate software problems connected with using COBOL and begin to allow uniformity among system specifications existing at various government agency offices.

2. COBOL Analyzer

The software industry today has developed many tools to assist in the design and development of application software. Almost every programming language can be found to have a software tool which can check coding methodology and textual utilization against pre-established criteria. The product resulting from these tools is supposed to be software which is superior to that which was initially developed. This concept is not new or unique and similar tools proliferate among the many vendors.

The COBOL Analyzer as described in this document will be used to validate COBOL source code against rather stringent standards and criteria. Errors will be identified, program structures will be evaluated, and software will be analyzed before, during, and after program execution. The following sections will go into greater detail pertaining to the structure and components of an analyzer. The one significant difference about this tool that distinguishes it from its predecessors will be its transportability. The goal is to obtain a COBOL Analyzer which will operate in a similar manner on any system supporting an ANS-74 COBOL Compiler. Given the opportunity of analyzing software written for a variety of systems and applications using common criteria, industry and government system managers will be in a much stronger position to evaluate software development and performance by utilizing a vendor and application independent methodology. The COBOL Analyzer as presented here is an attempt to define such a tool. The model and description are not meant to support every possible feature and variation of existing analyzers but rather a limited, well defined subset which can be applied to software running on any machine.

3. Functional Description

The section will attempt to identify and define the functional components and individual elements of a COBOL software Analyzer with an initial review of software quality techniques.

Usually COBOL software is tested only according to its developer's intuitions, if it is tested at all. Since the reliability of software is at least partially dependent upon the thoroughness of its testing, increased testing therefore contributes to increased reliability. Simple computer programs can be comprehensively tested without difficulty. When computer software becomes complex, usually by length of program or number of paths possible so that human intuition is inadequate to deal with its subtleties, the testing activity must be based on a systematic and rigorous methodology. Most COBOL software systems are lengthy or complex, so that the advantages of an automated verification system become pronounced and desirable.

The computer science community has recognized the problems concerning software correctness and has been developing systematic approaches to increase the reliability of software and simultaneously reduce the overall cost of producing it.

"Synthesis" techniques generally try to increase software quality by keeping software problems from happening in the first place. For example:

- Structured programming disciplines reduce the complexity of software and thereby enhance its quality and reliability by constraining the control structures of the programming language used.

- Chief programmer teams assign a talented person entire responsibility for all aspects of a software system, including its ultimate effectiveness and reliability.

- Software design methodologies such as "top down" or "bottom up" systematize the production of software and thereby improve the quality of the programs.

The alternative, to deal with software which has already been developed (or is in the final stages of development), involves two primary "analysis" approaches:

- Program proof demonstrates the correctness of programs by treating them as if they were mathematical theorems. An automated theorem prover is often used to assist in the construction of proofs.

- Automated Verification Systems (AVS) increase the practical reliability of software by increasing the level of "testedness" achieved.

Although advances are being made, program proving through logical or mathematical theorems is impractical today for programs of any size. Further, there is still discussion as to whether this mode of testing is "more correct" than other methods.

An Automated Verification System, however, is a valuable tool. The role of the AVS is to assure that software testing meets some criterion of completeness. Comprehensive exercise of a software system does not guarantee that it is error-free, but practical experience indicates that thorough exercise will locate a very high proportion of errors. Hence, testing with an AVS as an approximation to full program verification, along with proper system design, is a practical and valuable methodology.

A COBOL Analyzer is an example of an AVS. We shall use these terms interchangeably throughout the remainder of this document.

The common concept uniting this study is that software verification is a combination of separate techniques that, when applied together, form a good base methodology for testing. These techniques are (1) systematic design methodologies, (2) documentation, (3) static testing, and (4) dynamic testing and performance measurements.

3.1 Systematic Design Methodologies

It is imperative that software design be efficient, logical, and correct. Bad design increases programming time, programming errors, execution time, and maintenance frustrations. The technique of structured, modular design has been shown by working experience to be of great value in reducing these problems.

If a system has been designed and implemented in a structured fashion (top-down, bottom-up) using structured constructs and dividing program tasks into modules, the design, coding, and testing can be done in small steps. Further, enhancements or changes to the system can be done with ease and efficiency.

Most COBOL programs will need to be changed as the needs of the user change. It is therefore valuable to design a program so that it can be modified or maintained. Good planning and structure early in the design phase plays a large role in this. An automated verification system should encourage the use of structured programming and increase the value of the program written in modular form.

3.2 Documentation

Because maintenance of COBOL software has become such a large concern, there is a need for good comprehensive documentation. Personnel turnover, constant modification of programs, and high cost of programming time make it imperative that documentation of the system, from design through maintenance, be up-to-date and complete. Examples of automated documentation include cross-reference listings, program management systems, and printed reports such as those produced by DAS[1] and DCD II[2], or by an Automated Verification System such as FAVS for FORTRAN and JAVS for JOVIAL.

3.3 Static Program Analysis

One class of methods for software quality enhancement can be categorized as "static analysis." These methods scan the source text of a program for errors in syntax and semantics which can be detected without running the program on a computer. They provide consistency checking and documentation about the definition, reference, and communication of data within the program. They identify programming constructs which may be legal but risky; and they provide global, organized information about the identifiers used in the program. Static analysis expands upon the sort of diagnosis performed by a a typical compiler. In general, static analyzers are most useful in debugging.

3.4 Dynamic Program Analysis

Two basic types of dynamic program analysis are analysis of statement-level behavior and analysis of execution coverage. Both are well-known, general-purpose testing aids.

3.4.1 Statement-Level Analysis

In statement-level analysis, all program statements are instrumented in order to obtain detailed information concerning the program's internal behavior. This technique produces information that is more detailed and more closely related to the source program information than such earlier techniques as hardware monitoring, software monitoring ("snapshots"), and simulation techniques. Typically, a statement-level preprocessor automatically augments each source program statement with a "software probe" -- added statements or the invocation of a subroutine which takes measurements while the program is running. These measurements usually include the values of selected program variables and the number and types of branches taken.

When the program terminates, summary reports are printed which show the ranges of the program's intermediate variable values, which branches were taken and with what frequency, and which statements in the program were not executed.

3.4.2 Execution Coverage Analysis

This technique gathers information on the run-time sequencing of a program and the flow of control among the programs that make up a programming system. This sequencing information can be represented at various levels of detail. At the lowest level it may be a trace of the statements executed by a program when run with a particular testcase, or the sequence of branches executed by the program. At a higher level, the actual program flows traversed by the program may be collected or, at a still higher level, the dynamic calling sequence of procedures and subroutines in a programming system may be monitored.

The technique for implementing execution coverage analysis is the same as that for statement-level analysis; that is, placing software probes in the programs at the level at which monitoring information is to be gathered. The added statements are simply invocations of run-time auditing procedures which record which procedure and which control sequence or statement is being executed at the time of

monitoring. A post-processor can then reproduce the dynamic flow of control through a single program or a group of programs at whatever level is desired. This information is useful in determining which control flows and procedures were exercised by which test cases as a guide to what testing remains to be done.

4. Existing Methods and Procedures

For the most part, software verification is still a manual process. Tools and techniques exist, but this area of software engineering is in its infancy. Most of the tools and methodologies have severe restrictions or require highly-skilled persons to make their application successful.

4.1 Requirements

Requirements state what a computer system should do from the user's viewpoint. Manual systems exist which aid system decomposition via graphical techniques (SADT from SofTech and AXES from Higher Order Software) and which label requirements so the labels can be inserted in the design and code (THREADS from Computer Sciences Corporation) for tracing requirements to the code.

4.2 Specifications

At least two languages and tools exist for stating detailed specifications (Requirements Specification Language - RSL - from TRW and SPECIAL from SRI). Both provide a rigorous means of stating specifications which can be used to detect inconsistencies. Both are expensive to use and are best utilized on small programs only.

4.3 HIPO (Hierarchy plus Input-Process-Output) charts are a manual means of stating software specifications in the context of program structure.

4.4 Design

There are many design methodologies based upon decomposition, structure, data relationships, and top-down and bottom-up development. There are also systems and languages such as Process Design System (PDS from System Development Corporation) and Process Design Language (PDL). PDL is a control-structure keyword recognizer.

4.5 Functional and Performance Testing

Manual functional, and performance testing are assisted by deriving data from HIPO charts, using simulations, obtaining execution-time intermediate-value printout, and running stress or boundary tests. Boundary, or special value testing, is a strategy which exercises a program using certain values important to the control flow of the program. Predicates of logical expressions, and values which activate one or more conditions in a complex logical expression, are good candidates for special value testing. Stress testing requires that the tester manually identify areas in the program which are critical to its function. These areas are then subjected to intensive testing using special values and other methods.

Functional testing treats functional components of a program as separate programs, with their own input, output, and processing requirements. Assertions can be used to determine if the requirements for each of these functions are being met. Using assertions, the execution-time behavior of these functions can be checked for:

- Inconsistencies between the specified and actual contents of variables.
- Time required to execute a function.
- Contents of passed parameters upon entering and leaving a function.
- Changes in a function's behavior when the input values are systematically changed (as in the case of General Research's Adaptive Tester).

4.5.1 Structure-based Testing

This testing concept has been very popular for providing a measure of testing completeness, test data generation, error location, and finding structural anomalies. There are a number of automated tools which perform branch testing (RXVP, JAVS, FAVS, SQLAB, and TAP from GRC, NODAL from TRW, PET from McDonnel Douglas, Test Coverage Analyzer from Boeing) or execute user-specified sequences of statements (SADAT from Kernforschungszentrum Karlsruhe GmbH).

Algorithms are being developed to circumvent the impossible goal of testing all control paths in a program. Some of these techniques are (1) identifying strongly-connected components of a directed graph (Tarjan, Ramamoorthy), (2) partitioning the program graph into subschemes which are single-entry/single-exit structures (Sullivan), (3) identifying strongly-connected subgraphs which are single-entry/multiple-exit, called intervals (Hecht and Ullman). and (4) partitioning the program graph in terms of its interaction level, called level-i paths (Miller). Manual structure-based testing can be assisted by deriving decision tables (Goodenough and Gerhart) and choosing input data accordingly.

Structural anomalies such as dead

code, potential infinite loops, and infeasible paths can be determined by some current AVS tools (ATDG from TRW, SADAT, JAVS).

4.5.2 Consistency Checking

The most common techniques used to determine the consistency of variables and interfaces are:

- Adding assertions that define expected use (SQLAB from GRC, ACES from UC Berkeley).
- Employing static analysis (AMPIC from Logicon, DAVE from University of Colorado, FACES from UC Berkley, RXVP, FAVS, and SQLAB from GRC.
- Using data flow analysis to find uninitialized variables and interface inconsistencies (DAVE, RXVP, SQLAB).

4.6 Test Data Generation

A great deal of research energy has been expended on developing test data generators. So far, these systems (such as ATTEST at the University of Massachusetts) are still research tools and have had to back off from original goals. Other tools such as test harnesses or the Adaptive Tester require input boundaries and invariances between variables to be specified.

For manual test data generation, Howden suggests that input data be chosen to reflect special values for the program. Ostrand and Weyuker suggest deriving data in two phases based upon likely errors for the particular program's function and likely errors for the control structures used in the program.

4.7 Formal Verification

Automated formal verification systems (EFFIGY from IBM, PROGRAM VERIFIER from USC/ISI, SID from the University of Texas at Austin, SQLAB from GRC, SELECT from SRI) take user-supplied assertions (called verification conditions) usually at each branch, and symbolically execute them. The systems attempt to prove each verification condition as it is symbolically executed. The process involves simplification of inequalities and, in the case of interactive provers, the input of occasional rules to aid simplification. Formal verification is still reserved for small programs. Most of the implemented systems are based on LISP.

4.8 Program Modification

Tools which utilize a database system and save interface descriptions or other such system-wide information can be helpful to support program modification and maintenance activities. Valuable information for these activities are module interaction reports, detection of global changes, and local updates. Some of the tools that provide this assistance are the Boeing Support Software, SID, JAVS, FAVS, and SQLAB.

4.9 Documentation

Automatically-generated reports which provide information about program structure, calling hierarchy, local and global symbol usage, and input and output statement location are very useful during program development, testing, and maintenance. Most AVS tools provide some or all of these reporting capabilities.

4.10 Software Tools

COBOL is the most popular computer language in use today. There are many tools available to the COBOL programmer and analyst. Certain tools not applicable to COBOL are nevertheless useful in providing new techniques to consider.

- There are a number of text-editors such as MENTEXT, University of Maryland's editor for the Univac 1100 series, and DEC SOS. Some of the editors are powerful, others have fullscreen editing capabilities and program reformatting features.
- Object-code optimizers such as CAPEX's OPTIMIZER are useful tools which reduce core requirements, eliminate unused code, reformat the compiler listing, and permit faster execution time on IBM S/370 systems.
- Tools which assist in testing and debugging (CAPEX Analyzer/Detector, FAVS, RXVP80, QUALIFIER) are available for certain computer systems.
- Additional tools are test data generators (PRO/TEST, DATAMACS), instrumentation packages (QUALIFIER, FAVS, JAVS, PET), and data cros-reference and documentation packages (DAS, DCDII).

Many of these tools perform worthwhile functions and serve a selected market well. However, on researching the tools, the following disadvantages presented themselves:

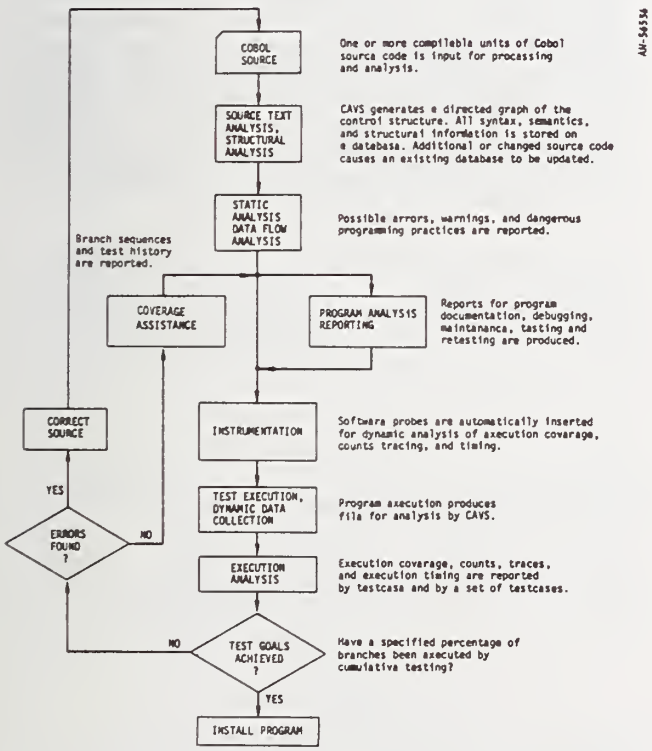
- Most are oriented to IBM-compatible hardware, and some are operating system dependent.
- Some require modification of the software for system 'fit'.
- Few of the tools actually support and encourage structured programming.
- There are many vendors, each offering

a tool for a specific function. Tool command languages differ; obtaining a comprehensive tool requires procuring many packages, and learning many operating languages and methods of utilization.

- Program debugging still requires extensive use of core dumps.

- Most of the software tools which incorporate static analysis and instrumentation testing have been developed for Fortran or another scientific language, usually at research facilities (PACE at TRW, PET at McDonnell Douglas, FAVS and JAVS at General Research). There are very few COBOL static analysis or instrumentation tools.

Exhibit 1



Many of the non-COBOL tools seem to have originated as research projects and, as a result, perform general program analyses which often include building a database and a program graph. This broad base of information allows these tools (with some overhead expense) to be extended in capability.

Tools created to assist COBOL programmers are generally smaller software packages which address areas other than program verification. There are tools to automatically create the COBOL statements which are continually used in the COBOL program. This alleviates the tedious task of typing in the wordy areas of a program, which are usually the same in every COBOL program in that data processing department. When the code has been created, there are tools to automatically reformat the printed source code. The COBOL program, with the selections and paragraphs aligned and indented, is easier to read. There are tools to create flowchart pictures of the program logic, and tools to create cross-reference listings of data elements.

COBOL has been the language of business. The emphasis has been placed upon speed and efficiency of programming. Tools were developed to assist the programmer in the task of completing programs in a short time with the least amount of difficulty. Just recently, the need for quality assurance tools for COBOL programs has been recognized, and software verification tools to analyze COBOL code are beginning to be developed.

The recognition of this need throughout the government has provided the Catalyst for the development of a COBOL Analyzer. The Office of Software Development within GSA has been tasked with the responsibility of developing tools and instituting procedures which will provide a much higher level of quality assurance to minimize development and maintenance costs thereby providing a greater return on software investment.

A COBOL Analyzer can include many functions which if comprised in their entirety could overly expand as well as degrade performance of the proposed product. Other functions and options fall into the category of vendor dependent and therefore cannot be included in a functional design. The following description is a distillation of the COBOL Automated Validation System as proposed by General Research Corporation. Only those components which appear to be vendor independent will be presented.

A COBOL Analyzer should provide or support software evaluation in two clearly defined modes; (1) static and (2) dyamic. Both of these areas have been previously discussed. Drawing from what has been covered, a summary of each area would be as follows:

Static Analysis

1. Source Text Scanning

    a. Syntax
    b. Semantics

2. Consistency Checking & Documentation

    a. Definition of Data
    b. Reference of Data
    c. Communication of Data

3. Identification of Programming Constructs

    a. Acceptable/Nonacceptable
    b. Global Identifier Analysis

Dynamic Analysis

1. Statement Level Behavior

    a. Statement Instrumentation
    b. Actual Program Experience (Run Time)
    c. Insertion of Statement Probes
    d. Addition of Sub-Routines
    e. Selection of Representative Variables
    f. Generation of Summary Reports Showing

        (1) Intermediate Variable Values
        (2) Branching Activity/Frequency
        (3) Unexecuted Code

2. Execution Coverage Analysis

    b. Control Flow Identification
    c. Dynamic Calling Sequences

        (1) Procedures
        (2) Sub-Routines

    d. Passed parameter identification

The important point to remember in reviewing this material is that the Analyzer should be portable between systems. Any feature which detracts from this portability should be removed from the functional description and model as presented in section 4.

The Office of Software Development is responsible for testing compilers for adherence to FIPS Standards throughout the government. The COBOL Analyzer will help enforce FIPS 21-1, Federal Standard (COBOL 74).

5. Proposed Model

A COBOL Analyzer as described in this document should include six major kinds of software development tools: (1) static analysis, (2) instrumentation, (3) testing analysis, (4) coverage assistance, (5) documentation and (6) reformatting. User interface could be through both batch and interactive terminals.

A coded program is submitted to the COBOL Analyzer illustrated in Exhibit 1. Any of the Static Analysis, Documentation, and Testing Analysis Functions, or Instrumentation, could be chosen by the programmer. Errors revealed by AVS could be corrected at each step of the process. This would continue until the software is ready for production testing. The user retains control of the amount of testing coverage to be performed, choosing from a selection of AVS functions. Exhibit 1 illustrates the usual sequence of events to be followed in using AVS.

Generally, static analyses are performed first. Reports can be generated showing the results of those functions. Once errors have been eliminated and the user is satisfied, the program can undergo dynamic testing analysis. Software probes could be automatically inserted for dynamic analysis of execution coverage, for tracing execution sequences, for counting execution of segments, and for timing execution of segments. Program execution could produce a data collection trace file for analysis by AVS, and listings could be generated displaying the information gathered.

Once the needs of the user have been determined and the capabilities of the software package identified, the most important task is to create a design of the system. The design must demonstrate that the tool (1) will operate correctly and satisfy the user's requirements, (2) is written in language which is familiar and portable, (3) will execute with economic efficiency and within reasonable execution times, and (4) by its design, is easy to modify to keep up with change and is adaptable for enhancement.

AVS's primary input is a collection of COBOL source text, which is recognized, parsed, and stored on the data base (composed of multilinked table structures). In this sense, it can perform some of the functions of a compiler, but for the most part its purpose and operation are different. Unlike a compiler, the AVS stores the source code in various representations (such as blank-delimited text form, statement token strings, and graphical representations). Attributes of the program (individual statements, parameters, symbols, etc.), should be saved in the data base and reconstructed in modified forms such as with testing coverage probes. The stored attributes should be examined on a program-by-program basis or across program boundaries in order to evaluate the semantic consistency of the code or to generate summary and documentation information about the program.

The AVS data base comprises the collection of program data in a set of tables. The system should be designed to handle large programs consisting of many subroutines, with the potential for run-to-run retention of data tables on auxiliary storage. The large data base should be maintained in random access files called libraries, with each library holding a collection of tables. In core the working storage should consist of allocated blocks of storage which contain active module data tables. Data transfers between the libraries and the working storage area, and between the working storage and analysis program, should be controlled by a Library Manager.

5.1 Table Structures

The tables that contain module information have a generalized structure. Access to table information should be made through a section of the library manager called the access interface.

5.2 Token

The functions of AVS require manipulation of the COBOL source text and in many cases involve accessing specific elements of text such as variable names, keywords, operators, etc. Therefore AVS should store text on its data base by breaking the text into its smallest meaningful elements (tokens).

5.3 Command

The AVS program should be divided into a group of functional segments. Similar or sequential activities can be combined in a segment and the acivities and options can be controlled by a set of segment commands. The first word of each command could signify which segment receives the command. Each segment could contain a command recognition routine which processes each command sent to the segment to determine which options and activities are being requested.

5.4 Storage

The Nucleus or data base would make up the core-resident root of the system although to minimize storage requirements, some nucleus routines should be loaded from secondary storage when needed. Each of the function segments would reside in secondary storage until called and loaded by the storage controller.

5.5 Functional Segments

The following is a brief description of each functional segment:

5.5.1 Command Decoding and Control

Process user input commands, output interactive response, and successively return each command to the overlay contoller.

5.5.2 Initialization and Wrapup

Upon run initialization, open files, initiate execution of the storage manager, and set various global data; upon run termination, close files and (for batch mode) produce report index.

5.5.3 COBOL Source Text Analysis

Read COBOL 68/74 source and perform lexical scan, token recognition, symbol classification, and structural pointer construction.

5.5.4 Structural Analysis

Build program graph, store branches, and compute single-entry/single-exit reduction history used in data flow analysis.

5.5.5 Supplementary Table Building

Build tables needed for module dependence reporting and cross references.

5.5.6 Program Analysis Reporting

Produce selected reports at user command.

5.5.7 Instrumentation

Insert probes at program unit entries, exits, branches (depending upon type of instrumentation selected); define new testcase or end of all testcases.

5.5.8 Structural Testing Analysis

Analyze run-time execution trace file, produce coverage and trace reports, and update test history table.

5.5.9 Execution Timing Analysis

Analyze run-time execution trace and produce timing report.

5.5.10 Print Services

Print the contents of specified database tables.

In order to create a COBOL AVS which is portable, the AVS should be written in a subset of ANSI-COBOL 1974 which is compilable on any system supporting an ANSI-74 COBOL Compiler. Some software changes may be required to fully adapt the AVS to the host system. A goal of this project is to minimize these changes while ensuring the reliability and transparency of AVS between unique host systems.

6. Summary

The paper was an attempt to identify, describe and explain the functional components, characteristics and proposed operation of a COBOL Automated Validation System (AVS). Although many products currently exist on the market for COBOL management tools, they are usually targeted for unique mainframes and for the most part contain dissimilar evaluation criteria. AVS on the other hand is the result of analysis of many packages. The most important and universal characteristics of each package were identified and subsequently used in the compilation of the AVS functional description.

BIBLIOGRAPHY

Alberts, D., "The Economics of Software Quality Assurance", Proceedings of COMPSAC 77, Computer Software and Applications Conference, November 1977, p. 222.

Andrews, D. M., Benson, J. P., Advanced Software Quality Assurance, Software Quality Laborstory User's Manual, General Research Corporation, CR-4-770, May 1978.

Benson, J. P., et. al., Software Verification: A State-of-the-Art Report, GRC, CR-1-638, March 1978.

Boyer, R. S., Elspas, B., Levitt, K. N., Select--A System for Testing and Debugging Programs by Symbolic Execution," Submitted to the 1975 International Conference on Reliable Software, April 1975.

Brooks, N. B., Gannon, C., JAVS, Jovial Automated Verification System, Vol. 3, General Research Corporation, CR-1-722, December 1976.

Brooks, N. B., Gannon, C., JAVS Jovial Automated Verification System, Vol. 2, General Research Corporation, CR-1-722/1, June 1978.

Brown, J. R., Lipnow, M., "Testing for Software Reliability, Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 21.

Clarke, L. A., "A System to Generate Test Data and Symbolically Execute Programs" IEEE Transactions on Software Engineering, Vol. SE-2, No. 3, September 1976.

Fischer, K. F., "Software Quality Assurance Tools: Recent Experience and Future Requirements," Software Quality and Assurance Workshop, San Diego, November 1978.

Gerhart, S., Yelowitz, L., "Observations of Fallibility in Applications of Modern Programming Methodologies", Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 86.

Glass, R. L., Real Time Software Debugging and Testing: Introduction and Summary, The Boeing Compsny, September 1979.

Holden, M. T., "Semi-Automatic Documentation of B-1 Avionics Flight Software Global Data," Naecon 1978 Record.

Howden, W. E. "Effectiveness of Software Validation Methods," Infotech: Software Testing, Vol. 2, 1979.

Howden, W. E., "Reliability of the Path Analysis Testing Strategy", Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 99.

Howden, W. E., "An Evaluation of the Effectiveness of Symbolic Testing," Software - Practice and Experience, Vol. 8, 1978.

Howden, W. E., "Theoretical and Empirical Studies of Program Testing," IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, July 1978.

King, J., "Symbolic Execution and Program Testing", Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 191.

Miller, E. F., Jr., Methodology for Comprehensive Software Testing, General Research Corporation, CR-1-465, February 1975.

Miller, E. F., Jr., Paige, M., Bendon, J., Wisehart, W., "Structural Techniques of Program Validation", Proceedings of COMPSAC 77 Computer Software Applications Conference, November 1977, p. 179.

Miller, E. F., Jr., "Toward Automated Software Testing: Problems and Payoffs", Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 16.

Moriconi, M. S., A System for Incrementally Designing and Verifying Programs, Vol. 1, USC/Information Sciences Institute, November 1977.

Ramamoorthy, C. V., Ho, S.F., "Testing Large Software with Automated Software Evaluation System", Proceedings of COMPSAC 77 Computer Software and Applications Conference, November 1977, p. 121.

Stucki, L. G., et al, Software Automated Verification System Study, McDonnell Douglas Astronautics Company, January 1974.

"SURVAYOR, The Set-Use of Routine Variables Analysis Program," TRW Brochure, 1975.

"Improving Organizational Productivity"

Computer Performance Management

SESSION OVERVIEW

CAPACITY MANAGEMENT -- FROM CONCEPT TO IMPLEMENTATION

Major Charles Gausche

Pentagon AFDSC
Washington, D.C. 20330

The session's objective is to develop a coherent view of how to plan, monitor and execute a capacity management effort. It takes a top-down view of the process. The first paper, Development of a Standard Performance Management Strategy for the U.S. Navy, provides an account of the Navy's experience in developing an organization whose sole purpose is capacity management, growing a staff to perform the required functions and defining the methodologies required to meet organizational objectives.

The second paper, Development of a Methodology for the Analysis of System Performance Indicators, expands on the methodologies required to analyze, correct and report capacity management related variables. It provides a logical, phased approach for developing a total capacity management system. The major phases of the approach are to analyze performance data, develop a data storage/retrieval architecture and then develop techniques for evaluation and forecasting.

The third paper, Computer Data Needed for Capacity Planning, provides nuts and bolts advice on what kind of data is needed for capacity planning and prediction and where to find it. Best/1, BGS System's commercial analytic modeling package, is used to demonstrate how to use the collected data in a structured way.

DEVELOPMENT OF A STANDARD PERFORMANCE MANAGEMENT STRATEGY FOR THE U.S. NAVY

S. B. Olson

Navy Regional Data Automation Center, Pensacola
Technical Support Department
Planning and Analysis Division, (Code 312)
Naval Air Station, Pensacola, FL 32508

The purpose of this paper is to inform the audience of the efforts of NARDAC Pensacola in building a Performance Management program for the U. S. Navy.

The major point of emphasis will be the location of the Performance Management staff as a management staff function and the goal of providing management planning information rather than being a solely technical group engaged in trouble shooting. Also stressed will be the need for skilled people to gain experience with an operating system/hardware suite in various settings to be able to correctly interpret data in the extremely complex environment in which a large scale computer system operates.

The Naval Data Automation Command was formed in the mid-1970's to improve the efficiency of Navy non-tactical ADP through centralized management and standardized procedures.

Central to this goal was the establishment of Technical Support Departments within the seven Navy Regional Data Automation Centers. These departments are to provide technical support not only to the co-located Data Processing Installation, but to provide support in specific areas initially to all NAVDAC activities, and ultimately to Navy-wide ADP activities. The Navy Regional Data Automation Center, Pensacola, has as one of its assigned technical management areas, Computer Performance Evaluation and Configuration Management.

The paper will describe the organization briefly, and outline the steps taken to develop the expertise necessary to provide both tools and consulting services to Navy Customers.

Key words: Capacity management; computer performance evaluation; Navy non-tactical data processing; performance management strategy

1. Background

The Naval Data Automation Command was formed in the mid 1970's as a result of studies of the Navy's non-tactical Automated Data Processing. The term non-tactical refers to the business-type data processing functions of the Navy such as inventories, payroll, accounting, etc., as opposed to tactical, i.e., weapons systems related, functions.

The reports of these investigations were critical. Central to the criticism was a common theme; the many Navy ADP installations around the United States and the world operated practically as separate entities, usually under direction of a local commander, and shared little if any common procedures, hardware, software, or management philosophy.

This situation was perceived as wasteful in that opportunities for shared efforts and benefit of common resources were not realized. In particular, it was realized that in many cases,

very similar tasks were being accomplished at several activities in different ways with development and support costs being duplicated.

It was determined that a single Naval Data Automation Command having cognizance over all non-tactical ADP and able to promulgate technical standards throughout the Navy would provide the vehicle for economies of standardization and shared benefits.

As an initial implementation of this concept, several large Navy Data Processing Centers located throughout the country were designated Navy Regional Data Automation Centers, or in Navy jargon, NARDAC's. Because of their locations, these sites were in ideal positions to provide services to Navy activities throughout the United States.

A major competitive procurement was under way to replace the suite of hardware used in these Data Centers, and all of the centers would be starting fresh with new, and more importantly, common hardware and software. This provided an ideal stage from which to launch the proposed Naval Data Automation Command.

It is common in any data processing facility to have a group or department which provides technical systems support to the operating departments. Termed Internal Systems, Systems Programming, or Technical Support, this group is typically responsible for the care and feeding of Operating Systems and Communications Systems Software, and the associated system generations, troubleshooting, etc., which are a normal part of doing business. The specialists in these departments are frequently looked to for expert advice to management on technical matters, for analysis of problems, and for inputs into management plans.

When several sites share common suites of hardware, this becomes an area of potential saving through the sharing of the talents of persons whose skills and experience are very difficult to obtain.

2. NARDAC, Pensacola

Located on the historic Naval Air Station, Pensacola, Florida, Birthplace of Naval Aviation, the Navy Regional Data Automation Center, Pensacola provides Automated Data Processing services to many activities and commands within its geographic area. Its Data Processing Installation Department operates three large-scale Univac U-1100 systems servicing a variety of workloads, as well as a Burroughs B-4700 and a Burroughs B-4800.

As with other Regional Data Automation Centers, NARDAC, Pensacola includes in its organization a Technical Support Department.

These Technical Support Departments at each of the NARDAC's serve two major functions. First, they provide technical support and expertise to the local Data Processing Installation, and, second, they provide support to all Naval Data Automation Command, and ultimately to multiple activities throughout the Navy, in select technical management areas. Included in the technical responsibilities assigned to Navy Regional Data Automation Center, Pensacola, are Computer and Teleprocessing Measurement and Evaluation, and Automated Data Processing Equipment Configuration Management. These technical management responsibilities are frequently referred to within the NARDAC's as 'lead assignments'.

Under the tasking of this 'lead assignment', then, NARDAC, Pensacola has responsibility to provide tools, methods, and procedures for Computer Performance Evaluation/Capacity Management, as well as responsibility to provide training and consultation services to Navy activities as required.

In early 1981, a team was formed at NARDAC Pensacola to aggressively attack lead technical management responsibility of providing Computer Performance Evaluation and Capacity Planning services to Navy activities.

The first fact to come to light was that research indicated that effective Performance Management functions were simply not being accomplished within the Navy data processing activities. In fact, evidence suggested that throughout industry as well, there was relatively little well-organized and systematic Performance Evaluation or Capacity Planning being done.

One of the reasons perceived as hampering good Performance and Capacity Management was the general lack of a stand-alone specialized group whose primary responsibility was the performance management effort. Typically, when studies were made or data collected, the effort was usually assigned, perhaps by default, to a person or group that had as a primary responsibility, support of day-to-day production at the data processing center.

Because of this, the task frequently took on a lowered priority to the daily crises until, usually as a result of user complaints, the performance or capacity of the system itself became a crisis. This would temporarily elevate the performance management effort to a sufficient priority to get some results. Lacking in this situation was a sustained evaluation effort, the failure to develop a comprehensive overall plan, and the failure to accumulate data over a long period of time for a trend analysis.

In light of these historic problems, the Navy placed the Performance Management function in the Planning and Analysis Division of the Technical Support Department. In this way, we see a group whose primary responsibility is

Performance/Capacity support to the activity, and in the case of Pensacola, the development of an overall program for the Navy, without the pressures of daily production problems.

In this manner, the emphasis on longer range management goals, and the participation in organization planning is assured. The perfor-ance people are not required to respond to the usual firefighting nor the imperatives of production schedules.

When the Computer Performance Evaluation group was initially formed it found itself in a position similar to that of organizations in like circumstances. That is, lack of skills, training, and experience on the installed suite of hardware and software, as well as a lack of performance evaluation skills in general. A factor that further complicated this lack of experience was that the hardware/software suite that was the object of our first efforts was the Univac U-1100. There was, and is, relatively little in the way of tools, skills, and training available for U-1100 systems. Team members were scheduled to attend classroom training in U-1100 operating systems and available software monitoring systems as a way of getting started.

What lay ahead was the task of taking people with a variety of backgrounds and experience levels and equipping them in a wide range of knowledge areas including operations, software development, diagnoses and troubleshooting, and management analysis. In addition to these technical skills, the performance management team members would require good technical writing and oral presentation skills, and the ability to present statistical information effectively with charts and graphs in order to be able to communicate with and prepare reports for management. Added to the personal development requirements facing us, were the requirements to develop the necessary tools, methods and procedures to get the job done.

3. Development

In addition to receiving classroom training, the CPE personnel needed to gain some actual experience as quickly as possible. To this end, the Federal Computer Performance Evaluation and Simulation Center was contacted. This activity, known as FEDSIM, provides Performance Evaluation and Simulation services on a contract basis to all Federal activities. The intent of our contract with FEDSIM was to give the Pensacola CPE team actual experience in the conduct of an evaluation of a computer system. The performance evaluation was to be conducted largely by Navy personnel, with close supervision and guidance from the FEDSIM analysts.

In early 1981, the first detailed analysis of the computers installed at Pensacola was conducted with FEDSIM's help. The strategy was successful. Having seen the entire process from initial planning through execution, we had a good feel for the structure and methodology needed to conduct a complete evaluation.

At the outset, one fact became obvious to even the neophyte performance analysts: that when using software tools to gather system performance and workload information, it is extremely easy to become buried in mountains of numbers! Various methods are available to accumulate and reduce the large volumes of data produced, and any performance management strategy must include tools to accomplish this task.

For U-1100 computer systems, there are two major sources of performance workload data. A software monitor which is an integral part of OS-1100 called the Software Instrumentation Package, or SIP, is an event-driven vehicle for gathering key internal performance data elements. Workload data are gathered from the OS-1100 system log files. Program products are available from the vendor to manipulate these files. The first area attacked was the SIP software monitor data.

Two approaches can be taken in the gathering and analysis of the software monitor data. In one, the monitor can be 'turned on' or activated for a long period of time, and a single report for that period produced. This has the disadvantage of effectively averaging the inherent variations in the many data elements over time. Significant peaks and lows will not be seen. The opposite approach is to collect many very short data periods, or 'snapshots'. This gives the advantage of revealing the range of variation in the data elements, and when done over a long period of time, say a month, gives very good statistical confidence that we are seeing the complete picture. The bad side of this option is that over an extended period of time we will see literally hundreds of data sets or cases accumulated. The second approach is clearly the most desirable from the standpoint of the validity of the results. The problem then; provide tools to manage the numbers.

An initial attempt consisted of modifying the vendor-supplied report generation programs to collect key data elements in a Mass Storage file. A simple statistical package was then used to produce meaningful data reduction.

While workable, the technique has a serious flaw. Any modification of vendor software is not only difficult to accomplish, but brings with it the requirement to be forever alert to changes in the vendor software and to verify in each case that the system still performs as intended.

A more acceptable technique is to allow the report programs to operate as intended, and extract the desired elements from report files. This is a popular technique on the U-1100, and

is one that is utilized by the NASA computer complex at Slidell, LA through a package called SCAR, for SIP Collection Analysis and Reporting. A copy of this package was obtained from NASA, and modified to suit our needs. Pertinent data elements are collected in a Mass Storage (disk) file, and reports are produced by a general purpose statistical package. The Navy version of the package is called SPAR for SIP/PAR Analysis and Reporting.

Data elements are extracted as desired from the file by the statistical package, statistical calculations performed, and reports formulated in accordance with parameters supplied by the analyst. This provides a very flexible and easy to use reporting tool, allowing ad hoc reports, or changes to existing reports, to be provided quickly and without programming effort.

There are several parameter driven, general purpose statistical report software packages available for a variety of hardware/software suites. Of those available on the U-1100, the Statistical Package for the Social Sciences (SPSS) seemed to offer the most in terms of flexibility, ease of use, and report presentation. SPSS has been selected as the report generation package for the current NAVDAC performance management statistical package.

After modifications and documentation were completed, the data collection portion of the NAVDAC version of the SPAR package was sent to all of the Regional Data Automation Centers.

A project is currently underway to develop a package to produce a similar capability for U-1100 workload data using elements produced by the vendor supplied Log Analyzer.

In addition, as the next phase of our Navy-wide role and in order to broaden support of all multiple Navy Automated Data Processing facilities, software tools are being developed to produce similar reports on the Burroughs hardware/software suites in use by the Navy.

4. Present Status

At NARDAC Pensacola there are currently several performance analysts who are gaining confidence daily in the analysis of U-1100 systems performance. With three large systems installed on-site and a tasking from management to provide reports on a continuous basis, experience on a variety of computer/workload environments is readily at hand. Part of our responsibility under the technical management assignment, and inherent in our overall strategy, is providing assistance and consultation to Navy activities in performance analysis studies with participation from local personnel, and to provide consultation and training to local site personnel in performance management analysis.

The software tools that are now available to collect and reduce the Operating System Software Monitor data gathered by the Software Instrumentation Package and initially reduced by the Performance Analysis Routines provide a picture very quickly of the general health and capacity of our computer systems.

The collection and reporting of workload data, an essential part of a complete Capacity Management effort, is the next area to receive attention. At present, workload information identifying work types, major accounts, and transaction volumes, is manipulated in a largely manual manner, using software tools, and utilizing data extracted from system log files by vendor supplied Log Analysis programs. Efforts are currently under way to develop an automated workload data collection system to parallel the SPAR system package currently in use for Software Monitor data.

5. Future

It is questionable whether the point can ever be reached where we can say that the effort has been completed. It is a multi-faceted effort, and events and requirements are always changing.

Within immediate sight is the first major goal. Milestones reached at this point include only part of the components of a Performance/Capacity Management Program.

Four major components comprise the program, and one is a further development requirement for NARDAC Pensacola. For effective capacity management we must have the ability to measure and evaluate current system performance, characterize current workload, obtain realistic projections of future workload, and finally combine all these data in simulation or model of future workload and proposed configurations to evaluate capacity requirements.

Although the third item in the list, the prediction of future workload, is the customer's responsibility, it is incumbent upon the performance management function to provide accurate and useable data to the customer on his current workloads. This provides the essential baseline from which to develop the needed projections. The development of useable tools and procedures for modeling future workload/configuration combinations is the next new area to be addressed.

6. Management Strategy

The key point in the management strategy implementing a performance management program is the location of the function in an area whose primary responsibility is the support of Performance/Capacity Management. The purpose of the effort is to provide management information, not just troubleshooting or system tuning. As mentioned in the discussion of the organization,

this helps insulate the performance personnel from the firefighting and the day-to-day crisis situation that usually exists in the production support area.

It is becoming increasingly evident that in order to measure and plan for computer capacity, the performance of the system in relation to stated performance levels must be considered. The true capacity of a system is the amount of workload the system can handle while meeting the required performance objectives. While the computer may be able to process additional workload, if the performance falls below agreed upon levels, we would have to say that it is operating beyond its capacity. An important part then, of the overall management strategy must be the formulation of Level of Performance agreements between the data processing service activity and its customers.

Data processing activities have been reluctant in the past to address firm performance requirements. There is some sense in the feeling that if no set standard exists, we can never be held accountable for not meeting it. On the other hand, we can also never demonstrate that the service being provided is in fact, as agreed upon. Without clear goals, and data reflecting whether or not those goals are being met, management is in a weak position to support budget issues for additional resources, or to justify increased charges for cost-reimbursable customers to cover the costs of additional resources.

One of the 'lessons learned', if one can use that rather trite phrase, is that the development of expertise in the study and analysis of computer systems and their capacity and performance is a very time-consuming process. It sounds cliche to say that large scale computer systems are very complex and the environments and workloads that they exist with are highly variable, but it is certainly quite true. Even with some formal training, the need for extensive experience with a hardware/software suite is essential, preferably under varied conditions. A frequent comment is that each new set of data gathered and analyzed is more likely to raise new questions than it is to provide complete answers. The age-old questions about what is a good number and what is a bad number are highly influenced by the system configuration and its workload makeup. A performance measure that indicates a healthy system on one machine may indicate serious problems on another. Only through experience with a hardware/software suite and its workload can realistic capacity decisions be made.

In summary, our plan has been to assign people to the task full-time, to develop the methodology, tools and procedures for Computer Performance/Capacity Management, and to allow the time to gain experience necessary to perform effectively.

In support of overall Navy objectives, we are now able to provide methodology, tools and procedures, as well as training and consulting services to other activities.

Appendix A

This appendix contains some samples of SPAR output reports. Each case represents the average value of the variable shown for a five minute period. The periods were measured once an hour twenty-four hours a day Monday thru Friday in these examples. A total of 279 observations was taken.

Figure 1 is a frequency distribution of average number of batch runs in memory during each five minute observation.

Figure 2 displays the same information on a histogram.

Figures 3 & 4 are scattergrams showing the relationship between two variables in each diagram. The numerics indicate the plotting of 2 or more occurences at the same point. Figure 3 shows the relationship between virtual memory swap rate and total number of runs open. Figure 4 shows average number of demand terminal runs open plotted against time of day.

Scattergrams are particularly useful for illustrating the relationship between two variables. They provide good intuitive understanding, even for people with little technical background.

07/26/82 FILE - PSIP$DTE - CREATED 07/26/82

PATCHCOR AVG PATCH IN CORE

CATEGORY LABEL	CODE	ABSOLUTE FREQUENCY	RELATIVE FREQUENCY (PERCENT)	ADJUSTED FREQUENCY (PERCENT)	CUMULATIVE ADJ FREQ (PERCENT)
	1	1	.4	.4	.4
	2	4	1.4	1.4	1.8
	3	9	3.2	3.2	5.0
	4	83	29.7	29.7	34.8
	5	73	26.2	26.2	60.9
	6	79	28.3	28.3	89.2
	7	23	8.2	8.2	97.5
	8	5	1.8	1.8	99.3
	9	2	.7	.7	100.0
	TOTAL	279	100.0	100.0	

Figure 1

07/26/82 FILE - PSIP$DTE - CREATED 07/26/82

```
PATCHCOR  AVG PATCH IN CORE
    CODE
          I
       1  ** (      1)
          I
          I
          I
       2  *** (      4)
          I
          I
          I
       3  ******* (      9)
          I
          I
          I
       4  ********************************************** (     83)
          I
          I
          I
       5  ****************************************** (     73)
          I
          I
          I
       6  ********************************************** (     79)
          I
          I
          I
       7  ************** (     23)
          I
          I
          I
       8  ***** (      5)
          I
          I
          I
       9  ** (      2)
          I
          I
          I.........I.........I.........I.........I.........I
          0         20        40        60        80        100
          FREQUENCY
```

MEAN 5.111 MINIMUM 1.000 MAXIMUM 9.000

VALID CASES 279 MISSING CASES 0

Figure 2

FILE PSIP$DTE (CREATION DATE = 07/26/82) ****STUDY OF SYSTEM C ALL****
SCATTERGRAM OF (DOWN) SWAPRATE (ACROSS) RUNSOPN

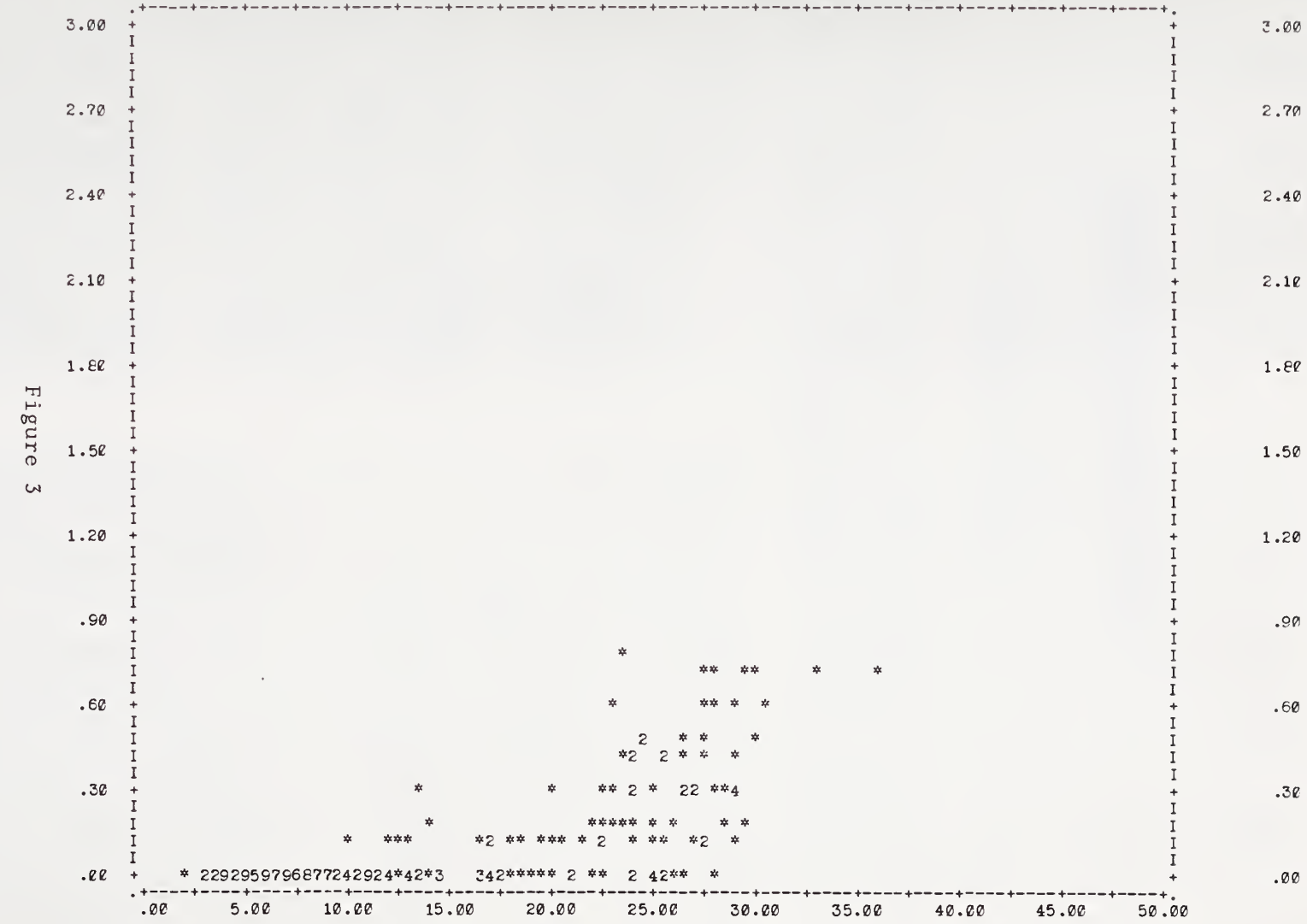


Figure 3

FILE PSIP$DIF (CREATION DATE = 07/26/82) ****STUDY OF SYSTEM C ALL****
SCATTERGRAM OF (DOWN) DEMNDOPN AVG DEMAND OPEN (ACROSS) TOD TIME OF DAY

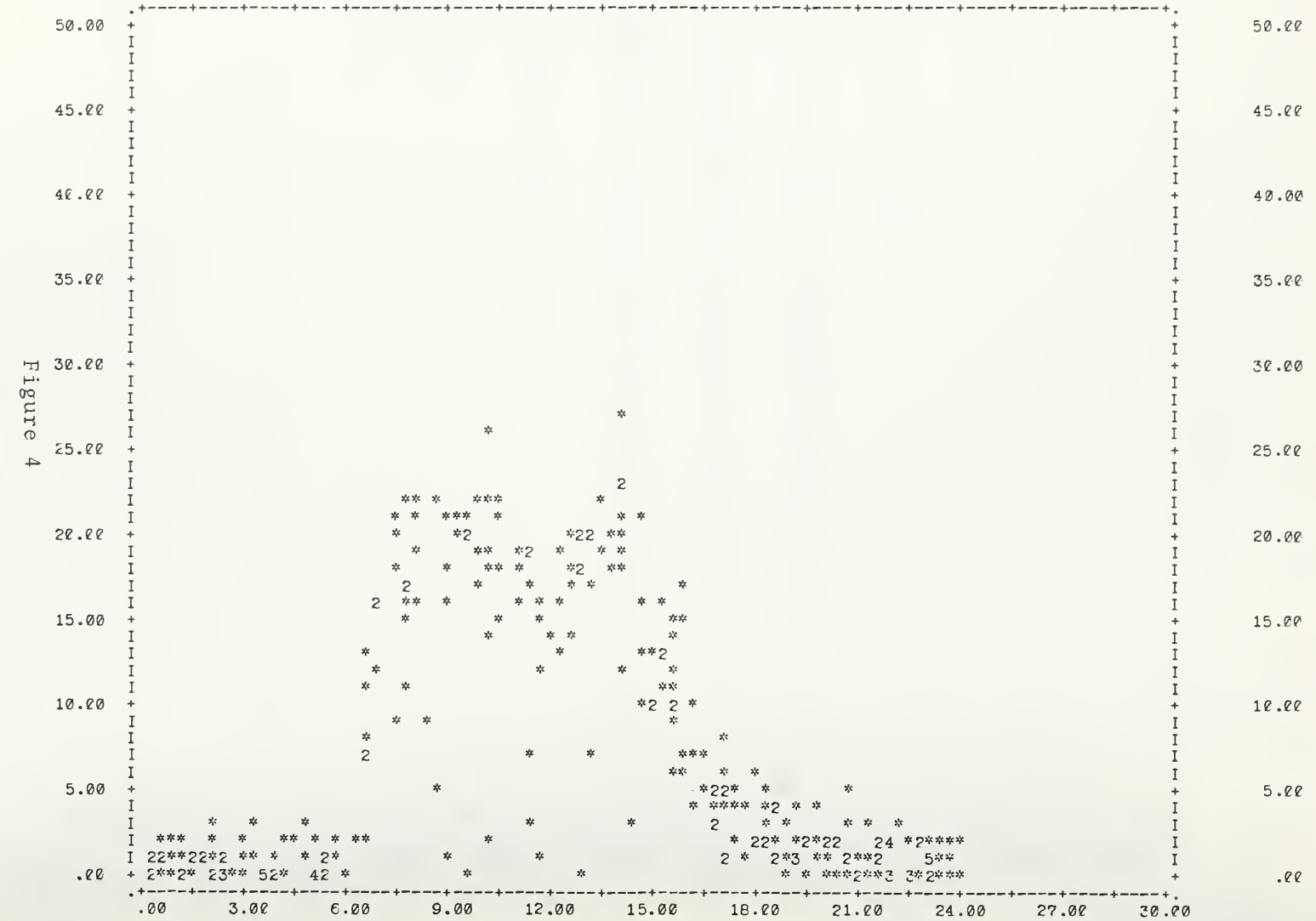


Figure 4

DEVELOPMENT OF A METHODOLOGY FOR THE ANALYSIS OF SYSTEM PERFORMANCE INDICATORS

Paul Chandler, CDP

Wilson Hill Associates
1025 Vermont Ave, N.W. #900
Washington, D.C. 20005

Many organizations are faced with the performance management of old or over-utilized computing resources. Often there is no performance improvement plan which addresses the efficient usage of these resources. The development of a performance improvement plan has value to the organization in that a solidly based plan can be developed which seeks to quantify the computer performance. This paper presents a method by which this plan can be developed.

1. Introduction

Many organizations operate and maintain vintage data processing equipment without a full appreciation of the processing potential of that equipment. Additionally, funds may be lacking for the upgrade of present equipment or the purchase of new hardware. How then can such an organization better manage its computing resources?

Additionally, all too often commercial packages are bought without a complete understanding of the actual requirements of the organization beforehand. The subject of this paper presents not an answer or itemization of relevant factors, but presents instead, a generalized method by which relevant system performance indicators can be identified. Also, this paper describes a way for the organization to anticipate its planning process through the use of these indicators by a careful consideration of the use that will be made of these indicators.

In a worst case situation let us presume an operational environment in which we find both 360/40's and 50's under DOS and 370/145's under OS. These systems produce unlike accounting data streams of GRASP and SMF data, respectively. How then can operations management assimilate and analyze the magnitude of this performance data from the installation necessary to monitor and improve performance of both the installation and selected application program? This paper will present a method by which an in-depth analysis of the accounting data and a statistical analysis of that data can be performed.

Specifically, this paper describes a methodology by which the user can:

- Plan for and control the application of ADP resources to workloads in order to improve systems performance, and
- Report to users and upper management in a manner that facilitates understanding of the progress made in performance management.

2. Background

Computer performance evaluation (CPE) in the simplest terms is the measurement of how well the software is using the hardware for a given job mix. The measurements can be taken directly by hardware monitors attached to the physical configuration, or by software monitors resident within the operating system itself. In any case, the information gathered is evaluated by a highly skilled analyst who makes recommendations that will improve performance or effect greater operating economies. A CPE can reduce long-range costs, attain better job billing, and provide management with more meaningful information concerning data processing performance.

For the purpose of this project, software-monitor-generated data will be utilized. The data provided is event driven and is SMF dependent upon the completion of a job step or issuance of a system command. These conditions include the beginning or ending of a job step and contain such detailed information as job I/Os, job tape I/Os, CPU seconds, block size, core usage, etc. This breakdown provides a better idea of where resources are being used and allows an analysis of computer performance to determine if resources are being used in an efficient manner.

Performance data provided by a software monitor in this project will consist of SMF data generated from OS systems and GRASP data generated from DOS systems. Although both contain similar types of data, the record formats and detailed data content are quite different. The ability of this data to indicate performance levels stems from the overall view of the job and its continuous operation. These data can be summarized to provide a "snapshot" of the system at any point in time, and compared to a baseline performance profile to obtain a quick gross measurement of how the system is performing. This technique can be extended from a collection of programs to a detailed analysis of any individual program so that a detailed picture of resource utilization by individual application systems can be seen.

Job accounting data normally will be used in two ways. First, general data may be collected over a representative period of time. The definition of a representative period is often very difficult; however, a representative period should cover one data processing cycle, which is the period of time that includes at least one occurrence of all major events of interest (such as in a given application program. Second, based on the general analysis, more specific data can be collected for in-depth analysis. This collection of data is reduced to provide an overall set of system and application program indicators, some scalar and some vector, which are analyzed.

Performance data must be used in conjunction with other factors related to job frequency. For instance, if during a given time period a job appears to utilize a high proportion of computing resources, it would be imperative to know the frequency of execution of this job; if the job is run only once per year, it may not be worth pursuing, and would be misleading in the management of the data center in any case.

The overriding consideration when performing a CPE study must be the overall benefit to the organization. If the goal is to cut expenses by optimizing the use of current resources, the study will provide the facts needed to do it. If the goal is to increase job throughput regardless of the money involved in buying additional equipment or staff, the study will provide the facts. Generally, a CPE study will show that changes are required in one or all of the following areas: equipment, procedures, or programs.

Equipment. Configuration changes usually involve adding or deleting peripherals, memory, or I/O channels. Other changes will take the form of upgrading existing gear, such as more memory or CPU. Another form of equipment change involves the reconfiguration of existing equipment. The study may show that some disk drives are underutilized while others are nearly at capacity. Thus it would be desirable to have as many channels as possible access the heavily utilized drives. It is assumed that the operations center plans no massive upgrades, but rather seeks to "fine tune" its existing systems. Therefore, although new disk drives might be a consideration, new CPU's certainly would not be.

Procedures. The interaction of resources and programs is prescribed in the data center operating procedures. The way jobs are scheduled, combined with systems rules for resolving contention for resources among those jobs already running, can sometimes affect throughput as much as a shortage of resources. Another scheduling improvement would be the redefinition of job classes to assure an equitable distribution of resources according to actual requirements. Resource allocation priorities are often described by a CPE study such that significant increases in throughput can be achieved by manipulating the priorities assigned to various program requests for resources. Dynamic load balancing of I/O devices can also be accomplished by running a system overhead program such as GRASP or other product.

Programs. Another approach to poor device utilization is to change the way that individual application programs use the devices, rather than changing the way the devices are configured. Specific information of value to a programmer in terms of his own programs is record blocking information and file media analysis. Blocking can have tremendous impact on a program's running time. The number of files that can be concurrently read or searched is often dependent on file location. Additionally the division of a single file into many smaller pieces, called "fragmentation," can be spotted through a detailed CPE study as well.

The process of CPE requires an interaction between the data analyst performing the study and the data center operations management. Top managers must set up procedures to monitor and track progress in implementing the recommendations it accepts after they are made the CPE team. This tracking system must be regular, automatic and publicized to help assure that the CPE project yields its anticipated bene-

fits. Other user involvement includes the input from senior systems analysts who provide preliminary software organization information, such as the location of system files. They may also be called upon to assist in implementing software monitors or triggering additional data gathering. These "hidden" personnel considerations are critical to the success of the CPE project.

3. Overview of Method

Development of a methodology for active computer performance control requires a careful consideration of the data available, as well as the development of a sound methodology for data analysis. It should be noted that no methodology will generate hard benefits directly. Benefits are realized through the effective use of the information developed from this analysis, and this effectiveness depends on the skills and motivation of individual operations managers. More specifically, direct benefits will be derived in expanded capabilities related to:

- Ability to measure actual performance versus goals and established targets
- Diagnostic analysis of problem areas; and
- "Threshold" profile reporting to develop sensitivity to selected areas of performance control.

First, utilization of computer performance data will provide the ability to measure actual performance, and to establish performance goals and objectives against which actual operations may be measured. At the present time, only broad performance measures are available from incompatible data formats (SMF versus GRASP), thus restricting analysis and development of performance profiles. In response to "are we doing better?" operations management personnel must respond in many instances with either "I think so" or "I don't know." The ability to measure actual performance will thus have direct results in a quantitative and meaningful manner. Additionally, having this ability to measure installation and application performance will allow the manager to take a more active role in computer center operations management.

Second, performance information is needed for diagnostic analysis of data center problems and operations. For example, an analysis of CPU utilization versus job priority can reveal that high priority CPU-bound jobs are blocking access for lower priority IO-bound jobs to computer peripherals, thus ineffectively utilizing computing resources. At the present time, the manager may not be able to conduct this type of analysis despite the availability of performance data due to the volume, complexity and diverse record types of data reported.

Third, comprehensive tracking and comparison of performance data is needed for overall management control of computing resources. In the longer run, significant efficiencies and economies will be obtained by anticipating workload demands and planning to meet these demands with a rationally developed operations management program. Currently HQ management control is exercised on an exception basis and no single source of comprehensive data is readily available for planning analysis or other purposes. It is necessary to accurately characterize each operations center according to its unique set of operating constraints in order to increase utilization and throughput of application programs.

In summary, the manager seeks expanded analysis of performance data. This analysis should serve to maximize the productivity of operations centers in the near-term future, applications programs in a longer horizon, and can be achieved within the scope of the study.

4. Project Methodology

The project methodology presented in this section presents an outline of the work tasks which should be accomplished. The plan is divided into three major phases:

1. Analysis of Performance Data
2. Development of Data Storage/Retrieval Architecture
3. Development of Evaluation/Forecasting Technique.

Phase 1. Analysis of Performance Data

The purpose of this phase is to review and select a subset of the performance data available from processing installations and application systems in order to develop workload profiles. Both the range and variety of data generated at the installation will be reviewed and considered. This will occur in three primary steps.

Step 1.1 Review Available Performance Data

In order to effectively introduce expanded performance measurement/data management on a production basis, a full understanding of existing work activities must be obtained. The primary source of this information is obtained through a detailed review of job accounting data provided from SMF and GRASP programs currently in place. This manual review requires the guidance of center personnel to identify major record types and field descrip-

tions and to identify the volume and frequency of such data produced. The range and variation of performance data values must be identified. During this step, a review of application program JCL and documentation is also made to identify major application system characteristics. Additionally, the manager must come to a fuller understanding of the processing configuration used by the operations center.

Step 1.2 Develop Preliminary Workload Profiles

This second step uses the data identified in Step 1.1 and seeks to characterize the operating environment through the preliminary construction of workload profiles. These profiles must consider and interrelate key performance data by which the operations center and application programs can be characterized on a gross level. These profiles are intended to provide an overview to the manager of the primary performance characteristics of the systems reviewed.

Step 1.3 Select a Set of Relevant Performance Indicators

This step presents the actual recommendations of the project team for a subset of performance data which will be used to characterize and describe both the operations center and application programs covered by the scope of this study. From the range of job accounting data reviewed, and the trial construction of representative sample profiles, specific key performance indioators can be identified, and primary derived performance indicators can be specified. Additional performance indicators may also be identified as being of secondary importance. Some data currently generated from the job accounting systems will be excluded at this point from further analysis.

Phase 2. Development of Data Storage/Retrieval Architecture

The primary purpose of this phase is to provide programming specifications by which the independent and unrelated data sources of SMF and GRASP job accounting data can be consolidated into a single data base, and specifications for a variable report generating program. This phase is divided into three steps.

Step 2.1 Identify System Characteristics

This step reviews and identifies a set of system characteristics which define the overall structure of the program specifications developed. Specifically, a set of programming considerations is defined which may include such factors as: table driven program, modular design, user friendly, inverted file structures, variable report format, single standard data base record. These are considered to be important program design considerations, and upper level management should be asked to concur.

Step 2.2 Produce File Design

Specific input record formats are defined and a single output record is formulated. The logical data base design is outlined and file interrelationships defined. Additionally during this step, a master reporting format is designed to present a user selected set of performance data together with standard identification data. Several basic report formats can be considered.

Step 2.3 Develop Program Specifications

During this step the operational flow sequencing of the data storage and retrieval requirements are developed and defined. The physical data base design is completed. Using structured coding techniques, and in accordance with established documentation requirements, program specifications are developed for the processing and storage and job accounting data, and for the selective retrieval and report generation of data from the data base.

Phase 3. Development of Evaluation/Forecasting Technique

The final phase of this project consists of a technique by which standardized performance data can be utilized for the tuning, evaluation and forecasting of major workload characteristics for an individual operation center over time, and for multiple centers with each other. Additionally, a method of profile characterization is developed which allows application programs to be described and evaluated against "standard" profiles for comparison over time to detect major operating errors or trends, and between application systems to identify and locate inefficient utilization of resources at a particular operations center, aberrant implementation, or isolation of problems due to unexpected and unusual data conditions. The methodology should contain a description of statistical techniques by which the performance indicator analyzed can be evaluated and data center operational management effected. This project phase is divided into four steps.

Step 3.1 Develop Performance Technique

The development of a performance technique is broken into the three primary areas of: workload profile, profile standards, variance analysis. Each step will be described in detail below.

Define Workload Profiles. From the preliminary workload profiles developed in Step 1.2, a final selection of relevant performance indicators and indicator interrelationships is made. Presentation techniques which are most

applicable to the operations environment and management control and review are identified and presented to management for concurrence. Some illustrative types of presentation techniques which can be considered include:

- Kiviat diagrams: a series of interelated performance indicators is presented on an eight axis graph such that a "star" pattern is formed. For each set of interrelated indicators, standard profile patterns can be derived. This method allows several indicators to be monitored simultaneously.

- Point and Plot Diagrams: for an identified performance indicator, the indicator is point plotted against a constant variable (such as time) such that a trend can be identified. This presentation is easily amenable to statistical analysis and extrapolation.

- Bar Graph: in this method a series of "steps" is presented for a selected variable. This technique is useful for broadly characterizing an operating system for a specific indicator, or for illustrating relative comparison over equal periods of time. Frequency distributions can be easily displayed in this way.

- Vector Derivatives: a series of performance indicators is combined to form a single number which represents the composite performance for a given application system. Weighted values for each of several components indicators can be generated, and a utility function computed which combines the indicators into a single number. The method is applicable in instances where several cost factors of a job are combined into a single user charge.

Establish Profile Standards. The role of CPE assumes that there exists a desired target or goal in the management of computing resources. One of the purposes of this step is to provide for the definition of management generated target areas against which actually measured indicators can be compared. This target generation is an interactive process in which the data center manager continually reviews and updates his target goals based on the performance of the data center. For instance, if a monitored performance indicator shows that only 30% of the CPU is being utilized, the data center manager might select a target utilization rate of 50%. As this rate is achieved, the target utilization rate might be later modified. The setting of the target goals involves not only a review of the current operating environment as characterized by the performance indicator reviewed, but must reflect the judgment and experience of the data center manager as well. This step does not actually produce target values, but provides the place for them to be used.

Describe Variance Analysis. This step is the description of the method by which measured performance profiles are compared against the management-provided standards and a variance profile generated. A variance profile is used here to indicate the simple difference, either positive or negative, between the measured and standard values. The method provides for the comparison of an operations center profile with itself, the comparison an operations center against another center profile, and similiar intercomparisons of application system profiles against themselves and one another.

Step 3.2 Develop Statistical Technique

This step should provide a discussion of descriptive statistical techniques which can be used to interpret the performance indicators measured. This discussion should provide a groundwork for establishing indicator means, computing standard deviations, and determining confidence intervals for management provided limits. This will provide management the ability to produce production control guidelines containing upper and lower limits between which each operations center or relevant application system should operate for maximum efficiency. Additionally, the statistical groundwork should be described by which selected performance indicators can be forecasted using simple linear regression methods.

Step 3.3 Develop Indicator Forecasting and Evaluation Techniques

Using the statistical methods developed in the previous step, illustrative forecasting scenarios are described in which management can participate in the future planning and utilization of computing resources in a positive, active way. This will prepare the way for the development of contingency operations and resource utilization.

First, the major management information presented, for each of the major performance indicators and profiles developed in Step 1.3, is characterized and interpreted. That is, for each indicator a description of the use of that indicator in performance management is provided; for each profile, a description of the major profile types that can be expected is provided, together with rules of thumb for management interpretation. Additionally, for each indicator and profile, one or more "red flag" conditions can be identified which may require immediate management action.

Second, for each major performance indicator, a method should be developed which will allow these indicators to be forecasted over a defined period of time. These factors should consider seasonal trends, harmonic components, and factor interdependencies to the extent appropriate. Several forecasting methods may be considered including linear regression, moving averages and exponential smoothing. For each of the methods presented, limitations of the method should be explained.

Step 3.4 Define Assumptions and Limitations

Computer performance indicators cover only a portion of the factors within the realm of management control. Equipment operation, data management policies, resource planning, resource control, and job scheduling can all be identified indirectly by a review of performance indicators, but may not be addressed directly. This step includes the identification those areas of management control to which the highest degree of success in controlling the utilization of computing resources might be expected to be achieved.

Additionally, CPE limitations include the fact that operating system overhead may be excluded from consideration, since the job accounting facility is typically resident as part of the operating system itself. There is also a machine configuration difference between DOS and OS operating systems which can not be reconciled. Finally, certain statistical assumptions may be identified, such as an abnormal distribution of data, and so on.

4. Discussion

For the purposes of the methodology developed, it is important to consider the extent to which management is able to control and influence a change in user behavior as well as data center operations. Although outside the scope of this study, a consideration of relevant factors of user behavior which may serve to limit the effectiveness of management control of resource utilization and implementation of resource control strategies should be made.

It should be further emphasized that a study using the methodologies described is but the first of several steps in a successful implementation of an operations center audit. Later steps must include:

- Establishment of operating standards
- Application of established standards to actual operations center performance
- Use of the evaluation and forecasting method derived
- Determination of relevant information to be reported to upper level management, and
- Identification of responsible operations management and user management, to whom regular reporting is to be made.

The methodology presented in this paper, then, is but the first of several steps in a successful CPE effort. It may be mentioned that a largely unspoken benefit of any method such as this is the development of a realistic estimate of the actual resources needed to conduct, and implement, a CPE study.

The author is grateful to both Christine S. Wenk and Thomas A. Mink for their inspiration and guidance during the course of this study.

Bibliography

Merrill, H. W. Merrill's Guide to Computer Performance Evaluation, SAS Institute, Inc. Cary, NC 1980.

GRASP User's Guide, SDI, Inc, San Mateo, CA 1978.

OS SMF, GC28-6712-7, IBM Corp., Armonk, NY 1973.

COMPUTER SYSTEM DATA NEEDED FOR CAPACITY PLANNING

Dr. John T. Peterson

BGS Systems, Inc.
One University Office Park
Waltham, MA 02154

The data needed for capacity planning comprises a short list, and can be collected with standard software monitors from the systems of the major main-frame vendors and their PCM's. However, most articles on capacity planning do not discuss the data needed in any detail, and many of those that do, do so only for a single vendor, such as IBM. Since the data is the same for any vendor, this paper discusses the data in generic terms, with examples from several vendors. The paper also explains some shortcomings of vendor-supplied software monitors.

Key words: Analytic modeling; capacity planning; computer performance; modeling, models, software monitors.

1. Introduction

Capacity Planning consists of two main activities: Workload Forecasting and Performance Prediction. The capacity planner must also understand, and make some use of, related aspects such as chargeback systems, configuration tuning, performance reporting systems, etc.. These aspects are too wide a subject for this paper, and have been well documented already. This paper will concentrate on Performance Prediction in particular. The Workload Forecasting part of Capacity Planning requires two types of data: data from the computer system, which is the same as that required for Performance Prediction; and data on user activities, which varies from agency to agency and does not concern us here.

Performance Prediction includes three steps:

1. Predicting the performance of future workloads on various future configurations;
2. Evaluating alternative configurations based on costs and the performance predictions; and
3. Developing a plan for configuration changes and application changes based on the results of the first two steps.

Steps two and three involve tradeoffs between costs, management priorities, and risks. The data needed for these steps varies greatly from agency to agency and from situation to situation; its gathering and use are relatively straightforward - if management makes up its mind. Step one is the most difficult of the three, and usually involves highly technical information and insight. The data required for this step is the main subject of this presentation.

There are three main methods of predicting the performance of future workloads on future configurations: benchmarks, simulations, and analytic models. Simulations and analytic models usually require the same types of information, with simulations typically requiring more detail. Benchmarks are a sleeper: they appear to require little information. You just select a few "typical" jobs, put them together and you have a benchmark. However, certain practical considerations may cause randomly selected "typical" jobs to be an atypical benchmark - one that does not properly represent your workload. The only way to insure that the benchmark is "typical", or representative, is to collect data similar to that required for modeling, run the benchmark, and adjust the benchmark until it reproduces the collected data. So benchmarks actually require much of the same data that analytic and simulation models require. Because the data requirements are similar, and because more and more people are using analytic models to predict performance, this paper will concentrate on the data required for analytic models. Most simulations will require more detailed data; most benchmarks will require less detailed data.

2. Data Required for Performance Prediction

Before we talk about individual data items, there are two observations to be made about the context of data collection. All three methods of performance prediction assume a particular time of day, or "typical" period, has been selected. The benchmark or model is to represent the workload at that time. In addition, most computer systems process several workloads at once: batch, on-line program development, data-base inquiry, and CAD/CAM are examples of typical workloads that might be active on the same system at the same time. The capacity planner will determine which workloads must receive separate attention based on which workloads are similar in resource usage, or will grow at a similar rate, and which workloads are treated alike by the computer system.

2.1 CPU Utilization

The capacity planner must know the total CPU utilization (or total CPU seconds used, or percent CPU idle) during the period selected. This total must include not only billable time, but also any overhead that kept the CPU busy. This measure is easily collected from most types of medium and large scale computers by software or hardware monitors.

The capacity planner must also know how much of the CPU utilization is used by each workload. For most systems, software monitors or accounting packages report billable CPU time by workload. But he must also know how much overhead was due to each workload, because some workloads cause far more overhead than others. Both hardware and software monitors are rarely able to report usage by workload. However, there are usually ways to make accurate estimates. Certain software monitors for UNIVAC, Burroughs and Honeywell systems divide overhead into parts that can be easily divided among workloads: I/Os and dispatching overhead by number of swaps, memory allocation overhead by number of allocations needed per workload, etc.. IBM's RMF monitor does not subdivide overhead, so methods there have ranged from simple rules of thumb ("capture ratios") to sophisticated use of RMF and SMF data with known path lengths. The latter approach is the one used in the Capture/MVS model generator.[1]

2.2 I/O Utilization

The capacity planner must know the total utilization (number of seconds, percent idle) of each disk device and tape channel during the measured period. This can be measured by hardware or software monitors and is usually no problem. Some software monitors, particularly for smaller machines, may report only disk channel busy. This is not sufficient, because it is not divided out by device and because the channel is busy for only a small part of the time that its devices are busy.

As with CPU time, the capacity planner must also know how much of the disk device time is due to each workload. This is one of the harder measures to obtain for smaller machines. Trace-driven software monitors record this (GMF for Honeywell, GTF for IBM, I/O trace for UNIVAC, and DMS Trace for Burroughs). It can also be estimated in various other ways, for example, based on your knowledge of which files reside on each device or file. Models of IBM machines usually make use of SMF accounting data for this purpose.

2.3 Level of Concurrency

The capacity planner must know how many jobs/transactions of each workload could be active at once. The higher the concurrency is, the less idle time there will be. The limit on the number of active jobs/transactions may depend on how many could fit into memory at once, or on a parameter set by the systems programmers or operators. (Examples of the latter are MAXOPN as a limit on active batch jobs for UNIVAC, and domain targets for IBM MVS systems.) Memory usage is nearly always reported in enough detail to be used for these purposes, either by software monitors, or accounting data, or both. The values of the system parameters involved can be obtained from systems programmers. Knowing which parameters are important is information that comes from systems programmers, the operating system vendor, the modeling package vendor, or from other sites that have been active in modeling or tuning.

2.4 Transaction Rates

The above information will usually suffice to produce utilizations and relative throughputs. (An example of a relative throughput result is, "An upgrade to a multiprocessor will result in a 30% improvement in batch work completed.") In order to predict response times and absolute throughput, the capacity planner must know the number of transactions or jobs from each workload that were completed during the interval. This information is easy to obtain; it is usually contained in accounting data or log tapes. It is also contained in some software monitors. This information will be used with the utilizations to calculate the CPU and I/O times per job or transaction.

2.5 Validation

Once the model is built, the capacity planner will want to validate it to make sure that when given today's workloads and configurations, the model predicts the response times and throughputs actually observed during

---

[1] Capture/MVS User's Guide, Release 2.0, BGS Systems, Inc., Waltham, MA

measurement. This step insures that arithmetic and/or measurement errors have not occured. In order to do this, the capacity planner needs measured response times/elapsed times/throughputs for comparison. These are usually measured by standard software monitors or contained in log tapes. Examples are: IBM TSO response, measured by RMF; and UNIVAC TIP response, contained in the TIP log records. Occasionally the measurements are biased by a particular definition of response. This is the case with UNIVAC Demand terminals, where the response time measure used by SIP is defined in such a way that it typically reports average response times half that experienced by terminal users. (Work is going on in that area to find better ways to measure response.) In this case, validation can be done to the extent that the model produces "reasonable" response times for that workload (based on user experience and the biased response measurements) and produces accurate response times and throughputs for the other workloads. More accurate response measures can be made by special software or hardware monitors, and alternative ways to calculate response can be used (such as using Voluntary Delay Time to calculate Demand response times).

2.6 Optional Data

In addition to the necessary data discussed above, the following data is sometimes needed depending on the computer system and workload forecasts:

a. CPU Priorities. Usually, some workloads have a higher priority for CPU usage than others. A workload's priority may be set by a system parameter (as in MVS), assumed by the operating system (as in UNIVAC's EXEC), or regulated by the operating system in response to various system parameters and measurements. Typical examples of the latter give higher priority to workloads that use little CPU time compared to I/O time. Information on CPU priorities is like that on concurrency parameters (obtainable from the systems programmers in most cases). Depending on your workloads, CPU priority may or may not affect your system's performance.

b. Channel Utilizations. If disk channel utilizations are expected to change greatly, (such that they would exceed 30% busy), increased accuracy may be obtained by including disk channel utilizations and number of I/O's per channel in the data collected. In most systems, the effects of heavily loaded channels are included already in the utilizations of the disk devices. However, if the channel utilizations will be significantly different in the future, the built-in effects will be off somewhat, and these may affect system performance significantly. In this case, one would model the channels also. The channel utilizations and numbers of I/O's are reported by most software monitors.

c. Number of Terminals. The capacity planner may want to include the number of terminals in the data collected if the number of terminals is small or if the workload forecasts are couched in terms of changes in the number of terminals. This number is easily obtained from software monitors or accounting data. Along with the number of terminals, one must specify the average "think time", the time the system spent waiting for the user to input the next command. While most monitors do not record think time explicitly, it can be calculated easily from other data already collected, such as the number of completed transactions and the number of terminals.

3. Uses for that Data

Figure 1 shows a sample analytic model created for use in BEST/1, BGS System's commercial analytic modeling package. The CPU and device utilizations and numbers of completed transactions were used to derive the "service times" listed in the matrix at the bottom of the figure (in milliseconds per transaction). The levels of concurrency were input as the average (attained) multiprogramming level and the maximum multiprogramming level. These levels can also be input as distributions. Arrival rates, where needed, were calculated from numbers of completed transactions (500 completed transactions in a half hour measurement period = 1000 transactions per hour arrival rate). CPU priorities and numbers of terminals have been specified for some workloads in this model. Channel utilizations could also be included.

An analytic modeling package like BEST/1 can calculate the response times and throughputs for this workload and configuration. This model can then be used to evaluate the effects of many different kinds of changes: CPU upgrades, disk device upgrades, memory upgrades, various tuning alternatives, workload additions or deletions, file residence changes, shared disk interference from other systems, workload changes of various types, etc.. With the addition of channel utilization data, channel, controller, and string reconfigurations can be evaluated. In each case, one or more of the inputs is changed to reflect the faster CPU, larger memory, etc.. Then, the new utilization, response times, and throughputs are calculated from the modified model. BEST/1 calculation time is short enough to allow many different alternatives to be evaluated in a short terminal session. Simulations can evaluate all the same alternatives, but the CPU time required per alternative would usually limit the number of alternatives that could be considered to a small, pre-selected set. Benchmarks could

```
-----WORKLOAD   1----DESCRIPTORS-----

     LABEL   BATCH PROCESSING
        BP   WORKLOAD TYPE
      1.00   PRIORITY LEVEL
      3.80   ATTAINED MPL

-----WORKLOAD   2----DESCRIPTORS-----

     LABEL   DATA BASE TRANS
        TP   WORKLOAD TYPE
   3200.00   ARRIVAL RATE (TRANS/HR)
      3.00   PRIORITY LEVEL
      4.00   MAXIMUM MPL
      1      DOMAIN ID

-----WORKLOAD   3----DESCRIPTORS-----

     LABEL   TIME SHARING USERS
        TS   WORKLOAD TYPE
     30      NUMBER OF TERMINALS
     25.00   THINK TIME (SECS)
      2.00   PRIORITY LEVEL
      6.00   MAXIMUM MPL

-----WORKLOAD   4----DESCRIPTORS-----

     LABEL   DATA BASE UPDATE
        TP   WORKLOAD TYPE
    800.00   ARRIVAL RATE (TRANS/HR)
      3.00   PRIORITY LEVEL
      4.00   MAXIMUM MPL
      1      DOMAIN ID
```

SERVER	WKL 1	WKL 2	WKL 3	WKL 4
1 CPU	13000.0	150.0	320.0	425.0
2 DISK - PAGE PACK 1	1099.0	75.0	92.0	115.0
3 DISK - PAGE PACK 2	1940.0	125.0	175.0	235.0
4 TAPE	395.0	0.0	0.0	0.0
5 TAPE	496.0	0.0	0.0	0.0
6 DISK - BATCH	343.4	0.0	0.0	0.0
7 DISK - BATCH	92.3	0.0	0.0	0.0
8 DISK - SYSTEM SPOOL	2800.7	7.8	48.3	7.8
9 DISK - SPOOL	8189.0	0.0	0.0	0.0
10 DISK - SPOOL	7723.6	0.0	0.0	0.0
11 DISK - SCRATCH	1514.7	12.2	25.7	146.9
12 DISK - SCRATCH	1418.3	11.5	26.4	131.9
13 DISK - SWAP	1834.2	60.5	60.5	211.6
14 DISK - PAGE PACK 3	1936.0	63.8	63.8	223.3
15 DISK - USER PACK	669.4	9.4	124.8	46.8
16 DISK - USER PACK	588.8	8.2	0.0	46.6
17 DISK - USER PACK	967.0	11.2	0.0	92.4
18 DISK - USER PACK	968.5	11.8	0.0	98.7
(TOTAL SERVICE TIME)	45975.9	546.4	936.4	1781.0

Figure 1. Sample BEST/1 Model

evaluate some of the above alternatives (i.e., only the ones involving hardware that already exists), but the requirement for large amounts of dedicated computer time would similarly limit the number of alternatives.

4. Conclusion

We have reviewed briefly the data a capacity planner typically requires from a computer system. This data is reported by software monitors and accounting files for the large mainframes of the major vendors (IBM, UNIVAC, Honeywell, Burroughs) and also for some of the smaller systems. The discussion in this article should enable capacity planners to make a first-cut determination if they have the system data necessary for capacity planning. Hopefully, as this list and the others are published, the vendors will take notice and plug the remaining holes in the data collected.

CPEUG82

"Improving Organizational Productivity"

A Tool for EDP Management: OMB Circular A-123

SESSION OVERVIEW

A TOOL FOR EDP MANAGEMENT: OMB CIRCULAR A-123

Theodore F. Gonter

U.S. General Accounting Office
Washington, D.C. 20548

The Budget and Accounting Procedures Act of 1950 placed the responsibility for establishing and maintaining adequate systems of accounting and internal control upon the head of each executive agency. The incorporation of adequate internal controls in management systems goes a long way towards preventing fraud, waste, and abuse, while improving the efficiency and effectiveness of government functions and programs. Yet, in the past decade a seemingly unending disclosure of fraud, waste, and abuse in government has surfaced a crisis of confidence in government programs and agencies.

The Office of Management and Budget Circular A-123, Internal Control Systems, is a step to resolving this crisis. The circular prescribes policies and standards to be followed by executive departments and agencies in establishing and maintaining internal controls in their program and administrative activities. The circular defines internal controls as the plan of organization and all of the methods and measures adopted within an agency to safeguard its resources, assure the accuracy and reliability of its information, assure adherence to applicable laws, regulations, and policies, and promote operational economy and efficiency. The Congress is going even further in an attempt to resolve the crisis by preparing legislation which will require on-going evaluations and reports on the adequacy of the systems of internal accounting and administrative control of each executive agency. The legislation, when passed, will be known as the Federal Managers' Accountability Act of 1982. The law will provide the permanence, priority, and continuity which the circular cannot provide.

The theme of internal control will be sounded many times over the next few years. When an agency head reports that there are adequate controls in an organization, he/she will be attesting to the adequacy of controls over the computer based systems which support most of the functions or programs for which he/she is responsible. In turn, an EDP manager must be aware of the internal controls at his/her disposal which will help assure the accuracy and reliability of the computer based systems, and the economy and efficiency of the environment in which those systems are developed and operated.

If an EDP manager is effectively managing, he/she will already have a good system of internal controls in place. However, if he/she is <u>not</u> managing effectively, the Federal Managers' Accountability Act and A-123 will provide the impetus for developing that system of internal controls and the basis for developing an efficient, effective, and sound operation.

DATA PROCESSING AND A-123

Sheila Brand

Department of Defense
Computer Security Center

In preparing for this talk I was struck by the straightforward but negative rationale given for promulgating A-123 in the Background section of the circular. I quote:

> Despite these statuatory requirements, there continue to be numerous instances of fraud, waste, and abuse of Government programs. These problems frequently result from weaknesses in internal controls or from breakdown in compliance with internal controls.

This background paragraph reflects concern that existing laws, in this case the Budget and Accounting Procedures Act of 1950 and the Anti-deficiency Act, were not effective in detering fraud, waste and abuse in government programs.

But government programs is a large target. More specific to this session - DATA PROCESSING - there have been numerous cries of concern in the past over the lack of effective controls in DP systems. In July 1978, OMB issued Circular A-71 Transmittal Number 1: "Security of Federal Automated Information Systems," which directed the establishment and institutionalization of security programs for protecting computerized portions of agency missions.

The Paperwork Reduction Act of 1980 also talks to controls over computerized systems and the establishment of standards and guidelines for implementing computer or system security protections for information collected or maintained by agencies.

In April of 1982, GAO issued a report: "Federal Information Systems Remain Highly Vulnerable to Fraudulent Wasteful, Abusive, and Illegal Practices," which blasts everyone out of the water for lack of guidance and control over computerized portions of Agency missions. This report says that OMB is not doing enough to lead in the area of computer security and that the agencies are not implementing security programs to protect DP resources.

With this growing list of legislation and regulation, and GAO's concern over the lack of controls IS IT POSSIBLE that the government is WORRIED ABOUT THE POSSIBILITY OF BEING RIPPED-OFF BY COMPUTER MANIPULATIONS?

What I seem to hear is: "DO SOMETHING, DO SOMETHING," from GAO, OMB, and Congress. "I DON'T KNOW WHAT TO DO," from the agencies.

The truth is no one knows the extent of fraud and abuse in computer systems. Our systems are not designed or managed to detect computer crime, nor are they, by and large, designed to prevent it. In spite of these hurdles there is a PCIE[1] project underway, led by the IG of HHS to do a government-wide survey to analyze the nature of computer crime in government programs. The objective is to address the agency comment: Is there a computer crime problem and if so what is its nature? Before leaving HHS I was manager of the project and can tell you. There is computer crime. All you have to do is go out and look for it. One agency alone told us of over 60 cases that were uncovered by a proactive, highly trained team of criminal investigator/computer specialists.

But I am digressing from the purpose of this session.

A discussion of computer crime is relevant to A-123 in that it will provide the agency manager a more focused rationale and strategy for implementing provisions of A-123 in terms of computerized systems. When GAO does its next review of the effectiveness of security programs perhaps the picture will have changed and improved.

As I mentioned earlier, the DP systems of today are not designed to deter or detect fraud. Or in terms of A-123, one could say that many DP systems today lack strong internal controls.

---

[1] President's Council on Integrity and Efficiency composed of all Executive Department Inspectors General and chaired by the Deputy Director of OMB.

The remainder of my remarks today will touch on the process by which DP application systems of the future should be designed if they are to be more responsive to assuring the same level of internal control that one expects in the manual or non-automated world.

In the time remaining I will touch on: the concepts of internal control versus computer security. Do they refer to the same or different things? I will then go on to discuss control objectives and why they are so valuable in the scheme of building a controlled system. I will then discuss management of the system throughout the system life cycle to assure a controlled environment for development of a controlled product. (I call this cradle-to-grave attention.) Transaction Flow analysis is a good method for analyzing the points of vulnerability in a computer system without getting overwhelmed by the complexity of the problem. And finally I will describe an audit that looks at all aspects of a system at all stages of the system life cycle. (I call this the Rotund Audit.)

Coming from the DP side of the house, I "grew up" with the term Computer Security. I did not understand the term Internal Control. That was an auditor's term and somehow had a different connotation. When I started reading audit reports and working in the IG's office the two terms began to converge in my minds eye. Today I think of them as meaning the same thing when talking about control in a computer system. I refer to table one to show how closely related the terms internal control and computer security are. If one recognized that the resources to be safeguarded merely refer to information that is maintained, manipulated or generated through use of a computer system and that assuring adherence to laws and regs translated pretty directly into being able to run that part of the agency mission on time, accurately, and without fear of compromise in the form of malicious alteration or destruction - one is talking about computer security.

Computer security has been a concern of a different "crowd" for many years - I must explain that I consider the crowd concerned with A-123 to be mainly financial in perspective with a generous sprinkling of budget and general management expertise. The crowd of computer security has traditionally been composed of computer analysts, system analysts, and some physical security types who have wandered into the computer sphere. And I must tell you ladies and gentlemen that the computer security crowd has not been very successful in getting top management to listen to their concerns. A-71, according to the April GAO report, has been a flop - and from my own personal observations so have most computer security initiatives. I think the problem is that we the computer types, have used the wrong sales pitch - it was too technical, it didn't show a bottom line profit, management has never taken it very seriously.

So when A-123 came along (I call it the Foreign Corrupt Practices Act in Government Clothing) I for one was overjoyed. Because maybe through a slightly different sales pitch the same goals sought by the computer systems professional will be achieved by the accountants, managers and budget people.

In addition to hearing auditors talk about internal controls another new term to me was CONTROL OBJECTIVES.

They are neat little items because they can do so much to foster a strategy for control.

TABLE 1. CONTROLS VS COMPUTER SECURITY

INTERNAL CONTROLS REFER TO:

- SAFEGUARDING RESOURCES
- ASSURING ACCURACY AND RELIABILITY OF INFORMATION
- ASSURING ADHERENCE TO LAWS, REGS, AND POLICIES
- PROMOTING OPERATIONAL ECONOMY

COMPUTER SECURITY REFERS TO MEASURES TAKEN TO:

- PROTECT AGAINST THREATS
- CONTROL TO ENSURE ACCURACY AND RELIABILITY
- MANAGE TO ENSURE ADP AVAILABILITY

ALL ASSETS AND RESOURCES WHICH ARE

- MAINTAINED, MANIPULATED OR GENERATED THROUGH USE OF A COMPUTER SYSTEM.

They set the stage so to speak by giving a lofty set of statements in rather general terms of management policy with respect to control and as a system takes shape, control objectives are broken down into more specific objectives for each part of the system - and finally in the operational system one finds the implementation of the control objectives translated into procedures, hardware, software and physical controls.

Control Objectives can also be used by the auditors to audit against - Does the organization comply with managements policy intent?

Figure 1 shows A-123's Control Objectives. I can translate the first one into a computer version by saying data and other information manipulated by the system must be safeguarded from unauthorized access or manipulation - by using read and write protect features offered by the operating system.

The second one can be translated to mean that only persons whose authorization are known to the computer system can execute certain transactions in that system.

The second-to-last objective is very interesting from a DP point of view - How does one program the computer to assure adherence to laws and regs? This can develop into a very complex translation. The SSI system is a case in point which went through many growing pains (translated into an extremely high payment error rate) before this control objective was achieved in the software system.

- To provide management with assurance that:
  - Resources are safeguarded from unauthorized use or disposition;
  - Transactions are executed in accordance with authorizations;
  - Records and reports are reliable;
  - Laws, regulations, and policies are adhered to;
  - Resources are effectively and efficiently managed.

Figure 1. A-123 Control Objectives

Using control objectives can only be effective if, as was mentioned earlier, they are applied throughout the system - at all stages of the system life cycle. If policy isn't translated into operating reality what good is it?

In terms of a computer system, the life cycle begins during the design or initiation phase, moves into the development phase, and reaches maturity during the operational phase.

Planning for control can be very expensive if not incorporated into the initial design of the system. So it is highly recommended that you don't think of security as an add-on. Right at the beginning ask control oriented questions or bring in the Auditors and Security experts to help you formulate the right question.

In addition to control objectives, one should be concerned as to whether or not source data collected will be accurate and complete enough to be processed by the computer system. In other words can you translate the law into a computer program and use the input data which will be available to carry out the intended mission.

Will personal accountability through automated means be possible - and will there be sufficient separation of duties to FORCE COLLUSION!

In other words, during the design phase of the system you must determine whether or not a computerized implementation of the mission can be controlled - or will you have to change the way you do business?

Of course all of the answers to these questions must be balanced by results of the vulnerability assessment done in a primitive way to sketch out initial "feelings" about threat variety, risks to be endured, etc.

During the development phase you will have a crisper picture of data types and, therefore, an assessment of its sensitivity and criticality to agency mission can be made. This is necessary for an accurate determination of the proper level of control in terms of data protection, back-ups and recovery requirements, edit and validity checks, etc.

Controls may appear to be a nuisance to the end user. If they are perceived as such people will try to get around them - like pasting the password on the terminal.

Human engineering refers to the set of techniques for making the machine/human user interface friendly. This is especially important in the area of controls where user acceptance is critical to success.

Design reviews and user involvement are often neglected with disastrous consequences. In order to assure accurate translation of functional requirement - and this includes control objectives - the system should undergo a number of different levels of review before putting a single line of code to paper. The first line of review should include a dialogue between the initial system designers who may start by translating a law into an agency mission, and the functional specification developers - than the next level of review should be between the functional specification developers and the computer system design spec developers - than between them and the programmers. At each stage of granularity

the system specs should be examined to assure accurate and complete translation of requirements. In this manner the area of control objectives and their translation to specific details will not be lost in the shuffle or mis-represented. Companies that use this approach, such as IBM, have found an enormous savings over the life cycle development costs.

Change control refers to assuring that changes to operational system do not take place "on-the-fly" as it were. This can have disastrous effects on the operation. During the early days of the SSI program programmers were allowed to come in at any and every emergency or even during non-emergencies and put "fixes" into the operational system, sometimes by patching in changes at the computer console. There was no audit trail and no documentation and at times it was quite difficult to tell what version of the various modules were running. The end result was that after a while there was no "institutional" memory and when problems occurred no one knew how to fix them. Not to mention the possibilities for inserting fraudulent code into a system, or inserting code to circumvent control software. The possibilities are endless! Change control is a must.

Test and validation do not refer to the same process. Testing refers to the debugging and system tests conducted by the programming staff. It is usually piece meal and does not put the system through its paces in a live setting. Validation refers to testing by the end user to assure the system performs according to specifications. This would include testing the control features to see how they operate in all kinds of settings.

The operational phase begins once the system has been accepted by its intended users and they become dependent on it to fulfill their mission and responsibilities.

For analytical purposes control objectives of this phase can be broken down by transaction flow, i.e. in terms of: transaction initiation, transaction entry, transaction transmission, computer processing, and file integrity.

Questions that should be asked about transaction initiation include the following:

Are there written procedures and manuals for instructing user on data collection?

Do manuals provide complete instructions on how to fill out forms, prepare input, regulate document flow, identify and correct errors?

Do application forms contain sequence numbers for tight control over use and disposition?

Are manual checks on source documents performed for accuracy, signature authorization, completeness?

Has consideration been given to establishing an independent control group to review applications and source documents for authorization of transactions?

At the close of business are source documents stored in a locked cabinet when not in use to prevent unauthorized modification or use of the data prior to input to the system?

Are only specified persons authorized and capable of initiating a transaction?

For large volume input-jobs are batching techniques and counts used to control the dispositions and flow of applications to insure that none get lost or processed more than once and that none can be added through unauthorized means?

Questions that should be asked about transaction entry include the following:

How effective are procedures and software for assuring accurate source data input, comprehensive source data validation and edits?

How effective are software and procedures for assuring that error-handling allows for error detection, correction, and timely resubmittal?

Is there a reconciliation performed daily to assure that all operations have been processed and nothing lost or added?

The area of access control would include some of these questions:

Are the data entry terminals which are used for updating the data base physically secure?

Are terminals in open work areas restricted via software to query traffic only?

Does the organization employ lists of authorized users?

Are the lists updated with change of personnel?

Are passwords or key/card systems employed?

Are they changed frequently?

Is each authorized terminal user restricted to a predetermined set of transactions?

Are these restrictions implemented in software through manual procedures?

Is an automatic software terminal lock used to prevent unauthorized access to the device? (a) after a predetermined number of unsuccessful attempts at input or (b) at close of business?

Does top management periodically review the different levels of transaction authorization and change them when needs arise?

In designing controls over transaction transmission these questions should be asked:

Do terminals have built-in (hardwired) identification devices to be checked by the host to assure that only authorized terminals can transmit applications for entitlement?

Does the system automatically log all messages?

Does the log contain a time and date stamp for each message?

Is a record count kept automatically of all messages?

Can the system log be used for restarting the system in emergency?

Is the network dedicated to this application?

Have contingency plans been developed so that if an input device is down messages can be rerouted to an alternate terminal?

Have software routines been developed to assure that messages do not get lost in transmission?

Have restart procedures been developed to assure that data can be identified and retransmitted if effected by a system problem, crash or line problems?

Are parity checks performed to assure complete message transmission?

Questions concerning computer processing should include:

Is data validation and editing performed as early as possible in the data flow to insure that the application rejects any incorrect transaction before its entry into the system?

Is data validation and editing performed for all input data fields even though an error may be detected in an earlier field of the same transaction?

Are the following checked for validity on all input transactions: individual and supervisor authorization or approval codes, characters, fields, combination or approval codes, characters, fields, combination of fields, transactions, calculations, missing data, extraneous data, amounts, units, composition, logic decisions, limit or reasonableness checks, signs, record matches, record mismatches, sequence, balancing of quantitative data, crossfooting of quantitative data?

Are special routines used which automatically validate and edit input transaction dates against a table of cutoff dates?

Are all persons prevented from overriding or bypassing data validation and editing problems?

If not, are the following true:

--Override capability is restricted to supervisors in only a limited number of acceptable circumstances?

--Every system override is automatically logged by the application so that these actions can be analyzed for appropriateness and correctness?

Are batch totals and record counts used by the application to validate the completeness of data input?

Do documented procedures exist that explain the process of identifying, correcting, and reprocessing data rejected by the application?

Are error messages displayed with clearly understood corrective actions for each type of error?

Are error messages produced for each transaction which contains data that does not meet edit requirements?

Are error messages produced for each data field which does not meet edit requirements?

Are all data that do not meet edit requirements rejected from further processing by the application?

Does the user department control data rejected by the application by using: turnaround transmittal documents, batching techniques, record counts, predetermined control totals or logging techniques?

Is all data rejected by the application automatically written on an automated suspense file and does it include: codes indicating error type, date and time the transaction was entered, and identity of the user who originated the transaction?

Are record counts automatically created by suspense file processing to control these rejected transactions?

Are predetermined control totals automatically created by suspense file processing to control these rejected transactions?

Is data from the automated suspense file used by management to analyze:

--Levels of transaction errors?

--Status of uncorrected transactions?

Are these analyses used by management to make sure that corrective action is taken when uncorrec transactions remain on the suspense file too long?

Are progressively higher levels of management reported to as these conditions worsen?

Are debit- and credit-type entries (as opposed to delete- or erase-type commands) used to correct rejected transactions?

Is the application designed so that it cannot accept a delete- or an erase-type command?

Do valid correction transactions purge the automated suspense file of corresponding rejected transactions?

Are invalid correction transactions added to the automated suspense file, along with the corresponding rejected transactions?

Are procedures for processing corrected transactions the same as those for processing original transactions with the addition of supervisory review and approval before reentry?

Does ultimate responsibility for the completeness and accuracy of all application processing remain with the user?

Questions concerning file integrity would include the following:

Are application programs prevented from accepting data from the computer console?

Are computer-generated control totals (run-to-run totals) automatically reconciled between jobs to assure completeness of processing?

Where computer files are used by the application as input, are there controls to verify that the proper version of the file is used?

Do all programs require and check internal file header labels before processing?

Can operators circumvent file checking routine?

Are internal trailer labels containing control totals, such as record counts, generated and used by the application to check that all records have been processed and no "extra records" have been processed?

Are all files maintained in a secure area?

Is a backup of the latest version also maintained and is it also in a secure area?

Is the backup maintained in a separate facility so that in an emergency, if the primary was destroyed the backup would not also be destroyed?

Are all development runs executed against a test data base or are the live files used?

In the area of personal accountability the following should be addressed:

Are audit trails maintained by the software on all transactions against each record in a file?

Does this trail include dates, changes, authorizing individual, and location of authorizer?

How long is this trail maintained?

Are audit trails kept of all override actions which allow processing though error or other exception indicator would ordinarily not allow processing of that record to continue?

Do audit trails of overrides identify the input clerk and authorizer of the override?

Do programs exist that can analyze audit trail data so that the reviewer can obtain useful and useable information without manually going through mountains of paper printouts?

CPEUG82

"Improving Organizational Productivity"

WWMCCS Network Performance Analysis

PERFORM - WWMCCS INTERCOMPUTER NETWORK (WIN) PERFORMANCE OPTIMIZATION RESEARCH MODEL

K. Chung*
O. A. Mowafi
K. A. Sohraby

Computer Sciences Corporation
Systems Division
6565 Arlington Boulevard
Falls Church, VA 22046

A hybrid modeling tool comprising both analytical and simulation models is described in this paper. The model, PERFORM, has been designed for WWMCCS Intercomputer Network (WIN) to be used in support of performance evaluation, capacity planning, and both software and hardware architectural studies for the host computers and the WIN subnetwork. Included in the discussion are details of the host and subnetwork models composing the PERFORM tool, software implementation, and future directions.

Key words: Analytical; capacity planning; central server; disk; main memory contention; modeling; packet switch; performance evaluation; simulation; trunk; WIN

1. Introduction

The WIN is a packet switched network designed to support the major WWMCCS communications functions, and to provide computer support to the National Command Agencies, to unified and specified commands, and to other DoD components.

The Director of the Defense Communications Agency (DCA) is tasked to secure network configuration management so as to provide planning, engineering and other technical support for WIN data transmission requirements, and to recommend policies, standards, and procedures for the WIN operation. To do so effectively, DCA must have a means to assess the network's performance, means based on both current and projected workloads and configurations. PERFORM, the WIN performance model described in this paper, is intended to provide DCA with this means.

PERFORM is a hybrid tool composed of analytical and simulation models - and is to be used essentially by WIN network managers and staff to analyze the efficiency and effectiveness of both the host and the network configurations under a variety of workloads and architectural conditions. The submodels support capacity planning, performance evaluation, hardware/software selection studies and improvement studies for the WIN subnetwork and host systems.

This paper is organized as follows: Section 2 presents a brief description of the model's features and the purpose of the model's development. Its organization and the host/subnetwork models supporting PERFORM are given in Section 3. In Section 4, host and subnetwork analytical and simulation models are detailed. Software implementation of the model is discussed briefly in Section 5. Section 6 outlines current efforts to improve the model, and concludes the paper.

2. Model Purpose and Main Features

PERFORM, [1, 2] is a software package built to model the operation of the WIN packet switched network and its attached host computers. PERFORM was necessitated by growth both in the WIN applications volume and in system utilization by its subscribers. As a sophisticated modeling tool, PERFORM can predict WIN performance and provide system planners, managers, and engineers with the required information to identify system bottlenecks, thus enabling early precautions of the required modifications to the system configuration or capacity.

*Currently with Bell Laboratories at Murray Hill, NJ, previously with Computer Sciences Corporation.

PERFORM is intended for use by moderately sophisticated users having some knowledge of the host computer's hardware and software structures, of WIN packet switches, of network protocols implemented on it, and some knowledge of the general communications network operation.

This model is designed to be executable on the Honeywell 6000 series at the DCA-Reston, Virginia; at the Pentagon through an interactive terminal; and other sites equipped with similar computer facilities. Written entirely in FORTRAN, the program is portable, with minimal modifications, to other computers .

PERFORM can be executed in real-time or batch mode at the user's option. Various modules constituting PERFORM's operational capability are analytical. The host memory contentions model is based on a pseudo-real-time simulator specifically designed to represent the WWMCCS host computer's operation.

3. Model Organization

The overall design of PERFORM is based on a hierarchical, modular structure. Its modules are mostly analytical with some minor exceptions incorporating simulations.

The model consists of five major subsystems, each fulfilling special functions and consisting of modules which are programmed with one or more program subroutines. The five major subsystems are:

1. User Interface
2. Workload Model
3. Host Modeling
4. Subnet Modeling
5. Report Generator.

The User Interface Program serves as the primary interface between the user and PERFORM. This program queries the user through a prompt/response session during which the user specifies the hardware configurations, workload characterizations, and model run options. The user is asked high-level questions. Most of the technical details and functional characteristics of host/subnetwork operation are incorporated in the model internally.

The Workload Model portion of PERFORM generates specific workload data for input to the various performance models to be used. The host system and subnetwork workload, as defined by the user, is translated into the basic units of system resources required for its execution. This translation accounts for system resources that are required directly for user program execution, and indirectly for overhead resources. (Overhead resources are those consumed by the operating system and other system software in execution of the user program.) The basic units of system resources and a traffic matrix indicating the subnetwork traffic volume to be modeled are output for use of subsequent component models.

The Host Modeling Subsystem models the performance of interconnected WWMCCS host computers and of local communications, including front-end processors. This subsystem is composed of several models which analyze the host components such as CPUs, IOMs, disks, main memories, front-end processors, and terminals. An executive program sequentially calls a setup routine and the various host models for each host configuration which has been specified. The Host Modeling Subsystem includes the following host models:

1. CPU Model
2. IOM Model
3. Disk Model
4. Central Server Model
5. Regional Communications Model
6. Front-End Processor Model
7. Main Memory Contention Model.

Details of the host models are discussed in Paragraph 4.1.

The Subnetwork Modeling Subsystem provides estimates of subnetwork activity performance. This subsystem consists of the following models which deal with various aspects of the backbone:

1. Throughput Model
2. Packet Switch Model
3. Network Performance Model
4. Buffer Model
5. Trunk Performance Model
6. Misdelivery Model.

Details and modeling methodology are provided in Paragraph 4.2.

The Report Generator is responsible for producing for the user formatted reports of performance metrics, and for workloads and configurations chosen for model runs. The programs in the Report Generator access the files created by other subsystems and output the several types of reports as selected by the user. The report categories include model run definition, configuration definition, workload characterization, host performance, and subnetwork performance.

4. Modeling Methodology

4.1 Host Models

The host models interact with one another as shown in Figure 4-1. The models are executed sequentially on the basis of both user inputs and output from the Workload Model. The arrows indicate that in most cases the output from one model is used as input to the subsequent models. Descriptions of each of the host models follow.

4.1.1 CPU Model

This model produces CPU service times of the various types of workloads for later use by the Central Server Model. Service times are

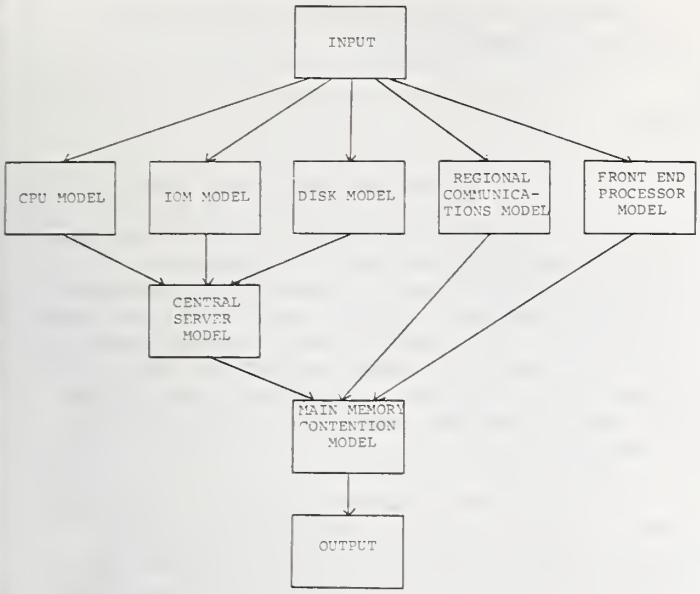


Figure 4-1. Host Model Interactions

derived by dividing the empirical, raw execution rates for single and multiple CPU processors into the machine-level language instructions counts.

The number of instructions executed by the host computer for subnetwork messages is calculated from the subnetwork-related message traffic and the average number of instructions required to process one message. The effective CPU execution rate available for non-subnetwork activities is then estimated by subtracting the expected CPU utilization due to subnetwork message processing.

4.1.2 IOM Model

The Input/Output Multiplexor (IOM) processing time for a transfer of one data block (320 words) is computed by dividing of the number of bytes in one block by the IOM processing rate given in the units of bytes per second. For contention effects, an M/M/1 queue is assumed at the IOM. The expected IOM utilization is obtained from the estimate of the data traffic through the IOM and the processing service rate. The response time through the IOM is calculated from the standard M/M/1 formula.

4.1.3 Disk Model

Based on user input configurations, workloads and branching probabilities (which may be input by the user), this model calculates the disk service time for each disk and the possible contention effects at the channel and disk controllers (MPCs).

The contention for disk services is addressed in the following Central Server Model from the results of the Disk Model and other models.

The disk service time is computed by summing the disk seek time, rotational latency and data transfer time. Furthermore, the time an I/O request waits for an open channel or a controller due to the RPS feature - is represented by P/(1-P), where P is the probability an I/O request shall have to wait an extra disk rotation because the channel or MPC that it must use is busy. Because P is the probability of both the union of the channel and the MPC being busy, the fraction of time each is busy must be calculated on the basis of the expected data traffic and the connectivity of the disk strings to channels.

4.1.4 Central Server Model

This model analyzes the contention among activities resident in the main memory to acquire CPU and disk resources. In order to represent different classes of jobs, the population maximum for each class, a multichain closed queueing network of single servers is formed as indicated in Figure 4-2.

Each server represents the CPU or a disk unit; the service time represents the mean service time at the CPU or disk for each type of job for each CPU burst service. The branching probabilities from the CPU to each device come from the Disk Model, depending on the disk file placement.

The queue discipline at the CPU is assumed to be processor-shared (PS), while the disk queues are first-in-first-out (FIFO). For CPU dispatches, no priorities are assumed for different classes. Furthermore, for a fast execution of the model, the different classes of jobs are bunched together to form only those batch and interactive types which are suitably weighted upon initial estimates of throughputs for the corresponding classes of jobs.

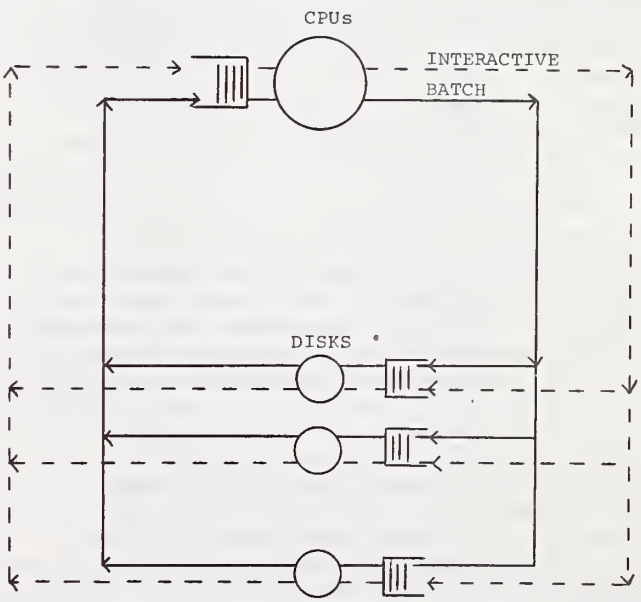


Figure 4-2. Central Server Model

The model calculates both the queue lengths at each device and the throughput for jobs in each job class - for each possible combination of job class population (0 through class maximum) and for all mixes.

The queueing problem is solved by a variation of the Reiser and Lavenberg Mean Value Algorithm [3].

For each combination of chain populations, the Lavenberg-Reiser Algorithm calculates throughputs by class, as well as queue lengths at each device and the CPU. The analysis exploits a relation between mean waiting time and the mean queue size of a system with one less customer.

4.1.5 Regional Communications Model

This model estimates the performance metrics related to the communications between remote terminals and host computers. The communications involving subnetwork nodes (IMPs) are analyzed within the Subnetwork Modeling Subsystem described in Paragraph 4.2.

The regional communications analysis is concerned mainly with the processing of jobs input from a remote terminal by the modems, the front-end processor (Datanet 355), and the CPU. Response time is defined here as the time between sending the job and the receipt of the first response back at the terminal. This definition excludes both user think and keying time at a terminal.

The model is based on a multichain closed queueing network. Each unique routing path of a job entered into a terminal corresponds to a closed path which includes terminals, modems, the FNP, and the CPU. The queueing network for this model consists of both single server queues with first-come- first-served (FCFS) discipline at the FNPs and the CPU, and Infinite Server (IS) queues, for the terminals and modems. Service times at single server queues are assumed to be distributed exponentially. Service times at the infinite servers have a general distribution.

Because the computational problems associated with a general queueing network model of many chains are severe, an approximate technique for the solution of a multichain queueing network had to be developed. The solution algorithm is a heuristic, derived in the style of mean value analysis of Reiser and Lavenberg. The heuristic used is derived after making two approximations [4, 5]. The first assumption is that the mean queue length of chain k jobs at queue j with N_i 1 jobs in chain i is nearly equal to the mean queue length with N_i jobs in chain i. The second approximation is that an arrival from chain i at queue j finds $N_{ij}(N_j-1)/N_i$ other jobs from chain i, where N_{ij} denotes the average number of chain i jobs at queue j. The iterative scheme converges to a satisfactory solution within several iterations.

In the current implementation, the model handles up to 100 terminals per host.

4.1.6 Front-End Processor Model

The purpose of this model is to provide performance estimates for the Front-End Processor (DN355) activities, including traffic through terminals connected to the FNP and subnetwork-related traffic. The model accounts for data traffic through the FNP in both directions; from the host to the packet switch and terminals; and from the packet switch and terminals to the host.

The average execution time of a stored FNP program is calculated from both the FNP cycle time and the average number of FNP cycles per instruction. The average processing time for servicing one character of the traffic to or from a terminal is estimated from the average number of instructions required to service one character for each terminal type. The transmission time per character is computed from the terminal line speed and the number of bits per character.

The average service time delay for a transaction from a terminal and from the FNP utilization requirements is obtained by means of Little's formula.

If a packet switch is connected to the FNP, the overhead involved with the switch protocol (GPLA or HDLC) is estimated as follows: An interpolation from an empirical linear function is made for each protocol which relates the service time for one subnetwork-related message with the average number of packets per message.

The total utilization for the FNP is obtained from the sum of FNP utilizations for terminal traffic and subnetwork traffic. Assuming an M/M/1 queue, the model calculates relevant response times, access times, and queue length.

4.1.7 Main Memory Contention Model

This model is concerned with the operating system (GCOS) storage management for batch and interactive jobs. It simulates the effects of contention between jobs for acquiring the main memory before execution by the processors.

Main memory must be allocated for an activity to be executed by the CPU. When there is more than one activity running concurrently, there may be contention for main memory depending on the available memory and the space required by the activities.

The queue discipline for the main memory involves job priorities, urgencies, and time quantum. Time quantum or time "slice" is the maximum CPU time an activity may use before task switching by the CPU.

For modeling purposes, an execution class is defined as a set of jobs with identical job type (batch or interactive), priority, and time quantum. Furthermore, jobs belonging to an execution class are clustered to have statistically common characteristics, such as work demand per transaction, working set sizes, CPU to I/O Workload ratios, and long wait times for interactive jobs (which includes operator think and keying time). The average values for an execution class are obtained by taking the means, weighted on the basis of the relative throughputs (estimated initially) over the jobs clustered in the same execution class.

For an activity to be processed at the host, it must be admitted into the multiprogramming set to acquire the necessary memory. If there is a waiting activity with a priority higher or equal to that of the activity reaching time quantum, then an activity which reaches time quantum while in the MPS will be taken out of the MPS and put to the end of the waiting queue for the priority assigned. If no qualified activity is waiting, the activity will refresh its time quantum.

The queue discipline at the host is head-of-line (HOL) with priority. When an activity exits from the MPS due to completion of the job or time quantum end, the activity with the highest priority which has been waiting for the longest time is admitted into the MPS, provided MPL rules are satisfied. This portion simulates the effects of urgencies, which change dynamically over time in GCOS.

Thus the queueing network model, which is based on Markovian processes for the main memory contention, has as many closed chains as the number of execution classes (so indicated in Figure 4-3). In the chain representing the interactive jobs, the delay times for IS queues are operator think and keying time plus the duration of communication delays for terminals. In the batch chain, the delay time is zero.

The state-dependent processing rates of the activities in the MPS are derived by obtaining the numbers of populations of interactive and batch jobs and determining the throughput for the corresponding chain populations of the Central Server Model.

Swapping rates depend on many factors: working set sizes of the jobs, total memory size, and the storage management algorithm for identifying the segments to swap out. The swapping effect is accounted for by incorporating the empirical curves (which correlate the swapping rates to the sum of active program working set sizes over the memory capacity). When inadequate memory is available thrashing

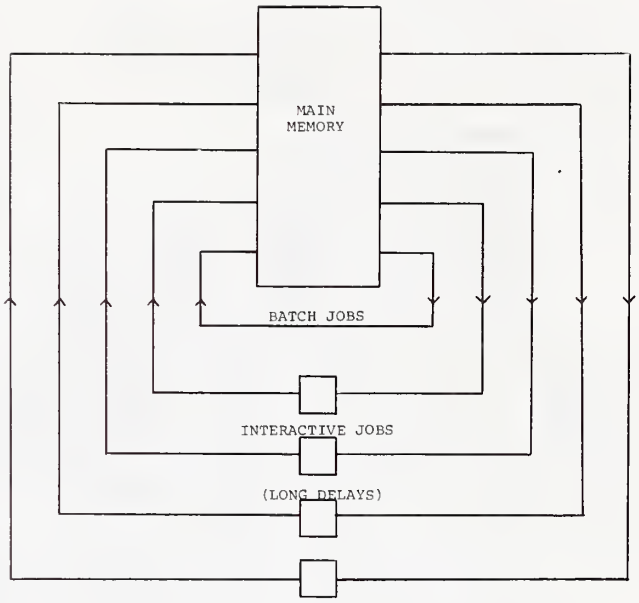


Figure 4-3. Main Memory Contention Model Queueing Network Structure

occurs, a phenomenon of excessive disk operations which are unproductive for the needs of system users. If there are swappings due to the overcommittment of memory, then processing rates are degraded by a state-dependent factor which takes into account additional disk operations. The degrading factor, due to thrashing, is estimated parametrically as shown in the following steps.

For each host state, an overcommittment ratio is computed as the allocated size of each storage pool divided by the total working set size of combined programs running in the corresponding pool. For a given overcommittment ratio, an experiment must be conducted to find the actual number of disk operations required. The processing degrading factor is thus derived by constructing the ratio of the required disk operations without thrashing over the actual disk operations.

The model is based on discrete, event-driven simulations. The style of the FORTRAN simulation is similar to an example for M/M/1 queue simulation [6].

However, due to the nature of the storage management scheme, the multichain closed queueing network with priorities and to state-dependent service rates, the Main Memory Contention Model is significantly more complex than M/M/1.

After reading proper input parameters, the program initializes simulation variables.

All activities are assumed to start from the long wait state (outside of main memory), except batch jobs. The results are insensitive to the initial state chosen due to both the stochastic nature and length of simulations.

4.2 Subnetwork Models

Functions of the following subnetwork models are detailed in this paragraph:

1. Throughput
2. Packet Switch
3. Misdelivery
4. Trunk Performance
5. Network Performance
6. Buffer.

The functional interface of these models in the PERFORM package are shown in Figure 4-4. As shown in the figure, subnet IMP connectivity, user traffic between different source-destination IMPs, network routing information and protocol structure, IMP processor configuration and its capacity, and trunk capacities, are inputs to the subnet models. Some of these inputs are obtained from the user, others from the host models.

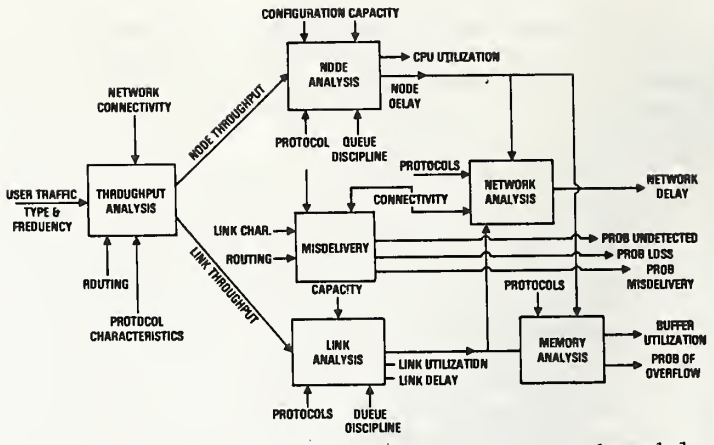


Figure 4-4. Functional Interface of Subnetwork models

The expected time of the next event of each type is computed from the random number generation and the subsequent choice of a value for a random variable satisfying exponential distribution. After comparison of the predicted time of events through a simple bubble-sorting algorithm, the earliest time is determined.

Each time an event occurs, the system queueing rate is revised and up-to-date statistics are collected. One important part of simulation is updating the CPU processing and swapping rates each time the queueing state changes.

After a period of simulation, sufficient statistics are collected and the program terminates; this occurs when the total number of state changes reaches a prescribed value, set currently at 20,000.

Finally, predicted performance metrics are computed from the accumulated simulation statistics.

Each subnetwork model is invoked by a standard FORTRAN subroutine call. Then, a system routine call is made which enables the subnetwork Executive to load the appropriate link overlay and to return to the next sequential statement. The order in which each model is called is very significant since the output of one model may be used as input to one or more of the other models. If an erroneous status is returned from any of the models, an appropriate error message is written to the terminal, both input and output files are closed, and all subnetwork activities are aborted.

4.2.1 Throughput Model

The Throughput model is part of the PERFORM Subnetwork Modeling Subsystem, responsible for calculating data and control packet throughputs. This model performs throughput calculations for both the packet switch nodes (IMPs) and the trunk that interconnect the IMPs - for a given network configuration and a set of WIN network traffic information. Traffic information is obtained from these transactions. Network protocol structure, routing, and various control packets affect network throughput values also. These structures are input by the model user.

The results of the throughput model are written to the Output File and used by the Packet Switch (IMP) Model, the Trunk Performance Model, the Buffer Model, the Network Performance Model and the report generator.

4.2.2 Packet Switch Model

The Packet Switch Model is a functional module that computes both packet processing delays and processor and channel utilizations at a given packet switch (IMP). Packet throughput and system description data of the IMP are inputs from the Subnetwork Output File and Input File, respectively. The computation of packet processing delays is performed in two phases. Phase I calculation uses a queueing model which portrays the task priority structure and the preemptive-resume dispatching system. Results

from this calculation include all the processing delays of each packet in different tasks, such as input, output and routing delays in the node processors. The effect of additional queueing on priority packets in the routing task is then analyzed in phase II calculation, using a queueing model which portrays the packet priority structure and the non-preemptive dispatching discipline. After the two-phase calculation, the processing delay for a particular packet type is computed from the task processing delays and then adjusted by the additional queueing effect corresponding to the packet priority.

This model also computes the processor and channel utilizations of the packet switch. The processor utilization is derived from the processor capacity of a Honeywell H716 packet switch (in instruction execution rate), and the total packet processing load (the sum of instructions executed for nodal traffic). The channel utilizations are computed for a Direct Memory Acces (DMA) channel and a Direct Multiplexor Channel Access (DMC) channel of the packet switch node. Host traffic and network traffic enter or exit by way of the DMC channel and the DMA channel, respectively; and the channel utilizations are derived from the channel capacity that is the total data transfer rate, and the total data transferred through the particular channel. The mean and variance of packet processing delays and associated process or and channel utilizations are written to the Subnetwork Output File. Figure 4-5 is the block diagram of the packet switch model.

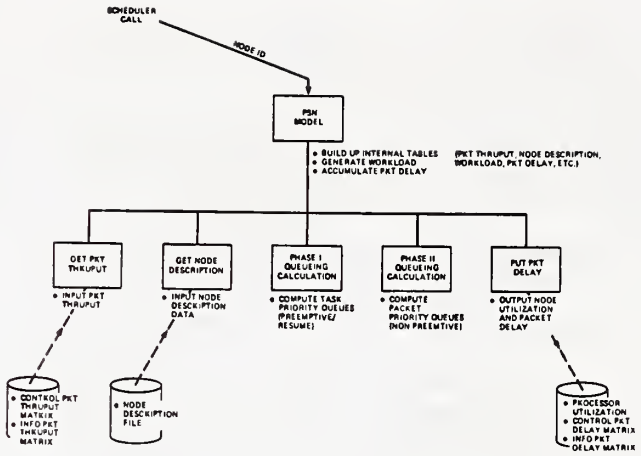


Figure 4-5. Block Diagram of Packet Switch Model

4.2.3 Misdelivery Model

The Misdelivery Model determines four major statistical measurements in the subnetwork model. They are:

1. Probability of Misdelivery - the probability that a destination IMP address in a packet originated from a source IMP and destined for a destination IMP is altered during transmission, and and that the errors are not detected by the error detection mechanism at the receiver side. The occurrence of the errors in the address field can cause the address digits to map to another IMP's address and the packet to be delivered to an incorrect destination.

2. Probability of Loss - the probability of an IMP not finding a match for the received address binary sequence on its address table .

3. Packet Error Probability - the probability of errors in a packet being detected by the cyclic redundancy check mechanism on a random error channel.

4. Probability of Retransmission - the percentage of traffic that is retransmission.

The most tractable mathematical model for characterizing the error channels is a Renewal Model (other models are Markov Processes). The simplest renewal channel is the binary symmetric channel. In a binary symmetric channel, errors on received codewords (packets) occur independently of each other. This type of error is applicable to random error channels.

When a zero is sent in a binary symmetric channel with bit error probability P, it is usually received as a zero. But occasionally, in a point-to-point communication, a zero will be received as a one or a one as a zero. When a packet is transmitted from a source IMP to a destination IMP, the binary digits may bounce back and forth from zero to one or one to zero if error occurs on the digit over different hops. The misdelivery model determines the probability that a received digit is in error after traversing N-hops through an N-cascaded binary channel.

4.2.4 Trunk Performance Model

The Subnetwork Trunk Model computes queueing delays (transmission delay + waiting delay) and utilizations of transmission trunks modeled in the Subnetwork by PERFORM. It obtains network information such as connectivity, line speeds or packet size from the Subnetwork Input File, and traffic information such as throughput, retransmission rate, which are computed by other models, from the Subnetwork Output File. It uses an M/G/1 priority (non-preemptive, first-come-first-serve) queueing model [7] to model the BSC link protocol.

4.2.5 Network Performance Model

The Network Model of the PERFORM Subnetwork Modeling Subsystem computes total backbone delays and their distribution. Using the average and variance of packet delays generated by the Packet Switch and Trunk Models, this model calculates the following performance parameters:

1. Average and variance of end-to-end packet delays for each type and priority of data and control packets

2. Average and variance of end-to-end packet delay for each type over all priority groups of data and control packets

3. Average and variance of end-to-end packet delay integrated for each priority group over all types of data and control packets

4. Average and variance of total network delay for each packet type

5. Average and variance of total network delay.

Outputs from this model are written to the PERFORM Subnetwork Output File for generating appropriate reports. They are also used by the PERCENTILE subroutine for calculating backbone delay distribution.

The Gamma function is suitable for the backbone delay distribution of most operational networks. Using the Gamma assumption, the PERCENTILE subroutine calculates the 20th, 50th, and 90th percentiles of the distribution for all source-destination IMP pairs, as well as the total backbone delay for the data packets [8]. Lists of the output from the PERCENTILE subroutine are given below:

1. Percentiles for end-to-end network delay for each type and priority

2. Percentiles for end-to-end network delay for each type over all priorities

3. Percentiles for end-to-end network delay for each priority over all types

4. Percentiles for end-to-end network delay for each packet type (regardless of the source-destination IMP)

5. Percentiles for the total network delay.

Output reports, in addition to the average and variance of each delay component, also provide the three percentiles. The percentile subroutine reads the Subnetwork Output File for the average and variance delays.

4.2.6 Buffer Model

The Subnetwork Buffer Model analyzes the store-and-forward buffer pool for each packet switch modeled in PERFORM. It obtains certain parameters from the Subnetwork Input File and traffic data from the Subnetwork Output File. The model assumes the buffer pool in each packet switch to be totally and dynamically shared by all the trunk input and output functions. (However, reassembly buffers are not analyzed by this model.) It computes the probability of buffer overflow for a given buffer pool size and the required buffer pool size for a given required probability of overflow. The mathematical model used is the M/M/C queueing model.

4.3 Integrated Models (Host and Subnetwork)

The User Interface Program queries the user for the type of model run. The user may choose to model the host only, the subnetwork only, or both the host and subnetwork.

Most aspects of the integrated models are imbedded in host and subnetwork modeling subsystems along with the code which is applicable to separate host or subnetwork models.

In the User Interface Program, the user is asked to identify the FNPs through which subnetwork-related messages are transmitted. The user must also provide the connectivity of the FNPs to network switches and the access line protocol (HDLC or GPLA). Message traffic rates between hosts are prompted for in the integrated run, while data packet traffic between nodes are input for modeling subnetwork only. All other prompts are the same as those for separate host or subnetwork runs.

A part of the Workload Model converts message traffic into data packet traffic on the basis of average message size. Also, the required data packet traffic between different nodes over the subnet is estimated from the host node connectivity and message traffic. Beyond this, the Subnetwork Modeling Subsystem execution is insensitive to the type of model runs.

For integrated model run reports, end-to-end delays are output after summing the access times of messages to the nodes and the data packet delivery times between appropriate nodes. In PERFORM modeling, there are several ways in which subnetwork-related activities impact the host activity performance and vice versa.

In the CPU model, the CPU service times for processing subnetwork data traffic will decrease the effective CPU processing rate for the host activities.

The regional data traffic to and from terminals attached to the FNP and subnetwork data traffic compete for FNP services. This type of contention is analyzed in the FNP Model.

Also, as coded in the Main Memory Contention Model, the NCP (or equivalent of it) occupies a portion of the main memory of the host so that the net storage space available to the host activities is reduced. This may increase swapping rate and result in performance deterioration of host activities.

Subnet reports for an integrated run include end-to-end subnet delays between source-destination IMPs, trunk utilization figures, throughput figures, store-and-forward buffer requirements and buffer overflow probabilities for various WIN IMPs, as well as misdelivery and packet error delivery statistics.

5. Software Implementation

PERFORM is programmed to be executable on the Honeywell 6000 series in either the real-time or batch mode. The entire program, including simulations, is written in FORTRAN and thus will be portable, with minimal modifications, to other computers.

The entire code consists of approximately 10,000 FORTRAN executable lines and requires over 200K bytes to be contained in the system at one time. In order to minimize these demands and enable the entire model to run in time-sharing, the model is divided into several overlays, enabling the entire model to be run within 32K bytes of core.

PERFORM is composed of five major subsystems and a controlling model driver, as shown in Figure 5-1. During the program run, the overlay mechanism is controlled by the PERFORM Driver at the top level, as determined by the user. The Driver calls modeling subsystems, which in turn call the modules of the component models.

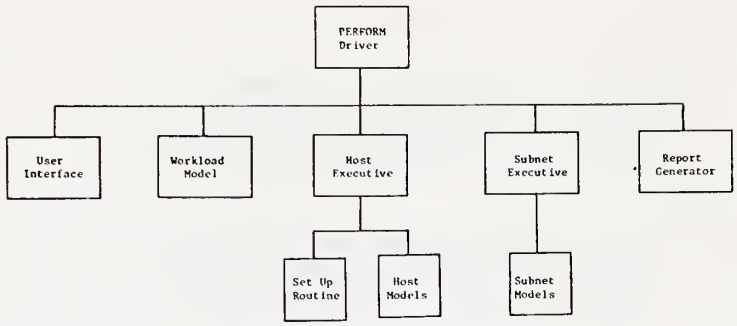


Figure 5-1. PERFORM Hierarchy

Both input and output variable data are passed through different modules through COMMON statements and data files. Data files are extensively used for communication between the major PERFORM subsystems. Additionally, data files are used to store equipment characteristics data for model input and to store typical user responses, host workloads, subnetwork traffic and configurations. The use of these data files will facilitate user interaction with PERFORM and permit flexibility in model use.

Workload Definition Files are used to store workload data relating to a predefined or typical host or subnetwork workload.

By referencing these files during the interactive portion of PERFORM, the user has a simplified means of defining host and subnetwork workloads. Public Response Files are used also to simplify the interactive session by allowing the user to reference previous responses and to make appropriate modifications.

As a result of model execution, a User Response File, Workload File, Host Output File, and Subnetwork Output File are generated. The output files are used for generation of reports.

The User Interface Program is designed for personnel having technical management and planning responsibilities. The user is expected to have some experience with computer and network performance evaluation to interpret the model's performance results. However, the user is not required to have detailed knowledge of the operating system, hardware characteristics, queueing theory, or modeling techniques. The parameters which influence performance but are not provided by the user are internally coded. Error messages are displayed when appropriate, to indicate inconsistent responses, followed by requests for the correct data.

In an attempt to limit the model memory requirements and also to keep the total model run-time within reasonable bounds, the maximum sizes and complexities of the modeled system configurations have to be restricted.

The current version can handle up to 20 host computers, 16 workload clusters per host, and 20 subnetwork nodes.

Up to 10 hosts - with moderate device configurations and workloads - and 10 switching nodes can be modeled within the CPU execution time of roughly 5 minutes on the Honeywell 6080.

6. Summary and Conclusions

PERFORM, a hybrid modeling tool composed of both analytical and simulation models, is described herein. It enables network designers, managers and planners to analyze both the efficiency and effectiveness of the WIN network, host configurations and network performance as WIN evolves.

Currently CSC is tasked by the DCA to validate the PERFORM model in terms of real-world performance data which is being generated by the GMF and other WIN monitoring tools available at the DCA.

Acknowledgements

The mathematical analyses presented here were developed under DCA contract DCA-#100-78-C-3098. The views in this paper are those of the authors and should not be interpreted as necessarily representing the views or official policies, expressed or implied of the Defense Communications Agency.

The Authors would like to acknowledge the contributions of their colleagues K. Chan, S. Hick, C. Omidyar and E. Yang to the development of PERFORM.

References

[1] Chung, K., Mowafi, O. A., Sohraby, K. A., Performance Modeling of WWMCCS Intercomputer Network, Proceedings of the IEEE Military Conference, Boston, 17-20 October 1982.

[2] WWMCCS Intercomputer Performance Research Model, Software Documentation Letter Report. Prepared by the Computer Sciences Corporation under Contract DCA-#100-78-C-3098, 30 November 1981.

[3] M. Reiser and S. Lavenberg, Mean-Value Analysis of Closed Multichain Queuing Networks, Jrl. of the ACM, Vol 27, No 2, 1981, pp 313-322.

[4] Y. Bard, Some Extensions to Multiclass Queueing Network Analysis, to be published.

[5] G. Stroebel, private communications.

[6] C. McMillan and R. Gonzalez, Chapter 12, Systems Analysis, A Computer Approach to Decision Models, Richard D. Irwin, Inc., 1968.

[7] Saaty T. L., Elements of Queueing Theory with Applications, McGraw-Hill Book, 1961.

[8] Martin, J., Systems Analysis for Data Transmission, Prentice-Hall, 1972.

A SIMULATION STUDY OF A LOCAL AREA NETWORK FOR A COMMAND AND CONTROL CENTER

Kathy K. Rebibo

The MITRE Corporation
McLean, Virginia 22102

A proposed architecture for integrating ADP resources at a military command and control center is a local area network. A computer simulation model was developed to study the throughput and performance characteristics of a local area network under various site configurations and workloads.

Keywords: Computer performance modeling; computer simulation; local area network

A possible future architecture for a military command and control center is a local area network. The local area network would connect the various data processing devices within each site. It could also contain a "gateway" to a long-haul network which would connect individual sites together.

To support system specification, an evaluation tool is needed to study the throughput and performance characteristics of a local area network at a site. MITRE chose discrete-event simulation modeling as one of the tools to study the network performance under various site configurations and workloads. The model, called WISSIM, measures the utilization of the network, response time to the user, and transfer rates of bulk data.

Experiments using the model were designed to determine the sensitivity and limits of the network by varying the workload on the network. Specific questions that were addressed in the analysis are:

- What is the maximum throughput of a single network interface unit (NIU)?
- What is the network channel (cable bus) utilization under various workloads?
- What is the NIU processor utilization under various workloads?
- What is the increase in response time as a function of network congestion?
- What is the file transfer time under various workloads?

CPEUG82

"Improving Organizational Productivity"

Workload Characterization

WORKLOAD CHARACTERIZATION USING IMAGE ACCOUNTING

Rajendra K. Jain

Digital Equipment Corporation
77 Reed Road (HL2-3/C09)
Hudson, MA 01749

Rollins Turner

Digital Equipment Corporation
1925 Andover Street (TW/D16)
Tewksbury, MA 01876

Most operating systems record system resource usage during a user session for accounting and billing. The user session consists of running many programs; however, the information is usually not broken down for individual programs. Image accounting consists of recording the usage information as each program image is activated or run. The data recorded by this facility tells us the relative importance of various programs, in terms of CPU usage, paging demands, I/O operations, and people's time. It also provides information on workload characteristics, such as the average number of characters written to a terminal with one output operation. Analysis of this data can provide useful insights on the potential for improvements in system level performance resulting from various possible optimizations. This paper describes techniques that can be used to exploit this information. The methodology is illustrated by actual examples of image accounting data collected from VAX/VMS installations at six universities.

Key Words: Computer accounting; Representative Workload; System monitoring; Workload characterization; Workload measurement;

1. Introduction

One of the major issues in most performance evaluation studies is selection of the workload [1,2][1]. It is important to use a workload for measurements or predictions that is representative of actual system workload. Often, however, there is no solid information on the characteristics of the actual workloads. It turns out that much of this desired information can be obtained by recording resource usage. Most computer systems record resource usage for accounting and billing. The accounting logs provide a record of the resources used, such as CPU time, connect time, and I/O operations for each user session. This information can be used to construct a representative user session[3]. However, if the accounting information were recorded for each program that a user runs, we could get some more useful statistics about workloads. Following is only a sample list of questions from system developers, performance analysts, marketing personnel, and system managers that can be answered using this data:

1. Which programs are most frequently used?
2. Which programs use most of the CPU time? People's time? Disk time?
3. What are the characteristics of the workload? What is the average CPU time? ... Number of disk I/O operations?
4. Which programs offer the best opportunities for system level impact on CPU utilization? ... on People's time?
5. How uniform is the workload among separate installations?

In order to provide such information as a basis for performance studies at Digital, an effort was undertaken to collect workload

[1] Figures in brackets indicate the literature references at the end of this paper.

operating system. Direct I/O is usually, but not always, disk I/O. The major use of buffered I/O is communication with user terminals. For both classes of I/O separate totals are maintained for input and output. For each type of transfer, the system keeps track of the number of I/O operations and the number of bytes transferred. The values of these eight counters are included in each image accounting record.

Paging information includes the number of read operations and the total number of pages read. The number of read operations is smaller than the number of pages read because VMS normally reads in several contiguous pages whenever a page read is necessary. The image accounting record also includes the number of page faults. This is normally a much larger number than the number of page read operations, because the required page is often found in one of the page caches maintained by VMS. These caches hold physical memory pages that have been removed from the page maps for the programs to which they belong but have not yet been allocated to other programs. When a fault occurs for such a page, the page is simply remapped to the program and no disk read is required. Table 1 summarizes the data items recorded by the VAX/VMS image accounting utility.

In designing a facility such as the one described here, one inevitably faces tradeoffs between cost and precision. Some of the tradeoffs in deciding what data items to record are as follows. If disk storage is constrained, program identification may consist of program name only; disk device name, directory and version numbers may be omitted. Instead of recording both start time and end time only the difference may be recorded. This will, however, not allow the analysis of activities that happened during a certain time period, say prime time. For I/O operations, the device utilization generally depends more on the number of I/Os than

Table 1: Data Recorded by VAX/VMS Image Accounting Utility

Name of the image file, device and directory
Program start time: date and time of the day
Program End time: date and time of the day
CPU time used by the program
Number of direct writes
Total direct write bytes
Number of direct reads
Total direct read bytes
Number of Buffered writes
Total buffered write bytes
Number of buffered reads
Total buffered read bytes
Number of Page read I/Os to the paging device
Number of Pages read from the paging device
Number of page faults

information from several VAX/VMS installations running their normal workloads. A facility known as "image accounting" was developed for VAX/VMS. The word "image" refers to program images, the executable form of programs in the VAX/VMS system. The image accounting facility provides information about each program execution during an interval over which it is enabled.

This paper discusses the issues of designing an image accounting facility, analyzing the data, and interpreting it. The methodology is general and is illustrated with our actual experience. Some interesting observations about workloads in educational environments are also presented in the paper.

2. Design of An Image Accounting Utility

While designing an image accounting utility there is considerable flexibility in terms of what data should be collected about each program. If more data is collected, more information can be obtained about the workload. However, more data also means more monitoring overhead and more use of disk storage. This section provides some guidelines to help make this tradeoff. First, the VAX/VMS image accounting utility is described; then the possible tradeoffs are pointed out.

The VAX/VMS image accounting utility can be easily enabled and disabled by the system manager. This helps keep the monitoring overhead under control. During the interval in which the utility is enabled, it records information about each program execution. The VMS operating system maintains resource usage information for each process. Image accounting writes a record into the system accounting file giving the current values of various counters and timers whenever a program begins execution and again when it terminates. By comparison of records for initiation and termination of a program we can determine resource usage and elapsed time for the program execution.

The data provided by the image accounting facility can be divided into three categories:

1. Identification and timing

2. Program I/O

3. Paging

Each record includes complete identification of the file from which the program image was obtained and the identification of the process in which the program was executed. Timing information includes the CPU time charged to the process and the actual clock time.

Program I/O information includes totals for the two classes of I/O that VMS provides, direct and buffered. In direct I/O the transfer takes place between the device and memory belonging to the program. In buffered I/O, the actual transfer is to or from a buffer supplied by the

on the size; the size may therefore be omitted. Distinction between input and output may or may not be necessary depending upon the symmetry of the device. However, as we point out later, inputs and outputs have generally different frequency and sizes. In some environments, the paging is global and the paging mechanism reads and writes pages belonging to several programs together. In such cases, paging done by an individual program may not be distinguishable and can not be recorded in the image record. For example, in VAX/VMS the page writes are done from global page caches only. The paging information on individual programs therefore consists of page read operations only. Like other I/O, number of paging operations is more important than the size. The user identification may not be recorded; however, recording it permits reconstruction of the sequence of programs run by a specific user or in a specific session. Also, the number of users simultaneously active on the system can be calculated from this.

3. Limitations of Image Accounting

There are a number of limitations to the data available through an image accounting facility. Some of these are specific to VAX/VMS image accounting and others are more general.

In VAX/VMS image accounting, the data is logged after the program completes. Many programs, particularly some device control programs, run indefinitely. Therefore, no data is logged for those programs. A periodic recording, say every 15 minutes, of resources consumed by such programs could alleviate this problem. Alternatively, the counters could be logged at the beginning and the end of the period during which image accounting is enabled.

The second known problem is that only the resources charged to a process are recorded. Some resources, for example, CPU time during interrupts, may not be charged to the responsible process. Thus, various programs may have taken slightly more or less CPU time than that indicated here. Similarly, I/O operations done by the device control programs (ACPs) on behalf of a user are not charged to the user.

The third problem is that VMS does not distinguish I/O operations by physical devices. It only distinguishes if the I/O is direct or buffered. This makes the physical device related analysis difficult. For example, size of an average terminal read can not be accurately determined, because even though the buffered I/Os are predominantly terminal I/Os, they also include I/Os to mail boxes, networks, magtapes, line printers, and card readers. Fortunately, for many commonly used programs it is possible to tell which device was being used.

The fourth problem with image accounting in general is that many data items depend upon system wide conditions along with the program behavior. For example, the elapsed time and the paging behavior of a program varies with the system load. Finally, the image accounting does not log any system level information such as queue lengths or device utilizations. The elapsed time includes service as well as queueing time at various resources, along with the user think times in many cases. It is not possible to separate the effect of queueing (system level phenomenon) from service times (program's demands). Many programs, including most editors, frequently wait for user input throughout execution. Again, there is no measure of the amount of time spent waiting for user inputs, and no way to distinguish between this delay and queueing delay.

4. Analysis and Organization of the Data

The first step in analyzing the image accounting data for workload characterization is to merge together all records for the same program. Unimportant programs can be either dropped off, or combined together as one group. For example, we concentrated solely on the programs taken from the system area. All user area programs were merged as one group. Defining program groups (e.g., a group of all editors, another group or all language translators) helps in studying the common characteristics of these groups. The groups do not have to be mutually exclusive. For example, in our analysis we defined six different groups:

COMPILERS	: All language translators including assemblers.
EDITORS	: All text editors.
DIGITAL	: All programs supplied by DEC.
SYSTEM	: All programs taken from the system directory.
USER	: All programs taken from the user directories.
TOTAVG	: All programs, both system as well as user written.

While merging, averages, variances, histograms, or other statistics such as median and percentiles may be calculated separately for each program or program group. We calculated only averages and coefficients of variation.

For each program, many different data items can be calculated. For example, the items that we calculated for each program are shown in table 2. These consists of 4 types of derived statistics on resource consumption: Per activation, Percentage of total resource consumed by all programs, per second resource consumption rate, and per CPU second resource consumption. The significance of these data items is described in the next section.

We found it useful to organize the same data in two forms of tables: data tables and image tables. The data tables presented information about a particular data item, say CPU time per activation, for all programs and program groups. The programs were listed in order of decreasing resource consumption. This sorting helped in identifying important programs. An image table, on the other hand, presented all data items about

a particular program in a single table. The image tables are of interest to developers and designers of individual programs, whereas, the system managers find data tables more useful.

5. Interpretation of the Data

Each data item provides useful information about the program and its users. Different people interpret the same data item in different ways. In this section, we explain how some of the derived data items can be used by developers, product managers, system managers, and others. The discussion is organized around the four categories of data items: per activation, percentage of total, per second, and per CPU second.

Table 2: List of Data Items Derived from the Image Accounting Log

Number of activations
% of total activations
CPU time per image activation
% of total CPU time
Percent CPU rate
Elapsed time per image activation
% of total elapsed time
Number of direct writes per image activation
Direct write size in bytes
Number of direct reads per image activation
Direct read size in bytes
Number of direct I/Os
Direct I/O size in bytes
% of total direct I/Os
% of total direct I/O bytes
Direct I/Os per second
Direct I/O bytes per second
Direct I/Os per CPU second
Direct I/O bytes per CPU second
Number of buffered writes per image activation
Buffered write size in bytes
Number of buffered reads per image activation
Buffered read size in bytes
Number of buffered I/Os per image activation
Buffered I/O size in bytes
% of total buffered I/Os
% of total buffered I/O bytes
Buffered I/Os per second
Buffered I/O bytes per second
Buffered I/Os per CPU second
Buffered I/O bytes per CPU second
Number of page reads per image activation
Pages read per image activation
Page read size in pages
% of total page reads
% of total pages read
page reads per second
Pages read per second
page reads per CPU second
Pages read per CPU second
Page faults per image activation
% of total page faults
Probability of page read per fault
Page faults per second
Page faults per CPU second

Per activation data is the average resource consumption per activation of the program. This information is useful for performance analysts in modeling program behavior, and in constructing synthetic workloads.

By "Percentage of Total" resource consumption we mean the resource consumed by all activations of a particular program expressed as a percentage of the total resource consumed by all activations of all programs. For example, if the percentage of total direct I/Os for program A is 10%, it indicates that 10% of all direct I/Os on the system can be attributed to program A. To developers and product managers, percentage of total data items show the opportunity that different programs offer for reducing consumption of that resource. If the programs are sorted in the order of decreasing "% of total" consumption of a particular resource, the programs on the top of the list will have a high impact and the programs on the bottom of the list will have little impact. Thus, if we find that the program A, consumes only 0.01 percent of the total CPU, it may not be worthwhile to devote manpower to optimize this program for better CPU efficiency. Since elapsed time is also people time, programs with high percentage of total elapsed time may provide high potential for improvements in worker productivity.

The "% of total" data items are also of interest to performance analysts. Analysts interested in analyzing a particular resource, say disk, should include in their workload programs consuming high percentage of that resource. For example, an analyst interested in studying a remote disk server should choose programs observed to consume a high percentage of disk I/Os. Programs consuming only say 0.01% of disk I/Os even though performing poorly on a system with remote disk server will have little impact on the total system performance. Similarly, programs with high percentage of total CPU time should be used to analyze the impact of new CPUs, programs with high percentage of total buffered I/Os should be used to analyze the impact of new terminal devices, remote terminal front ends, and network links connecting the terminals. Programs with high percentage of total faults should be used to study new virtual memory schemes.

Per second resource consumption rates are obtained by dividing the resource consumption by the elapsed time. These rates indicate the intensity of resource usage by the various programs. They can be used to calculate approximately the number of users that can be supported without making the resource a bottleneck. A higher number of users will increase the response time beyond the observed

value. For example, direct I/O operations per second and direct I/O bytes per second determine the disk utilization or network link utilization in systems with remote disk servers. System planners can use this information to determine the number and types of the disks required. Analysts and customers can use it to find out the programs that cause disks to become bottleneck. Similarly, high CPU rate (CPU time per second) indicates highly compute bound programs. A small number of highly compute bound programs may saturate the CPU. Buffered I/Os per second, and Buffered I/O bytes per second indicate utilization of terminals, remote terminal front ends, and network links to these front ends.

Per CPU second resource consumption is obtained by dividing the resource consumption by the CPU time consumed. Per CPU second consumption is less variable than per second consumption. This is because the elapsed time depends heavily on the system load which varies widely over time. Each CPU second represents the execution of a certain number of instructions. Per CPU second data, therefore, gives an idea of resource demand per instruction of the program. For example, page faults per CPU second give the number of instructions per page fault. Similarly, page read operations per CPU second indicate the number of instructions between successive page read operations. Performance analysts can use this data to compare their synthetic workloads with those measured on the system.

Per CPU second resource consumption also represents the ratio of the resource demand to CPU demand. Programs with higher per CPU second resource consumptions tend to impose higher demand on the particular resource than the CPU and tend to make the resource as the bottleneck. For these programs, therefore, the maximum number of simultaneous users is determined by the capacity of the corresponding resource. For example, the programs with high direct I/O per CPU second are generally disk bound; the maximum number of simultaneous users of such programs on a system is determined by the throughput of the disk.

The probability of page read on a page fault is obtained by dividing the number of page read operations by the number of page faults. Page read operations consume more CPU time than page faults satisfied in the memory. Therefore, the programs with low probability are in a sense better than those with high probability. However, this probability also depends upon the number of users sharing the program image and the elapsed time of the program. As the number of users sharing the image goes up, the probability of needing a read goes down.

Finally, a word about the number and amount of I/Os. Depending upon the I/O device, number of I/O operations or the total bytes transferred may be more meaningful. For disk devices in which seeks take much longer than data transfer, which is the case with practically all disk devices, the disk time depends heavily on the number of I/Os and is not influenced by the size of the I/O. Similarly, for remote disk servers using a message oriented protocol, such as DECnet, the number of packets on the link, as well as the CPU time consumed in the communication depends heavily on the number of I/Os rather than the size. In both these cases, direct I/Os per activation, direct I/Os per second, and direct I/Os per CPU second would provide more meaningful information than direct I/O bytes per activation, direct I/Os bytes per second, and direct I/O bytes per CPU second respectively. On the other hand, for some mass storage devices, set-up (seek and rotational latency) time is less than the transfer time, and for remote disk servers with stream oriented protocols, such as BSP, the link time depends upon the number of bytes transferred. In such cases, the number of bytes transferred provides more meaningful information than the number of I/Os. Similar arguments hold for buffered I/Os and paging operations.

6. Commonly Asked Questions and Their Answers

There are two ways to organize a doctor's prescription manual. One would be to list the diseases for each medicine. This is what we have done so far by describing how we can analyze the data and what information can we obtain from each data item. However, another alternative, probably a more useful one, is to list prescriptions for each disease. That is, given a performance question, which data items from the image accounting should be looked at? This section therefore restates the information described in the previous section in a different manner. Given below are some of the commonly asked questions that can be answered by the image accounting. These are sample questions, one can make and answer other similar questions by appropriate substitutions for items enclosed in angle brackets <>.

Q: Which programs are using most of the people time?

or,

Which programs should the programmers be trained to use more efficiently?

or,

Which programs provide the highest opportunity for better human interface?

A: The programs with high "% of total elapsed time" provide the best opportunity for training as well as better human interface. For example, in our data we found that at universities one-third of people's active time was being spent in editing. The universities may be able to improve productivity by training their users in efficient use of the editors.

Q: Which programs are good candidates for code optimization?

or,

Which programs should be included in a workload to be used in analyzing a new CPU's performance?

or,

Which programs are using most of the CPU time?

A: The programs with high "% of total CPU time" will have the highest impact on CPU usage. For developers, these programs offer opportunity for code optimization, For analysts, these programs provide the programs to be included in their workloads for CPU analysis. In our data, the programs on with high "% of total CPU time" were mostly compilers and editors.

Q: Which programs offer the highest opportunities for code restructuring to minimize the page faults?

or,

Which programs should be used to analyze a new paging device?

or,

Which programs are keeping the paging device busy?

A: The programs with high "% of total page faults", "% of total page read operations", and "% of total pages read" provide opportunity for the highest impact on page read operations. These programs do not necessarily have a poor paging behavior. However, even a small improvement in their paging behavior will have significant impact on the system.

Q: Which programs have a poor locality of reference?

A: The programs with high "Page faults per CPU second" have a poor locality of reference. These programs might benefit from code restructuring. However, given a set of limited resources, more benefit at the system level would be obtained by restructuring programs with high "% of total Page faults".

Q: How many <Program A> jobs can run simultaneously on one <CPU> without undue performance degradation?

A: The CPU rate (CPU time per second) as observed on a system with desired performance can be used to calculate the number of simultaneous users, provided the program is compute bound. For non-compute bound programs, the device with the highest utilization determines the number of supportable users.

Q: Which programs are <disk>-bound?

or,

Which programs' performance is highly dependent on the <disk> characteristics?

A: The programs with high "DIRECT I/Os per CPU second" generally impose more load on the disk than on the CPU. These programs are disk bound and their performance depends critically on the performance of the disk.

Q: What is the ratio of <disk> reads to writes?

or,

What is the average size of a <terminal> read?

or,

What is the average number of <disk requests per seconds>?

A: The data for average of all programs combined (TOTAVG) provides this information. In our data we found that the ratio of disk reads to writes was 3:1. The average size of a buffered read was 4.18 bytes.

Q: What is the average think time in <Program A>?

A: Although think time may not be exactly obtained from Image Accounting, we may get some idea of it from the mean time between terminal reads (reciprocal of terminal reads per second). The think time should be generally less than the time between terminal reads.

Q: Which programs should be used to analyze a new <terminal concentrator>?
or, to analyze a new protocol for remote terminal communications?
or, to analyze a new remote terminal communication link?
or, to analyze a new terminal controller?

or,

Which programs offer the highest opportunity for terminal I/O optimization?

A: All these questions refer to the devices that impact buffered I/Os. Therefore, the programs high on the "% of total buffered I/Os" or "% of total buffered I/O bytes" provide good candidates for these studies.

For devices using a message oriented protocols (such as DECnet), the number of packets on the network link and the CPU time consumed in the communication depends heavily on the number of I/Os rather than the size. In such as case, programs with high "% of total buffered I/Os" will have the highest impact on the device or communication link usage. For developers these programs offer opportunity for buffered I/Os optimization.

For devices using stream oriented protocol (such as BSP), the communication link time will depend on the total bytes transferred. In such a case, programs with high "% of total buffered I/O bytes" will have the highest impact on the communication link and the CPU consumption.

Q: What would be the <network link> utilization or <terminal concentrator> utilization when <n> users are simultaneously running <Program A>?

or,

How many <Program A> users can be supported with a <terminal concentrator>?

A: For terminal I/O bound programs (i.e., programs with high terminal I/Os per second, or high terminal I/O bytes per second) the terminal or link utilization can be determined given the rate and the device's speed or bandwidth. The reciprocal of the device utilization gives an approximate idea of the number of users supportable before the device would become the bottleneck.

Q: Which aspect of <Program A> offers the highest opportunity for optimization to impact the system performance.

A: The resources used by the program A as a percentage of total resources used by all programs should be compared for different resources. At the system level, the resources with high percentage will be impacted highly by program A.

Q: Is remote terminal emulator script of a given educational workload representative?

A: Compare the resource consumption rates, e.g., direct I/Os per CPU second, buffered I/Os per CPU second, etc. for the script with those for overall average for all programs (TOTAVG). If they are not far off, one can claim that the script is close to that observed in the field.

Q: What workload characteristics should be used in a <simulation> model?

A: It depends upon the purpose of the model. If the model is to analyze a particular new device, say a new terminal concentrator, the programs that use a high percentage of that resource should be included in the workload. However, if two different systems are being compared and a system level workload is needed, the overall average data (TOTAVG) provides the resource demands to be used.

7. Some Observed Workload Characteristics

In this section, we present a summary of some of the interesting observations from our data. The data was collected from VAX/VMS systems at six different universities. There was a variety of types of installations: instructional, research, and administrative. Several installations had substantial use in more than one category. Some also had a significant batch load. The complete database consists of a total of 265,673 image activations which add up to 175 hours of CPU time, and about 8 person-months of active usage time. The total usage time was more than 8 person-months because the time between successive image activations is not included in the total.

1. In a university environment, vendor supplied programs are used very extensively. DIGITAL supplied programs accounted for more than 50% of all resources consumed at these sites. As shown in figure 1, 83 out of every 100 programs run were DIGITAL supplied programs, like LOGINOUT, DELETE, TYPE, etc. Fifty four percent of the total CPU time used was spent on DIGITAL supplied programs. Fifty seven percent of time users spent in running programs was spent on DIGITAL programs. Again, the time between images is not included.

   This clearly shows the power that the vendors have in influencing overall system performance in these environments. For example, a 10% reduction in vendor software's CPU consumption will show up as at least 5% improvement in total system CPU consumption.

2. A large portion of the system resources was used on compilers. The percentage of resources consumed by all compilers (including MACRO translators) is shown in figure 2. As shown there, although COMPILERS constitute only 6% of all activations, they consume a large portion of system resources. They account for about one-fourth of the CPU consumption, one-third of all page faults, and about one-fifth of all page reads.

   The ratio of linker and task builder activations to compiler activations was 1:2.

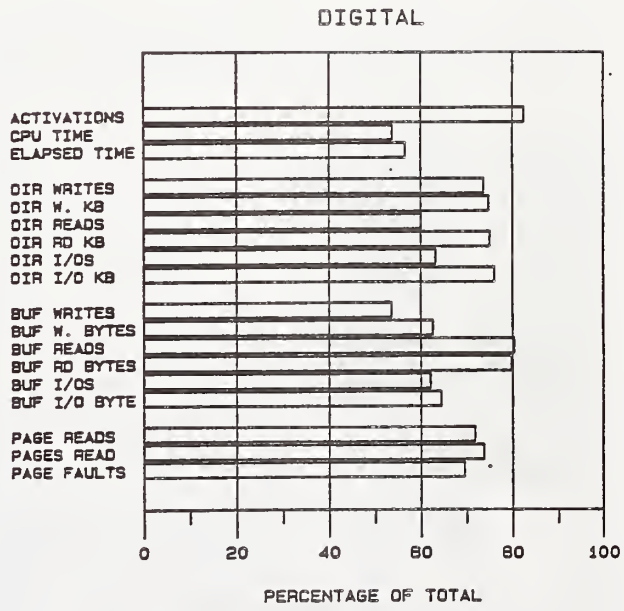


Figure 1. Percent of Resources Consumed by Vendor Supplied Programs.

That is, programs were linked, on the average after two compilations. This is remarkably low considering that the students might just be learning the language syntax.

3. About one-third of user time was spent on editors. The percentage of resources spent on EDITORS is shown in figure 3. This clearly shows the importance of editing. At all installations, a substantial portion of the users' time and system resources were devoted to editing. At installations making use of full screen editors, the resource usage was considerably greater than at those using line oriented editors. Since there appears to be strong trend toward more extensive use of full screen editors, it seems safe to assume that editing will continue to be a major workload ingredient, and probably even increase in importance. The educational institutions should, therefore, give good training to the students in using the editors. This will have significant impact on user productivity.

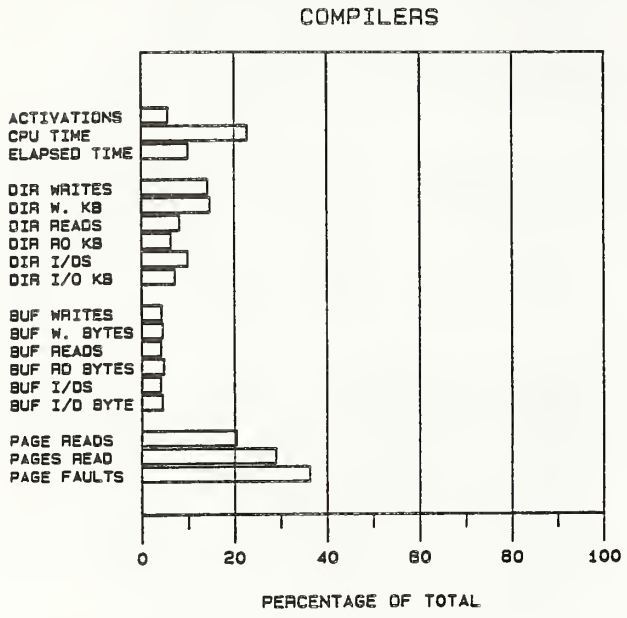


Figure 2. Percent of Resources Consumed by Language Translators

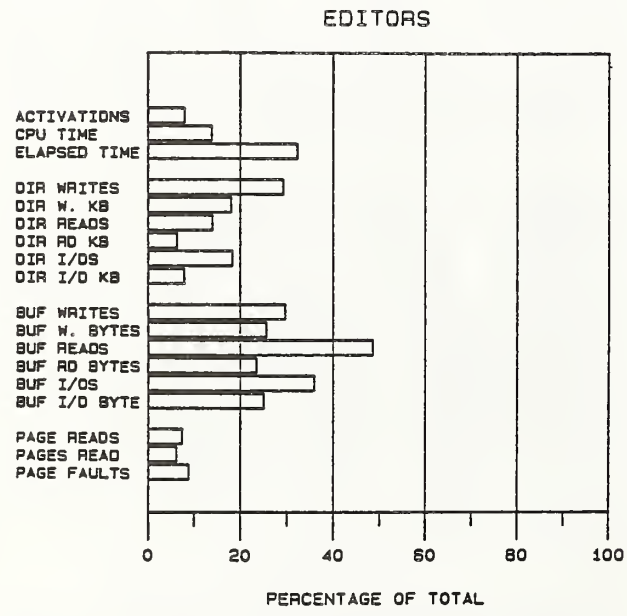


Figure 3. Percent of Resources Consumed by Editors

4. LOGINOUT is the most commonly executed program image. Roughly 18% of all activations are LOGINOUTs. Of these about one-third are due to batch jobs because SUBMIT (the utility to submit a batch job) constituted about 3% of all activations and since each SUBMIT results in two LOGINOUTs. Not all LOGINOUTs are successful, the counts include both successful and unsuccessful activations.

There are many reasons for frequent loginouts in the educational environments. Shortage of terminals requires users to logout. In many industrial environments every user has his own terminal; however, the terminals are dynamically connected to the system via a switch. In such environments, system managers often encourage users to logout and allow other users to use the limited number of lines coming out of the system. Even if the terminals are directly connected, logging out is encouraged to keep a heavily loaded system below the maximum login capacity.

The second highest in terms of activations is DELETE (file deletion utility) constituting about 10% of all activations. Shortage of disk space and purging of the old versions of files seem to be the reasons.

Other commonly activated programs and the percentage of their activations are as follows:

TYPE	9% (To type a file on the terminal)
DIRECTORY	9%
SET	6% (To set process parameters)
SOS	5% (Line oriented text editor)
SUBMIT	3% (To submit batch jobs)
EDT	3% (Screen oriented text editor)
LINK	3% (Linker)

5. The data confirmed the common belief of disk writes to disk reads having a ratio of 1:3. The different I/O ratios are as follows:

For direct I/O:

# of writes	: # of reads	::	1:3
Size of writes	: Size of reads	::	1:2.5
Total write bytes	: Total read bytes	::	1:7.5

And, for buffered I/O:

# of writes	: # of reads	::	2:1
Size of writes	: Size of reads	::	4:1
Total write bytes	: Total read bytes	::	8:1

Although, for other environments, the ratios may be different, the following inequalities should generally hold:

# of disk writes < # of disk reads
Size of disk writes < Size of disk reads
# of terminal reads < # of terminal writes
Size of term. reads < Size of term. writes

The argument is as follows. Disk reads are done automatically by the computer. Disk writes, on the other hand, require either creation of new information or modification of the old information. A human intervention is generally required. We humans read more and write less (be it a book, or a memo, or a data file). The creation of new information is, therefore, slow and always in smaller quantities than use of the old information. This reasoning, which we call "the law of slow creation" explains why disk reads would be generally more frequent than disk writes and why disk reads would be larger in size than disk writes.

In case of terminal reads and writes, a similar argument follows, although in reverse direction. Terminal writes are done automatically by the computer. Terminal reads, on the other hand, require human intervention. Thus, by the law of slow creation, terminal reads would be slower and smaller in size than terminal writes.

6. In VAX/VMS, 96.5% of the page faults are satisfied from the global page caches (free page list and modified page list), only 3.5% require a disk read. Other observations in this regard are:

   - Among the programs analyzed in detail LINK (linker) has the highest probability of finding the page in the global page caches.
   - Compilers, although causing more page faults than the average, also have a better chance of finding the page in the page caches.
   - The inter-site variation in number of page faults as well as page read operations was surprisingly low. The number of page faults depends upon so many system parameters (e.g., size of physical memory, working set size, free page and modified page list size), program parameters (e.g., program size, locality), and user load (number of users sharing the image, number of users on the system), that one would normally expect large variations. However, among the 45 programs analyzed in detail only two had inter-site coefficient of variation greater than 1. This gives the impression that a program's fault rate depends little on the system and can be considered intrinsic to the program.

7. Average session lengths of some of the interactive programs are:

BASIC	7 minutes	
SYSGEN	5 minutes	(To change Sys. Parameters)
SOS	5 minutes	(Text Editor)
EDT	4 minutes	(Text Editor)
TECO	4 minutes	(Text Editor)
CREATE	2 minutes	(File Creation Utility)
DISPLAY	2 minutes	(System Monitoring Utility)

8. Based on 285,673 program activations, the profile of an average program in an educational environment is as shown in table 3.

   This data can be used by analysts in modeling educational environments. Also, the resource consumption rates, (per second, as well as per CPU second) can be used to see the closeness of a synthetic workload (e.g., a remote terminal emulator script) to the measured data. The average and coefficient of variations can be used together to get an approximate distribution of the resource demands at a particular site.

   The coefficients of variation in table 3 are high because data from several sites has been combined together. For individual sites the coefficients of variation are only about half as high. The high coefficients of variation indicate that the resource demand characteristics of programs differ widely and, therefore, synthesis of an "average

Table 3: Characteristics of an Average Program

Data	Average		Coef. of Variation
CPU time (VAX-11/780)	2.19	seconds	40.23
Elapsed time	73.90	seconds	8.59
Number of direct writes	8.20		53.59
Direct write bytes	10.21	Kilo bytes	82.41
Size of direct writes	1.25	Kilo bytes	
Number of direct reads	22.64		25.65
Direct read bytes	49.70	Kilo bytes	21.01
Size of direct reads	2.20	Kilo bytes	
Number of buffered writes	52.84		11.80
Buffered write bytes	978.04	bytes	9.98
Size of buffered writes	18.51	bytes	
Number of buffered reads	38.52		101.02
Buffered read bytes	161.08	bytes	39.74
Size of buffered reads	4.18	bytes	
Number of page read I/Os	5.33		9.56
Number of pages read	34.86		10.23
Size of page reads	6.54	pages/read	
Number of page faults	214.06		26.51

representative program" may be difficult. However, the coefficients are much smaller when programs of a given category only, say compilers, or editors, are considered.

The profiles of an average compiler activation and an average editor activation are as shown in tables 4 and 5 respectively. These profiles are based on 15961 activations of compilers and 23579 activations of editors.

From these tables we see that in educational environments, there are on the average, 14 direct I/Os per CPU second and 27 direct kilo bytes per CPU second. COMPILERS do 2 to 4 times more computation than an average program before issuing a direct I/O. EDITORS do almost the same number of direct I/Os per CPU second as an average program. However, they transfer only one half as many direct I/Os bytes per CPU second as the average.

For buffered I/Os, the overall average in an educational environment is 42 buffered I/Os per CPU second, and 520 buffered bytes per CPU second. COMPILERS do only one-fifth as many buffered I/Os per instruction as an average program. EDITORS do four times as many buffered I/Os per instruction as an average program.

Table 4: Characteristics of an Average Compiler Activation

Data	Average	Coef. of Variation
CPU time (VAX-11/780)	4.81 seconds	3.09
Elapsed time	111.03 seconds	3.47
Number of direct writes	12.17	5.52
Direct write bytes	11.63 Kilo-bytes	3.32
Size of direct writes	0.96 kilo-bytes	
Number of direct reads	21.13	5.61
Direct read bytes	35.18 Kilo-bytes	3.96
Size of direct reads	1.67 kilo-bytes	
Number of buffered writes	28.94	25.96
Buffered write bytes	637.62 bytes	5.62
Size of buffered writes	22.03 bytes	
Number of buffered reads	14.40	1.51
Buffered read bytes	101.64 bytes	2.18
Size of buffered reads	7.06 bytes	
Number of page read I/Os	17.79	4.44
Number of pages read	184.75	4.20
Size of page reads	10.38 pages/read	
Number of page faults	021.95	3.16

Table 5: Characteristics of an Average Editing Session

Data	Average	Coef. of Variation
CPU time (VAX-11/780)	2.57 seconds	3.54
Elapsed time	265.45 seconds	2.34
Number of direct writes	19.74	4.33
Direct write bytes	13.46 Kilo-bytes	3.87
Size of direct writes	0.68 kilo-bytes	
Number of direct reads	37.77	3.73
Direct read bytes	36.93 Kilo-bytes	3.16
Size of direct reads	0.98 kilo-bytes	
Number of buffered writes	199.06	4.30
Buffered write bytes	314.95 bytes	3.04
Size of buffered writes	16.65 bytes	
Number of buffered reads	202.41	4.02
Buffered read bytes	351.98 bytes	2.62
Size of buffered reads	1.74 bytes	
Number of page read I/Os	4.49	1.75
Number of pages read	20.18	2.17
Size of page reads	4.49 pages/read	
Number of page faults	138.55	1.28

8. CONCLUSION

Image accounting provides useful information about workload characteristics. There are many tradeoffs that can be made in designing a image accounting utility. The paper described the design of a VAX/VMS image accounting utility. The use and interpretation of the data was discussed. Many interesting characteristics of workloads in educational environments were presented based on actual observations at six different university sites.

The initial image accounting package was developed by Steve Forgey. The project that collected the data used in this study was carried out by Ted Pollack. Peter Sheh wrote many of the analysis programs. Burton Leathers is currently maintaining the data and the programs. All of the above people were very helpful in this work.

References

[1] Agrawala, A. K., Mohr, J. M., Bryant, R. M., An Approach to the Workload Characterization Problem, Computer, June 1976, pp. 18-32.

[2] Prichard, E. L., and Kolence, K. W., Workload Types and Capacity Management Data Requirements, 3rd Int. Conf. Comput. Capacity Management, Chicago, IL, April 1981, pp. 25-31.

[3] Sreenivasan, K., and Kleinman, A. J., On the Construction of a Representative Synthetic Workload, Comm. of the ACM, March 1974, pp. 127-133.

METHODOLOGY FOR CHARACTERIZING A SCIENTIFIC WORKLOAD

Ingrid Y. Bucher and Joanne L. Martin

Computer Research and Applications Group
Los Alamos National Laboratory
Los Alamos, NM 87545

In the Los Alamos environment of large-scale scientific computing there is always a need for the fastest and largest machine on the market, whether scalar, vector, or parallel processor. Therefore, a determination must be made of the particular architecture most suitable for executing our diverse workload. This determination relies on both an accurate characterization of the current workload and a realistic assessment of future research requirements in computing. Studies are in progress to characterize the present and projected workloads of the major computer users within our facility. This paper describes our general approach to this characterization, which has three stages: (1) identification of major resource consumers among approximately 3000 scientific users; (2) qualitative analysis of the consumer workload to assess the nature and relative importance of current and projected codes, as well as the significance of such peripheral features as I/O and graphics capabilities; and (3) quantitative analysis of the primary codes to determine specific characteristics, such as ratio of vector to scalar operations, average vector length, number of floating point operations, and number of fetches from and stores to noncontiguous memory locations. The results of these analyses will permit a determination of the relative importance to our users of such machine characteristics as scalar, vector, and parallel processing speeds and capabilities before the next major procurement effort.

Key words: Amdahl's Law; benchmarking; computing environment; large-scale scientific computing; parallel processing; scientific workload; vector processing.

1. Introduction

The cost effectiveness of computer modeling experiments as opposed to conducting actual experiments generates a constant demand for increased computing power at the Los Alamos National Laboratory. It is therefore necessary for the Laboratory to be aware of the latest developments in computer architecture and their applicability to its user requirements. Understanding the relationship between new architectures and the applications of Los Alamos scientists relies on the realistic characterization of the Laboratory's computer workload.

Currently, major codes are run on vector processors. It appears likely that the next big machine acquired will be some type of parallel processor with several CPUs. Because a determination of the most suitable computer for the Laboratory will rely heavily on benchmark performance, it is crucial that the benchmark tests are representative of the workload that will be executing on the new machine. Thus, our workload characterization has the ultimate goal of allowing us to design a benchmark set that will model Laboratory computing needs. This set will consist of typical codes or code sections rather than synthetic benchmarks. For the complex architectures to be tested, the sequencing of instructions is as important as the correct instruction mix.

Our approach to the characterization of our workload may be divided into three stages:

(1) identification of major resource consumers,

(2) qualitative analysis of the consumer workload, and

(3) quantitative analysis of primary codes.

Although each of these stages involves information that is dynamic with respect to time, analysis of the combined results will permit us to determine the relative importance of various machine characteristics. Hence, we will have a framework for designing and weighting an appropriate benchmark set.

2. Environment

Currently, the computers in the Integrated Computing Network at Los Alamos National Laboratory support the computing needs of approximately 3000 users in an environment that is dominated by large-scale scientific computation. We define large-scale scientific computations as those problems that saturate all resources of any available computer [1]. The dominant physical problems at the Laboratory include radiation transport and diffusion processes, hydrodynamics, and the motion of charged particles in electromagnetic fields. The numerical methods employed in the solution of these problems include finite difference methods to solve partial differential equations, particle-in-cell (PIC) techniques, and Monte Carlo methods. Most of these calculations require a machine that provides 64-bit accuracy. Further, Fortran is used almost exclusively in the major Los Alamos codes.

Another important feature of the Los Alamos computing environment is the requirement that most programs be executable in not more than eight hours when consuming all resources of a machine. In the batch system this allows for overnight turnaround, which is acceptable to most users. However, if the codes were to require more computer time, the resulting delay in turnaround would have serious implications on user productivity. Currently, jobs are being tailored to sizes that will meet this criterion, sometimes at the expense of accuracy.

Because the majority of the large-scale scientific computation is done on the four Cray-1s within the network, their batch workload is the focus of this study. A computing system associated with this type of workload may be characterized by

- hardware that is very fast in performing floating point operations,
- large memory capacities (several million 64-bit words),
- large mass storage facilities,
- interactive graphics, and
- state-of-the-art technology [1].

The Cray-1 has hardware that is capable of performing up to 160 million floating point operations per second (MFLOPS). Further, the memory capacities on these machines range from one million to four million words. Large mass storage facilities and interactive graphics are available through the network. Finally, the Cray-1 is a vector processor; that is, it has hardwired instructions that operate on n-tuples of numbers.

Thus, Los Alamos has a large-scale scientific workload and a corresponding computing system. In fact, starting at the birth of electronic computation shortly after World War II, Los Alamos has operated one of the world's largest computing facilities and continues to be at the forefront of scientific computing [2]. Nevertheless, the nature of the computations currently being performed, as well as the goals for future computing capabilities, indicate the need for larger and faster machines. Here, "faster" refers to the entire computer system and not just to the CPU. The I/O requirements of the codes at the Laboratory are such that if the CPU were to gain speed without a corresponding gain in the speed of I/O operations, many CPU bound codes would become I/O bound.

The continual need for newer and faster machines implies the need for the regular monitoring of Laboratory computing workload in order to assess advances in computer architecture as they relate to this workload.

3. MIMD Architecture

Recent advances in technology indicate that the next major computer procurement might involve some type of MIMD architecture. According to Flynn's taxonomy, MIMD implies multiple-instruction and multiple-data streams, where an instruction stream is the sequence of instructions performed by the machine and a data stream is the sequence of data manipulated by the instruction stream [3]. In particular, in MIMD architectures, several central processors operate in parallel in an asynchronous manner; that is, they may execute different instruction streams and typically share access to a common memory. The performance of such machines could be modeled by the consideration of three processing speeds: sequential, synchronous parallel, and asynchronous parallel.

- Sequential speed characterizes the rate at which a machine can process code that must be processed in a sequential form, either for reasons of logic or because it is too costly to vectorize or parallelize it.

- Synchronous speed characterizes the rate at which a machine can process code that lends itself to either vectorization or synchronous parallel processing with small granularity. Vector processing permits a single operation to be performed on many elements of a vector in a pipelined fashion. Synchronous parallel processing is the processing of codes in which single instructions are allowed to operate simultaneously on multiple data.

- Asynchronous parallel speed characterizes the rate at which a machine can process code that is not suited to either vectorization or synchronous parallel processing but that can be parallel processed asynchronously in

large chunks. That is, sections of the code can be processed independently from one another with respect to time, allowing multiple instructions to execute simultaneously on multiple data streams. Examples of this category are Monte Carlo and PIC calculations.

These three factors should be considered when judging the applicability of a parallel processor. The determination of how these factors should be weighted relies heavily on understanding the degree to which Laboratory workload can be divided into the three categories.

4. Amdahl's Law and Its Extension

A major principle in large-scale scientific computing is Amdahl's Law [4]. It states that when a computer has two distinct modes of operation, one high-speed and one low-speed, the overall operation is dominated by the low-speed mode unless the fraction of work processed in the low-speed mode can be virtually eliminated. By extension, this remains true in computers with three modes of operation.

This principle increases the criticality of understanding the extent to which major Laboratory codes fit into the three categories of processing speeds. For example, even a machine that has infinitely fast synchronous and asynchronous parallel speeds, but slow sequential speed, would be ineffective on a workload dominated by sequential code.

The effective speed of a computer with all three processing speeds may be characterized in the following fashion. Let f(seq) be the fraction of the workload that can be processed in sequential mode only, f(syn) the fraction that can be either vectorized or processed by synchronous parallelism, and f(asyn) the fraction that can be processed by asynchronous parallelism; then the time required to run this workload is proportional to

$$t \sim \frac{f(seq)}{S(seq)} + \frac{f(syn)}{S(syn)} + \frac{f(asyn)}{S(asyn)} \quad \frac{1}{S(eff)}$$

where S(seq), S(syn), and S(asyn) represent sequential, synchronous, and asynchronous speeds, respectively, and S(eff) is the workload-dependent effective speed of the machine.

5. Methods for Characterization of Workload

Our approach to characterizing the workload at Los Alamos has had three stages.

In the first stage we used accounting records to identify the major resource consumers. For the identification we defined several base parameters and considered these parameters over a 16-month study period. In particular, we defined major users as those individuals who consistently used in excess of $2000 of Cray time per month in blocks of usage that exceeded three system resource units, where a system resource unit represents the basic unit of charge for computer use, including CPU, I/O, memory, and system times. This allowed us to determine the groups within the Laboratory on which to focus our attention and guided us to our second stage.

In the second stage we met with project leaders, in the groups which we had identified, to gain a qualitative understanding of broad code characteristics, general composition of current workload, the importance of peripheral features, and directions and goals for the future. We also obtained access to the major codes so that we could conduct some measurement tests on them.

In the third stage, we examined the codes to make quantitative assessments of the present workload. In particular, we measured CPU time, the number of floating point operations, current MFLOPS rates, percentage of vectorization, and average vector length. These measurements were accomplished by taking advantage of a Cray compiler option that compiles each floating point operation as a return jump to an externally supplied subroutine. The external subroutines we then used executed the arithmetic that otherwise would have been generated by the compiler, and then counted the floating point operations, sorting them by type of arithmetic operation and by whether they were vector or scalar functions. Intrinsic functions were counted separately by modifying existing library functions to include counters, and scale factors were applied to these because they are iterative. SIN and COS were each associated with 20 floating point operations, whereas EXP, ALOG, and SQRT were counted as taking 16 floating point operations each.

Of particular significance to the characterization of workload for future machines are the measurements of percentage of vectorization and average vector length. Recalling Amdahl's Law, we re-emphasize that a new machine with a vector unit that is considerably faster than that in current machines but with a scalar unit demonstrating little or no improvement will be ineffective in terms of total speed-up unless the codes can be vectorized at a very high percentage. Similarly, the average vector length is significant in that some machines will attain maximal speeds only on extremely long vectors. Thus, unless our codes have vectors that are long enough to take advantage of this type of architecture, it would not be the architecture most suitable for the Laboratory workload.

It should be pointed out that floating point operations, the traditional measure for the workload of supercomputers, constitute only part of the total work. Among others, branch and subroutine linkage instructions can influence performance significantly. In addition, the sequencing of instructions is of great importance.

6. Results

We obtained qualitative results in the second stage of our study and quantitative results in the third stage. These are summarized in this section.

6.1 Qualitative

Through discussions with the group and project leaders identified from accounting records, we were able to choose five large codes on which to focus our attention. The first three of these codes (Codes A-C) consume approximately 50% of the Laboratory's computer resources. The other two codes (Codes D and E) are PIC codes in which considerable effort at vectorization and optimization has been invested.

Codes A-C are extremely large. Typically, they have in excess of 100000 lines of Fortran, are overlaid codes with 10 to 30 overlays, and employ dynamic memory allocation techniques because of their enormous memory requirements. These features imply that a very large and very fast memory will be needed in future machines because memory access might be replacing CPU time as a major bottleneck. Further, there is a need for the ability to do rapid fetches and stores to memory for noncontiguous vectors. It is estimated that this type of scatter-gather operation consumes as much as 30% of code run time on these large codes. Having a hardwired scatter-gather operation would therefore be a tremendous advantage.

Codes D and E are smaller codes. In particular, each has approximately 30000 lines of Fortran. This smaller size is one of the factors that makes concentrated attempts at optimization and vectorization reasonable. Code E has a further advantage in that 90% of its CPU time is spent in a very small fraction of the total code. Thus, attention can be focused on those small segments with maximal benefits. Finally, although Code D is primarily written in Fortran, it does make use of Cray Assembly Language (CAL) subroutines to gain speed-up wherever possible. Code E has been extensively optimized and major portions have been written in CAL.

Dimensionality of the codes is another factor that influences the need for larger and faster machines. The transition from scalar to vector processing machines, with corresponding increase in memory capabilities, yielded sufficient speed-up to allow designers to work with two-dimensional rather than one-dimensional codes. By a one-dimensional code we mean a code that has one independent variable, as is the case with a geometry that demonstrates spherical symmetry. A two-dimensional code employs two independent variables as, for example, in the case of cylindrical symmetry. It is estimated that a speed-up factor of 10-100 in overall processing ability is required for the consideration of three-dimensional problems to be practical.

This change in emphasis from one- to two-dimensional codes is reflected in the codes we measured. Code A is an old one-dimensional code that previously ran on the CDC 7600 and has been converted to run on the Cray. Because it would require enormous effort to restructure and rewrite this code, little emphasis has been placed on its vectorization. Codes B and C are both two-dimensional codes that have been designed with vector architecture in mind and should reflect this consideration in the degree to which they vectorize.

It is significant that the decision was made not to invest the man-hours in vectorizing Code A, but rather to accept the level of vectorization automatically obtainable through the optimizing compiler. Similarly, it can be assumed that the same decision will be made regarding the conversion of current codes to a new architecture. Thus, there should be some automatic optimization that will allow for speed-up of existing codes in their current form.

Finally, we note that there is a relationship between workload and architecture that allows each to influence the other. In particular, Monte Carlo calculations cannot be vectorized effectively and are therefore extremely time-consuming and costly. Hence, they are not the method of choice in code design and comprise only about 5% of the current workload. If the next machine is a parallel processor, it is possible that the fraction of Monte Carlo runs would be increased because this architecture is considered amenable to Monte Carlo calculations,

6.2 Quantitative

We measured the number of floating point operations, current CPU times, percentage of vectorization, average vector length, and current MFLOP rates on Codes A through E. Codes A-C were each run for different problem setups in an effort to assess code performance on a variety of typical problems. Because different setups require access to different portions of the code, it is hoped that the results obtained through the combination of problems are representative of total code use.

With Code A, one input problem used only the hydrodynamics portion of the code, whereas the second problem included both hydrodynamics and neutronics transport cycles. In a third input problem, which also used both hydrodynamics and neutronics, we adjusted parameters to demonstrate the relationship between average vector length and MFLOP rate.

With Code B, the first input problem is mainly a hydrodynamics problem, and the second problem accesses the Monte Carlo portions of the code also. Finally, the first input problem used with Code C is a one-dimensional problem and the second is a two-dimensional problem. The results are summarized in the following table. Note that the average vector length has a maximum of 64

MEASURED CHARACTERISTICS OF CODES A-E

Code	Millions of Floating Point Operations	CPU Time (in (seconds)	MFLOPS	Average Vector Length	Percentage Vector	Percentage Scalar
Code A						
Problem 1	219.6	69.6	3.2	59	1.8	98.2
Problem 2	137.9	38.4	3.6	11	26.5	73.5
Problem 3a	5.5	1.9	2.9	8	29.1	70.9
Problem 3b	9.4	2.8	3.4	13	28.7	71.3
Problem 3c	16.9	4.5	3.8	21	28.4	71.6
Code B						
Problem 1	4432.8	232.6	19.1	32	98.6	1.4
Problem 2	8595.6	4314.5	2.0	31	10.7	89.3
Code C						
Problem 1	4215.2	902.2	4.7	26	36.3	63.7
Problem 2	1121.9	225.3	5.0	53	49.0	51.0
Code D	136.3	69.8	2.0	20	77.7	22.3
Code E	2427.1	85.5	28.4	63	69.5	30.5

because all measurements were made on the Cray. That is, when handling vectors with lengths that exceed 64, the Cray divides the total length into units of 64 plus a remainder and then processes each separately.

7. Summary

So far, we have learned several important lessons from this workload characterization. They are summarized as follows:

- This method of measuring codes and discussing code development with the designers can produce valuable information about Laboratory workload and will aid us greatly in modifying and extending our benchmark set. This will permit us to be assured of realistic test procedures for new architectures, relative to the needs at Los Alamos during the next decade. We will be able to evaluate computers with new capabilities reaching beyond those of the present generation. Special emphasis should be put on these capabilities.

- Other code characteristics are significant and should be measured as this study continues; for example, the number of noncontiguous memory accesses, the amount of I/O and memory used relative to the amount of CPU time, the ratio of memory accesses to the number of floating point operations, and the percentage of time spent in library routines.

- Getting actual codes to run at advertised levels of machine performance requires an enormous investment of time, effort, and funding. Therefore, in general, benchmarks should not be tight, highly optimized programs that demonstrate only the peak performance of the machine under test. Overall, actual codes are not highly optimized even after considerable effort has gone into their optimization.

- This effort at workload characterization can be beneficial to code developers as well as to us as we modify and prepare our benchmark set. That is, as we obtain measurements on the codes, we are able to feed that information back to the designers who, in turn, are able to modify their codes to obtain better performance. For example, on the Cray-1 vectors of length 64 produce optimal results and short vectors (less than length 5) yield lowest performance statistics. If the designer is able to see measurements that indicate that performance is suffering as a result of a small average vector length, then in some cases it is possible to modify loop counts in a manner that will increase the average vector length, thus producing higher MFLOP rates.

We would like to thank a number of people for their cooperation in running the codes we measured. In particular, we are grateful to Tony Warnock, John Romero, Art Dana, Bruce Wienke, Molly Mahaffey, Jim Painter, and Dave Forslund for all their help. We also want to thank Timothy Rudy of the Lawrence Livermore National Laboratory for many of the subroutines used in this study.

References

[1] B. L. Buzbee, "Large-Scale Scientific Computation at the Los Alamos Scientific Laboratory," Los Alamos National Laboratory report LA-7258-MS (June 1978).

[2] Los Alamos National Laboratory, "Institutional Plan, Long-Range Projections FY 1982-FY 1987," Los Alamos National Laboratory report LALP-82-10 (January 1982).

[3] M. Flynn, "Some Computer Organizations and Their Effectiveness," IEEE Trans. on Computers, C-21 No. 9, September 1972, pp. 948-960.

[4] J. Worlton, "A Philosophy of Supercomputers," Los Alamos National Laboratory report LA-8849-MS (June 1981).

CASE HISTORY: BUSINESS DRIVER METHODOLOGY IN A MANUFACTURING LOGISTICS APPLICATION

F. J. Machung

Department 943
Building 001-1
Poughkeepsie, NY

The business driver or key volume indicator methodology offers a means of forecasting data processing computer workload as a function of user planning indicators. This paper is a case history of one model developed in the manufacturing logistics area. The author assumes that the reader is familiar with the fundamental notions of correlation and linear regression.

1. System Environment

IBM Poughkeepsie Manufacturing mission covers a range of products from individual components all the way to large systems such as 3033 and 3081. The data processing services support the Manufacturing activities of product build, assembly, test and release. The data processing, services are grouped into three principal clusters of workload.

The cluster boundaries were intially established due to varying response and availability objectives. The Interactive support cluster supports the development and production of program development and end user developed interactive data base systems. The Manufacturing control cluster supports the process related functions for direct manufacturing assembly, build and test. The Administrative cluster supports the materials logistic, personnel and administrative functions for three sites: Kingston, Poughkeepsie and Brooklyn.

The Administrative cluster is composed of (2) 3033 and (1) 168 processors in a loosely coupled, JES-3 controlled environment. A 155 Telecommunication Support Processor System (TSPS) routes all terminal traffic from Kingston, Brooklyn and Poughkeepsie to the common Manufacturing Information System (CMIS), the principal materials logistics applications. The CMIS applications are distributed across the three processors. In addition, the Development and Production Records System, a large IMS service offering new product manufacturing logistics applications and miscellaneous personnel applications, reside on a single IMS processor.

Teleprocessing networks exist between the three clusters as well as to distributed satellite nodes.

In exces of 400 terminals are distributed across the CMIS and IMS services. The daily average transaction volumes for CMIS and IMS services exceed 250,000.

The on-line data bases, which are dispersed over 40 spindles, represent approximately 35% of 3330-11 spindles in the cluster.

Six major on-line and five major batch functional services committed by Service Level Agreements comprise most of the activity within the cluster. Generally, the minimum availability agreements are 95% for individual service components and 92% for interconnected service components.

2. Applications Environment

The applications within the Administrative Cluster support two major categories of activity:

1. Materials Logistics
2. Administration/Personnel

The materials logistics functions start with order receipt from IBM's Administrative Application Systems (AAS) and Consolidated Customer Order Processing (CCOP) systems. The orders are processed and disseminated through the Common Manufacturing Information System (CMIS). CMIS is concerned with the materials required to produce the product, support the planning, ordering and stocking or parts and sub-

assemblies, and controlling their distribution to the Manufacturing floor. Basically CMIS begins with the introduction of product schedules and ends with the shipment of product from the plant. The primary subsystems within materials logistics are:

1. Development and Production Records System (DPRS) which is the basic engineering and manufacturing records system. The records system supports product structure definition, process description, records progression and related engineering manufacturing services of a product to be procured or manufactured.

2. Operations Planning (OPS) which explodes the total product schedule into part and assembly orders.

3. Procurement (PROC) which provides vendor information to both purchasing and accounts payable.

4. Manufacturing Activity Release and Control (MARC) which handles the release and control of internal production orders, the determination of components required for these orders, and the allocation of available components.

5. Final Assembly Logistics Control (FALC) which supports the final assembly of products to customer order, providing customized assembly parts lists and instructions for each machine.

6. Warehousing (Whse) which controls the movement of parts, materials and supplies from receipt to disbursement or shipped. It maintains an inventory of parts in stock, in transit or assigned to vendors.

7. Purchase and billing which is a set of systems to control Accounts Payable to vendors, the purchase of components and traffic and transportation billing to customers.

Administrative/Personnel Systems perform the accounting, measuring, costing and personnel functions necessary to control plant operations.

The key volume indicator approach described in this paper focuses on a subset of the material logistics applications.

3. Methodology

The objective of the business driver approach is to provide a gross predictor of DP computer workload enabling the user of the DP resource to forecast in known natural terms. The operating premise is to utilize the highest level and the most accurate planning drivers to forecast this DP workload. This method enables the capacity planner to minimize the number of drivers and generate a macro view of the workload. Figure 1 reflects the potential basic key volume indicators for the Manufacturing site.

The following steps outline the procedure used to generate computer workload forecasts:

1. Identify application sets.

2. Classify workload within application.

3. Survey users and applications development.

4. Collect available historical data on potential user drivers.

5. Utilize regression analysis programs to determine potential indicator validity.

6. Determine which applications are interrelated to determine if macro modeling is possible.

7. Secure user forecasts and track monthly values.

Identify Application Sets

Utilizing the billing/utilization system individual computer jobs are categorized into application sets. Various resource and identifying parameters are collected for the reporting period. The task control block and system control block CPU seconds are the two key parameters used to describe the workload.

Classify Workload Within Application Groups

The classification of workload within application groups enables the capacity planner to describe the workload as a function of shift, e.g. prime, second, third or weekend as well as user functional relationships. This step enables the planner to describe those sets which are the more significant consumers of resources.

Survey Users and Applications Development

The critical element in the driver method required that the capacity planner survey the owner-user of the applications. The components of this survey are the following:

a. Briefly describe the application functions.

b. Itemize all key inputs and outputs.

c. Determine which of the primitive application functions can serve as drivers of workload.

d. Determine whether the indicator is predictable in terms of user functional workload.

e. Determine if the indicator is readily trackable in terms of forecasts and actuals.

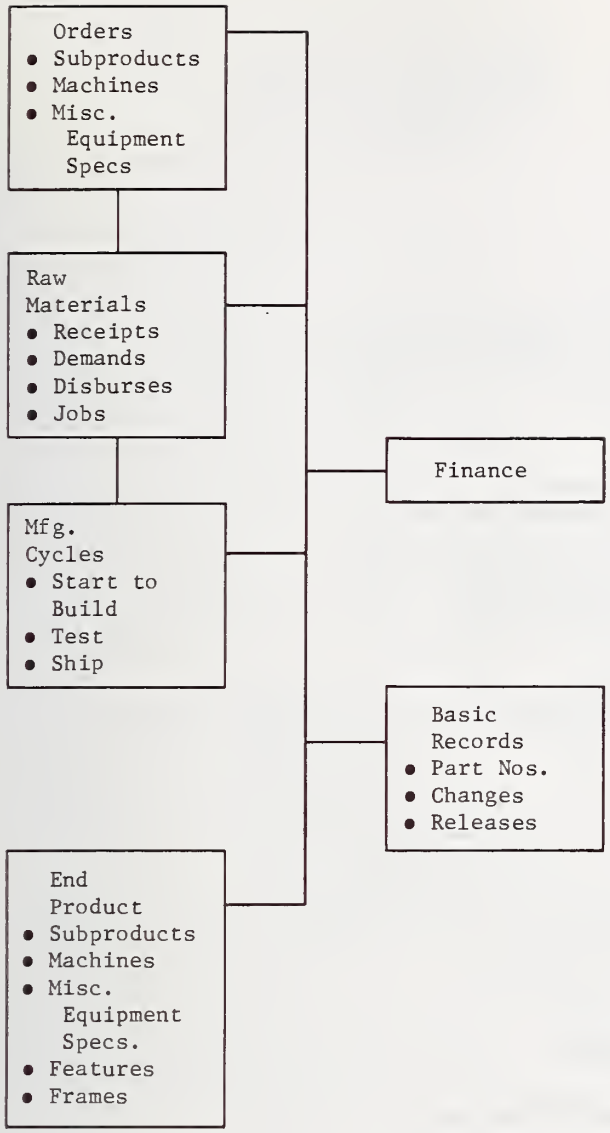


Figure 1. Key Volume Indicators Macro View (Potential Drivers)

f. Determine whether the prime inidcator is more readily trackable in terms of some more readily available derivatives.

The underlying purpose of the survey is to determine the availablility of "natural" user drivers and ensure an appropriate mechanism can be installed to ensure tracking. In those cases where drivers are not readily identifiable, programs may be required to extract the data from the application system. Figure 2 describes functional flow for one application group and its associated potential drivers.

Collect User Drivers

This part of the process requires the collection and classification of the drivers into an appropriate data base. This will enable future manipulation with the DP utilization data.

Regression

Once a sufficient data base of potential drivers and resource parameters are amassed, the search for predictable relationships begin. The regression programs provide for seeking "good mathematical fits" and coincidentally offer validation of the results. Our regression analysis is conducted on a local APL Interactive Service.

Macro Modeling

After several iterations of regression, some overall indicators may be apparent which provide for summarization of lower model groups. This will minimize the number of drivers which must be reported and tracked without comprising the prediction capability of the DP workload. Figure 3 depicts a classification schematic for one macro model.

User Forecast

Once the appropriate mathematical relationships of drivers to CPU resource are identified then a procedure is established to ensure timely forecasts. In addition feedback of actuals is required. The feedback process reports the actual driver variances as well as the CPU resource consumed by model group.

Follow-on Activities

On a minimum of a semi-annual basis new total forecasts are collected for the site operating plan. In addition, selected sensitivity analyses will be done to determine the adequacy of service levels and the possible performance characteristics of the workload. At present the performance prediction of service factors is not a direct part of the driver process. During these operating plan cycles, the user projected workload is converted into total system workload for all application sets running within the cluster. Based on derived or empirical capacity statements versus the projected workload, the required CPU resource is determined (Figure 4 highlights the overall capacity planning process) while Figure 5 highlights the tracking system.

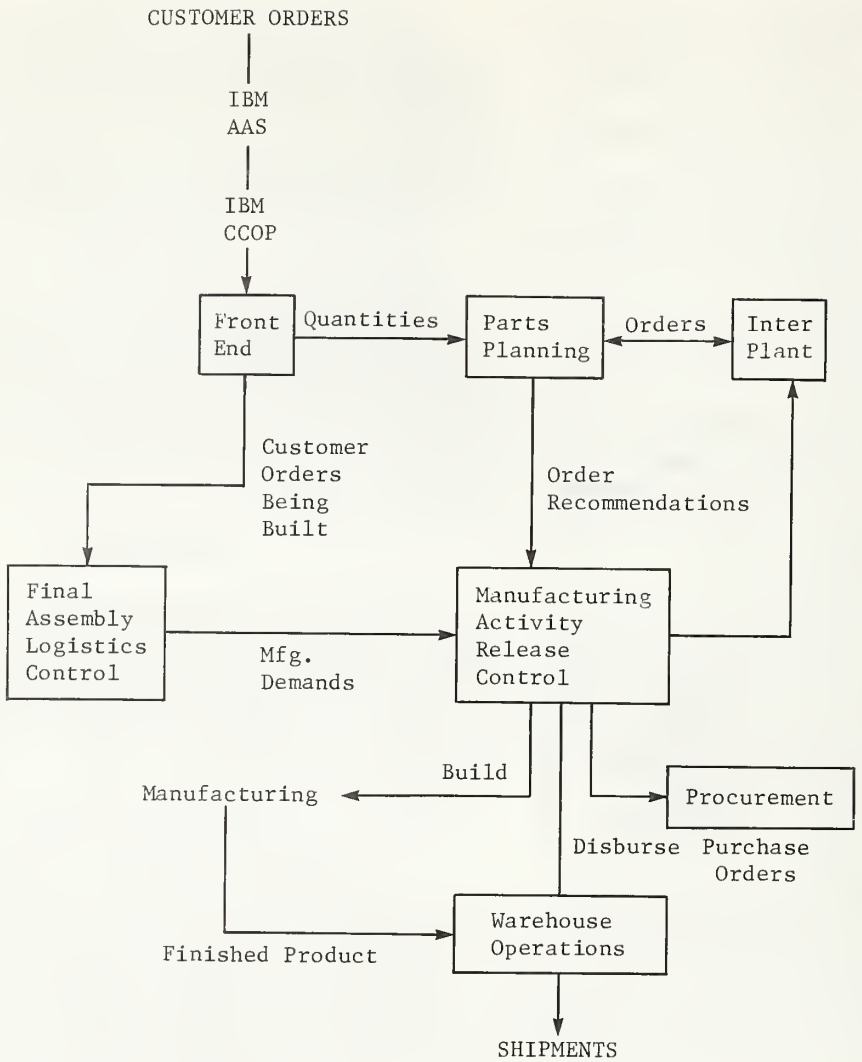


Figure 2. Applications Overview with General Drivers

Macro Model:	All Logistics			
Model Groups:	On-Line Logistics	Support Logistics-A	Basic Logistics	Support Logistics-B
Application Sets: (Acronyms)	5	34	10	15
Jobs/Day:	28	101	157	30
Approximate % Total Workload:	6	7	25	2

Note: The acronym designation is a standard established in the billing/utilization system
(e.g. MEAS001) is the job number and MEAS is the project/workscope)

Figure 3. Classification Schematic

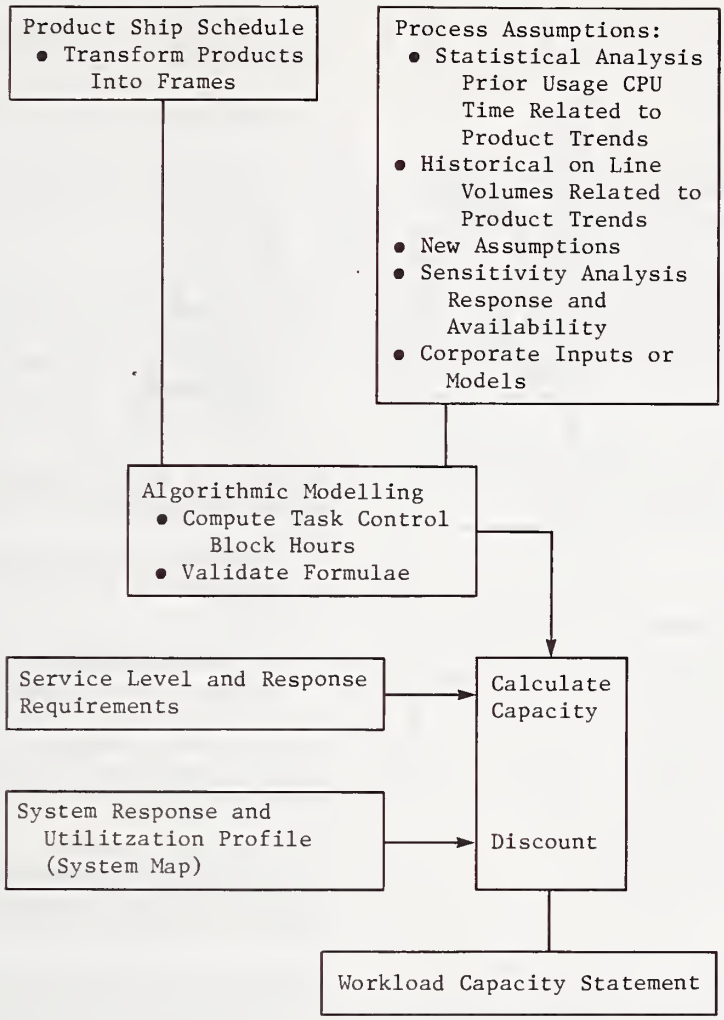


Figure 4. Capacity Workload Forecasting Example

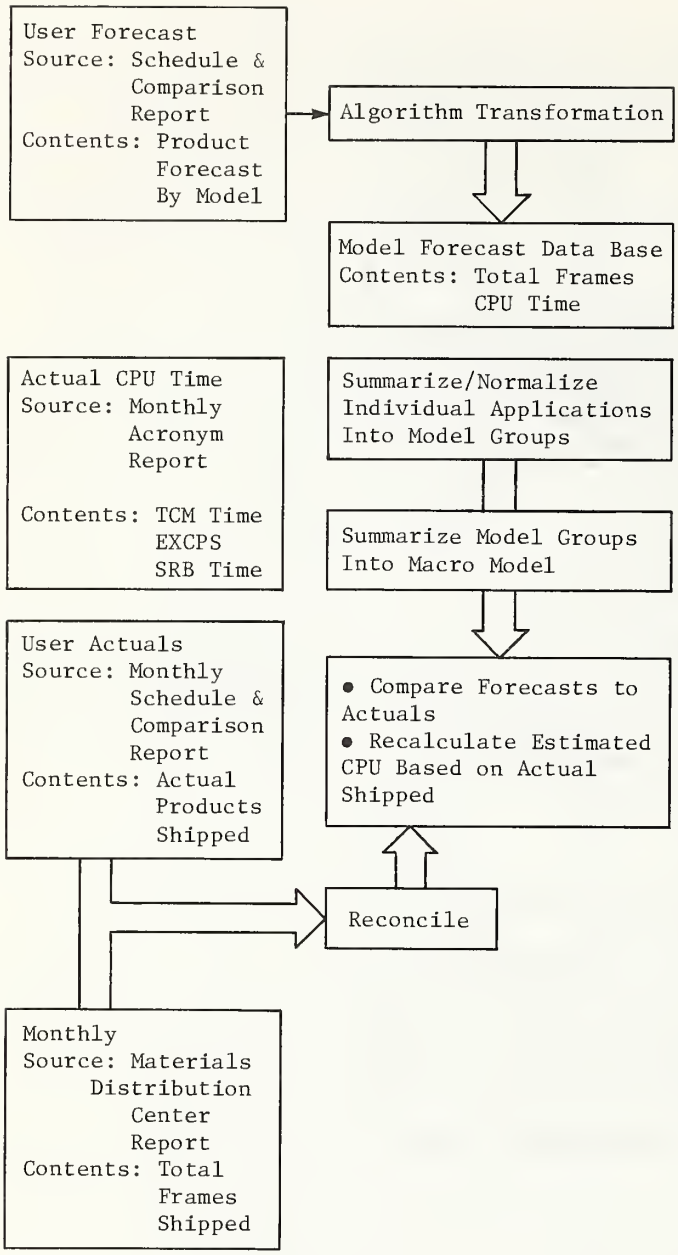


Figure 5. Tracking System Overview

The Model

The algorithm which predicts the workload for the "all logistics" macro mdoel is derived from the monthly product ship schedule. The product ship schedule is tranformed into "frames" by machine type. The frames shipped is summarized across th total product line. Figure 6 depicts various formulae and corresponding "data fits" of estimate to actuals.

Operating Experiences

After approximately two years in accumulating data and validating the model results, the following experiences are offered:

1. The regression approach requires a sufficiently large data base to offer sensitivity.

2. Many plausible relationships can be readily established but they may be invalid because:

   a. The data contains little variation.

   b. The historical driver may not be forecastable or it would be costly to establish a forecasting machanism.

   c. More definitive drivers have been established which provide more accurate historical fits but are unable to be utilized for forecasting purposes since the schedules are not provided at these lower levels.

3. Some of the indicators utilized may be technology dependent which require:

   a. A careful understanding of the historical data before future predictions are possible, or

   b. A transformation - approximation mechanism for the future product.

4. The horizon of predictability is sometimes limited resulting in the typical "planners droop".

5. The regression method is not intended to provide detailed understanding of the application jobstream characteristics. It offers a collective view of workload.

The key volume indicator or business driver approach offers a quick, flexible and realistic method of forecasting DP workload. The key elements required to sustain this method are:

1. A classification schematic or standard of the utilization data offering easy manipulation for the required regression analyses.

2. Data bases containing the utilization data and drivers.

3. A corresponding tracking system.

The key volume indicator method offers the following advantages:

1. User Deterministic: The user of the data processing service is required to estimate the business parameters which are eventually reflected in the capacity requirements and operating service level agreements.

2. Synthetic Approach: The method offers the ability to mathematically relate general drivers to overall data processing workload.

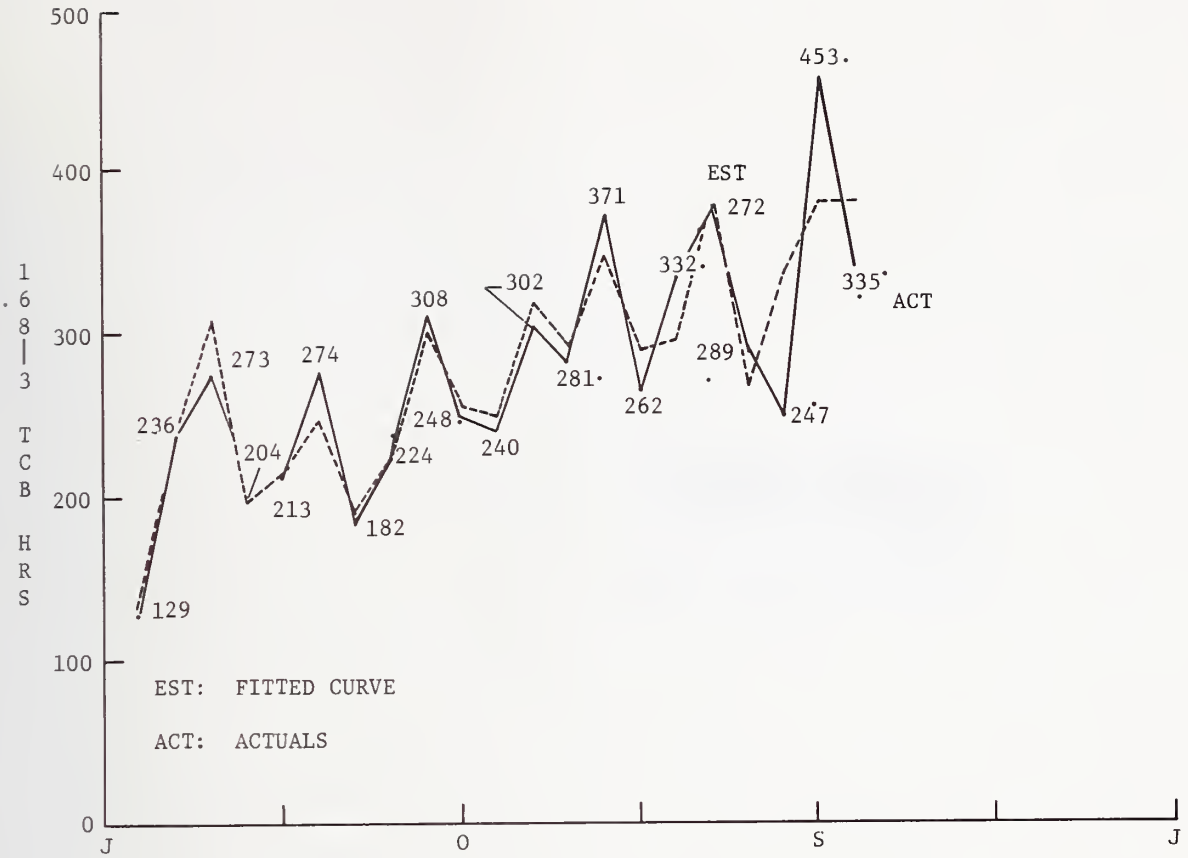


Figure 6. All Logistics

3. Heuristic: The driver approach provides for easy data manipulation and analysis once categorization, classification and collection structures are established and maintained.

4. Usable: The method offers a practical and non-labor intensive means of describing workload adaptable to a changing business environment.

5. Educational: The method requires that the data processing operations function and the users to understand the general business drivers and their relationship to data processing workload.

The disadvantages of the key volume indicator method are:

1. Generalization: The method is not intended to be used for a detailed analysis of individual jobstream or on-line performance analyses. To understand the functional operating characteristics of each application and jobstream, detailed models are required.

2. System Control Product Sensitive: The data base of utilization requires adjustments for operating system releases, new application functions or application software changes.

Due to some of the aforementioned operating experiences, Mid Hudson Valley Manufacturing Computer Center utilizes three approaches in estimating workload. These three approaches are:

1. The business driver or key volume indicator method.

2. Trendline method where little variation exists in the data or a historical trend is evident and satisfactorily projects workload.

3. Specific detailed application models done in conjunction with Corporate groups where common application development is undertaken and the jobstream or response characteristics must be individually assessed.

However, the business driver approach offers the DP Site the capability to understand the true relationship of the user driven workload and its translation into computer workload. It is readily adaptable to a changing business environment.

References

Sarna, David E.Y., Forecasting Computer Resource Utilization Using Key Volume Indicators, Price Waterhouse & Company, New York, New York; National Computer Conference 1979.

CPEUG82

"Improving Organizational Productivity"

Modeling Techniques

SESSION OVERVIEW

MODELING TECHNIQUES

K.C. Sevcik

University of Toronto
Toronto, Canada M5S1A1

Computer system modeling has come to be recognized as an effective technique for understanding and predicting computer system performance. Models based on only a few dozen parameters are capable of capturing the essential system aspects that most influence performance. Solution of these models can be based either on mathematical analysis or on simulation.

The purpose of this session will be to familiarize the attendees with some recent studies in which modeling techniques have been developed and investigated. Also, one paper will provide an overview of the components now available in typical software packages for analysis of models of computer systems.

Design of A Software Tool For Evaluation of Computer and Communication Systems

Ashok K. Agrawala
Satish K. Tripathi
Ashok K. Thareja

System Design and Analysis Group
Department of Computer Science
University of Maryland

The Systems Design and Analysis group at the University of Maryland is in the process of designing and implementing a software package aimed at providing the necessary tools for the performance studies of Computer and Communication Systems. Implemented in a user friendly environment, this package will provide the analyst easy access to the state-of-the-art performance modeling techniques including analytic, simulation, hybrid and approximation techniques. The design concepts of the package are presented in this paper.

Key words: Queuing Models; simulation; approximation techniques; systems performance; software package.

1. Introduction

As the complexity of computer systems is increasing, it is becoming more and more important that adequate tools be available to analyze and evaluate their performance. Queueing network models have been found to be very robust models for computer systems. The state-of-the-art tools incorporating queueing network models suffer from one of the several inadequacies: the solution techniques used may be outdated, software is specialized to some specific application, the software provides a poor user interface and is very difficult to be used by somebody other than the designers and implementors, the tool implements some specific solution techniques, or the tool is in a private domain.

The Systems Design and Analysis Group at the University of Maryland has undertaken a project to design and implement a comprehensive software package. The goal is to implement a tool that incorporates the whole range of queueing network modeling techniques, i.e., analytic, simulation, hybrid and approximation, under a common friendly user interface and is modularly expandable. The tool is to serve the needs of the users with varying degrees of knowledge of the performance evaluation techniques. This paper summarizes important design aspects of this package.

Sections 2, 3 and 4 provide a quick glance at the user and application environment assumed for package, the possible applications and some related design goals. These sections should be very helpful in assessing the applicability of the package to an application. Section 5 defines the class of systems to be analyzed and the modeling primitives used in the implementation of the package. This section is essential in understanding of the design details of the package.

An example of a realistic system is introduced in Section 5, where it is used for demonstrating the set of assumptions required to describe a system in terms of the modeling primitives of the package. The same example has been carried through the other sections of the paper to illustrate different aspects of the package. The example should be useful in understanding the way a realistic modeling problem may be formulated and analyzed using the package.

A model definition language, MDL, has been defined to communicate the model definitions across various cross-sections of the information flow path. A brief introduction to MDL is presented in Section 5. Sections 6, 7, and 8 describe the design details of the main modules implementing the package. Only the main concepts have been described in this paper.

2. Operational Environment for the Package

2.1 User Environment

Two basic skill levels have been assumed for the users of the package:

1) Performance Analysts: Users, who are familiar with the primitives used in implementation of the package to define queueing network models and are versed with the analytic techniques are assumed to possess skill level of a Performance Analyst. Such users would be able to interact with the package through an interface where they have an access to the full capabilities of the package, in terms of the range of systems that may be modeled and the type of analysis that may be performed. An intermediate level knowledge of queueing theory and familiarity with the implementational details of package would enhance the skills of a Performance Analyst.

2) Application Analysts: Not all users interested in solving problems of systems design, configuration planning, systems management or systems evaluations can be expected to possess the skills of a Performance Analyst. The users who are involved in systems with specific applications or in a specific aspect of the design, analysis, or management can be considered Application Analysts for our purpose. An application analyst will interact with the package through an application oriented interface that tailors the package to the requirements of the specific application. Thus a user with minimal knowledge of the inner workings of the package and queueing theory techniques should be able to use the package system for solving a specific problem.

An application analyst's view of the package would be more of a problem solver than that of a queueing network models analyzer. An application analyst would be able to interact with the package in terms of the terminologies of the application. The package would be able to retrieve predefined models or create models from predefined schema on the type of the system or application to be analyzed. The user then supplies information to define the parameters of the models and to carry out the other phases of the analysis.

2.2 Configuration Planning Using the Package

Consider the problem of planning the configuration of a computer system. Typically, the workload to be handled, the choice of components available, performance and cost constraints are known. Thus, the problem of planning involves selection of the different components; determining component capacities; identifying interconnection topology of the components and scheduling disciplines so that the resulting system can handle the given workload under specified performance and cost constraints. To keep the example simple, we will ignore factors such as cost, reliability, etc. So that, the main problem is to select a combination of choices with which the system offers the best performance.

An approach to solving the problem without using the package is to write a benchmark for the given workload and to run it on different configurations until we find one that offers the performance. This method, however, is expensive. The problem can be solved relatively easily and inexpensively using the package as follows.

The user begins by describing a base line configuration. If a user is interacting with the package through an interface tailored for this application, describing a base line configuration will involve describing different devices, their capacities, their interconnections, the scheduling disciplines and the performance measures of interest. The variable factors for which alternate choices exist will be described next. The interface will set up a queueing network model for the base line configuration. Relative to this base line configuration, the user describes the alternate choices. The package interface then initiates the execution of the solution and report generation phases successively to solve the model and provide comparative performance figures.

The user may choose to bypass the application oriented interface and work with the package at a performance analysts' level. If the application oriented interface has been implemented and the user just needs the standard application schematic, there should not be any reason for a user to do this. But we consider this possibility here to provide the performance analysts' view of the package. Working at this level, the user sets up the queueing network model for the base line configuration using MDL. The user creates a model definition file and using a text file editor available on the host system describes the queueing network model in term of MDL constructs. Alternatively, the user may retrieve the model from the model Archives if such a model has already been developed.

After building the model definition file, the user logs on to the package. The user identifies the performance quantities of interest such as throughput, response time, etc. The alternate choice with respect to the base line configuration are given more explicitly as the range values representing the capacities of individual components, and interconnections. The user may select a particular solution technique or leave this decision up to the package. In the latter case, the Monitor in the package determines the most appropriate solution technique based on the input model. The results obtained by solving the model are saved in an output data structure. The user may choose to sample the results as the solution of the model proceeds. The user may decide to prematurely terminate the execution after studying the sample results.

Next, the user interacts with the Monitor to specify a set of performance quantities (which may be a subset of those provided by the solution phase), and the formats for tabulation of these results. Reports in the form of tables or plots

of comparative performance measures for the alternate choices may thus be obtained.

After examining the performance reports, the user may choose to experiment with various types of components, different numbers of components, alternate interconnection topologies, or alternate scheduling disciplines. Experimentation may either involve changing the model in the model definition file or merely providing new values for some model variables. In the latter case, the user interacts with the Monitor to provide new values for the variables. To change the model the system text file editor is to be used to appropriately modify the model definition file. In either case, the process of solution and report generation will have to be repeated. A typical configuration planning phase will involve several iterations of this type.

3. Design Goals

In the process of any design a set of choices have to made. In order that the end product be similar to the expectations, the choices should be made based upon criteria that are consistent with the design goals. In this section, we outline the major design goals for the package.

1) Modularity of system design: The total package should be decomposable into a set of well defined modules. Modules should interact with each other only through data structures and a module need not know the internal operation of other modules. A module should correspond to some logical function of the package. This is required to facilitate independent design and implementation of the various modules.

2) Flexibility and Generality of the Model Definition: Much emphasis is to be placed upon the generality of the package in the classes of problems that can be solved and flexibility in extending its applications. More specifically, it is intended that

a) a parameterized definition of a wide class of systems of interest should be possible,
b) the class of systems to be analyzed should be defined independent of the solution techniques to be used,
c) a model be built independent of the technique to be used for solving the model
d) it should be possible to carry out small modeling exercises efficiently and quickly, while providing facilities to handle more complex modeling problems.
e) the model definition be adaptable to be solved by different solution methods:
   * exact solution techniques for analytic queueing network models,
   * approximate solution techniques for analytic queueing network models,
   * simulation techniques for generalized models,
   * hybrid solution techniques, (e.g., simulation and analytic methods) used in solving subsystems of the same model, and
   * solution techniques based upon hierarchical decomposition.

3) Ease of Use: It should be possible to use the package at one of the two levels: Application Analyst, and Performance Analyst. (See section 2. for more details on assumed user skill levels.) At either of the two skill levels, a user should be able to learn to use the package in a systematic manner. The package should provide basic guidance for the use of the package to a naive user to develop and solve relatively simple models. For the more experienced users, it should provide a quick reference to the complete set of capabilities of the package.

4. Functional Overview

From the view point of the package, a typical modeling analysis involves the following logical functions:

I. Information Extraction: Obtaining and assembling the information to define a) a base-line model for the system of interest, b) the performance measures of interest for which the model is to be evaluated, c) the variables and their values in the successive models to be solved, and d) miscellaneous information such as the precision of the results, format of the output reports, etc. The information may be extracted from several sources: a) directly from the user in interactive sessions, b) from the model archives in terms of some pre-defined models, and c) as set of assumptions pertinent to the application.

II. Model Solution: There are several solution techniques for queueing network models: convolution techniques for exact solutions, mean value analysis, simulation, hybrid solution techniques, different approximation technique, and the solution techniques based upon decomposition and aggregation. The suitability of a solution technique for a given model may depend upon several factors: the assumptions and details included in the model itself, the required precision of the results, the time and cost constraints, the number of different models to be solved for the analysis, etc. Moreover, at times it may be desirable to solve a model using more than one solution technique to validate the model and/or the appropriateness of a solution method.

III. Communication of the Results: The results obtained by the solution of a model or a series of models may be used in several different ways. The output of the analysis may be sampled, as the analysis is being performed, to affect the course of the current execution in particular and the whole modeling study in general. A summary of the results in terms of tabulation and

plots or simple listing of the performance statistics can be used to study the behavior of the model. The performance statistics of a model may be used as input parameters for another model, especially when analyzing a system using decomposition and aggregation techniques. And finally, the detailed output traces may be useful in carrying out a follow up analysis for further validating and characterizing the model.

In the following section, we describe the way these logical functions are mapped on to different modules implementing the package.

4.1 Overview of the Design

The architecture of the package is shown in Figure 1. There are three main structures for communication between the user and the package and between the different modules of the package. The subunits for the definition of a model are defined in terms of the Model Definition Language (MDL). A model is internally represented in terms of the Internal Data Structure (IDS). The results of the solution are represented in terms of the Output Data Structures (ODS).

To define the model all the information is collected in the MDL primitives. There are several ways in which the information required to define a model may be provided by a user. An application analyst may define the model in terms of a predefined schema for generating the class of models pertinent to the given application. A user may simply opt to retrieve a pre-defined model in the repository of model Archives. A performance analyst well versed with MDL primitives may define the model directly in those primitives. And finally, a user not so well versed with the MDL primitives or with the schema of an application may interact with the package in the Tutored Mode to define the model.

The functions for the user interface and coordination of different activities are implemented by the Monitor. The Monitor has been further structured into different submodules to implement these functions. These modules are shown in Figure 2.

The MDL Parser transforms the MDL primitives into the IDS to be used for the solution. A consequence of the MDL parsing is the verification of the consistency and completeness of the defined model.

Before the model can be presented to the solution modules for execution, yet another task has to be completed. There may be variables whose values are not bound to their names at this stage. Such a feature is provided to facilitate solution of the same model with different values of the parameters. The variable names may just be bound to the new values without reparsing the MDL definition. An explicit module called Binder carries out the binding process.

The solution techniques are implemented by the solution modules. That is, each of the solution methods such as convolution, mean value analysis, simulation and each of the approximation techniques are implemented by a separate module. The hybrid and decomposition-aggregation techniques are implemented by a module that systematically invokes the other solution techniques.

Most of the utility programs to interpret the formats, queries, sampling directives, and other output related functions are implemented in the report generator module.

In summary, the package may be structured into several modules and submodules each corresponding to a logical function. The Monitor provide an interface to the user and coordinates the flow of control as the package is executed for a modeling analysis.

5. The Classes of Models to be Solved

We would like to be as general as possible in defining the types of systems that the package may be used to model and analyze. The type of systems that we envisage the package to be used to evaluate, design and improve upon may be represented by generalized queueing network models.

Queueing network models are a general class of models which make use of simplifying assumptions to represent a complex system with a relatively small number of parameters. A queueing network model may be defined by:

1. Resources
2. A Population of jobs making requests for the resources, and
3. Rules which govern how the jobs proceed through the network.

A resource is viewed as a waiting room (queue), a set of servers, and a scheduling discipline describing the order in which the jobs in the queue are to receive service. Examples of system components generally representable by model resources include processors, memory, I/O channels, etc. Software systems entities such as file managers, data-base managers and I/O handlers may also be represented as resources in queueing network models.

Requests are described in terms of the average number of times a job requires a resource and the work demand placed upon that resource per request.

The rules which govern how the jobs proceed through the network represent the flow of jobs through the system. For example, a job upon completion of CPU service might require I/O activity seventy percent of the time and leave the system, (i.e., complete) thirty percent of the time. A job moves according to the routing rules through the network making requests at the resources. If

upon arrival at a resource, a job finds all servers busy, it waits in the queue until a server becomes free. The next job to be processed at the resource is then selected in accordance with the queueing discipline at that resource.

In the following, we describe the characteristics of the models and the systems to provide a feel for the kinds of systems that can be modeled with the primitives available in the package. We use the queueing network terminology for such description.

The package may be used to model two broad classes of systems, open systems and closed systems. These two classes of systems are modeled by open and closed queueing networks, respectively. Within each of these broad classes, a system may have different characteristics for the jobs and the way these are handled by the system, different features for the resources and a variety of classes of routing rules. Multiple class systems with/without class changes are handled by the package.

The queueing network models may be conveniently categorized into several classes based upon the solution techniques which offer least expensive and fastest analysis.

1. Product Form Networks: These types of networks lend themselves to solution using exact analytic techniques such as convolution algorithms and mean value analysis.

2. Non-Product Form Networks: Other analytic and non-analytic techniques are available for the models that do not satisfy all the assumptions required for the product form networks. Among these techniques are several Approximation techniques, Simulation, Hybrid techniques.

The package provides fairly general versions of all the techniques described above.

There exist several features in systems which are either hard to describe or are hard to analyze without sophisticated techniques. The generality of a performance analysis tool often depends upon which of such situations can be handled. In the following we describe some such features that can be modelled and analyzed using the package.

1. Blocking of Resources: This pertains to the situations in which a server may be blocked because the queue of some other server is filled to capacity. To allow a resource to block another resource often precludes the use of product form solution techniques.
2. Holding of Multiple Resources: The package includes the primitives to describe and analyze models which allow a job to hold multiple resources. An example of such models arises while modeling disk subsystems with unequal number of channels and devices. Again, such models are solvable with non-product form solution techniques only.
3. Scheduling Disciplines: Jobs may be scheduled for processing at a resource using one of several scheduling disciplines. A Variety of scheduling strategies may be represented in the models.
4. Service Requirements of Jobs: The class of systems that may be modeled are also reflected in the generality with which the service requirements of a resource may be specified. At one extreme, the service requirements of a resource may only be specified using some pre-defined distributions, whose parameters may not depend upon any system characteristics. At the other extreme, arbitrary distributions with parameter values depending upon the system state may be included. In the package, resource requirements of a resource may be defined using both pre-defined as well as user defined distributions. The parameters of such distribution do not have to be fixed, but may depend upon the queue lengths at that resource or the state of the system.
5. Routing Rules: The rules that govern the manner in which jobs proceed through the queueing network may be based on several factors. The standard means of specifying job flow through the system is by specifying the probability for a job to select each of the potential destinations, after leaving the present resource (i.e., Probabilistic Routing). In addition to the probabilistic routing, the package provides a generalized schematic to define arbitrary deterministic and probabilistic routing rules. Using this schema, routing rules may be specified in which the decision as to which destination a job selects after leaving the present resource may depend upon such factors as the state of the system, and the attributes of the job.

To allow for the representation of such a large variety of models, we need a rich set of model definition primitives. We present in the next section a set of primitives that provides such an ability and forms the basis of the model definition language of the package.

5.1 Model Definition Subunits

Resources and the jobs are the fundamental entities of a queueing network model and it is no surprise that most of the primitives required are for describing these two entities. Resources are often categorized into two classes: Active (i.e., the resources at which a job spends non-zero time, once it has acquired the resource) and Passive (i.e., the resources which a job has to merely acquire and it does not need any service from such resource). There are two types of factors which dictate how job behaves at a resource: those associated with the resource and the others that depend upon the attributes of the job. (Of

course, sometimes such a division is only superficial.) The factors which depend upon the job attributes are conveniently described by grouping jobs into classes and chains and then associating the attributes with the class or chains of jobs. Subunit set gates may be used to define dynamic changes of the attributes of a job.

The primitives for describing the flow of jobs through the system also constitute significant part of the model description. The definition of a job's flow through the system involves defining creation of the job, its routing from resource to resource, and its exit from the system. The subunits source, fission, and split gates are used for creating jobs, sink and fusion gates for describing the exiting of the jobs from the system, and the routing primitives are used to define the path a job would take as it travels from resource to resource.

There are some other miscellaneous primitives also. Initial state describes the state of the system as the system is started, for example, while simulating the system.

5.2 An Example System and Its Model

In this section we consider an example system to be modelled and analyzed using the package. We first describe the system and then subsequently its model in terms of the primitives and the subunits provided in the package.

The architecture of the system is shown in Figure 3. The relevant characteristics of the operating system are as follows.

There are two classes of users, "batch" and interactive ("demand"). A batch job may be in any of the following states:

1. Initiation. In this state the job is set up to be ready for execution if the needed resources are available.

2. Queued for memory: this state may be artificially separated from job the initiation state. Once a batch job acquires memory it is not swapped out.

3. Holding memory while queued for CPU usage,

4. Holding memory while using the CPU,

5. Holding memory while queued for I/O (a small amount of CPU interaction is considered negligible in the I/O transactions),

6. Holding memory while using an I/O resource, and

7. Batch job termination. In this state the resources that the job is holding are retrieved and its existence in the system in terms of job control block and other information is destroyed.

A demand job may step through the following states:

1. Receiving terminal input: In this the job does not tie up any of the major system resources. The duration of time for which the job stays in this state is determined by user's behavior.

2. Queued for memory: When demand jobs are waiting for user input they may be swapped out of memory, so upon receiving user input, the job may have to wait for its memory reallocation.

The other states are same as the states 3 to 7 for a batch job.

The job scheduler in the operating system functions as follows. Whenever permitted by the available work load a constant number of batch and interactive jobs are maintained in the active state. Alternatively, it may be assumed that whenever an interactive or a batch job completes, a similar job is introduced into the set of active jobs.

The quantitative values for the workload for each job are obtained using the performance monitors available with the operating system. Examples of such quantitative measures include average CPU and I/O burst time (i.e., service times), the number of times an average job visits each of the resource (i.e., visit ratios), the number of jobs that are on an average in the active set (i.e., degree of multiprogramming or DMP), etc.

A queueing network model for this system is shown in Figure 4. The model imitates the flow of jobs of the real system. All of the important resources are reflected in the model. The CPU, the Terminals, and the I/O devices are each represented by a FCFS queues. The finite memory is represented by a Passive resource called memory.

The jobs are distinguished as batch and interactive, with respect to their resource demands and the path of flow. The interactive jobs alternate between the service at the terminal and the rest of the system. While a batch job when once enters leaves the system only after the service is complete.

Upon entering the system a job requests memory allocation at the allocate gate. The amount of memory required by a job is represented by the number of memory tokens requested, which is defined by the token demand distribution. After obtaining the memory, the job receives service from the CPU and an I/O device. The branching probabilities d_p1, d_p2, d_p3 represent the probability that a demand job would select IO1, IO2 and IO3 for I/O service, following a service at the CPU. b_p1, b_p2, and b_p3 represent the corresponding values for a batch job.

As a job completes service at the I/O, it releases the memory at the mem-release gate. An interactive job, at this time, returns to the terminal for an amount of time defined by the think time distribution. A batch job, however, leaves the system. It is assumed that the operating system maintains a fixed number of batch jobs

active at all times. This is reflected in the models by the reset gate, which introduces a batch job into the network as soon as one exits.

Here, we leave the example at this stage. In the following sections, as we describe other aspects of the package, we shall use this example as an illustration.

6. Model Definition Language

In the last section, we described the subunits of a queueing network model. To describe a model to the package, such as the one given in the illustration of Section 5.3, a generalized Model Definition Language (MDL) has been developed. Here, we present an overview of the language and an illustration of its use. The Model Definition Language(MDL) is like a procedure oriented high level language.

A model definition is organized into seven sections, as follows:

1) File identification section
2) Preamble section
3) Job description section
4) Subsystem specification section
5) Resource description section
6) Job flow description section
7) Initial system state specification section

Here we illustrate the MDL constructs by an example. We may point out that there are several features of the MDL that are not included in the example.

Consider the model developed in Section 5.3. In MDL constructs, the model is described in Figure 5.

7. The Monitor

The monitor is the main control module in the package. There are two types of activities for which the monitor is responsible: providing a friendly interface to the user and coordinating the actions of different modules of the package. In the following we describe each of these activities.

7.1 User Interface

In order that both the advanced and novice users can use the package effectively, two types of interfaces are provided by the package: Command Mode and Tutored Input Mode. A user (whether performance analyst or an application analyst) who is well versed with the primitives and the options available can directly define and execute the models efficiently using the Command Mode. A relatively novice user can be led through the structures and the options of the package with the Tutored Input Mode and can effectively use the package without having to acquire a detail knowledge of the inner-working.

The interaction with the package through Command Mode is implemented by defining several set of commands for various phases of the analysis. The commands may be divided into several categories:

1. For Model Definition: The purpose of the model definition commands is to provide information regarding the model definition. A model may be defined in several ways: by giving the name of file with MDL definition, accessing a definition from the Archives, collating multiple definitions, or may be generated internally from some predefined schema. Commands are provided to cover each of these definition methods.

2. Input/Examine Variable Values: The values for the variables may be input in several different ways. A set of values may be assigned to the variables for each execution of the model. Alternatively, a set of values may be assigned for each variable and the package be directed to execute the model by using different sets of values for each of the variables. Commands are defined to examine the values of the variables and of other parameters of the model.

3. Output and Results: There are four types of actions that may carried out upon the results obtained through execution of the model. The results may be sampled as computation proceeds. Secondly, when the execution of a model is complete a summary of the results may be obtained in terms of listings, tabulations and plots. A set of performance measures may be identified for which the information is to be stored for a follow up analysis using the data base and the associated query system. And finally, the package may be informed that the output of the current execution may be used as an input for another model. The commands are defined to specify the performance measures and their formats for each of these categories.

4. Model Archiving: Commands are provided to save the model so that it may be retrieved at a later moment.

The Tutored Mode has been planned to circumvent the need for a user to know the format of these commands. A menu driven implementation of this mode has been proposed. The information is organized in a tree structure. The information at each node of the tree is defined at several levels of details. This allows for a balance in user interaction to achieve clarity without overabundant prose. For traversing the tree, the user is presented with a series of questions, most of which have a selection of numbered answers.

There are a number of desirable features to be added to this basic question-answer dialogue. The user should have the capability to view the partially defined model, make changes, reverse previous decisions, or ask for further explanation on a query. The flexibility of machine interaction through a human engineered interface is the ambitious goal in the design of this mode

of user interface.

7.2 Example of Interaction With the Monitor

In section 6 we described the MDL representation of an example model to be solved. After completing the model definition in MDL, a user will typically proceed to execute the solution phase of the model. At this time the user starts interaction with the package. In the following, we identify at a functional level the interaction of the user with the package and the activities of the Monitor.

1. At entry the user is asked to make a series of choices for the package to determine the skill level of the user, the mode of operation, i.e., Tutored Mode or Command Mode, the user terminal type, and the degree of detail for the prompts. A default set of options are invoked, if user wished to bypass this stage.

2. The information is extracted from the user that will lead to a complete definition of the model in MDL. The user may simply give the name of a model file or may invoke the model Archives Management to retrieve a predefined model. The details of the interactions with the Archives Management are described in section 8.

3. The user is asked to select a suitable method for the model solution As a default, the Monitor may select the suitable solution technique. In this case, the simulation will be an appropriate technique.

4. At this stage, upon user's request the monitor invokes the Input phase submodule to parse the model. The Parser produces a data structure of the model, i.e., IDS. If there are any inconsistencies in the model definition the user is informed and the process is aborted if the inconsistencies are fatal.

5. At this stage the package requests the values for the unbounded variables. In the example at hand, the user specifies the values for variables "batchno", demandno", "d_cycle" and "b_cycle". The user may in addition review the model and the values of the parameters.

6. The user at this time provides the information about the performance statistics to sample, to be summarized at the end of the execution, to abstract to link this model to some other model, and for possible interest in the follow up analysis.

7. The Solution phase module of the Monitor invokes the Binder to bind the model parameter values for all occurrences of the relevant variables in the model. The control is then passed to the appropriate solution module. This module solves the model, writing its output to the Output Data Structure(ODS) and signaling completion to the monitor.

8. The solution submodules may run into some contingencies or may need to communicate with the user to display the sample results. In either case, the control is systematically passed to the Monitor which coordinates the appropriate activity.

9. After the execution is complete the user may choose to do one of several things: save the model and suspend the execution for now, execute the model with different values for the parameters, or carry on a follow up data analysis using the query system.

8. Archives Management

The Archives Management implements the basic facility to store and retrieve the predefined models. The models are stored in the form of MDL. The archived models may be the models for complete systems or merely the components of a system that may be collated to create a complete model.

The MDL files may be archived in one of the following ways:

1. A user types up a model in a file, using the system editor, and want the package to maintain the model so that he or some other users may be able to use this model.

2. The user suspends the editing session for defining a model to resume it later. The partially defined model may be saved in the Archives.

3. Using an application schema the package created the model. The user may also archive such a model.

4. During the model execution the variables in the model may have been assigned some values that are different than the ones provided as the initial values in the MDL definition. An option is provided where the model definition may be archived with the set of values current to the execution.

Several pieces of information is stored along with the model definitions at the archiving time. This may include:

1. Status indicating whether the model has been executed at least once,

2. A symbol table giving the number, type, and name of all of the model parameters. This is provided especially if the model is archived with the option mentioned in 4) above.

3. Version control information. This information is needed for addressing and protection of the elements archived. At present

only a modest protection system is proposed, since the extent of the protection will significantly depend upon the protection facilities available in the host system. The retrieval of the model is possible using more than one keys. For this purpose, an Archives index is maintained with the name of the system modelled, name of the author, the version number, and an additional optional key.

9. Concluding Remarks

We have presented the design level details of the package under development. As noted above, the goal is to create a flexible and expandable environment for system performance studies. New techniques and methodologies, as they are reported would be easy to incorporate in the package.

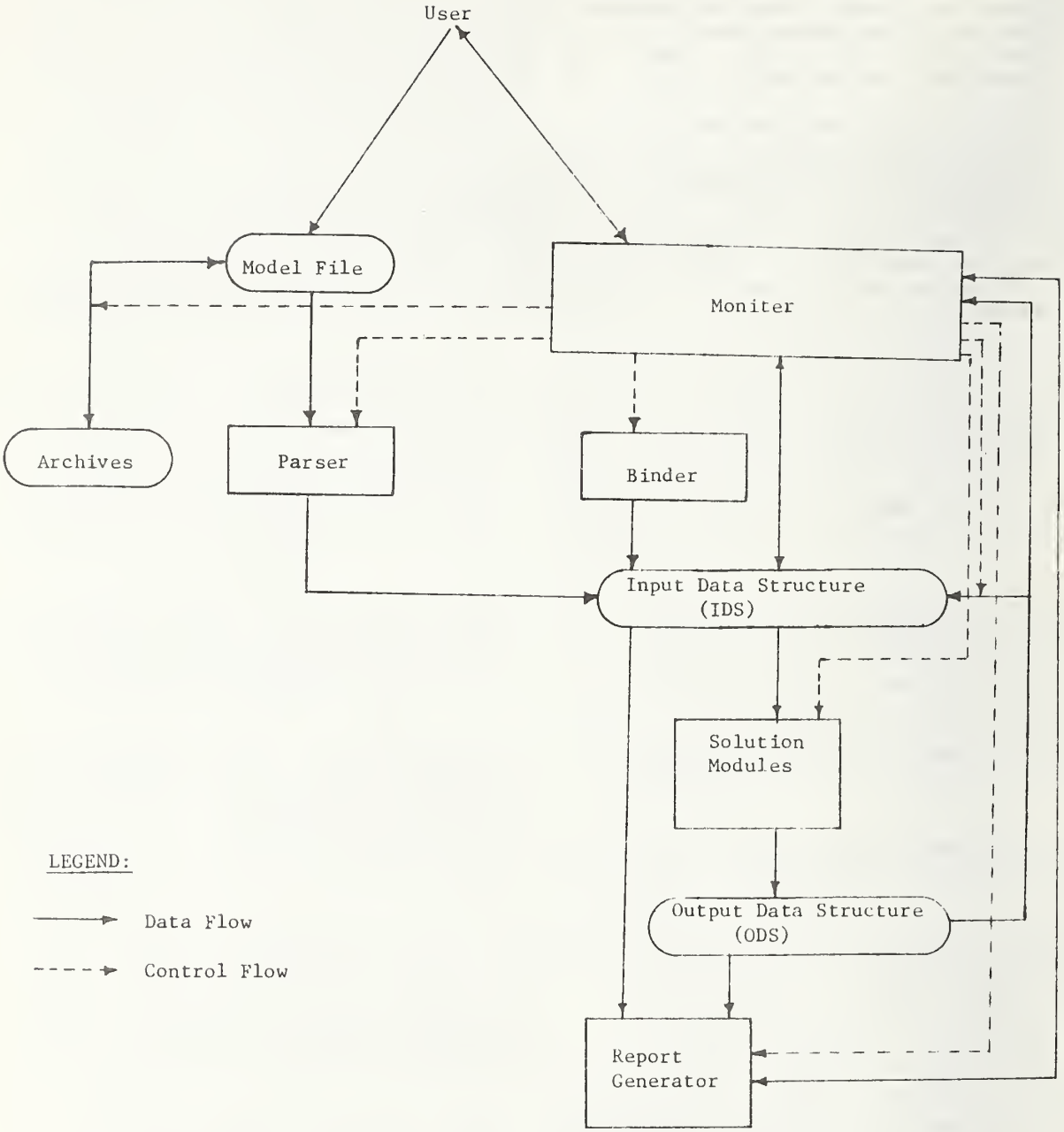


Figure 1

Simplified System Block Diagram

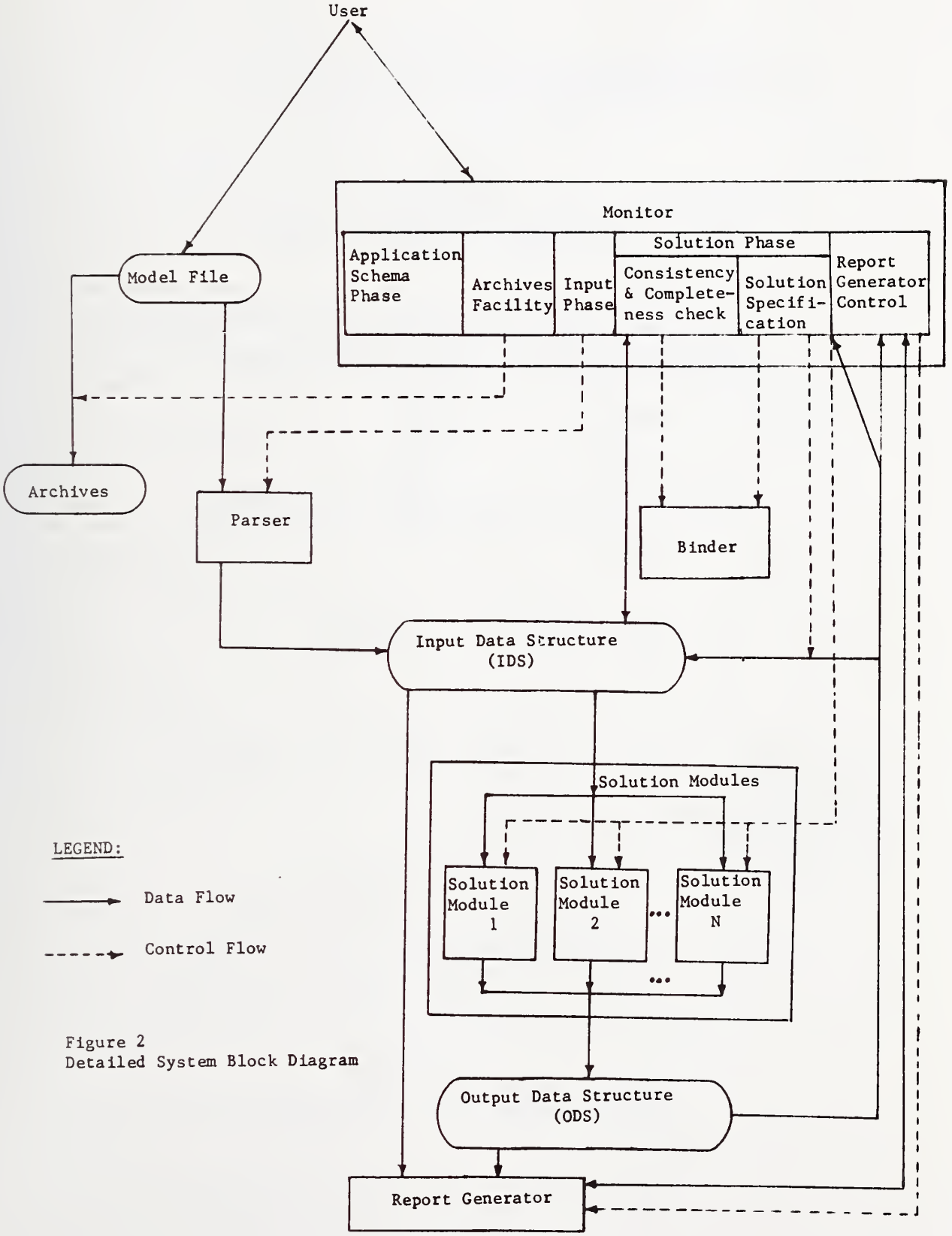


Figure 2
Detailed System Block Diagram

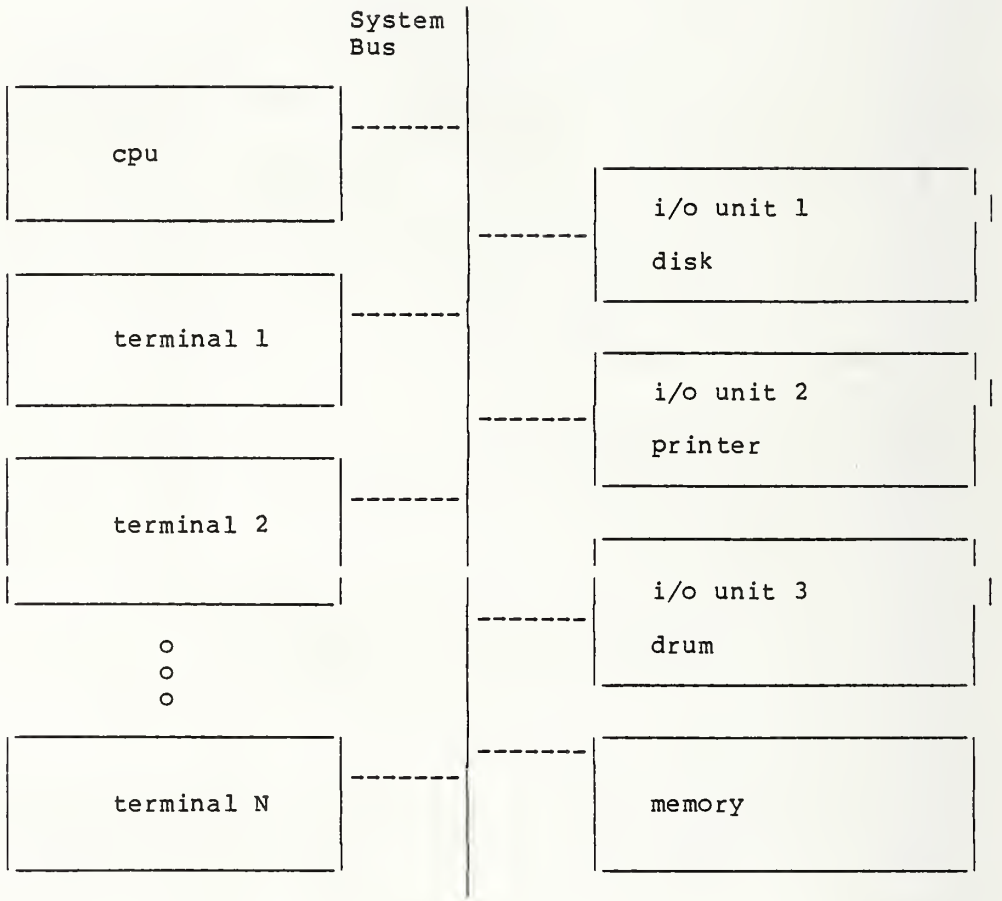


Figure 3
Architecture of the Example System

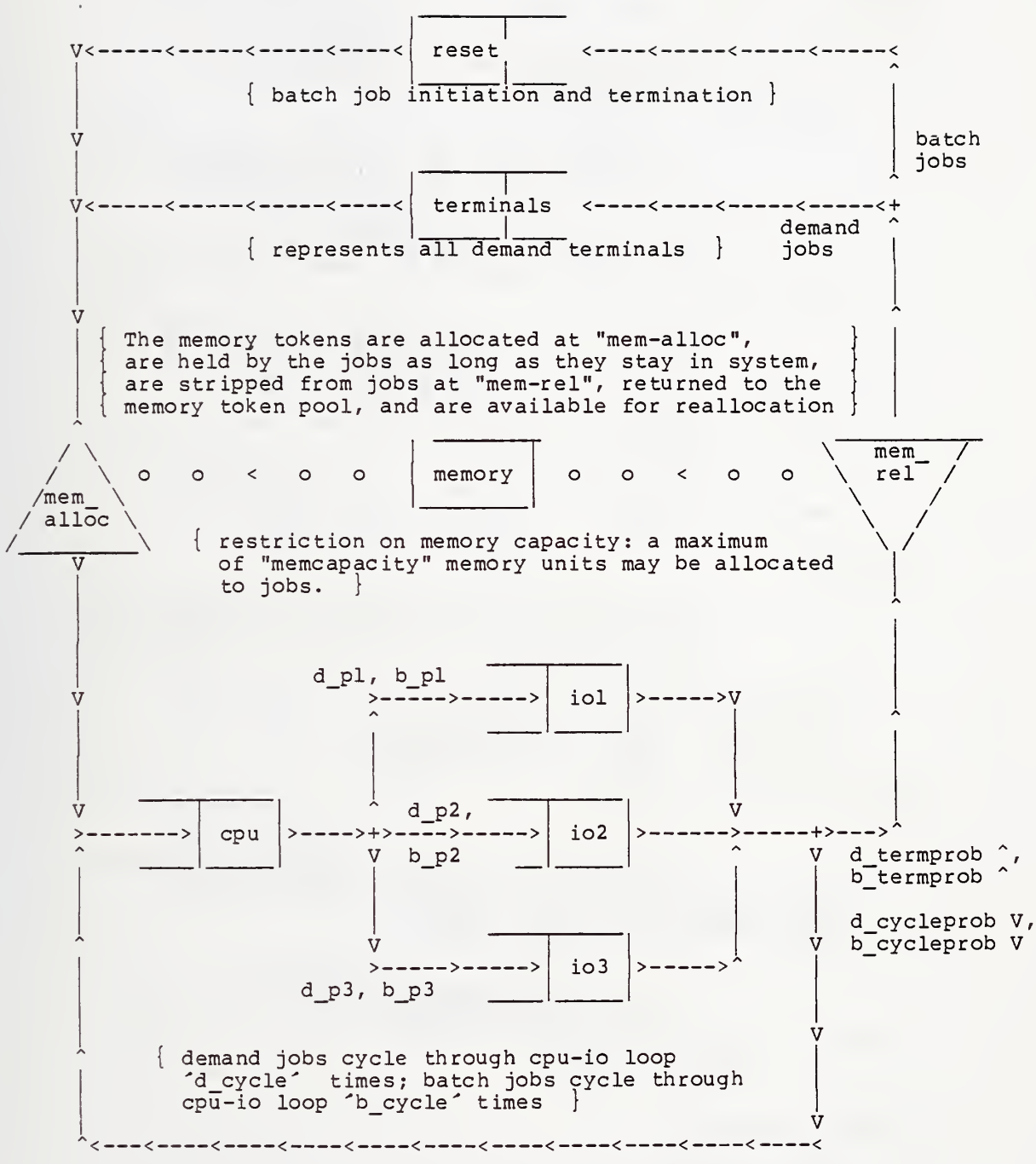


Figure 4

Queueing Network Model of the Example System

Figure 5

```
MODEL      OurModel

VERSION       4.1

VARIABLES
                  { The number of memory subunits available
                    for jobs is: }
     memcapacity : INTEGER := 64 ;

                  { The probabilities of "demand" users branching
                    from cpu to iol and io2 respectively are: }
     d_pl, d_p2 : REAL := 0.3433 ;

                  { The probabilities of "batch" users branching
                    from cpu to iol and io2 respectively are: }
     b_pl, b_p2 : REAL := 0.3573 ;

                  { The number of times "batch" and "demand" users
                    execute a cpu-io cycle is }
     b_cycle, d_cycle : REAL ;

                  { The number of users of class "batch" is }
     batchno : INTEGER ;

                  { The number of users of class "demand" is }
     demandno : INTEGER


RELATIONS
                  { "b_outprob", "b_cycleprob", "d_outprob",
                    and "d_cycleprob" are branching probabilities
                    which depend upon the variables "b_cycle" and
                    "d_cycle". }
     b_outprob : REAL := 1/ b_cycle ;
     d_outprob : REAL := 1/ d_cycle ;
     b_cycleprob : REAL := 1- b_outprob ;
     d_cycleprob : REAL := 1- d_outprob ;
```



```
                  { The probabilities of branching for "demand"
                    and "batch" users from cpu to io3 : }
     b_p3 : REAL := 1 - b_pl - b_p2;
     d_p3 : REAL := 1 - d_pl - d_p2;


                  { Specification of number of jobs of class
                    "batch" and "demand" }
CLASS       batch : batchno ;
            demand : demandno


ACTIVE   terminals

     QUEUE FCFS
     RATES DEPENDENT demand  EXPONENTIAL(#)
          UPTO demandno
          FIRST 0.5
          DELTA 0.5

ACTIVE   restart
         { This resource is for batch job accounting : to
           indicate overhead for a job of class "batch"
           entering and leaving the network.  }
```

Figure 5 (cont.)

```
     QUEUE FCFS
     RATES DEPENDENT batch  EXPONENTIAL(#)
          UPTO batchno
          FIRST 0.1
          DELTA 0.1

ACTIVE   cpu

     QUEUE PS
     RATES INDEPENDENT
           batch  EXPONENTIAL(10),
           demand EXPONENTIAL(8)

ACTIVE  io1

     QUEUE FCFS
     RATES INDEPENDENT
           batch  EXPONENTIAL(5),
           demand EXPONENTIAL(3)

ACTIVE  io2

     QUEUE FCFS
     RATES INDEPENDENT
           batch  EXPONENTIAL(3),
           demand EXPONENTIAL(8)

ACTIVE  io3
```

Model Definition Language

```
     QUEUE FCFS
     RATES INDEPENDENT
           batch, demand  EXPONENTIAL(4),

PASSIVE memory
     ALLOCATE mem_alloc
     RELEASE  mem_rel
     QUEUE  FCFS
     TOKENS memcapacity
               { The token demand distribution represents
                 the integer number of memory subunits in
                 each job's working set }
          TOKENDEMAND
               batch EXPONENTIAL (10),
               demand EXPONENTIAL (6)
```

Figure 5 (cont.)

```
ROUTING

     CLASS  demand

          terminals ->
               mem_alloc ->
               cpu ->
               (d_p1, d_p2, d_p3 : io1, io2, io3 ) ->
               (d_cycleprob, d_outprob : cpu, mem_rel);

          mem_rel -> terminals;

     CLASS  batch

          restart ->
               mem_alloc ->
               cpu ->
               (b_p1, b_p2, b_p3 : io1, io2, io3 ) ->
               (b_cycleprob, b_outprob : cpu, mem_rel);

          mem_rel -> reset;

EOM     {End Of Model }
```

A PERFORMANCE BOUND FOR MULTIPROGRAMMED VIRTUAL MEMORY SYSTEMS

Rollins Turner

Digital Equipment Corporation
Distributed Systems Performance Evaluation
Tewksbury, MA 01876

Optimal Memory Allocation and Scheduling of two programs running in a memory constrained environment is investigated. A simple mathematical model, representing the system as a controlled Markov process, is defined. The problem of jointly optimal memory allocation and scheduling is solved for this model using a technique from stochastic control theory. Several improvements in computation time are shown, and a technique for ensuring convergence of the iterative solution algorithm is developed. Numerical examples are given, showing several possibly counterintuitive results.

1. Introduction

The performance of a multiprogrammed virtual memory computer system depends on numerous decisions made by its operating system. We can group many of these decisions into two major categories: program scheduling and memory management. Scheduling means deciding which program to run at any given time. Memory management involves two kinds of decisions: deciding how much memory to allocate to each active program at any time, and deciding which of a program's pages to hold in its allocated space. The designers of a virtual memory operating system face the problem of determining how the operating system will make these decisions. In this paper a simple and abstract version of that problem is formulated in terms that permit mathematical treatment, and solved by a technique from stochastic control theory. The resulting control policy is not claimed to be of practical value for application in actual systems. However it provides a bound for the performance of a broad class of control policies. It could therefore be of use as a benchmark against which the performance of practical control algorithms can be compared or in a proof of optimality of another, more practical, algorithm. In addition, the relatively simple system model used in these solutions permits certain insights that might be obscured by the complexity of a more realistic model.

1.1 Overview

We attack the problem of running two programs as quickly as possible in a limited amount of memory. In doing this, we must decide how much of the memory to allocate to each program, which program to run any time, and when to do page reads. Section 2 of the paper provides definitions and background information, and then it gives a mathematical formulation of the problem. The system is represented as a controlled Markov process, and the problem is defined as determining an optimal control, or parameter value, for the process. The cost function to be minimized is the expected time to reach a terminal state, which represents completion of the two programs.

Section 3 describes a solution to the problem defined in Section 2, using a basic technique of stochastic control theory known as "iteration in policy space." With this technique an initial approximation is made for the expected time until completion from each system state. On the basis of the approximate finish times, the best control decision at each state is determined.

Using these decisions, new approximations are determined for the minimal expected finish times. This cycle is repeated until it converges to a fixed set of decisions and expected finish times, which are guaranteed to be optimal. Section 3 describes this solution technique in detail, including practical considerations of computation time and determining when convergence has occurred. Also, Section 3 provides an intuitively meaningful interpretation for the intermediate results of the iteration. It makes use of this interpretation to develop a technique for determining successively better upper and lower bounds for the minimal finish times at each state.

Section 4 provides several numerical examples, each intended to illustrate a particular point. We might hope to find some simple rules by which optimal decisions can be determined without having to go through the extensive calculations described in SEction 3. No such rules have been found at this time. The examples in Section 4 refute several seemingly reasonable principles that might be thought to lead to such rules.

Section 5 gives a summary and conclusions.

1.2 Relation to Prior Work

The memory management problem has been studied extensively over the past ten years. (See [COFF77], [FERR78], and [SPIR77], and bibliographies therein.) Scheduling has also been studied extensively, both in reference to computer systems and in a more general context. (See [COFF73], [COFF76], and KLEI76], and references therein.) However, the treatment of scheduling and memory management as a joint optimzation problem is not well developed. This paper deals with the joint optimization problem, within the framework of stochastic control theory.

Application of control theory to analysis of computer systems is not a new idea. Ashany [ASHA72] formulated several problems of computer performance analysis within a state space framework, and used control theory techniques to solve them. Wilkes [WILK73] used a control theory setting for a discussion of memory management policies, including general descriptions of several existing systems. Lew [LEW76] formulated the page replacement problem as a multistage decision process, solvable by dynamic programming. Arnold [ARNO75] developed a memory management algorithm using a Wiener filter predictor. Raj Jain [JAIN78] developed a more general predictor based on an ARIMA (Autoregressive Integrated Moving Average) model of the program's memory reference behavior and the forecasting methods of Box and Jenkins [BOXJ76].

This paper differs from the previous work that applied stochastic control theory to memory management, and from most prior work in memory management, by considering the issues of memory management and scheduling as a single problem with a system level cost function. All of the above mentioned papers that treat memory management mathematically optimize cost functions defined for individual programs, such as page fault rates. We shall not be concerned with the performance of individual programs or with page replacement algorithms. We assume that some page replacement algorithm has been chosen and that the resulting page fault rates are known for each program running in each possible amount of memory. We assume that these fault rates are statistically stationary, and that increasing a program's memory allocation does not increase its fault rate. Within that framework, we concentrate on how to divide the memory among the programs, and which program to run when both are ready.

2. Problem Formulation

In this section we define a simpler and somewhat more abstract mathematical problem based on the very complex and ill-defined actual problem of designing an optimal scheduling and memory management policy for an

operating system. Our goal is to capture the essence of the real problem in a tractable mathematical form. We begin with a model of program behavior, described in Section 2.1. Section 2.2 then defines the system within which the programs are to be executed. Section 2.3 describes a representation of the overall system as a controlled Markov process, which is the context for our optimization problem. Section 2.4 gives the problem statement.

2.1 The Program Behavior Model

We represent the execution of a program as a continuous time, finite space, stochastic process. When a program is running, it produces page faults (references to pages not in memory) at random intervals, with the average fault rate depending on the program's memory allocation. If the program occupies N pages of memory, its execution time between page faults is exponentially distributed, with an average rate of mu(N) faults per second. We assume that mu is a nonincreasing function of N. We also assume that the total execution time for a program is exponentially distributed, with mean completion rate nu. All probabilities are assumed to be time independent.

Notation: We use Greek letters for parameters of stochastic processes, which have arbitrary values, but which will be known when any specific instance of the problem is to be solved. In this draft, the English names of the Greek letters will be spelled out, as in the above paragraph. When it is necessary to distinguish between parameters of separate processes, numeric indices will be used, either within parentheses or as a suffix to the letter name. Thus mu(1,N1) denotes the fault rate of program 1 running with N1 pages in memory. The symbol mu1 will be used for the same purpose when the amount of memory is understood.

2.2 System Description

We shall be concerned with a multiprogramming system in which two programs are to be executed. This section describes the rules according to which we assume that system operates.

Whenever a reference is made to a page that is not in memory, the page must be read in before the program can continue. A page cannot be read in except as a result of such an event, which is called a page fault. The time required to read in a page is an exponentially distributed random variable with known mean (1/lambda), and is independently and identically distributed for all pages and both programs. Only one read may be in progress at any time, and once started the read must go to completion. Page reads may be overlapped with execution for the other program so long as the other program is executable. Only one program may be executing at any time.

At various times in the operation of the system just described, decisions must be made. An actual system will incorporate a set of rules by which these decisions are made in response to various system events. There are three types of decisions that must be made:

1. Which program to run at any time.

2. When to read in a required page.

3. Which page to displace with a page being read in.

We shall limit our concern with page replacement to deciding from which program to remove a page when a replacement is necessary. An unspecified page replacement algorithm makes the decision of which particular page to remove from that program. The page fault function, mu(N), for each program reflects the page replacement algorithm as well as the program's behavior.

As we have defined the system, decisions are only required at certain discrete points in time. Decisions are required when the following events occur:

1. A page fault.

2. Completion of a read.

3. Termination of a program.

2.3 Representation as a Controlled Markov Process

The system just described can be represented by a controlled Markov process [KUSH71]. A controlled Markov is identical to an ordinary Markov process except that the state transition probabilities can depend on a parameter, which is called the control. For any particular choice of the parameter value, the state transition probabilities are fixed values, and the resulting process is an ordinary markov process. Thus we can think of a controlled Markov process as a family of ordinary Markov processes with a common state space. By specifying the parameter value, or control, we select one member of the family. For the problem of interest here, we need only consider controls with a finite number of values (i.e. finite set of Markov processes), and processes for which the transition probabilities corresponding to any particular control are fixed over time. The Markov process corresponding to any particular control will be a discrete space continuous time process, within a finite number of states. For each state, the probability distribution function for the time until a state transition will be known, although it may depend on the control as well as the state.

In an optimization problem we always have a cost function. In the case at hand, the cost function is the expected time to reach the terminal state. The optimization problem consists of determining a control (i.e. a specific member of the family of Markov processes) for which the cost function is minimal. In the originally described problem we were concerned with making decisions. Specifically, we had to make decisions about how much memory to allocate to each program and which one to run. We think of these decisions as controlling the system. However, in fact, they do not determine precisely how the system will behave in the future, but only the probabilities for how it will behave. Hence a complete set of decision rules is a control in exactly the same sense as the word is used in "controlled Markov process."

When the system is represented as a controlled Markov process, each possible set of decision rules corresponds to a particular control. Or, in other words, each possible set of decision rules maps into a set of state transition probabilities governing a specific Markov process. An optimal control for the controlled Markov process can be interpreted within the original system as a set of decision rules for which the expected completion time is minimal.

We emphasize that the system we have described is stochastic, and that the completion time is a random variable. If the system were actually implemented as described and many experiments carried out with a particular set of decision rules, the observed completion times would vary from run to run. The observed completion times for an optimal control would not always be less than those for a non-optimal control. However the average, over many independent runs, of the finish times for an optimal control would with high probability be smaller than the average for a non-optimal control. In a stochastic system, that is the best you can do.

The state space of a Markov process representing the system we have described can be defined in terms of two components for each program. We call these components the program's execution state and its memory state.

The execution state has four values: Ready, Read Wait, Read in Progress, and Done. In state transition diagrams these will be abbreviated R, RW, RIP, and D. In the Ready state the program is executable; otherwise it is not. Read Wait means that a page fault has occurred but a read has not been started. Read in Progress means that a page required by the program before execution can continue is being read. Done means that the program has reached completion. A program's memory state tells how many pages of that program are in memory, or how many page frames it occupies.

The system state consists of the states of the two programs. For example system state (R,3,RIP,2) means that program 1 is executable and occupies three page frames, while program 2 has a read in progress and

occupies two page frames (including the page being used for the read.) Not all combinations of program states are possible. For example, it is impossible for both programs to be in the Read in Progress state at the same time. Considering only the execution states of each program, the following states and transitions are possible:

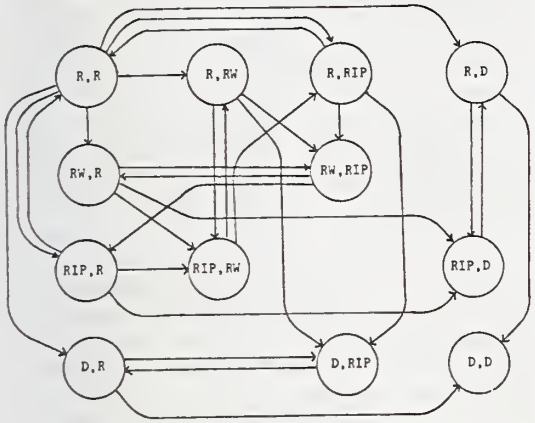


Figure 2.3-1

The complete system state space consists of an entire family of states corresponding to each of the circles above -- one state for each possible allocation of memory to the two programs. It is helpful to visualize state diagrams such as that in Figure 2.3-1 printed on a number of sheets of plexiglass and stacked up. For a fixed memory allocation, the arrows stay on the same sheet. When the memory allocations change, an arrow representing the state transition has to go up or down to another sheet.

Memory allocations can change only at the beginning of a read. If there is unallocated memory when a read is started, one of those page frames will be used. If all page frames are in use, a page must be released from one of the programs to provide space for the page being read. Thus when a read is started for a program, that program's memory allocation may increase by one or may remain the same. The other program's memory allocation may decrease by one or remain the same. These are the only possible one step changes in memory state.

In general, a control alternative for a given state has three components:

1. Which program to run.

2. Whether to start a read upon the next transition.

3. Where to get the page frame for the read, if a read is to be done.

In most states one or more of the three components will not be applicable. The first applies only in (R,R) states. Otherwise there is no choice; if one program is ready it should always be run. The second component applies for (R,R) and (RW,RIP) states. The third component applies only when all memory is in use and each program has a nonzero allocation.

It is sometimes confusing as to whether "a control" applies to the entire Markov process or to a single state. Strictly speaking a control specifies the state transition probabilities for each state, and therefore applies to the process as a whole. However, since a control specifies the transition probabilities for every state, it is meaningful to talk about "a control" for a particular state. Within the context of a discussion of a single state and its successors, a control corresponds to a set of transition probabilities for that state.

A definition of the controlled Markov process representing our system is given as an appendix. For each state, there is a set of one or more control alternatives. For each control alternative, the state transition probabilities are given, along with the identification of the event causing each possible transition. The expected time until the transition is also given for each state under each control.

Now that we have defined a representation of the system within a mathematical framework, we are able to restate the original problem in more precise terms. That is done in the following paragraph.

2.4 Problem Statement

Given the parameters describing each program's fault rate for each amount of memory (the mu's), each program's completion rate (the nu's), the page read rate (lambda), and the number of page frames available (N), determine for each system state the program to execute, whether to start a read on the state transition, and which program to take a page from to make room for the page to be read (each as applicable), so as to minimize the expected overall completion time for the two programs.

3. Solution by Policy Iteration

In this section we describe the solution of the problem just defined by a standard technique of stochastic control theory. Section 3.1 explains the general technique, known as "iteration in policy space." Section 3.2 describes some improvements that can be made to the general technique based on the structure of the particular problem that we are solving. Section 3.3 describes the set of equations that we have to solve, and gives specific examples of several of them. Section 3.4 discusses the problem of convergence. Section 3.4 gives an intuitively meaningful interpretation of the intermediate results of the iteration. This interpretation is used in order to develop a technique for homing in on the solution values from above and below simultaneously. This technique solves the problem of determining when convergence has occurred.

3.1 Iteration in Policy Space

We can in principle determine an optimal control for any controlled Markov process using a standard technique of stochastic control theory known as "iteration in policy space" [KUSH71]. For each control, we can write an equation for the expected finish time at each system state.

(3.1-1)

$$FINc(S) = TRTIMc(S) + \sum_{S'} Pc(S,S') * FINc(S')$$

where FINc(S) = Expected finish time from state S under control C,

TRTIMc(s) = Expected time until state transition in State S under control C,

Pc(S,S') = Probability that state S' follows state S under control C,
and the summation is over all states that can possibly follow state S.

This equation simply says that under any given control the expected finish time from any state is the expected time until the next state transition plus the expected time from that transition until completion. The second expectation is the sum of the expected times until completion for all possible success states weighted by the probabilities that they will be the successor under the control being used. There will be one such equation for each system state, with an equal number of unknowns. This system of linear equations can be solved to determine the expected finish time for every state under the given control. By solving the set of equations for each control we could determine an optimal control.

In fact, it is not necessary to solve the set of equations separately for each control. The expected finish time equations are normally a large, sparse set of equations. Sets of equations of this form are most easily solved by an iterative technique such as the Guass-Seidel iteration. The determination of the optimal control can be combined with the solution of its expected finish times in a single iterative process. We start with an initial guess at the finish time for each state. On each iteration, we compute a new approximation of the optimal expected finish time for each state according to the equation:

(3.1-2)

$$FIN(S) = \min_{C} \left(TRTIMEc(S) + \sum_{S'} Pc(S,S') * FIN(S') \right)$$

where FIN(S) = current approximation to optimal expected finish time from state S,

the MIN is taken over all control alternatives aplicable to state S,

and the summation is over all states that can possibly follow state S.

It can be shown that this process will converge and that the resulting solution values are the expected finish times for an optimal control. [KUSH71, P108]. The optimal decision for each state is also determined by this process. The optimal control alternative for each state is the one resulting in the minimum value in the MIN operation for that state.

3.2 Computation Time Improvement

The above solution technique can be applied to any controlled Markov process. The particular problem that we are solving has some properties that allow us to reduce substantially the amount of computation. For any controls worth considering the total amount of memory in use never decreases until a program finishes. (A control that unnecessarily throws away useful pages is clearly not optimal.) We know therefore that states in which all pages are in use have as successors other such states plus states in which one program is finished. As will be shown in the following section, we can easily compute the expected time to completion for all states in which one program is finished. Hence these values may be assumed to be known before starting the iteration. In the iteration, we can solve for the expected finish times for just those states in which all memory is in use. The values for states in which not all memory is in use, and both programs are still running, need not be considered since the system will never re-enter those states. This reduces the number of states over which we iterate from a multiple of N squared to a multiple of N, where N is the number of pages of memory available.

For states with not all memory in use, the minimum expected finish time and optimal control can be determined by dynamic programming [HOWA60]. When a fault occurs in any of these states and we start a read for the required page, it is always better to allocate an additional page to the program for the page being read in than to use a page that is already in use. The only decisions that must be made in these states are the scheduling decisions. All states in which not all memory is in use can be arranged into a list with the property that all possible successor states for any given state preceed that state in the list. We go down the list computing the minimum expected finish time for each state in turn. The expected finish time for each state with each possible control can be computed one and for all when we reach that state, because we then know the exact value of the minimum expected finish time for all successor states. We can then determine an optimal control and minimum expected finish time for that state by means of a few comparisons. Working back from the set of states with all memory in use to the initial state, we determine the minimum expected finish time from a "cold start" and the set of optimal controls for all states.

3.3. Optimal Control Equations

In this section we show several examples of the equations that are used in the solution technique just discussed. The complete set of equations includes one or more equations for each system state. For each system state there will be one equation for each control applicable to that state. The equation will have the form shown in equation (3.1-1), and will use values of TRTIMc(S) and Pc(S) given for that current state and control in the appendix.

In each state, under any given control, the expected time until the next state transition is the reciprocal of the sum of the rates for the processes that are active in that state under that control. This is a result of the fact that all processes involved in this problem are Poisson processes, and the fact that merging any number of Poisson processes yields another Poisson process with an event rate equal to the sum of the rates of the component processes [ALLE78]. For example in state (R,N1,RIP,N2) there is only one applicable control. Under that control, program 1 has a fault rate of mu(1,N1) and a completion rate of nu(1). At the same time, there is a page read in progress with completion rate lambda. The overall event rate is the sum mu(1,N1) + nu(1) + lambda, and the expected time until a state transition is

(3.3-1)

$$\frac{1}{mu(1,N1) + nu(1) + lambda}$$

The probability that a particular type of event will be the first to occur is its rate divided by the total rate.

Thus we have

$$P[\text{Exit due to a fault}] = \frac{mu(1,N1)}{mu(1,N1) + nu(1) + lambda} \tag{3.3-2}$$

$$P[\text{Exit due to pgm completion}] = \frac{nu(1)}{mu(1,N1) + nu(1) + lambda} \tag{3.3-3}$$

$$P[\text{Exit due to read completion}] = \frac{lambda}{mu(1,N1) + nu(1) + lambda} \tag{3.3-4}$$

For this state, there are no decisions to make. The next state is uniquely determined for each possible event leading to a state transition. The equation for expected finish time is

$$FIN(R,N1,RIP,N2) = \frac{1}{mu(1,N1) + nu(1) + lambda} * \tag{3.3-5}$$

$$[\ 1 + mu(1,N1)*FIN(RW,N1,RIP,N2) + nu(1)*FIN(D,0,RIP,N2) + lambda*FIN(R,N1,R,N2)\ ]$$

In other states the successor state depends on a control. There is an equation of the above form for each possible control, and the acutal expected finish time is the minimum of these values.

For example, from state (RW,N1,RIP,N2) we have the choice of starting a read for program 1 or not doing so upon completion of the read for program 2. If we do start a read, we have the choice of taking a page from program 2 or using a page already allocated to program 1 (assuming all pages are in use.) In this case there is only one process in progress, the page read. Thus the event leading to the state transition is known in advance; it will be a read completion. The expected time until a state transition is the expected read time. The equation for expected finish time from this state is

$$FIN(RW,N1,RIP,N2) = \frac{1}{lambda} + \tag{3.3-6}$$

$$\min \begin{bmatrix} FIN(RIP,N1,R,N2), \\ FIN(RIP,N1+1,R,N2-1), \\ FIN(RW,N1,R,N2) \end{bmatrix}$$

In general the next state depends on both the control and a random event. The expected finish time is the minimum of several values, each of which is a weighted sum reflecting the state transition probabilities under a particular control. For example, in state (R,N1,R,N2) with all memory allocated and each program having a nonzero allocation, we have three choices to make:

1. Which program to run. (1 or 2)

2. Whether to start a read when a fault occurs. (Read, Wait)

3. Which program to take a page from if a read is started. (1 or 2)

The state transition probabilities under the various controls are the following:

Control			Event	Next State	Probability
Run	Read|Wait	Page From			
1	Read	1	Fault Finish	(RIP,N1,R,N2) (D,0,R,N2)	mu(1,N1)/(mu(1,N1) + nu(1)) nu(1) /(mu(1,N1) + nu(1))
1	Read	2	Fault Finish	(RIP,N1+1,R,N2-1) (D,0,R,N2)	mu(1,N1)/(mu(1,N1) + nu(1)) nu(1) /(mu(1,N1) + nu(1))
1	Wait	-	Fault Finish	(RW,N1,R,N2) (D,0,R,N2)	mu(1,N1)/(mu(1,N1) + nu(1)) nu(1) /(mu(1,N1) + nu(1))
2	Read	1	Fault Finish	(R,N1-1,RIP,N2+1) (R,N1,D,0)	mu(2,N2)/(mu(2,N2) + nu(2)) nu(2) /(mu(2,N2) + nu(2))
2	Read	2	Fault Finish	(R,N1,RIP,N2) (R,N1,D,0)	mu(2,N2)/(mu(2,N2) + nu(2)) nu(2) /(mu(2,N2) + nu(2))
2	Wait	-	Fault Finish	(R,N1,RW,N2) (R,N1,D,0)	mu(2,N2)/(mu(2,N2) + nu(2)) nu(2) /(mu(2,N2) + nu(2))

Table 3.3-1

The optimal control equation is

(3.3-7) FIN(R,N1,R,N2) = MIN of:

$$\frac{1}{mu(1,N1) + nu(1)} * (1 + mu(1,N1)*FIN(RIP,N1,R,N2) + nu(1)*FIN(D,0,R,N2))$$

$$\frac{1}{mu(1,N1) + nu(1)} * (1 + mu(1,N1)*FIN(RIP,N1+1,R,N2-1) + nu(1)*FIN(D,0,R,N2))$$

$$\frac{1}{mu(1,N1) + nu(1)} * (1 + mu(1,N1)*FIN(RW,N1,R,N2) + nu(1)*FIN(D,0,R,N2))$$

$$\frac{1}{mu(2,N2) + nu(2)} * (1 + mu(2,N2)*FIN(R,N1-1,RIP,N2+1) + nu(2)*FIN(R,N1,D,0))$$

$$\frac{1}{mu(2,N2) + nu(2)} * (1 + mu(2,N2)*FIN(R,N1,RIP,N2) + nu(2)*FIN(R,N1,D,0))$$

$$\frac{1}{mu(2,N2) + nu(2)} * (1 + mu(2,N2)*FIN(R,N1,RW,N2) + nu(2)*FIN(R,N1,D,0))$$

Given an approximation for all of the expected finish times, we compute an improved approximation from equations of the form shown above. When the iteration has converged, so that the values computed by these equations equal the assumed values used as inputs, each minimization determines the optimal control for the corresponding state.

3.4. Convergence Considerations

The theorem from stochastic control theory [KUSH71,p108] guarantees that iteration in policy space will converge, regardless of the initial approximation, but it does not say anything about how fast it will converge or how to know when you have arrived. In fact, the convergence can be arbitrarily slow. Furthermore, if you test for convergence by comparing the maximum change in expected finish time over all states to some small nonzero threshold, cases can always be found in which the convergence test is met and the current approximations are arbitrarily far from the true solution values. Hence it is important to have some degree of understanding of the particular problem being solved in order to have any confidence in the results.

It is possible to choose the initial approximations in such a way that the successive approximations have a meaningful interpretation within the context of the original problem.

Suppose we choose a control policy for which we can determine the expected finish time from every state as a closed form expression. One such example is the following:

Given a memory allocation (N1,N2), select the program with the smaller fault rate to run first. Throw out all pages belonging to the other program, and let the selected program run, with no overlap, until it completes. Give the running program an additional page on each fault until it occupies all of memory. Upon completion of the first program, let the other program run to completion, starting with no pages in memory and giving it an additional page on each fault until it occupies all of memory.

Now this may not be a particularly good policy, but it does have the virtue that we can compute the expected finish time exactly for every state. It is also a policy that can be implemented for all possible amounts of memory and all possible sets of program parameters. We shall take the expected finish times under this policy as our initial approximation for each state, and refer to this policy as the "default policy" as it will be the policy that we fall back on after a planned multistep optimal policy.

On each iteration we compute the best control decision and the corresponding expected finish time, assuming the expected finish time for each possible successor state is the value that we were given as an input. For the initial approximation just described, the results of the first iteration will be the acutal exact values for the expected finish time if we made the (stochastically) best decision on the first state transition and then followed the default policy from that time on. The results of the second iteration will be the actual expected values if we made optimal decisions for the first two state transitions and then followed the default policy. And so forth. The results of the Kth iteration will be the actual expected values for finish times given that we make K optimal decisions and then follow the default policy. We can now see intuitively that the iteration has to converge when (and only when) K becomes large enough that the probability of going through K state transitions before both programs finish is very small.

Having seen how the convergence of the iteration can reflect actual results of K-step optimal decisions, we shall now apply that result in a technique to determine when convergence within a given threshold has occurred. This will be done by determining separate initial approximations for which the successive approximations are guaranteed to converge to the actual values from above and from below. When the values obtained from the two different initial approximations are all within the specified value of each other, we can stop the iteration.

If we guessed the optimal expected finish times and used them as our starting values, the results of the Kth iteration would be exactly the same values. The expected finish times for the case in which we follow an optimal policy for K steps, given that we switch to a specific nonoptimal policy after K steps, can only be greater than the expected finish times for the case in which we follow an optimal policy until completion. Hence if we start with the exact values for the default policy, the values produced on each iteration will be greater than the actual solution values. They will, of course, eventually come arbitrarily close to the solution values. In a similar vein, if we switched to a ficticious "better than optimal" policy after K steps, following the optimal policy given that fact for the first K steps, the resulting values could only be better than the solution values. Thus if we start with initial approximations all known to be smaller than the optimal expected finish times, the iteration will converge to the solution from below. We therefore have a technique for computing a sequence of upper and lower bounds for the solution values. Each sequence is guarenteed to converge to the solution. If we desire the solution within a certain tolerance, we can be sure of reaching it by continuing the iteration until the upper and lower bounds for each value are within that distance of each other.

4. Examples and Observations

This section gives several examples of solutions for optimal control for specific parameter values. Each example was choosen to illustrate a particular point, and the results were computed using the solution technique described in Section 3.

4.1. Example 1

In some cases it is better not to start a read needed for one program even though the other program is runnable. Thus taking all possible overlap is not always an optimal policy. The following example is one such case.

Program 1		Program 2	
Memory	Fault Rate	Memory	Fault Rate
1	100	1	100
2	100	2	100
3	0	3	100
Expected Exectuion Time = 10		Expected Execution Time = 1	

There are five pages of memory available for the programs. The expected read time is 50 milliseconds.

The minimum expected finish time for these two programs running in five pages is 12.852 seconds. The optimal control calls for reading in the three pages for Program 1 as quickly as possible, and deferring the first read for Program 2 until after these three pages are in memory. Once Program 1's pages are in memory, the optimal control runs Program 2 whenever it is runnable, and runs Program 1 during the page reads for Program 2.

Note that once Program 1's three pages are in memory, it will never fault. Thus all of Program 2's reads will be fully overlapped (unless it finishes prior to Program 1.) Reading a page for Program 2 at any earlier time will not improve anything; it will be an additional read that is almost completely nonoverlapped, and we will still have to do the same essentially nonoverlapped reads for Program 1. The minimum expected finish time under a policy that starts a read as soon as possible is 12.881 seconds, or 29 milliseconds more than the unrestricted optimum.

4.2. Example 2

In the preceeding example we saw the benefit of running the program

that is likely to fault when both are runnable, and overlapping its page reads with execution of the other program. One might conjecture that it is always best the run the program with the higher fault rate when both are runnable. The next example shows that this is not the case.

Program 1		Program 2	
Memory	Fault Rate	Memory	Fault Rate
1	100	1	100
2	100	2	10
3			
Expected Exectuion Time = 1		Expected Execution Time = 100	

There are four pages of memory available for the programs. The expected read time is 50 milliseconds.

The minimum expected finish time for these two programs running in four pages is 152.247 seconds. The optimal control calls for running Program 2, the one with the lower fault rate, when both programs have their pages in memory and both are runnable. The reason for this is the difference in expected execution times. Because Program 2 will run for so much longer than Program 1, it will have far more page faults before finishing than Program 2 will, even though Program 2's fault rate is higher. By running Program 2 when both are runnable we are able to overlap more faults than if we ran Program 1 when both are runnable.

4.3. Example 3

If there is enough memory for each program individually to run with a zero fault rate, but not enough for both to run with zero fault rates, then for sufficiently large expected execution times it is always better to run the programs one at a time. The reason for this is that the cost (number of nonoverlapped reads) of getting all of a program's pages into memory is fixed, and independent of run time. The cost of a steady state nonzero fault rate for both programs increases linearly with the expected time until the first completion. Thus there has to be a value of the expected run times such that the overlap policy is inferior to a single thread policy. The following example illustrates this point.

Program 1		Program 2	
Memory	Fault Rate	Memory	Fault Rate
1	100	1	100
2	5	2	5
3	5	3	5
4	0	4	0
Expected Exectuion Time = 0.1		Expected Execution Time = 0.1	

There are four pages of memory available for the programs. The expected read time is 50 milliseconds.

For this example the minimum expected run time is 0.324 seconds. The optimal allocation of memory is to give two pages to each program, and the optimal control will force the system into this configuration unless one of the programs finishes before it is reached. Note that even though we get a high degree of overlap with this division, there is a nonzero probability of getting into a state in which the processor is idle.

If we increase the expected run time for each program to 10 seconds, the optimal division of memory is to give all four pages to one of the programs. In this case, the programs will be running long enough that it is better to give up the potential overlap in order to reduce the fault rate. The optimal control still attempts to overlap reads with execution during the startup transient. However, once one of the programs has two pages in memory, the best policy is to let it take over all of memory.

5. Conclusions

Optimal control of a virtual memory operating system is a complex problem. We have looked at the actual problem and derived a much simpler related problem that is mathematically tractable. We have used that problem as a vehicle for exploring solution techniques and gaining a better understanding of the problem space. A standard technique of stochastic control theory, known as interation in policy space, was found to provide a solution to the problem of allocating memory and scheduling the processor in this simplified system model. Several improvements were made to the basic technique,

taking advantage of the special properties of this particular problem to improve the computation time and to guarantee convergence within specified limits. The solution techniques may be extensible to more realistic problems, but that remains to be seen. The mathematical framework, in which we represent the multiprogrammed virtual memory system as a controlled Markov process, seems to be a useful way of looking at the problem.

References

[ALLE78] A.O. Allen, Probability, Statistics, and Queueing Theory, With Computer Science Applications, Academic Press, New York 1978.

[ARNO75] C.R. Arnold, "A control theoretic approach to memory management," Proc. of Ninth Asilmor Conf. on Circuits, Systems and Computers, Pacific Grove, CA, Nov. 1975.

[ASHA72] R. Ashany, "Application of control theory techniques to performance analysis of computer systems," Proc. of Sixth Asilmor Conf. on Circuits, Systems, and Computers, Pacific Grove, CA, Nov. 1972, pp. 90-101.

[BOXJ76] G.E.P. Box and G.M. Jenkins, Time Series Analysis Forecasting and Control," Holden-Day, San Francisco, 1976.

[COFF73] E.G. Coffman Jr. and P.J. Denning, Operating System Theory, Prentice-Hall, Englewood Cliffs, NJ 1973.

[COFF76] E.G. Coffman Jr., Editor, Computer and Job Shop Scheduling Theory, John Wiley and Sons, 1976.

[FERR78] D. Ferrari, Computer Systems Performance Evaluation, Prentice-Hall, Englewood Cliffs, NJ, 1978.

[HOWA60] R.A. Howard, Dynamic Programming and Markov Processes, The M.I.T. Press, Cambridge, MA, 1960.

[JAIN78] R.K. Jain, Control-Theoretic Formulation of Operating Systems Resource Management Problems, Harvard PhD Thesis, May 1978. Also published in "Outstanding Dissertations in Computer Science Series," Garland Publishing, New York, 1979.

[KLEI75] L. Kleinrock, Queueing Systems, Volume 2: Computer Applications, Wiley-Interscience, 1976.

[KUSH71] H. Kushner, Introduction to Stochastic Control, Holt, Rinehart, and Winston, New York, 1971.

[LEW76] A. Lew, "Optimal control of demand-paging systems," Information Sciences, Vol. 10, Nr. 4 (1976) pp.319-330.

[SPIR77] J.R. Spirn, Program Behavior: Models and Measurement, Elsevier North-Holland, New York, 1977.

[WILK73] M.V. Wilkes, "The dynamics of paging," Computer Journal, Vol. 16, Nr. 1, (Feb. 1973) pp.4-9.

APPENDIX

The following tables give a detailed definition of the controlled Markov process described in the text. See Section 2.1 through 2.3 for definitions of the symbols.

States with Not All Memory in Use

Current State	Control: Run	Control: Read / Wait	Expected Time Until State Transition	Event	Probability	Next State
R,N1,R,N2	1	Read	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1)}$	RIP,N1+1,R,N2
				Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,R,N2
	1	Wait	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1)}$	RW,N1,R,N2
				Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,R,N2
	2	Read	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	R,N1,RIP,N2+1
				Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	R,N1,D,0
	2	Wait	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	R,N1,RW,N2
				Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	R,N1,D,0

R,N1,RIP,N2	1	----	$\frac{1}{mu(1,N1) + nu(1) + lambda}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1) + lambda}$	RW,N1,RIP,N2
				Finish	$\frac{nu(1)}{mu(1,N1) + nu(1) + lambda}$	D,0,RIP,N2
				IO Done	$\frac{lambda}{mu(1,N1) + nu(1) + lambda}$	R,N1,R,N2
RW,N1,RIP,N2	-	Read	$\frac{1}{lambda}$	IO Done	1	RIP,N1+1,R,N2
	-	Wait	$\frac{1}{lambda}$	IO Done	1	RW,N1,R,N2
RW,N1,R,N2	-	Read	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	RW,N1,RIP,N2+1
				Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	RIP,N1+1,D,0
R,N1,D,0	1	Read	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,n1)}{mu(1,N1) + nu(1)}$	RIP,N1+1,D,0
				Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,D,0
RIP,N1,D,0	-	----	$\frac{1}{lambda}$	IO Done	1	R,N1,D,0

States with All Memory in Use

Current State	Control Run	Control Read/Wait	Control Page From	Expected Time Until State Transition	Event	Probability	Next State
R,N1,R,N2	1	Read	1	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1)}$	RIP,N1,R,N2
					Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,R,N2
	1	Read	2	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,n1)}{mu(1,N1) + nu(1)}$	RIP,N1+1,R,N2-1
					Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,R,N2
	1	Wait	-	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1)}$	RW,N1,R,N2
					Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,R,N2
	2	Read	1	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	R,N1-1,RIP,N2+1
					Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	R,N1,D,0

R,N1,R,N2 (cont)	2 Read 2	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	R,N1,RIP,N2
			Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	R,N1,D,0
	2 Wait -	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	R,N1,RW,N2
			Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	R,N1,D,0
R,N1,RIP,N2	1 ---- -	$\frac{1}{mu(1,N1) + nu(1) + lambda}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1) + lambda}$	RW,N1,RIP,N2
			Finish	$\frac{nu(1)}{mu(1,N1) + nu(1) + lambda}$	D,0,RIP,N2
			IO Done	$\frac{lambda}{mu(1,N1) + nu(1) + lambda}$	R,N1,R,N2
RW,N1,RIP,N2	- Read 1	$\frac{1}{lambda}$	IO Done	1	RIP,N1,R,N2
	- Read 2	$\frac{1}{lambda}$	IO Done	1	RIP,N1+1,R,N2-1
	- Wait -	$\frac{1}{lambda}$	IO Done	1	RW,N1,R,N2

States with All Memory in Use

Current State	Control Run	Control Read/Wait	Control Page From	Expected Time Until State Transition	Event	Probability	Next State
RW,N1,R,N2	2	Read	1	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	RW,N1-1,RIP,N2+1
					Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	RIP,N1+1,D,0
	2	Read	2	$\frac{1}{mu(2,N2) + nu(2)}$	Fault	$\frac{mu(2,N2)}{mu(2,N2) + nu(2)}$	RW,N1,RIP,N2
					Finish	$\frac{nu(2)}{mu(2,N2) + nu(2)}$	RIP,N1+1,D,0
R,N1,D,0	1	Read	1	$\frac{1}{mu(1,N1) + nu(1)}$	Fault	$\frac{mu(1,N1)}{mu(1,N1) + nu(1)}$	RIP,N1,D,0
					Finish	$\frac{nu(1)}{mu(1,N1) + nu(1)}$	D,0,D,0
RIP,N1,D,0	-	----	-	$\frac{1}{lambda}$	IO Done	1	R,N1,D,0

Notes

1. Definitions for states not shown in the tables can be obtained by symmetry from those that are shown.

2. Certain definitions will produce invalid memory amounts for particular values of N1 or N2. For example, if N1 is equal to the amount of memory available, state (RIP,N1+1,R,N2-1) is not a valid state. It should be understood that controls resulting in invalid states for when one program has all of memory are not possible for those values of N1 and N2.

An Efficient Capacity Assignment Algorithm for Computer Communication Networks with a Tree Topology

Roberta Klibaner
The College of Staten Island
Dept. of Computer Science
715 Ocean Terrace
Staten Island, N.Y. 10301

Chaim Ziegler
Brooklyn College
Dept. of Computer & Info. Science
Bedford Ave. & Ave. H
Brooklyn, N.Y. 11210

An efficient capacity assignment algorithm for a centralized computer communication network having a binary tree topology is designed. The queueing model used consists of Poisson inputs, constant nodal service times, and fixed length packets. Using more exact and realistic analysis methods for the actual queueing time and/or system time that a packet experiences at each merger node, we have been able to design a more efficient capacity assignment algorithm. The results using this algorithm are compared with an existing algorithm to show the improvement that would occur in the design of a centralized computer communication network.

1. INTRODUCTION:

The effectiveness of a computer communication network can be measured by cost-performance parameters. More specifically, we are concerned with the cost of the various links used in the network, and performance, as measured by the mean queueing time or mean system time experienced by users.

Capacity assignment algorithms seek to efficiently assign capacities to the various links in a network while minimizing both delay and cost. We have developed a capacity assignment algorithm for a fixed length, packet switched, centralized, computer communication network having a binary, tree-like topology, as shown in Figure 1.

Fixed length packets are permitted to enter the network from external sources at all nodes, except node 1, the central CPU. Nodes 4-7, of Figure 1, are the leaves of this network, and are known as external nodes. Nodes 2 and 3, have multiple input links and are an example of merger nodes.

A finite set of line capacities, each with a fixed cost per unit link distance, is available from which the user may choose. External input into the network is assumed to be Poisson in nature. Every packet at node i, when node i is assigned capacity k, requires and receives a constant service time of s_{ik} (sec./packet), i=2,3,..N.

An algorithm was developed to produce a cost/performance table and graph which

---

This work was partially supported by the National Science Foundation under grant ECS-8105963 and by PSC-CUNY Research Award 13655.

would aid in the actual selection of capacities to be used in the development and implementation of a centralized computer communication network. For a given cost we have determined the unique capacity assignment to be made in order to minimize the maximum delay over the entire network. Equivalently, for a given maximum acceptable delay, we can determine the minimum cost network.

Specific examples are then used to compare this algorithm with results produced by an existing algorithm. In all cases our results are shown to be tighter approximations to the actual queueing time and system time, resulting in improved design.

2. NETWORK MODEL:

The network model employed in the development of our algorithm is that of a centralized computer network represented by a binary tree-like topology, as in Figure 1. This structure can be thought of as a series of processors (the nodes), and feeder lines (the links). External input to the system may enter at any point, but not at the central processor, the CPU.

The nodes are numbered sequentially from 1 to N, where N is the number of nodes in the network. Node 1, the root, is the CPU. Packets (customers) are permitted to enter the network at an external node or a merger node. A packet is processed at its entry node and then recursively transmitted to its father (the node next closest to the CPU) until it reaches the CPU where it is subsumed. The network which we modeled can be complete, as in Figure 1, (every node has a left and right son) or unbalanced (some nodes have missing sons), as in Figure 2.

Fixed length packets, of length $1/\mu$ (bits/packet), enter the network randomly. The arrival process into node i is assumed to be Poisson with a mean arrival rate of λ_i (packets/sec.), $\lambda_i \geq 0$. A finite number of capacities (C_j, $j=1,2,\ldots k$), each with a fixed cost per unit link distance are available for selection on each link of the network. A given link may be assigned any of these discrete capacities, if this assignment leads to an optimal capacity assignment for the entire network.

With fixed length packets, we assume that all packets entering a given node, when that node is currently assigned capacity C_j, would require the same amount of service to be processed. Thus, a packet at node i requires and receives a constant service time of s_{ij} (sec./packet) given by

$$s_{ij} = 1/\mu C_j \qquad (1)$$

$i=2,3,\ldots N;\ j=1,2,\ldots k$

The distance between nodes, D_i, is the link distance between node i and its father. Thus, link cost is a function of the capacity selected for a given link and the length of that link, and can be expressed as

$$\text{Cost(of Link i)} = \text{COST(of selecting capacity j)} * D_i \qquad (2)$$

The queue at each node is assumed to have infinite storage available. In addition, a first-come, first-served (FCFS) queueing discipline exists at each node.

The capacity assignment problem which we have addressed seeks to minimize the maximum delay from anywhere in the network to the CPU while keeping cost at a minimum. We have developed an algorithm for producing a network delay/cost array which shows the maximum system or queueing delay associated with a given cost. The delay/cost array which is produced indicates what capacity must be selected for each link to produce this network design.

3. AN EXISTING ALGORITHM:

The algorithm first outlined by Frank, Frisch, Chou, and Van Slyke[1] enabled one to select link capacities for a specified tree structure which minimized the delay while selecting only cost effective capacity assignments. In this algorithm, an initial DELAY/COST characteristic table is developed for each node i in the network, $i=2,3,\ldots N$. COST is determined by equation (2). DELAY is the total mean system time, $T^{(i)}$, experienced by a packet at node i. A delay-cost array is created from the combination of these two lists.

From these initial delay-cost arrays, a series of parallel and serial merges starting at the external entry nodes are undertaken. Each merge produces a new delay-cost array which can be considered an equivalent link replacing the arrays of the merged nodes. Equivalent link arrays and the initial arrays are continuously merged until a final, network delay-cost array is produced.

The application of this algorithm to a centralized, computer communication network has been studied by Schwartz [4] who applied the algorithm to exponentially distributed packets of mean

length $1/\mu$ (bits/packet), following a Poisson arrival process. Hence, the service time at each node was also exponential. The independence assumption, introduced by Kleinrock [2] was used in the analysis of the network; each node was treated as an independent M/M/1 queueing system.

We have expanded the application of this algorithm to a centralized network servicing fixed length packets with constant service time at each node. Thus, for fixed length packets, the mean system time at node i, $T^{(i)}$, arising from the selection of capacity C_j can be determined from

$$T^{(i)} = (\rho_{ij}s_{ij})/[2(1-\rho_{ij})] + s_{ij} \qquad (3)$$

where

$$\rho_{ij} = \lambda_i s_{ij} = \text{utilization of node i} \qquad (4)$$

and λ_i = arrival rate into node i from all sources.

To insure maximum flexibility to a user, we have introduced a capacity assignment process based either on mean system time/cost or mean queueing time/cost considerations. We note that the mean queueing time, $W^{(i)}$, at node i, arising from the selection of capacity C_j, is given by

$$W^{(i)} = (\rho_{ij}s_{ij})/[2(1-\rho_{ij})] \qquad (5)$$

Using (3) or (5), as appropriate, one can then apply the Frank, et. al. algorithm, treating each node as an independent M/D/1 queueing system, to produce the set of possible network designs (a detailed M/M/1 example can be found in Schwartz [4]).

The problem with this algorithm, we believe, lies in use of the independence assumption. In the next section we will remove the independence assumption in an attempt to get a more realistic picture of the actual delay experienced by a packet.

4. THE NEW ALGORITHM:

The existing algorithm, as we have applied it to the 15-node network of Figure 3, visualizes each node as an independent M/D/1 queueing system. The arrival process into merger nodes is assumed to be Poisson in nature, by virtue of the independence assumption developed by Kleinrock [2]. Implicit in this assumption is the assumption that the capacity assigned to one link in no way affects the actual delay over other links. It has been shown, by Rubin [3], Ziegler and Schilling [5,6] and others, that at merger nodes, this yields at best a loose upper bound to the actual delays within the system.

We have removed the independence assumption at merger nodes and will now outline our new recursive algorithm and the analysis involved in its development.

Capacity Assignment Algorithm (Recursive)

1. Traverse the left subtree in postorder
2. Calculate delay at leaf
3. Traverse the right subtree in postorder
4. Calculate delay at leaf
5. Visit the root (if not central CPU)
   a. Perform parallel merge as outlined in [1]
   b. Calculate delay at root
   c. Perform serial merge as outlined in [1]

5'. Visit the root (central CPU)
   a. Perform final parallel merge

Note the major difference between this algorithm and the Frank and Frisch algorithm involves the calculation of the delay at the root. We cannot calculate the delay at the root until the left and right subtrees have been assigned capacities that we employ in the computation of the delay. The Frank and Frisch algorithm computes all the delay/cost arrays prior to the parallel and serial merges.

Traffic into external nodes, nodes 8-15, only arrives from external sources, thus, these nodes, the leaves of the tree, are indeed independent M/D/1 queueing systems and formulas (3) and (5) apply.

The merger nodes, nodes 2-7, do not conform to this analysis because of the non-Poisson nature of their input traffic. Input at a merger node is the output of previous nodes and may also include traffic from external sources. We will now analyze the subnetwork consisting of nodes 4, 8, and 9. We must decompose this subnetwork into the following subcases:

Subcase 1: $s_4 \geq s_8$ and $s_4 \geq s_9$

For the three node network described, exact results have been derived in [5]. At node 4, the steady state mean queueing time, $W^{(4)}$, is given by

$$W^{(4)} = \frac{\rho_{44}s_4}{2(1-\rho_{44})} - \frac{\rho_{88}s_8}{2(1-\rho_{88})}\left(\frac{\lambda_8}{\Lambda_4}\right) - \frac{\rho_{99}s_9}{2(1-\rho_{99})}\left(\frac{\lambda_9}{\Lambda_4}\right) \quad (6)$$

where

$$\rho_{44} = \Lambda_4 s_4; \quad \Lambda_4 = \lambda_4 + \lambda_8 + \lambda_9$$

and, at node 4, the steady state mean system time, $T^{(4)}$, is given by

$$T^{(4)} = W^{(4)} + s_4 \quad (7)$$

The delay at node 4, given by (6) or (7), arising from the removal of the independence assumption is always less than the steady state values given by (3) or (5) when independence is assumed.

Subcase 2: $s_4 \leq .5 \min(s_8, s_9)$

If no external traffic enters at node 4, the work in [6] yields exact results. The steady state mean queueing time, $W^{(4)}$, is given by

$$W^{(4)} = (\rho_{84}\rho_{94})/(\lambda_8 + \lambda_9) \quad (8)$$

and the steady state mean system time, $T^{(4)}$ is once again given by (7).

If external traffic is permitted to enter at the merger node, node 4, these results are no longer applicable and the approximation technique introduced in Subcase 3 must be employed.

Subcase 3: $s_4 \leq s_i$ for at least one value of i, i=8,9

In this region exact results have not yet been obtained. We employ an approximation technique developed in [7]. We approximate the effect of the service time of the slower feeder node(s) on the merger node by setting its service time equal to the service time of the merger node; i.e., by replacing each $s_i > s_4$ by s_4, i=8,9. This lower bounds the delay at the entry node, thus upper bounding the delay at the merger node. The upper bound that this approximation yields is tighter than the upper bound obtained using the independence assumption.

For example, if $s_4 > s_8$ and $s_4 < s_9$, we set $s_9 = s_4$. The steady state mean queueing time, $W^{(4)}$, is given by

$$W^{(4)} = \frac{\rho_{44}s_4}{2(1-\rho_{44})} - \frac{\rho_{88}s_8}{2(1-\rho_{88})}\left(\frac{\lambda_8}{\Lambda_4}\right) - \frac{\rho_{94}s_4}{2(1-\rho_{94})}\left(\frac{\lambda_9}{\Lambda_4}\right) \quad (9)$$

This analysis also applies to the subnetworks rooted at nodes 5, 6 and 7. Traveling up the tree, we now consider the subnetworks rooted at 2 and 3.

We will now analyze the subnetwork rooted at node 2, a network consisting of nodes 2, 4, 5, 8, 9, 10, and 11. At this point we utilize a simplifying assumption. We assume that the subnetworks rooted at nodes 4 and 5 can be viewed as an equivalent network consisting of a single node with Poisson input having a mean arrival rate Λ_i (packets/sec), i=4,5 and constant service time s_{ij}, i=4,5; j=1,2,3,...K. This assumption yields, for all cases, at worst a lower bound on the delay that a fixed length packet would experience traversing the network through that node, and at best exact results. As an example consider the following cases:

Case 1: $s_2 \geq s_4$, $s_2 \geq s_8$, $s_4 \geq s_8$, $s_4 \geq s_9$, $s_5 \geq s_{10}$, $s_5 \geq s_{11}$

The approximation technique yields a steady state mean queueing time, $W^{(2)}$, at node 2, given by

$$W^{(2)} = \frac{\rho_{22}s_2}{2(1-\rho_{22})} - \frac{\rho_{44}s_4}{2(1-\rho_{44})}\left(\frac{\Lambda_4}{\Lambda_2}\right) - \frac{\rho_{55}s_5}{2(1-\rho_{55})}\left(\frac{\Lambda_5}{\Lambda_2}\right) \quad (10)$$

while the actual queueing time at node 2 would be

$$W^{(a)} = \frac{\rho_{22}s_2}{2(1-\rho_{22})} - \left(\left(\frac{\rho_{44}s_4}{2(1-\rho_{44})} - \frac{\rho_{88}s_8}{2(1-\rho_{88})}\frac{\lambda_8}{\lambda_4} - \frac{\rho_{99}s_9}{2(1-\rho_{99})}\frac{\lambda_9}{\lambda_4}\right)\frac{\lambda_4}{\lambda_2} + \left(\frac{\rho_{55}s_5}{2(1-\rho_{55})} - \frac{\rho_{10,10}s_{10}}{2(1-\rho_{10,10})}\frac{\lambda_{10}}{\lambda_5}\right) - \left(\frac{\rho_{11,11}s_{11}}{2(1-\rho_{11,11})}\frac{\lambda_{11}}{\lambda_5}\right)\frac{\lambda_5}{\lambda_2} + \sum_{i=8}^{11}\frac{\rho_{ii}s_i}{2(1-\rho_{ii})}\frac{\lambda_i}{\lambda_2}\right\}$$

$$= \frac{\rho_{22}s_2}{2(1-\rho_{22})} - \frac{\rho_{44}s_4}{2(1-\rho_{44})}\frac{\lambda_4}{\lambda_2} - \frac{\rho_{55}s_5}{2(1-\rho_{55})}\frac{\lambda_5}{\lambda_2} \qquad (11)$$

Note, for this case our approximation yields an exact result.

Case 2: $s_2=\max(s_2,s_4,s_5,s_8,s_9,s_{10},s_{11})$, $s_4 \geq .5\min(s_8,s_9)$, $s_5 \geq .5\min(s_{10},s_{11})$

Our approximation yields (10), while the actual queueing time at node 2 is

$$W^{(a)} = \frac{\rho_{22}s_2}{2(1-\rho_{22})} - \frac{\rho_{84}\rho_{94}}{\lambda_8+\lambda_9}\frac{\lambda_4}{\lambda_2} - \frac{\rho_{10,5}\rho_{11,5}}{\lambda_{10}+\lambda_{11}}\frac{\lambda_5}{\lambda_2} - \sum_{i=8}^{11}\frac{\rho_{ii}s_i}{2(1-\rho_{ii})}\frac{\lambda_i}{\lambda_2}$$

$$= \frac{\rho_{22}s_2}{2(1-\rho_{22})} - \frac{\rho_{84}\rho_{94}}{\lambda_2} - \frac{\rho_{10,5}\rho_{11,5}}{\lambda_2} - \sum_{i=8}^{11}\frac{\rho_{ii}s_i}{2(1-\rho_{ii})}\frac{\lambda_i}{\lambda_2} \qquad (12)$$

where it is easily shown that (12) must always be less than (10). Our approximation is an upper bound to the actual delay.

Thus, by viewing the subnetworks that feed a merger node as an equivalent network consisting of a single node we have been able to reduce the multi-node subnetworks at a merger node to a 3 node equivalent network as seen in Figure 4. These three node networks can then be analyzed as previously described, in all cases yielding, at worst, an upper bound to the delay at a merger node.

At the root, the central CPU, a final parallel merge of the left and right subnetworks is performed yielding the capacity assignments available for this network.

5. NUMERICAL EXAMPLES:

The algorithm just described has been coded in PL/I and run on the CUNY/UCC Computer System, New York City.

The numerical results in this section refer to the design of Example 1, a 15-node network shown in Figure 3; Example 2, a 10-node network shown in Figure 5; and Example 3, a 15-node network shown in Figure 6. Fixed length packets of length 120 bits/packet were used as a parameter in all network designs. Link distances and external input rates are indicated within the figures.

Table 1 is a list of the available link capacities and monthly charges assumed for the networks of Figures 3 and 6. For Figure 3, Table 2 is the optimal system time/cost array produced by the new algorithm and Table 3 is the system time/cost array produced using the old algorithm. A graph comparing the results is shown in Table 4.

The available link capacity table assigns to each capacity a number, for example, 450 bps has been assigned the number 1, etc. A line in the delay/cost array may be interpreted as follows: the system time (or queueing time) is the amount of delay that must be tolerated for the given cost. The numbers which appear under the various links indicate what capacity number has been selected at each link to produce the delay/cost entry shown. The graph plots the delay vs. the cost produced by the two algorithms.

Note that of the 53 possible capacity assignments produced by our algorithm, only 12 are identical to those produced by the old algorithm. We show, with the set of capacities given, for a given cost, it is possible to design a network that is at least 10% faster than any network designed by the old algorithm. When the slowest capacity is selected for each link, there is a dramatic difference in system time.

Table 5 is the available link capacities and monthly charges assumed for the unbalanced network of Figure 5. Tables 6, 7, and 8 are the outputs produced by our program for this network.

Once again, note the different design choices produced by the two algorithms and improved designs that the new procedure yields. For example, a $301 monthly tariff using the new algorithm, will produce a network design that is 15% more efficient.

For users who are designing a system based upon queueing time, Tables 9, 10, and 11 are the output produced by our program for the network in Figure 6. Note, for all network designs based upon queueing time our algorithm will produce a system that is *at least 50%* faster for a given cost. In addition, note the cost savings for a given value of queueing time (e.g., 0.097).

6. CONCLUSION:

In this paper, we have introduced a new algorithm for use in the capacity assignment problem for centralized computer networks. We found the new algorithm to be of definite value as it consistently produced more efficient network designs than previous methods. It should be viewed as a practical instrument for the selection of capacities in a centralized computer communications network.

We recognize that in only limited cases are the results produced using this algorithm exact . Nevertheless, in all instances we produce upper bound results. We are presently working on tightening these bounds through an improvement of the delay calculation at point 5b of the new algorithm. The procedure will involve removal of the simplifying assumption mentioned above. The results of this effort will be presented at a future time.

REFERENCES:

[1] H. Frank, I.T. Frisch, R. Van Slyke, and W.S. Chou, "Optimal Design of Centralized Computer Networks," Networks 1, No. 1, pp. 43-57, John Wiley & Sons, New York, 1971.

[2] L. Kleinrock, Communication Nets, McGraw Hill, New York, 1964, Dover, New York, 1972.

[3] I. Rubin, "Tandem Queues with Constant Channel Service Times and Group Arrivals," UCLA Tech. Rep. UCLA-ENG-7417, March 1974.

[4] M. Schwartz, Computer Communication Network Design and Analysis, Prentice-Hall Englewood Cliffs, New Jersey, 1977.

[5] C. Ziegler and D.L. Schilling, "Delay Decomposition at a Single Server Queue with Constant Service Time and Multiple Inputs," IEEE Trans. Commun., Vol. COM-26, pp. 290-295, Feb. 1978.

[6] C. Ziegler and D.L. Schilling, "Waiting Time at Fast Merger Nodes," Conference Record of ICC '80, pp. 23.2-1-23.2-6, June 1980.

[7] C. Ziegler, Analytic Methods for Delay Analysis at Packet-Switched Merger and Separation Nodes, Ph.D. Dissertation, C.U.N.Y., 1978.

ACKNOWLEDGEMENTS:

The authors wish to acknowledge the assistance of Mr. Joseph Arfin and Miss Amalia Kletzky in the preparation of this paper.

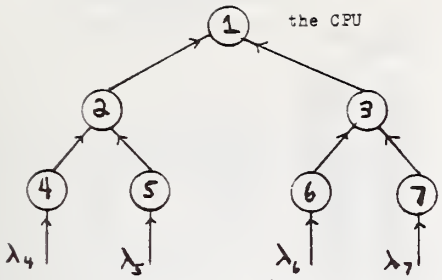


Figure 1
Network having a Binary Tree Topology
with Input at External Nodes Only

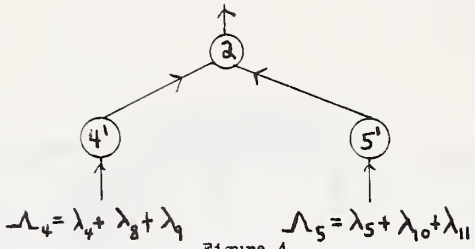


Figure 4
Three Node Equivalent Network
Produced by Simplifying Assumption

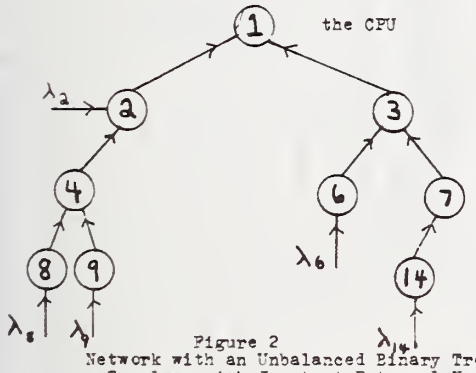


Figure 2
Network with an Unbalanced Binary Tree
Topology with Input at External Nodes
and Merger Nodes

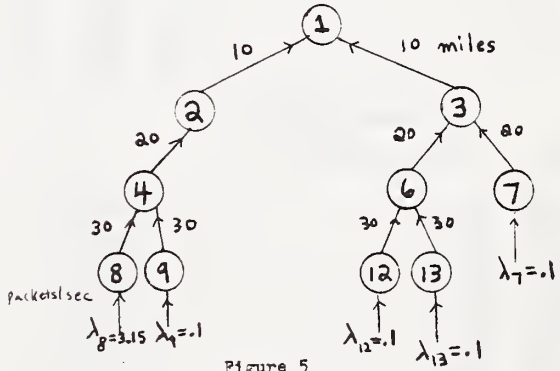


Figure 5
Ten Node Network used in Example 2

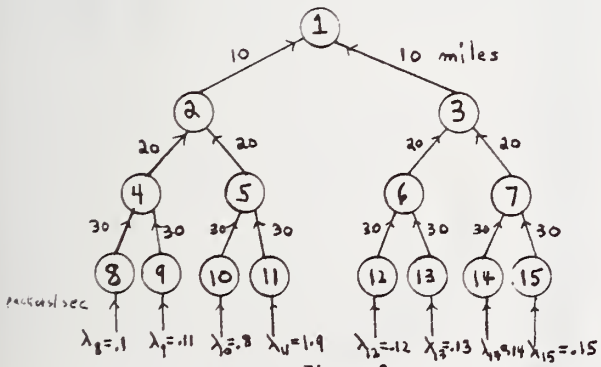


Figure 3
Fifteen Node Network used in Example 1

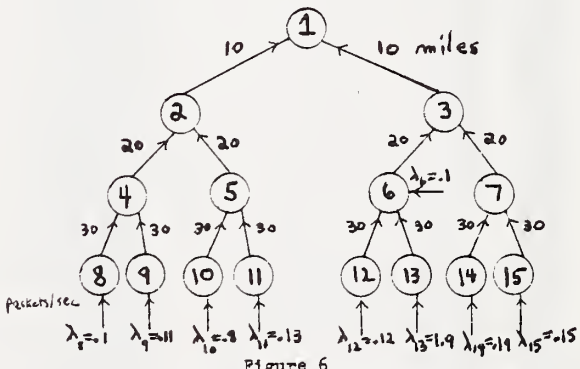


Figure 6
Fifteen Node Network used in Example 2

POSSIBLE LINK CAPACITIES

CAPACITY #	CAPACITY(BPS)	COST/MONTH/MILE	SERVICE TIME /CAPACITY
1	450	1.40	0.2666000
2	600	2.70	0.2000000
3	750	3.88	0.1690000
4	900	5.35	0.1332999

Table 1

SYSTEM-TIME/COST CHARACTERISTIC TABLE
NEW ALGORITHM

CHOICE #	SYSTEM TIME(sec)	COST/MW	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.449044	1554.40	4	4	4	4	4	4	3	3	4	4	3	3	3	3
2	0.459364	1525.00	4	4	4	4	3	4	3	3	4	4	3	3	3	3
3	0.459924	1495.60	4	4	4	4	3	3	3	3	4	4	3	3	3	3
4	0.463472	1466.20	4	4	3	4	3	3	3	3	4	4	3	3	3	3
5	0.464882	1422.10	4	4	3	4	3	3	3	3	3	4	3	3	3	3
6	0.472892	1416.10	4	4	3	4	4	3	3	3	4	2	3	3	3	3
7	0.473102	1380.70	4	4	3	4	4	3	3	3	3	4	2	2	3	3
8	0.473512	1374.70	4	4	3	4	4	4	3	3	3	4	2	2	2	3
9	0.473724	1339.30	4	4	3	4	4	4	3	3	3	4	2	2	2	2
10	0.476940	1333.30	4	4	4	4	4	4	2	3	3	4	2	2	2	2
11	0.477148	1297.90	4	4	4	4	4	4	2	2	3	4	2	2	2	2
12	0.488081	1253.80	4	4	4	4	4	4	2	2	3	3	2	2	2	2
13	0.500334	1224.40	4	4	4	4	3	4	2	2	3	3	2	2	2	2
14	0.501049	1195.00	4	4	4	4	3	3	2	2	3	3	2	2	2	2
15	0.504268	1165.60	4	4	3	4	3	3	2	2	3	3	2	2	2	2
16	0.512186	1130.20	4	4	3	4	3	3	2	2	2	3	2	2	2	2
17	0.529015	1115.50	4	3	3	4	3	3	2	2	2	3	2	2	2	2
18	0.541331	1106.60	4	4	3	4	2	3	2	2	2	3	2	2	2	2
19	0.541761	1081.60	4	4	3	4	4	3	2	2	2	3	1	1	2	2
20	0.542226	1058.00	4	4	3	4	4	2	2	2	2	3	1	1	2	2
21	0.542706	1033.00	4	4	3	4	4	4	2	2	2	3	1	1	1	1
22	0.545111	1009.40	4	4	2	4	4	4	2	2	2	3	1	1	1	1
23	0.545426	984.40	4	4	4	4	4	4	1	1	2	3	1	1	1	1
24	0.552452	955.00	4	4	4	3	4	4	1	1	2	3	1	1	1	1
25	0.554429	949.00	4	4	4	4	4	4	1	1	2	2	1	1	1	1
26	0.569050	919.60	4	4	4	4	3	4	1	1	2	2	1	1	1	1
27	0.570109	890.20	4	4	4	4	3	3	1	1	2	2	1	1	1	1
28	0.572666	860.80	4	4	3	4	3	3	1	1	2	2	1	1	1	1
29	0.594694	831.40	4	4	3	3	3	3	1	1	2	2	1	1	1	1
30	0.595876	821.80	4	4	3	4	3	3	1	1	1	2	1	1	1	1
31	0.598075	807.10	4	3	3	4	3	3	1	1	1	2	1	1	1	1
32	0.610047	798.20	4	4	3	4	2	3	1	1	1	2	1	1	1	1
33	0.611286	774.50	4	4	3	4	2	2	1	1	1	2	1	1	1	1
34	0.613489	751.00	4	4	2	4	2	2	1	1	1	2	1	1	1	1
35	0.636142	721.60	4	4	2	3	2	2	1	1	1	2	1	1	1	1
36	0.639251	706.90	4	3	2	3	2	2	1	1	1	2	1	1	1	1
37	0.668559	692.20	3	3	2	3	2	2	1	1	1	2	1	1	1	1
38	0.678868	680.90	3	4	2	3	1	2	1	1	1	2	1	1	1	1
39	0.680241	654.90	3	4	2	3	1	1	1	1	1	2	1	1	1	1
40	0.681912	643.60	4	4	1	3	1	1	1	1	1	2	1	1	1	1
41	0.687364	634.00	4	4	1	4	1	1	1	1	1	1	1	1	1	1
42	0.708487	619.30	4	3	1	4	1	1	1	1	1	1	1	1	1	1
43	0.711982	605.30	4	3	1	2	1	1	1	1	1	2	1	1	1	1
44	0.736837	589.90	4	3	1	3	1	1	1	1	1	1	1	1	1	1
45	0.750919	578.10	4	2	1	3	1	1	1	1	1	1	1	1	1	1
46	0.769255	563.40	3	2	1	3	1	1	1	1	1	1	1	1	1	1
47	0.812678	554.50	4	2	1	2	1	1	1	1	1	1	1	1	1	1
48	0.823167	541.50	4	1	1	2	1	1	1	1	1	1	1	1	1	1
49	0.845096	526.80	3	1	1	2	1	1	1	1	1	1	1	1	1	1
50	0.902048	515.00	2	1	1	2	1	1	1	1	1	1	1	1	1	1
51	1.078546	500.80	3	1	1	1	1	1	1	1	1	1	1	1	1	1
52	1.135498	489.00	2	1	1	1	1	1	1	1	1	1	1	1	1	1
53	1.315050	476.00	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 2

SYSTEM-TIME/COST CHARACTERISTIC TABLE
OLD ALGORITHM

CHOICE #	SYSTEM TIME(sec)	COST/MW	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.502219	1383.40	4	4	3	4	4	4	3	3	4	4	2	2	2	2
2	0.504472	1354.00	4	4	3	4	3	4	3	3	4	4	2	2	2	2
3	0.505455	1324.60	4	4	3	4	3	3	3	3	4	4	2	2	2	2
4	0.512798	1318.60	4	4	4	4	3	3	2	3	4	4	2	2	2	2
5	0.513007	1283.20	4	4	4	4	3	3	2	2	4	4	2	2	2	2
6	0.518057	1239.10	4	4	4	4	3	3	2	2	3	4	2	2	2	2
7	0.534551	1224.40	4	3	4	4	3	3	2	2	3	4	2	2	2	2
8	0.540569	1195.00	4	3	3	4	3	3	2	2	3	4	2	2	2	2
9	0.541256	1150.90	4	3	3	4	3	3	2	2	3	3	2	2	2	2
10	0.545453	1117.00	4	4	3	4	4	3	2	2	3	3	1	1	2	2
11	0.546218	1107.40	4	4	3	4	4	4	2	2	3	3	1	1	1	2
12	0.546602	1068.40	4	4	3	4	4	4	2	2	3	3	1	1	1	1
13	0.565361	1033.00	4	4	3	4	4	4	2	2	2	3	1	1	1	1
14	0.573189	1003.60	4	4	3	4	3	4	2	2	2	3	1	1	1	1
15	0.574515	974.20	4	4	3	4	3	3	2	2	2	3	1	1	1	1
16	0.581008	964.60	4	4	4	4	3	3	1	2	2	3	1	1	1	1
17	0.581384	925.60	4	4	4	4	3	3	1	1	2	3	1	1	1	1
18	0.603611	910.90	4	3	4	4	3	3	1	1	2	3	1	1	1	1
19	0.607604	875.50	4	3	4	4	3	3	1	1	2	2	1	1	1	1
20	0.608946	846.10	4	3	3	4	3	3	1	1	2	2	1	1	1	1
21	0.615118	837.20	4	4	3	4	2	3	1	1	2	2	1	1	1	1
22	0.616779	813.60	4	4	3	4	2	2	1	1	2	2	1	1	1	1
23	0.645875	798.90	4	3	3	4	2	2	1	1	2	2	1	1	1	1
24	0.649051	759.90	4	3	3	4	2	2	1	1	1	2	1	1	1	1
25	0.650549	736.30	4	3	2	4	2	2	1	1	1	2	1	1	1	1
26	0.685974	725.00	4	4	2	4	1	2	1	1	1	2	1	1	1	1
27	0.688392	699.00	4	4	2	4	1	1	1	1	1	2	1	1	1	1
28	0.699120	669.60	4	4	2	3	1	1	1	1	1	2	1	1	1	1
29	0.717488	654.90	4	3	2	3	1	1	1	1	1	2	1	1	1	1
30	0.720670	628.90	4	3	1	3	1	1	1	1	1	2	1	1	1	1
31	0.749747	619.30	4	3	1	4	1	1	1	1	1	1	1	1	1	1
32	0.762030	607.50	4	2	1	4	1	1	1	1	1	1	1	1	1	1
33	0.774833	602.40	3	2	1	3	1	1	1	1	1	2	1	1	1	1
34	0.795867	593.50	4	2	1	2	1	1	1	1	1	2	1	1	1	1
35	0.799816	578.10	4	2	1	3	1	1	1	1	1	1	1	1	1	1
36	0.838940	565.10	4	1	1	3	1	1	1	1	1	1	1	1	1	1
37	0.853978	550.40	3	1	1	3	1	1	1	1	1	1	1	1	1	1
38	0.896362	541.50	4	1	1	2	1	1	1	1	1	1	1	1	1	1
39	0.950525	526.80	3	1	1	2	1	1	1	1	1	1	1	1	1	1
40	1.060058	515.00	2	1	1	2	1	1	1	1	1	1	1	1	1	1
41	1.242197	500.80	3	1	1	1	1	1	1	1	1	1	1	1	1	1
42	1.351730	489.00	2	1	1	1	1	1	1	1	1	1	1	1	1	1
43	1.740369	476.00	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 3

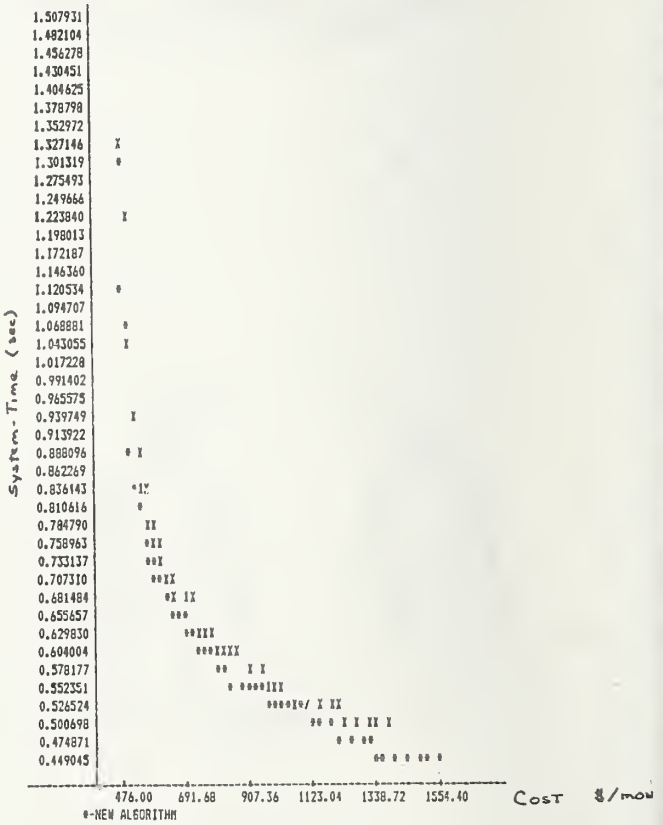


Table 4

POSSIBLE LINK CAPACITIES

CAPACITY #	CAPACITY(BPS)	COST/MONTH/MILE	SERVICE TIME /CAPACITY
1	450	1.40	0.2666000
2	600	1.70	0.2000000
3	750	1.88	0.1600000
4	900	2.00	0.1332999

Table 5

SYSTEM-TIME/COST CHARACTERISTIC TABLE
NEW ALGORITHM

CHOICE #	SYSTEM TIME(sec)	COST/MN	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.452302	377.20	4	4	4	0	4	1	4	3	0	0	3	3	0	0
2	0.457204	374.80	4	4	4	0	3	1	4	3	0	0	3	3	0	0
3	0.470825	366.40	4	4	4	0	4	1	4	3	0	0	2	2	0	0
4	0.472799	361.00	4	4	4	0	4	1	4	2	0	0	2	2	0	0
5	0.497944	358.60	4	4	4	0	3	1	4	2	0	0	2	2	0	0
6	0.512049	355.00	4	4	4	0	3	1	3	2	0	0	2	2	0	0
7	0.525221	353.80	4	3	4	0	3	1	3	2	0	0	2	2	0	0
8	0.538726	351.40	4	4	4	0	2	1	3	2	0	0	2	2	0	0
9	0.538998	339.40	4	4	4	0	4	1	3	2	0	0	1	1	0	0
10	0.541010	330.40	4	4	4	0	4	1	3	1	0	0	1	1	0	0
11	0.566154	328.00	4	4	4	0	3	1	3	1	0	0	1	1	0	0
12	0.571388	325.60	4	4	3	0	3	1	3	1	0	0	1	1	0	0
13	0.593432	324.40	4	3	3	0	3	1	3	1	0	0	1	1	0	0
14	0.598088	323.20	3	3	3	0	3	1	3	1	0	0	1	1	0	0
15	0.606936	320.80	3	4	3	0	2	1	3	1	0	0	1	1	0	0
16	0.634214	319.60	3	3	3	0	2	1	3	1	0	0	1	1	0	0
17	0.641029	317.80	4	3	4	0	2	1	2	1	0	0	1	1	0	0
18	0.671407	315.40	4	3	3	0	2	1	2	1	0	0	1	1	0	0
19	0.675192	310.60	4	4	3	0	1	1	2	1	0	0	1	1	0	0
20	0.698107	309.40	3	4	3	0	1	1	2	1	0	0	1	1	0	0
21	0.702545	308.20	3	3	3	0	1	1	2	1	0	0	1	1	0	0
22	0.724190	305.80	4	3	2	0	1	1	2	1	0	0	1	1	0	0
23	0.743633	304.00	4	2	2	0	1	1	2	1	0	0	1	1	0	0
24	0.750890	302.80	3	2	2	0	1	1	2	1	0	0	1	1	0	0
25	0.790890	301.00	2	2	2	0	1	1	2	1	0	0	1	1	0	0
26	0.812674	298.00	2	1	2	0	1	1	2	1	0	0	1	1	0	0
27	1.266468	295.60	4	1	3	0	1	1	1	1	0	0	1	1	0	0
28	1.293167	294.40	3	1	3	0	1	1	1	1	0	0	1	1	0	0
29	1.319251	292.00	4	1	2	0	1	1	1	1	0	0	1	1	0	0
30	1.345950	290.80	3	1	2	0	1	1	1	1	0	0	1	1	0	0
31	1.385950	289.00	2	1	2	0	1	1	1	1	0	0	1	1	0	0
32	1.552710	286.00	4	1	1	0	1	1	1	1	0	0	1	1	0	0
33	1.579409	284.80	3	1	1	0	1	1	1	1	0	0	1	1	0	0
34	1.619409	283.00	2	1	1	0	1	1	1	1	0	0	1	1	0	0
35	1.686008	280.00	1	1	1	0	1	1	1	1	0	0	1	1	0	0

Table 6

SYSTEM-TIME/COST CHARACTERISTIC TABLE
OLD ALGORITHM

CHOICE #	SYSTEM TIME(sec)	COST/MN	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.530033	348.40	4	4	4	0	4	1	4	3	0	0	1	1	0	0
2	0.568972	346.00	4	4	4	0	3	1	4	3	0	0	1	1	0	0
3	0.570531	340.60	4	4	4	0	3	1	4	2	0	0	1	1	0	0
4	0.596929	339.40	4	3	4	0	3	1	4	2	0	0	1	1	0	0
5	0.609780	335.80	4	3	4	0	3	1	3	2	0	0	1	1	0	0
6	0.610494	333.40	4	4	4	0	2	1	3	2	0	0	1	1	0	0
7	0.638451	332.20	4	3	4	0	2	1	3	2	0	0	1	1	0	0
8	0.638741	323.20	4	3	4	0	2	1	3	1	0	0	1	1	0	0
9	0.680435	318.40	4	4	4	0	1	1	3	1	0	0	1	1	0	0
10	0.701162	316.00	4	4	3	0	1	1	3	1	0	0	1	1	0	0
11	0.708392	314.80	4	3	3	0	1	1	3	1	0	0	1	1	0	0
12	0.738760	311.80	4	3	4	0	1	1	2	1	0	0	1	1	0	0
13	0.750742	310.00	4	2	4	0	1	1	2	1	0	0	1	1	0	0
14	0.801191	307.60	4	2	3	0	1	1	2	1	0	0	1	1	0	0
15	0.822547	304.60	4	1	3	0	1	1	2	1	0	0	1	1	0	0
16	0.863603	303.40	3	1	3	0	1	1	2	1	0	0	1	1	0	0
17	0.940229	301.00	4	1	2	0	1	1	2	1	0	0	1	1	0	0
18	1.002650	299.80	3	1	2	0	1	1	2	1	0	0	1	1	0	0
19	1.141698	298.00	2	1	2	0	1	1	2	1	0	0	1	1	0	0
20	1.396241	295.60	4	1	3	0	1	1	1	1	0	0	1	1	0	0
21	1.458663	294.40	3	1	3	0	1	1	1	1	0	0	1	1	0	0
22	1.535289	292.00	4	1	2	0	1	1	1	1	0	0	1	1	0	0
23	1.597711	290.80	3	1	2	0	1	1	1	1	0	0	1	1	0	0
24	1.736758	289.60	2	1	2	0	1	1	1	1	0	0	1	1	0	0
25	2.280999	286.00	4	1	1	0	1	1	1	1	0	0	1	1	0	0
26	2.343421	284.80	3	1	1	0	1	1	1	1	0	0	1	1	0	0
27	2.482469	283.00	2	1	1	0	1	1	1	1	0	0	1	1	0	0
28	3.228179	280.00	1	1	1	0	1	1	1	1	0	0	1	1	0	0

Table 7

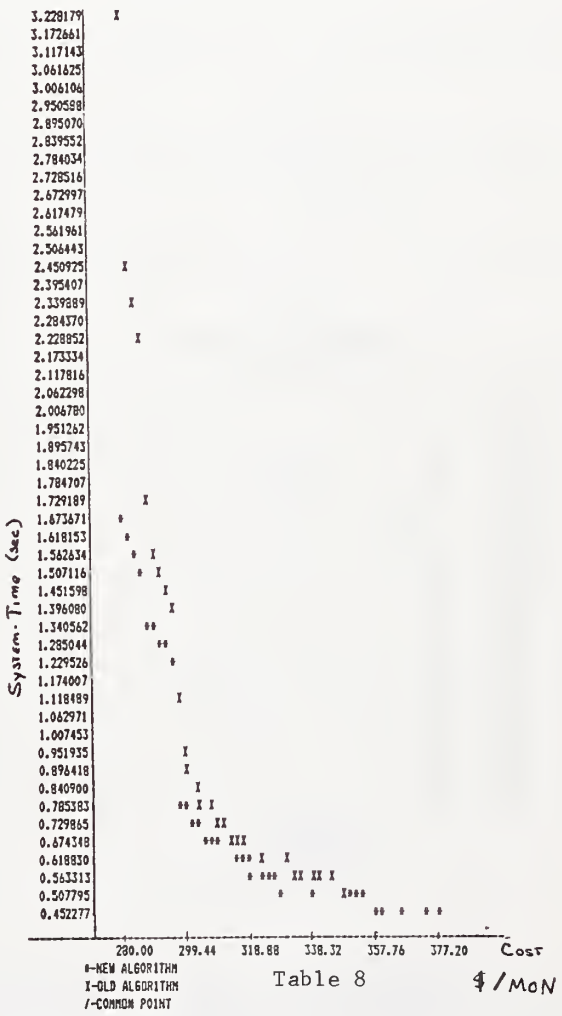


Table 8

QUEUEING-TIME/COSt CHARACTERISTIC TABLE
NEW ALGORITHM

CHOICE #	QUEUEING TIME(sec)	COST/MIN	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.036667	791.00	2	4	1	2	4	1	1	1	2	1	1	4	1	1
2	0.045710	787.40	3	4	1	3	4	1	1	1	1	1	1	4	1	1
3	0.046125	778.50	4	4	1	2	4	1	1	1	1	1	1	4	1	1
4	0.048156	763.80	3	4	1	2	4	1	1	1	1	1	1	4	1	1
5	0.049004	719.70	3	4	1	2	4	1	1	1	1	1	1	3	1	1
6	0.051373	708.40	4	4	1	1	4	1	1	1	1	1	1	3	1	1
7	0.052315	707.90	2	4	1	2	4	1	1	1	1	1	1	3	1	1
8	0.052731	678.50	2	4	1	2	3	1	1	1	1	1	1	3	1	1
9	0.054255	664.30	3	4	1	1	3	1	1	1	1	1	1	3	1	1
10	0.058219	649.60	3	3	1	1	3	1	1	1	1	1	1	3	1	1
11	0.058414	637.80	2	3	1	1	3	1	1	1	1	1	1	3	1	1
12	0.069223	624.80	1	3	1	1	3	1	1	1	1	1	1	3	1	1
13	0.079078	604.10	1	4	1	1	3	1	1	1	1	1	1	2	1	1
14	0.084566	589.40	1	3	1	1	3	1	1	1	1	1	1	2	1	1
15	0.087960	580.50	1	4	1	1	2	1	1	1	1	1	1	2	1	1
16	0.093448	565.80	1	3	1	1	2	1	1	1	1	1	1	2	1	1
17	0.107388	554.00	1	2	1	1	2	1	1	1	1	1	1	2	1	1
18	0.160109	550.40	1	3	1	1	3	1	1	1	1	1	1	1	1	1
19	0.163503	541.50	1	4	1	1	2	1	1	1	1	1	1	1	1	1
20	0.168991	526.80	1	3	1	1	2	1	1	1	1	1	1	1	1	1
21	0.182931	515.00	1	2	1	1	2	1	1	1	1	1	1	1	1	1
22	0.200838	500.80	1	3	1	1	1	1	1	1	1	1	1	1	1	1
23	0.214778	489.00	1	2	1	1	1	1	1	1	1	1	1	1	1	1
24	0.273029	476.00	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 9

QUEUEING-TIME/COST CHARACTERISTIC TABLE
OLD ALGORITHM

CHOICE #	QUEUEING TIME(sec)	COST/MIN	CAPACITY SELECTED AT EACH LINK 2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.080406	763.80	3	4	1	2	4	1	1	1	1	1	1	4	1	1
2	0.088521	752.00	2	4	1	2	4	1	1	1	1	1	1	4	1	1
3	0.092742	707.90	2	4	1	2	4	1	1	1	1	1	1	3	1	1
4	0.097931	693.70	3	4	1	1	4	1	1	1	1	1	1	3	1	1
5	0.107533	664.30	3	4	1	1	3	1	1	1	1	1	1	3	1	1
6	0.109617	652.50	2	4	1	1	3	1	1	1	1	1	1	3	1	1
7	0.119090	646.50	2	4	1	1	4	1	1	1	1	1	1	2	1	1
8	0.126216	637.80	2	3	1	1	3	1	1	1	1	1	1	3	1	1
9	0.133901	617.10	2	4	1	1	3	1	1	1	1	1	1	2	1	1
10	0.138286	604.10	1	4	1	1	3	1	1	1	1	1	1	2	1	1
11	0.152564	589.40	1	3	1	1	3	1	1	1	1	1	1	2	1	1
12	0.166447	580.50	1	4	1	1	2	1	1	1	1	1	1	2	1	1
13	0.185109	565.80	1	3	1	1	2	1	1	1	1	1	1	2	1	1
14	0.209444	565.10	1	4	1	1	3	1	1	1	1	1	1	1	1	1
15	0.227951	554.00	1	2	1	1	2	1	1	1	1	1	1	2	1	1
16	0.228107	550.40	1	3	1	1	3	1	1	1	1	1	1	1	1	1
17	0.241990	541.50	1	4	1	1	2	1	1	1	1	1	1	1	1	1
18	0.260652	526.80	1	3	1	1	2	1	1	1	1	1	1	1	1	1
19	0.303494	515.00	1	2	1	1	2	1	1	1	1	1	1	1	1	1
20	0.360313	500.80	1	3	1	1	1	1	1	1	1	1	1	1	1	1
21	0.403155	489.00	1	2	1	1	1	1	1	1	1	1	1	1	1	1
22	0.549678	476.00	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 10

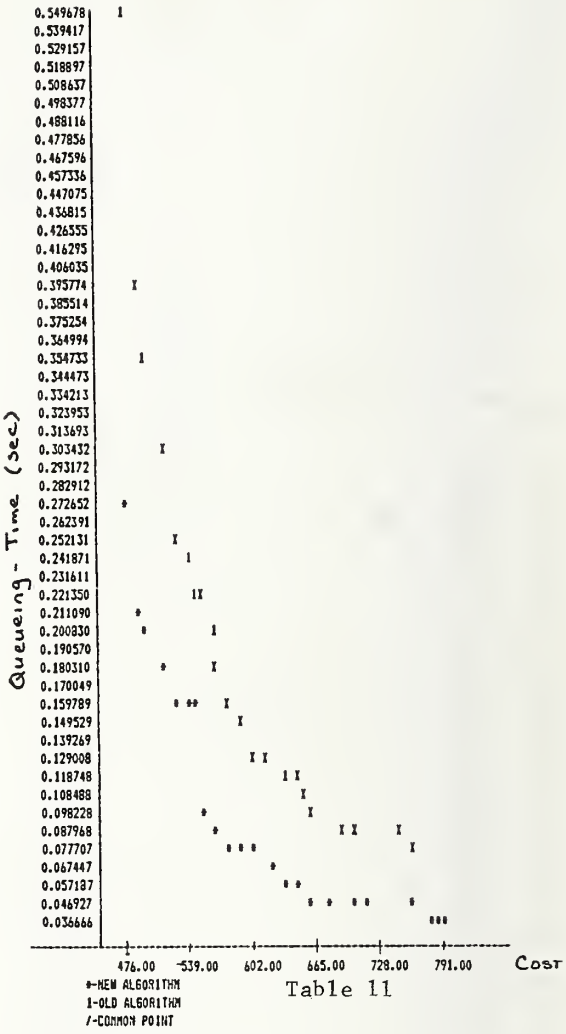


Table 11

Components Of Software Packages For The Solution Of Queueing Network Models

G.S. Graham*
E.D. Lazowska*
K.C. Sevcik*

Quantitative System Performance
7215 30th Avenue N.W.
Seattle, Washington
U.S.A. 98117

We describe the various components that are present in most current software packages for the analysis of queueing network models of computer systems. Each type of component serves a specific function. We survey some of the alternative approaches that can be used in implementing each component.

1. Introduction

In the past ten years, many organizations have come to use queueing network models as essential elements of their system sizing and capacity planning activities. They use, on a day-to-day basis, software packages for analysing queueing network models. Several such packages have been developed. Some are available commercially, while others are used on an experimental or research basis by a limited number of installations.

In this paper, we attempt to provide insight into the structure of software packages for queueing network model analysis. We shall enumerate the major components and their functions, and describe some of the alternative approaches for implementing each one.

Many software packages for queueing network model solution can be viewed as having the structure shown in figure 1. There is a sequence of software layers, each layer transforming its input from the layer above into a form suitable for the layer below. While the number and content of the layers may vary from package to package, the layered form is still present. We shall use this structure to describe the major component types in the next section. Section three describes various possible front end interfaces to such packages, some for the general performance environment and others tailored to specific application areas.

2. Basic Components of Queueing Network Analysis Packages

A queueing network analysis package has, at its core, a software routine that evaluates queueing network models. But there may be many layers of software between the user wanting to solve a performance problem and this basic computational engine.

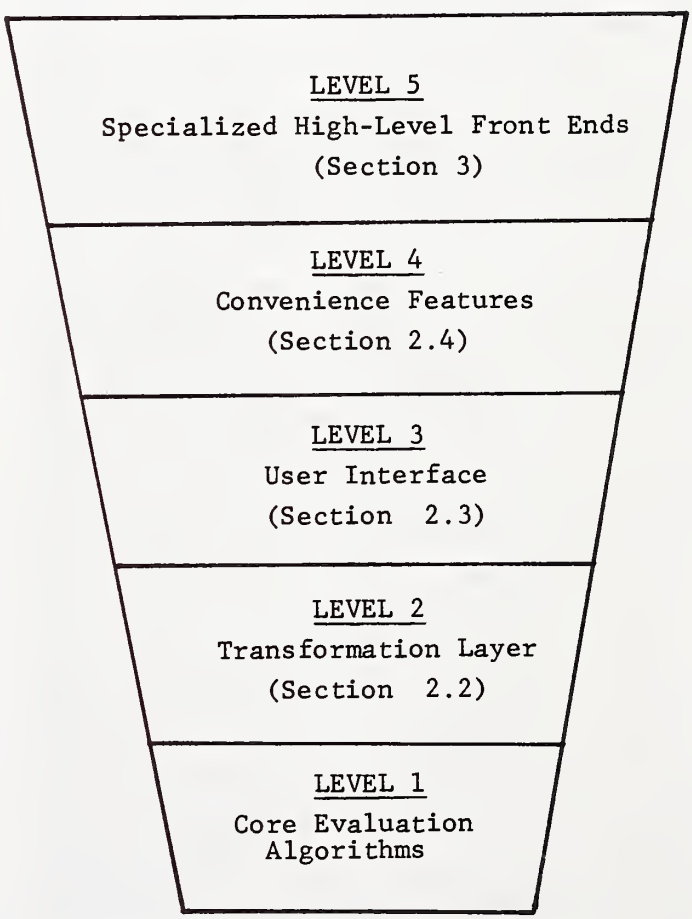


Figure 1. Basic components of a queueing network analysis package.

*University affiliations: G.S. Graham, Computer Systems Research Group, University of Toronto, Toronto, Ontario. E.D. Lazowska, Department of Computer Science, University of Washington, Seattle, Washington. K.C. Sevcik, Computer Systems Research Group, University of Toronto, Toronto, Ontario.

Some layers are there for the convenience of the user, who may not be dealing directly with the standard queueing network model elements of classes, devices, and loadings; this is discussed in more detail in section three. Some layers are there for the convenience of the package implementor, who may have to solve a number of models to produce a single set of answers.

2.1 The Core Computational Algorithm

Situated at the lowest level in figure 1, the core routine that comprises the computational engine may be based on exact algorithms (such as mean value analysis [ReLa80] or a related technique, convolution [Buze73,SaCh81]) or on approximate algorithms (as described by Bard [Bard79], Schweitzer [Schw79], and Neuse and Chandy [NeCh81]). The selection of a particular algorithm is based on storage requirements, execution time requirements, and error tolerances. Either type of core algorithm solves only those networks that satisfy certain assumptions that cause their state probability expression to be *separable* (or to have *product form*). These assumptions involve restrictions on the scheduling disciplines allowed and on the patterns in which jobs move from one resource to another in the system. Intuitively, activities (e.g., rates of service) at one device cannot depend on conditions at other devices. Also, scheduling cannot discriminate among jobs on the basis of their service times.

The job of the core computational algorithm is simply stated. Given a model that conforms to the assumptions leading to a separable model, from the definitions of classes, devices, and loadings, the algorithm must produce such performance measures as throughput, mean response time, device utilization, and mean device queue lengths.

2.2 Approximation Transformations

Some of the models posed to a queueing network analysis package may not be separable. It is the purpose of the software layers above the computational engine to translate these models into a form that is suitable for the basic core routine. Issues such as memory constraints, priority scheduling, and advanced I/O modeling lead to non-separable models, and thus are some of the items treated by the approximation transformation layer (level 2 of figure 1).

Memory constraints have an effect when the number of ready-to-run jobs of a class is greater than the number of memory partitions assigned to that class. A batch class operates with a specified average multiprogramming level and as such the memory constraint is automatically handled in a separable model. Terminal-driven classes or open classes do not have this property; when either of these classes is memory constrained, a non-separable model results. We illustrate how this can be handled conveniently by treating a terminal-driven class. Suppose the model has M terminals and a central subsystem multiprogramming limit of N jobs (N<M). The transformation layer software can call the core routine to solve the separable central subsystem model for multiprogramming levels from 1 to N. The solution of this model establishes the load-dependent service rates of a flow equivalent composite server that would represent the throughput of the central subsystem in a hierarchical model with M terminals. Again, the transformation layer calls the core routine to solve this separable model for mean response time and to determine the average number of terminal requests seeking service. The assumption that the load-dependent throughput rates of the central subsystem are a good representation of the throughput of the central subsystem causes the final result to be an approximation to the solution of the original model. With this approach, the user needs only to request a solution to the memory constrained model; he is generally oblivious to the fact that two separable models were solved and an approximation made. This type of approach can be generalized to permit the specification of constraints on populations of arbitrary combinations of workload classes (i.e., limits on domain capacities) [LaZa82].

Priority scheduling is a dominant feature of CPU dispatching in many current operating systems. To model such a feature, the user typically identifies any device(s) scheduled by priority and assigns priority values to the job classes. The resulting non-separable model can be solved in a variety of ways, but a common method is an iterative solution. The transformation layer solves a sequence of models in which each priority class has a separate cpu device. (This is known as the *shadow cpu* technique [Sevc77].) Consider the case of two priority classes. By giving the lower priority class its own cpu, we avoid the interference that would otherwise be encountered by the high priority class at the cpu. But, to represent the delays at the cpu suffered by the lower priority class, the service rate of the shadow cpu is reduced by an amount related to the cpu utilization achieved by the high priority class. This leads naturally to an iterative solution in which better and better estimates of the cpu utilization of the high priority class are obtained from solutions of the separable model containing both the actual cpu and the shadow cpu. Again, the package user simply requests a solution of the model involving priority scheduling. To obtain this, the transformation layer calls on the core computational algorithm to solve several separable models containing an additional artificial device.

Finally, advanced I/O subsystem modeling is sometimes important when details of multipathing and shared storage devices need to be taken into account. In IBM I/O architectures, the delay due to channel contention is not represented explicitly in most queueing network models. The central subsystem devices modeled are typically just the CPU and individual disk devices. To account for channel contention, a software layer in the package needs to know about which control units are attached to which channels, which channels are on which logical channels, and which disks are on which logical channels. This layer can then estimate the amount of wasted time due to interference at the physical channel (for example, in missed RPS reconnects). Because channels are not usually represented explicitly as devices in queueing network models, the wasted time of a job would be assigned to the disks in the form of a service time expansion. Thus, the disk loading consists of an intrinsic I/O resource demand plus a busy time due to channel contention. This model, with loadings that may differ from those submitted by the performance analyst, is a separable model acceptable by the core routine. Sometimes iteration is necessary between the software layers, but the core computational algorithm need never treat channels or missed RPS re-connects.

The basic difficulty in modelling I/O subsystems is that jobs must simultaneously hold and use several resources at once (i.e., disk, control unit, and channel). This simultaneous resource possession cannot be reflected directly in a queueing network model, but general approaches to obtaining iterative solutions by repeated calls on the core computational algorithm have been developed [JaLa82].

Another situation in which the transformation layer produces a model solution by manipulation of the results of one or more calls to the core algorithm is when some model parameter is specified as a distribution rather than as a single value. Two examples are multiprogramming level of a class or priority assigned to a class. The core algorithm is called upon to solve the basic model for each possible value of the parameter specified by its distribution. The transformation layer then produces a weighted combination of these results as the final answer.

2.3 User Interface

Software packages for queueing network model solution allow users to express their performance models conveniently. In addition to being designed in accordance with the usual human and software engineering principles, the package front ends use the normal language of the performance analyst. There are explicit counterparts to the concepts of a system workload component, a job, a device, and a resource demand. The user is allowed some flexibility in determining what exactly constitutes a workload com-

ponent. In IBM's MVS operating system, either a performance group or a performance period within a group may be used as the basic workload component. The choice can be made according to whether or not the performance measures on a performance group basis are sufficiently specific to answer the questions being addressed with the model.

Most packages deal with the concepts of classes, devices, and loadings directly, although the syntax may vary (e.g., a BATCH class in MAP has TYPE BP in BEST/1). The user interface layer, situated on level 3 of figure 1, provides the input to the transformation layer. When advanced I/O modeling is done, for example, the user interface language is used to describe channels, controllers, and logical channels, and the transformation layer converts this information into inflated device loadings, creating a separable model to be solved by the core routine.

2.4 Convenience Features

Additional features, provided by the next software layer (level 4 in figure 1), enhance the convenience of using the package by increasing the performance analyst's ability to work quickly and efficiently. We shall discuss three examples.

The first involves providing a data base for storing models and their results. Models that have been defined during an interactive session with a package can be saved on auxiliary storage. This is useful because a baseline model, once created and validated, can be recalled many times for modification of individual components. Model modification is a major activity in interacting with a queueing network analysis package. It requires the ability to work on several models and versions during a session, and this requires the ability to save models. Similarly, output reports need to be stored. This is useful in a parametric study in which the output values from a series of output reports are plotted against an independent variable that is being varied across the series of models.

Model outputs can also serve as inputs to other models in hierarchical model evaluation. Components of models can be isolated, evaluated, and replaced by a flow equivalent composite server in a higher level model. Both model inputs and outputs need to be stored in such a case.

A second convenience feature permits the user to describe service centers as specific commercial hardware devices. Most service centers in a queueing network model correspond directly to a hardware processor, either a CPU or a storage device. It is convenient for a user to be able to associate a hardware type with a device name. For example, a device may have the name CPU and the TYPE 3033 or V7 (the type in this case having a mnemonic purpose). A software package may permit a user to change a device's TYPE and thereby automatically modify all the loadings at that device on the basis of relative speed factors stored inside the package. For example, if the device SYS001 has TYPE 3350 in a baseline model, changing its TYPE to 3380 alters all its loadings, for all classes, by dividing them by the associated speed conversion factor (approximately 1.6 in this case). The relative speed factors do depend on the character of the particular workload, but for an initial performance study, the standard stored factors are suffuciently accurate. This added level of convenience comes from a software layer that allows specification of device TYPEs and provides device loadings to the core routine.

Thirdly, some packages permit a user to carry out an essentially arbitrary sequence of model experiments. This is done through a limited capability programming language. For example, the user may want to perform a parametric study (say on the effect of increasing the number of TSO terminals). It is certainly possible for him to interact with a package directly and issue solution commands with a new number of terminals each time. However, this is cumbersome and is not a good way for a performance analyst to spend his time. Instead, he may write a driver routine that, when interpreted, accomplishes what he wants automatically. This driver routine may vary the number of TSO terminals from 20 to 30 in steps of 1, obtaining model solutions each time. The command to a package might be VARY_TERM, which is just the name of a file that contains other commands. The syntax for VARY_TERM specific to the MAP solution package is:

```
        CLASS TSO
        SET NT 20
:LOOP   PERFORMANCE
        SET NT 1+
        IF LE NT 30 LOOP
        RETURN
```

(The class context is TSO; the number of terminals is initialized to 20; a system-level PERFORMANCE report is generated; the number of terminals is increased by one; if the number of terminals is less than or equal to 30, go to the label LOOP; otherwise, RETURN to the package.)

This example illustrates how a user can program iterations of model solutions. It is usual that several closely-related versions of a model need to be solved. A simple programming capability that can be implemented in a layer of software can be of great benefit in this situation.

3. High-Level Front Ends

The interactions between user and package that were discussed in the last section are typical of those involved in performance validation and prediction problems. The issue is the performance of an existing system subjected to new workloads and configurations. The language accepted by the front end is thus the language of capacity planning and tuning: new workloads and devices. Because the major work in using a package is performance prediction, packages allow the user to modify easily classes, devices, and loadings.

Constructing a model initially can be a tedious process. After system operation is understood sufficiently well and a model is outlined, measurement data must be gathered and reduced to yield parameter values for input to a queueing network analysis package. A side benefit of this process is that a deeper understanding of the system can be gained. However, there are at least three disadvantages to performing this routine manually. It takes an inordinate amount of time to gather and reduce the data for any reasonable system. Because of the large volume of data, the process is prone to calculation errors. Finally, many of the actions are repetitive; measurement data from many time periods may be used as model input. There is a clear need to automate the process of parameterizing a model of an existing system.

The basis for such an automatic process is the existence of archived time-stamped measurement data. Instead of using only standard performance reports as the source of model inputs, a special program can be used to identify events over a specified time interval. The program can be directed to form classes from workload components in a particular way. For example, if all batch workloads are to be treated as a single class, then the program can aggregate all resource demands associated with these workloads. In IBM's MVS operating system, a natural workload component is the performance group. All CPU and storage devices that appear in the archived records are normally treated as individual servers in the generated queueing network model. Loadings are calculated using standard parameter estimation procedures. Models constructed from such an automatic procedure can be stored on auxiliary storage, but are otherwise identical to models entered by users interacting with a package. The user can direct the package to read a file that contains the model; both model preparation time and input time are greatly reduced. The normal mode of model modification can then be initiated.

The automatic construction of a model is one example of a front end (level 5 of figure 1) for a package, designed for the per-

formance evaluation of an existing system. Other application areas for which specialized front ends are useful are the performance evaluation of a future computer system, of a data base system, and of a communication network.

The performance evaluation of a future computer system differs from the evaluation of an existing system in the lack of actual measurement data. The system does not exist, so measurements do not exist. But system design specifications, flowcharts, and rough estimates of gross resource usage are usually available soon after the beginning of the development project. The performance evaluation effort can make use of this information during system implementation. The front end interface should therefore be based on a system designer's point of view.

Instruction execution rates for processors can be closely estimated before they are operational, and software designers can estimate mean cpu path lengths for each major activity of the software under development. But even information at this high level of detail can be used to establish loadings for a queueing network model. For example, a CPU path length of a certain number of instructions on a CPU of a certain instruction execution rate can be transformed into a CPU loading that is input to a core computational routine. Similarly, specifications of the number of file reads to each file, together with a projected assignment of files to storage devices of specified speeds are sufficient to project storage device loadings. From these, the core solution algorithm can then produce estimated average response time and other performance measures that the system designer can use to guide the project evolution.

Evaluation of either computer-communication networks or data base systems can also be accomplished by placing suitable high-level front ends on existing queueing network analysis packages. In the case of networks, the front end accepts system descriptions in terms of such entities as terminal clusters, line speeds, multiplexors, and concentrators. Also, various communication protocols can be specified as governing the interactions between the terminals and the communications controller. With this information specified, the delays incurred in the communications network can be incorporated into queueing network models as service requirements at additional devices (e.g., the communications controller).

In the case of data base systems, a high-level front end can permit the user to investigate data base design decisions as well as computer system management decisions. The data base environment is specified in terms of the dominant transaction types and the data base components to which they require access. This information, together with file placement and buffer handling strategies, permits calculation of the expected number of device accesses per transaction, and, knowing device characteristics, these can be converted to loadings. (A more detailed description of this kind of activity is given elsewhere [Sevc81].)

A particularly useful form of high-level front end for data base performance modelling is one that is specialized to a specific data base management system (such as S2000 or IMS). The advantage of such specialization is that a great deal of system-specific information can be incorporated into the front end. That is, general strategies for scheduling, buffer management, and record access would be known to the front end, and the user would be required only to supply values for parameters used to control the general strategies.

4. Choosing a Package

One decision faced by an organization interested in modeling performance is the choice of a queueing network analysis package: Should it be developed in-house, or should it be obtained from a commercial vendor in the business of producing such packages? Most of the arguments are in favor of a vendor-produced package. Basic computational algorithms are widely known and published. However, developing a complete software package locally is a major software development project with all the associated problems.

A vendor-produced package can allow performance analysts to concentrate on what they do best: capacity planning, tuning, and performance evaluation. The package is a tool to be used as a performance calculation component of a planning methodology, rather than as an end goal itself. It is certainly desirable for an analyst to understand the fundamental ideas of how the analysis works (e.g., the general strategy of mean value analysis), but he need not become involved with the details of the core algorithm implementation.

There are other advantages to using vendor-produced packages. Because they typically have a larger user base than an in-house package, one can have more faith in their implementations. In some cases, user groups exist to share information on using a particular package effectively and on applying it in non-standard situations. Some packages have matured to the point where they have interfaces to standard statistical analysis reporting packages, which may be used elsewhere in the organization. A vendor-produced package is more likely to contain various application-specific or system-specific front ends. For example, there may be an IBM-compatible front end based on "channels" and "control units", and a CDC front end based on "peripheral processors".

One of the difficulties of producing a queueing network analysis package is maintaining it to keep up with the rapid technological advances in queueing network modeling. In 1971, the initial efficient computational algorithms were developed for single class separable models. These were followed in 1975 by the multiple class algorithms and in 1978 by the heuristic iterative algorithms based on mean value analysis. Modeling strategies for specific situations, such as advanced I/O modeling, have appeared only in recent years. This brief historical summary shows that rapid and extensive changes have taken place in the lower level software layers of queueing network analysis packages in just over ten years. The working performance analyst should not be expected to track these developments and determine their significance.

5. Conclusions

Software packages for queueing network model solution typically consist of several layers. An important design goal for such packages is to permit each user to interact with the package at the most convenient level for his particular problems, without needing even to be aware of other layers. Nonetheless, by understanding the general structure of such packages at the level of detail presented in this paper, a performance analyst is in a better position to understand the significance of answers obtained from such packages and the feasibility of broadening a package to handle additional computer system features.

References

Bard79 Y. Bard, Some extensions to multiclass queueing network analysis, Fourth Int. Symp. on Modeling and Perf. Eval. of Computer Systems, Vienna (February, 1979).

Buze73 J.P. Buzen, Computational algorithms for closed queueing networks with exponential servers, *Comm. ACM 16*, 9 (September, 1973), 527-31.

JaLa82 P.A. Jacobson and E.D. Lazowska, Analyzing queueing networks with simultaneous resource possession, *Comm. ACM 25*, 2 (February 1982) 142-51.

LaZa82 E.D. Lazowska and J. Zahorjan, Multiple class memory constrained queueing networks, Proc. ACM SIGMETRICS Conf. on Meas. and Mod. of Computer Systems, Seattle (August 1982).

NeCh81 D. Neuse and K.M. Chandy, SCAT: A Heuristic algorithm for queueing network models of computer systems, *Perf. Eval. Rev. 10*, 3 (Fall, 1981), 59-79.

SaCh80 C.H. Sauer and K.M. Chandy, *Computer Systems Performance Modeling*, Prentice-Hall (1981).

ReLa80 M. Reiser and S.S. Lavenberg, Mean value analysis of closed multichain queueing networks, *J.ACM 27*, 2 (April 1980), 313-22.

Schw79 P. Schweitzer, Approximate analysis of multiclass closed networks of queues, Int. Conf. on Stoch. Control and Optimization, Amsterdam (1979).

Sevc77 K.C. Sevcik, Priority scheduling disciplines in queueing network models of computer systems, *Proc. IFIP 77* (August, 1977), 565-70.

Sevc81 K.C. Sevcik, Data base system performance prediction using an analytical model, Proc. 7th VLDB Conf., Cannes (September 1981), 182-89.

"Improving Organizational Productivity"

Performance Monitoring Techniques

DESIGN OF EMBEDDED COMPUTER MONITORING SYSTEM

Abundio Alvarez
Naval Ocean Systems Center
San Diego CA

The design and adaptation of an embedded monitoring system in a military processor has been quite a challenge. Accurate definitions and analysis of monitoring requirements for the embedded system has been one of the more difficult issues confronting the system performance analysts and designers. Designing an embedded Computer Monitoring System for the standard Navy computer is further compounded by the constraints on size and environmental protection.

1. Introduction

Computer systems currently being introduced in the military community have one thing in common; they have virtually no provisions by which performance measurements can be taken using commercially available hardware monitors. The Naval Ocean System Center therefore had to define and develop a new monitoring system. This system was designed to be a high impedence embedded monitoring system for the purpose of providing passive monitoring of selected signals of the Navy's AN/UYK-20 computer and for providing these signal sources to the Monitor Interface Adaptor (MIA) component of a large Data Extraction/Data Reduction Processor.

2. Design Objectives

The following design objectives were established as requirements and constraints governing the design of the Embedded Monitoring System (hereafter referred to as monitor).

a. The monitor shall have no impact nor affect to the normal operation of the AN/UYK-20 computer.

b. The monitor shall be housed within the frame (with no external protrusions) of the AN/UYK-20 computer.

c. There shall be no physical modifications nor structured changes made to the AN/UYK-20 computer to accommodate the monitor.

d. The monitor shall be capable of providing a signal output sufficient to drive 50 feet of twisted pair cable while maintaining TTL compatible signal recognition characteristics.

e. The monitor shall provide for the passive monitoring of 256 discrete signals from the AN/UYK-20 computer plus 16 address lines and an address strobe line.

3. Design Approach

Receiver Selection

The low power Schottky Octal Buffer circuits that were selected provided the required TTL output signal at the highest input impedence (Z) of any currently available integrated circuit. These integrated circuits were used as receive circuits for the sensing of selected AN/UYK-20 computer data. The high density packaging and tri-state output were another reason for the selection of this particular integrated circuit chip. The high input impedence of these integrated circuits also allowed for the multiplexing of data outputs in a bus arrangement.

3.1 Power Independence

The monitor was designed using an independent power supply to provide the required voltage to the circuit boards. The power supply is also configured with an in line filter to eliminate external primary power source line noise. Implementation of a separate power supply for the monitor assures no common mode noise injection into the AN/UYK-20 computer circuits and imposes no current load to the AN/UYK-20 computer internal power supplies.

3.2 Logical Organization

The monitor was designed to accommodate a total of 256 discrete data signals for transmission to the MIA. This is in addition to the 16 address (address register) lines and the address strobe line. The monitor is organized in a multiplex fashion wherein there are 8 groups of 32 bits each which may be selected for data acquisition. That is, the MIA is restricted in selecting only one group of 32 bits of data for monitoring at any one time. This constraint was imposed on the designer by the number of connector pins available on the AN/UYK-20 DMA plug which is used as the I/O port to the MIA.

3.3 Physical Organization

Figures 1 through 3 depict the monitor and its interconnection with the CPU/IOC chassis of the AN/UYK-20 computer.

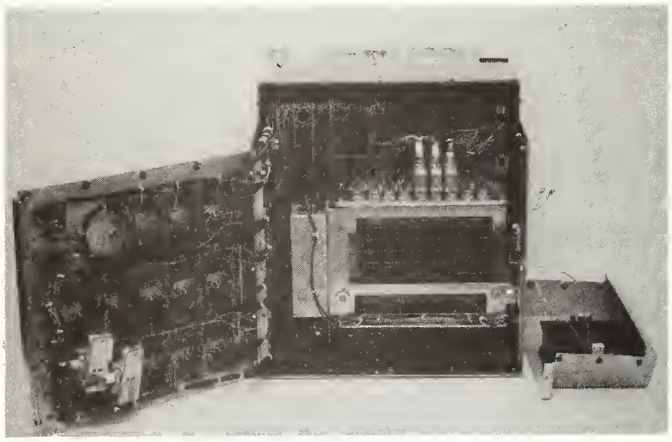


Figure 1

Figure 2

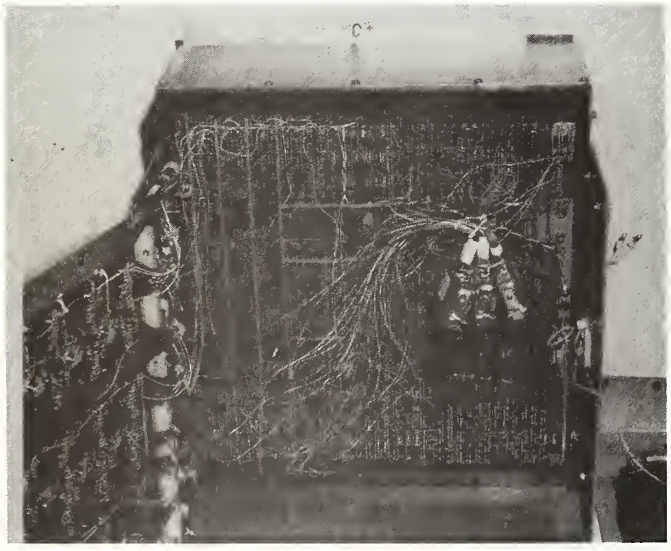


Figure 3

The monitor is comprised of two general purpose IC boards with ground and voltage planes on each board. There are 60 ICs mounted on the two boards. This population provides for the full 256 discrete signal multiplexing capability, the address data and for some additional AN/UYK-20 signals reconstitution. Connection between the monitor, the AN/UYK-20 computer and the MIA is provided via the nine, 50-pin connectors mounted on top of the monitor. The first two connectors provide for communication between the monitor and the MIA. The remaining seven connectors are available for signals acquisition from the AN/UYK-20 computer. As can be seen from Figure 3 only partial AN/UYK-20 signals acquisition has been implemented.

Monitor connection to the AN/UYK-20 computer is by means of hand wire wrapping to selected data points on the backplane of the CPU/IOC chassis. Due to the critical nature of the high repetition rate and pulse duration of the address Register and address Strobe signals these points were wired using twisted pair solid conductor 28ga. wire. All other data points were acquired using single conductor 28ga. wire.

Primary power for the monitor was brought in to the AN/UYK-20 chassis through the existing external Real Time Clock (RTC) access port. The RTC connector was replaced with an appropriate power connector. No chassis modification was required. The I/O connection between the monitor and MIA uses the existing AN/UYK-20 Direct Memory Access (DMA) port. No AN/UYK-20 computer modifications were allowed or required.

The prototype monitor is a wire wrap assemblage between the various 14, 16 and 20 pin DIP ICs. The monitor is attached to the frame of the AN/UYK-20 computer CPU/IOC chassis by means of existing AN/UYK-20 mounting screws.

3.4 Signals Reconstruction

Approximately one-half of the signals that were selected for data extraction from the AN/UYK-20 computer were not available on the backplane of the computer in the composite form that they were required. Therefore, the monitor design provided for reconstitution of these signals through implementation of logical operations (i.e., "ANDs", and "ORs", latches, decoders, etc.). Some of those signals which required reconstitution were:

a. IDAs for channels 0 – 17

b. EFAs for channels 0 – 17

c. Jump and Link Register (42) instruction

d. Branch Satisfied condition

3.5 Data Transmission Techniques

The driver circuits used in the monitor for transmitting the acquired data to the MIA are the same low power Schottky Octal Buffers that were used as data receivers. The low output impedence of this chip provided a high current drive capability of single ended TTL level transmission lines. Tests were conducted to determine the drive capability of the circuit using 60 feet of 26ga. twisted pair cable without any appreciable degradation to the signal.

3.6 Selection of Data Points

The data points of the AN/UYK-20 computer selected for monitoring and their associated bit positions were grouped into multiplexer groups for transmission to the MIA. Some of the discrete signals identified were selected rather arbitrarily but many were believed to be able to provide important indicators of computer performance. As can be seen, there is room for many more data points to be wired to the AN/UYK-20 computer. At present only 83 of the possible 256 discrete points were wired. Also wired are the 16 address Register address bits and address Strobe independent of the 256 multiplexer bits.

The I/O signals selected for monitoring provide for acquiring the four possible types of activities which are available on each of the 16 I/O channels, i.e.,

a. Input data

b. Output data

c. External functions

d. External interrupts

Through these signals the MIA can:

a. Count number of words input

b. Count number of words output

c. Count number of external functions of any or all channels

d. Count number of interrupts on any or all channels

e. Time all of the above events and determine percentages of specific I/O activity

4. Operational Validation

The monitor was bench tested prior to installation within the AN/UYK-20 computer. Subsequent to this, the monitor was tested on-line in conjunction with the AN/UYK-20 computer, the MIA and the MIA Test Fixture (Future Data Microcomputer System). The monitor (Figure 4) was debugged and final testing validated proper operational performance, as verified by MIA test report outputs matched with AN/UYK-20 simulation test program results.

On-line testing of the monitor with the MIA system was performed and documented.

5. Future Direction

The PCOTES Hardware Monitoring System shown in Figure 5 depicts the final objectives of the implementation of the Embedded Monitoring System.

That portion outlined by the dotted line was completed and tested. The Navy program responsible for the development of the overall system was cancelled. This system was to have included four DEC VAX 11/780 systems with approximately thirty MIA units, along with ten MUX and HPM port processors with a number of support devices working on a 50 Mh Hyperchannel Bus.

NOSC has since developed the MIA into a dedicated stand alone Hardware Monitoring System for the USQ-20 computer. The development of this unit cost the government about 160K, or two man-years, for both hardware and software. NOSC is now in the process of developing a new system with enhanced capabilities for around 50K a copy.

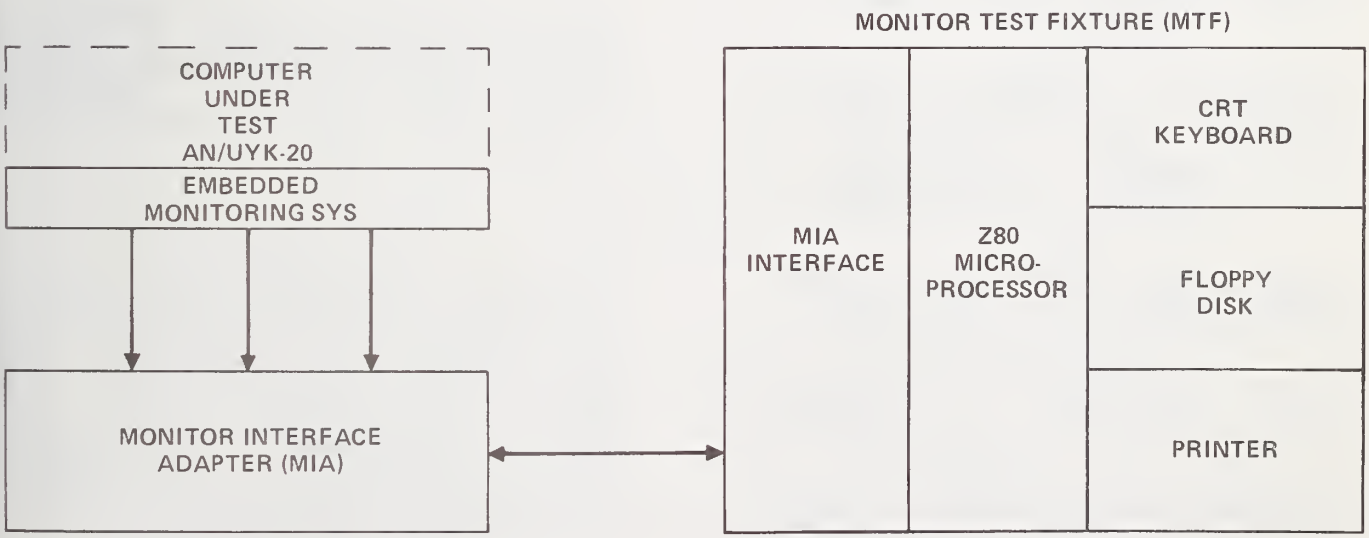


Figure 4. NOSC Micro Monitor

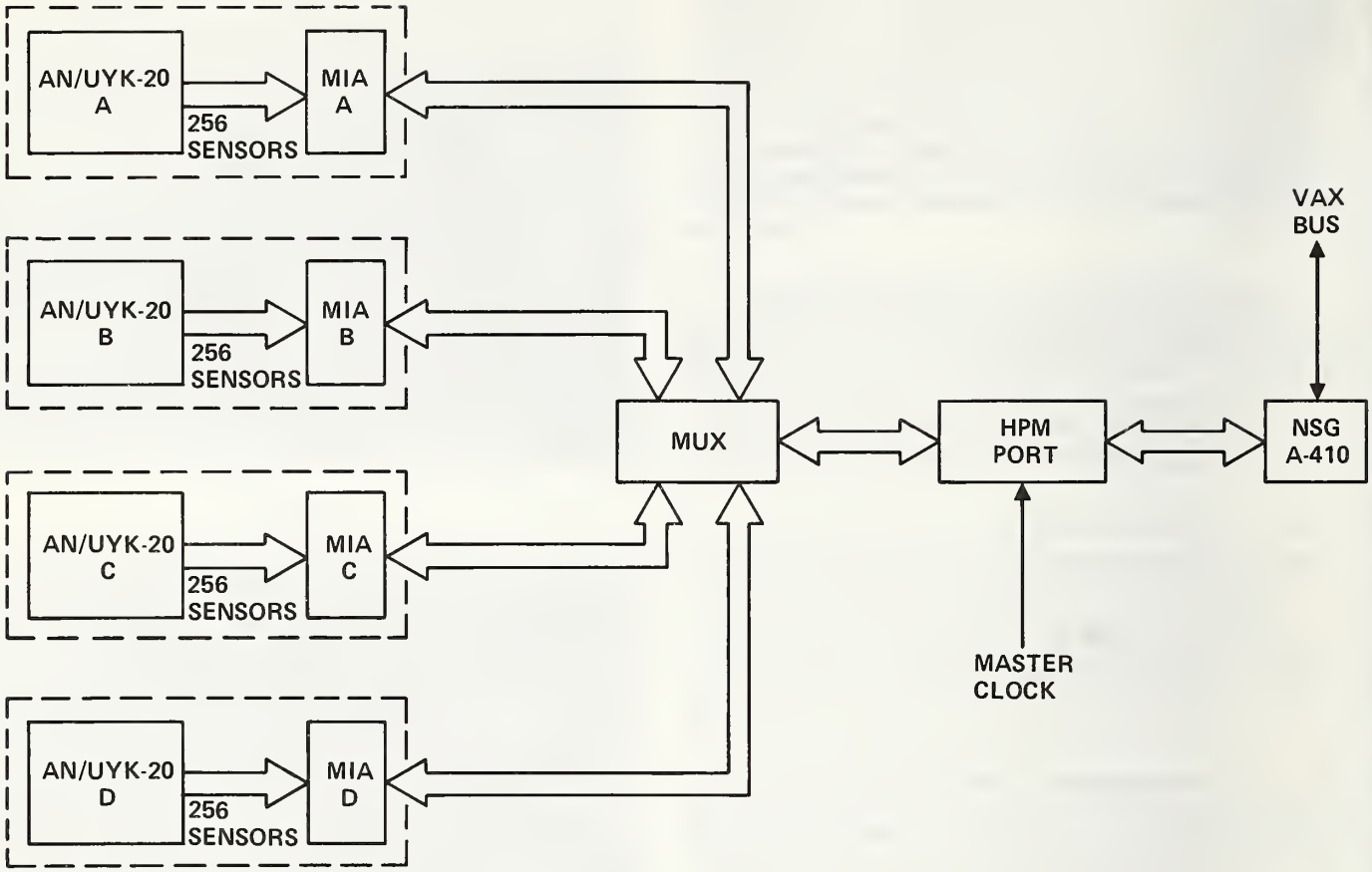


Figure 5. PCOTES Hardware Monitoring System

PROGRAM INSTRUMENTATION TECHNIQUES

Raymond C. Houghton, Jr.

National Bureau of Standards
Institute for Computer Sciences and Technology
Washington, DC 20234

Many tools that perform dynamic analysis of computer programs modify (or instrument) the programs by inserting probes. Typical analyses that are performed by these tools include coverage, tuning, timing, tracing, and assertion analysis. There are four common instrumentation techniques for higher level languages: global, local, trace, and buffered trace. These four techniques are examined for performance differences on a DECSYSTEM-10.

Key words: Coverage analysis; dynamic analysis; performance monitoring; program analysis; program instrumentation; software tools.

1. Introduction

Several dynamic analysis features [Houg81] of software development tools can be implemented by instrumentation techniques. Table 1 lists these features along with their definitions.

Note: Certain commercially developed products are identified in this paper in order to provide a characterization of their instrumentation features for certain types of programs. In no case does such identification imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the product identified is necessarily the best for the purpose.

Assertion Checking - checking of user-embedded statements that assert relationships among elements of a program.

Coverage Analysis - determining and assessing measures associated with program structural elements to determine the adequacy of a test run.

Timing - reporting actual CPU, wall-clock, or other times associated with parts of a program's execution.

Tracing - monitoring the historical record of execution of a program.

Tuning - determining what parts of a program are being executed the most.

Table 1. Dynamic Analysis Features

The most common feature that is familiar to most programmers is tracing. Tracing is provided as a feature by many compilers. Compilers instrument software by including breakpoints in a user's program so that execution can be interrupted to examine program behavior. This type of instrumentation is added to the object level of a program and requires close cooperation with the run-time system. Consequently, this technique is very reliant on the hardware characteristics of the host system.

Tracing can also be implemented by using a machine-independent instrumentation technique where the original program is transformed to an instrumented program that is written at the same level as the original program. This type of instrumentation is performed by a tool that is commonly called a pre-processor. Figure 1 shows the typical relationship with a compiler. The figure shows that the pre-processor produces a modified version of the subject program which when compiled produces output which is analyzed by a tool called a post-processor. If tracing information is requested by the user, then the post-processor sorts through the trace data produced by the modified version of the subject program, formats the information, and presents it to the user.

Whereas tracing is a feature that is used for debugging programs, coverage analysis is a feature that is used for testing them. Although there are many testing techniques [Adri81], a common technique is to cover a certain percentage of program statements, branches, or paths [Mill77]. A guideline used by many software tool developers is to obtain 85 per cent branch coverage in the software that they develop. To determine measures such as this, there are many coverage analysis tools available. [Houg82] lists 40 tools that measure coverage for several programming languages including FORTRAN, JOVIAL, COMPASS, COBOL, PL/1, AND Pascal.

2. Instrumentation Techniques

There are four common instrumentation techniques for modifying a subject program to collect data for features such as coverage analysis. These are classified as trace, buffered trace, local, and global instrumentation. These instrumentation techniques should not be confused with the features that they implement. For example, trace instrumentation can be used to implement the feature called "tracing," but it can also be used to implement "assertion checking," "coverage analysis," "timing," and "tuning."

Figure 1. Block Diagram of a Dynamic Analysis Tool

2.1 Trace Instrumenation

Example 2 shows the trace instrumented version of the program shown in example 1. In example 2, a FORTRAN program is instrumented so that a subroutine called TZZ4QX is called with trace data passed as a parameter. The function performed by this subroutine is to output the trace data. After a completed execution of the instrumented program, a file is generated that is composed of trace data produced by this subroutine. A post-processor uses this file or a number of these files to produce reports that characterize the execution of the subject program.

```
C    PROGRAM THAT COMPUTES THE MEAN AND THE STANDARD DEVIATION OF
C    A SEQUENCE OF NON-ZERO NUMBERS TERMINATED BY A NEGATIVE NUMBER.
C
      PROGRAM COMPUTE

      REAL  X,MEAN,STDEV,SUM1,SUM2
      INTEGER  K
C
      SUM1 = 0.0
      SUM2 = 0.0
      K = 0
    1 READ(5,100) X
  100 FORMAT(F10.3)
      IF (X .LT. 0.0) GOTO 2
      K = K + 1
      SUM1 = SUM1 + X
      SUM2 = SUM2 + (X ** 2.0)
      GOTO 1
    2 MEAN = SUM1 / K
      STDEV = SQRT( (SUM2 / K) - (MEAN ** 2.0))
      WRITE(6,200) MEAN,STDEV
  200 FORMAT(' THE MEAN IS ',F10.3,'   THE STANDARD
     *       DEVIATION IS ',F10.3)
      STOP
      END
```

Example 1. An Uninstrumented Program

```
      PROGRAM COMPUTE
      REAL  X,MEAN,STDEV,SUM1,SUM2
      INTEGER  K
      CALL TZZ4QX(    1)
      SUM1 = 0.0
      SUM2 = 0.0
      K = 0
    1 CALL TZZ4QX(    2)
      READ(5,100) X
  100 FORMAT(F10.3)
      IF(X.LT.0.0)THEN
      CALL TZZ4QX(    3)
      GOTO 2
      ENDIF
      CALL TZZ4QX(    4)
      K = K + 1
      SUM1 = SUM1 + X
      SUM2 = SUM2 + (X ** 2.0)
      GOTO 1
    2 CALL TZZ4QX(    5)
      MEAN = SUM1 / K
      STDEV = SQRT( (SUM2 / K) - (MEAN ** 2.0))
      WRITE(6,200) MEAN,STDEV
  200 FORMAT(' THE MEAN IS ',F10.3,'   THE STANDARD
     *       DEVIATION IS ',F10.3)
      CALL RZZ4QX
      END
      SUBROUTINE TZZ4QX(NSEG)
      WRITE(13) NSEG
      RETURN
      END
```

Example 2. Trace Instrumentation

Trace instrumentation is the simplest of all instrumentation techniques because it requires a minimal transformation of the subject program. No storage or processing of trace data is required by the instrumented program, and all data is provided to the post-processor for analysis. Since the instrumented program does not have a storage requirement, it is generally assumed that the resulting instrumented program is not as bulky as the other three techniques. The impact of this storage requirement is examined in section 3.1.

2.2 Buffered Trace Instrumentation

This technique introduces storage requirements in the instrumented program. Rather than perform output at each probe in a program, trace data is stored in a buffer until it is filled. The information is then output. Example 3 shows buffered trace instrumentation. Buffered trace complicates instrumentation because the pre-processor must identify the

termination points in a program. This is necessary so that the buffer can be purged prior to the actual termination of the instrumented program. The advantage, however, is a reduction in output processing overhead. Section 3.2 examines this reduction further.

```
      PROGRAM COMPUTE
      REAL  X,MEAN,STDEV,SUM1,SUM2
      INTEGER  K
     CALL TZZ4QX(    1)
      SUM1 = 0.0
      SUM2 = 0.0
      K = 0
    1 CALL TZZ4QX(    2)
      READ(5,100) X
  100 FORMAT(F10.3)
     IF(X.LT.0.0)THEN
     CALL TZZ4QX(    3)
      GOTO 2
     ENDIF
     CALL TZZ4QX(    4)
      K = K + 1
      SUM1 = SUM1 + X
      SUM2 = SUM2 + (X ** 2.0)
      GOTO 1
    2 CALL TZZ4QX(    5)
      MEAN = SUM1 / K
      STDEV = SQRT( (SUM2 / K) - (MEAN ** 2.0))
      WRITE(6,200) MEAN,STDEV
  200 FORMAT(' THE MEAN IS ',F10.3,'   THE STANDARD
     *       DEVIATION IS ',F10.3)
     CALL RZZ4QX
     END
     SUBROUTINE TZZ4QX(NSEG)
     DIMENSION ICIRCL( 100)
     DATA ICOUNT / 1 /
     ICIRCL(ICOUNT)=NSEG
     IF (ICOUNT.EQ.100) THEN
     WRITE(13) (ICIRCL(I),I=1,100)
     ICOUNT=1
     END IF
     RETURN
     END
```

Example 3. Buffered Trace Instrumentation

2.3 Local Instrumentation

For certain types of dynamic analysis such as coverage analysis, tuning, and assertion checking, it is possible to perform additional processing of the trace data within the instrumented program. This eliminates the need to output trace data. Example 4 shows that in the monitoring routine a counter is incremented each time the routine is invoked. Each counter is stored locally in the subroutine and may represent the number of times a statement, branch, or false assertion is executed. Consequently, the monitoring subroutine may have a storage requirement equal to the number of statements, branches, or assertions in the subject program. The impact of this storage requirement for branch coverage is examined further in section 3.1. Like buffered trace instrumentation, local instrumentation requires the identification of termination points.

```
      PROGRAM COMPUTE
      REAL  X,MEAN,STDEV,SUM1,SUM2
      INTEGER  K
     CALL TZZ4QX(    1)
      SUM1 = 0.0
      SUM2 = 0.0
      K = 0
    1 CALL TZZ4QX(    2)
      READ(5,100) X
  100 FORMAT(F10.3)
     IF(X.LT.0.0)THEN
     CALL TZZ4QX(    3)
      GOTO 2
     ENDIF
     CALL TZZ4QX(    4)
      K = K + 1
      SUM1 = SUM1 + X
      SUM2 = SUM2 + (X ** 2.0)
      GOTO 1
    2 CALL TZZ4QX(    5)
      MEAN = SUM1 / K
      STDEV = SQRT( (SUM2 / K) - (MEAN ** 2.0))
      WRITE(6,200) MEAN,STDEV
  200 FORMAT(' THE MEAN IS ',F10.3,'   THE STANDARD
     *       DEVIATION IS ',F10.3)
     CALL RZZ4QX
     END
     SUBROUTINE TZZ4QX(NSEG)
     COMMON / CZZ4QX / IZZ4QX(    5)
     INTEGER IZZ4QX
     IZZ4QX(NSEG)=IZZ4QX(NSEG)+1
     RETURN
     END
```

Example 4. Local Instrumentation

2.4 Global Instrumentation

Unlike local instrumentation where counters are stored locally in the monitoring subroutine, global instrumentation stores the counters globally and does not require calls to a monitoring subroutine. Consequently, global instrumentation eliminates the linkage overhead that is incurred by the other instrumentation techniques during the execution of the instrumented program. Global instrumentaion, on the other hand, is the most complicated of the techniques. Besides the insertion of probes in the instrumented program, global data specifications (Block Common in FORTRAN) must also be inserted throughout the instrumented program.

```
      PROGRAM COMPUTE
     COMMON / CZZ4QX / IZZ4QX(    5)
     INTEGER IZZ4QX
     SAVE /CZZ4QX/
      REAL  X,MEAN,STDEV,SUM1,SUM2
      INTEGER  K
     IZZ4QX(    1) = IZZ4QX(    1) + 1
      SUM1 = 0.0
      SUM2 = 0.0
      K = 0
   1 IZZ4QX(    2) = IZZ4QX(    2) + 1
      READ(5,100) X
 100  FORMAT(F10.3)
     IF(X.LT.0.0)THEN
     IZZ4QX(    3) = IZZ4QX(    3) + 1
      GOTO 2
     ENDIF
     IZZ4QX(    4) = IZZ4QX(    4) + 1
      K = K + 1
      SUM1 = SUM1 + X
      SUM2 = SUM2 + (X ** 2.0)
      GOTO 1
   2 IZZ4QX(    5) = IZZ4QX(    5) + 1
      MEAN = SUM1 / K
      STDEV = SQRT( (SUM2 / K) - (MEAN ** 2.0))
      WRITE(6,200) MEAN,STDEV
 200 FORMAT(' THE MEAN IS ',F10.3,'   THE STANDARD
     *        DEVIATION IS ',F10.3)
     CALL RZZ4QX
     END
```

Example 5. Global Instrumentation

3. Performance of Instrumentation Techniques

This section discusses the performance of the four types of instrumentation techniques for certain subject programs. Three tools were used to instrument the programs. Trace instrumentation statistics were obtained using the Automatic Testing Analyzer (ATA) developed by Science Applications, Inc. Global instrumentation and buffered trace instrumentation* data were obtained using the Node Determination and Analysis Program (NODAL) developed by TRW, Inc. Versions of both of these tools have been developed under contract to various agencies within the Government. Local instrumentation statistics were obtained using the NBS analyzer [Lyon74].

Table 2 provides some of the characteristics of the subject programs that were examined. All programs and tools were executed on a DECSYSTEM-10 under the TOPS-10 Operating System. The subject programs were selected from various experimental programs and various software tools available on the DECSYSTEM-10. Because of various language and portability constraints, data was not obtained for every subject program using every instrumentation technique.

During the experimentation and subsequent analysis, an attempt was made to obtain generalized results. However, it was found that each subject program displayed such different execution

---

* Buffered trace statistics were obtained using a slightly modified version of the instrumentation that is provided by NODAL for tracing.

Subject Program No.	No. of Statements	Per cent Comments	No. of Segments	Internal Data Storage
3	38	0%	14	Very Large
2	49	0%	27	Small
4	65	8%	32	Small
7	410	32%	170	Very Large
8	749	55%	177	Very Large
6	2290	40%	792	Small
11	9040	50%	1992	Large

Table 2. Characteristics of the Subject FORTRAN Programs

characteristics that statistically significant results could not be obtained for the generalizations. Consequently, the sections that follow discuss typical cases from the experiments along with a brief discussion of some of the worst case results.

3.1 Storage Effects

Table 3 provides the storage requirements for the instrumented and uninstrumented subject programs. Programs 6 and 2 show a larger percentage increase in storage for the instrumented programs because they have smaller internal data storage than programs 7 and 8 (see table 2). The first trend to notice is that the storage requirements for local instrumentation is consistently higher than global instrumentation. This trend makes sense because local instrumentation performs a call to a monitoring routine at each branch. Each call along with its associated parameters requires extra storage. The next trend to notice is that for programs 6 and 8, buffered trace takes less storage than trace instrumentation. Although this trend does not seem logical, it can be explained by the slightly different instrumentation techniques used by NODAL and ATA. The most important trend to notice is that global instrumentation does not necessarily require more storage than trace instrumentation. This seems counter-intuitive because, as it was pointed out in section 2.1, trace instrumentaton does not have a storage requirement for the execution statistics. This trend seems to be based on a stand-off between the storage required for subroutine calls versus the storage required for counters.

3.2 Output Effects

Table 4 provides output statistics for the instrumented and uninstrumented subject programs. The important trend to notice here is the large percentage increase in output produced by trace instrumentation. This trend is expected since the monitoring routine outputs the trace data each time it is called. Also notice that buffered trace instrumentation does reduce the amount of output. The output from program 8 was cut by 75 per cent. Program 4 was cut by 86 per cent. Since output from local and global instrumentation is relatively fixed, the effect on the instrumented program will depend on the uninstrumented output. Program 3 shows the worst case that was found in the data. This is a small program that produces very little output and has a relatively large run-time.

Program No.	No Inst.	Trace Inst.	% Incr.	Buff Trace Inst.	% Incr.	Local Inst.	% Incr.	Global Inst.	% Incr.
6	20531	32253	57.09	31342	52.66	37116	80.78	27377	33.34
2	11016	14456	31.23	-----	-----	15737	42.86	13565	23.14
7	27766	31011	15.86	-----	-----	32247	20.48	31242	16.72
8	352125	355015	.82	354443	.66	356451	1.23	354420	.65

Table 3. Instrumentation Storage Effects

Program No.	No Inst.	Trace Inst.	% Incr.	Buff Trace Inst.	% Incr.	Local Inst.	% Incr.	Global Inst.	% Incr.
4	6	2468	41200	345	5650	7	17	10	67
8	123	19728	15939	4903	3886	130	6	126	2
7	49	360	635	----	----	60	22	59	20
11	417	-----	-----	8314	1893	---	--	504	21
3	1	-----	-----	6316	631500	6	500	8	700

Table 4. Instrumentation Output Effects

3.3 CPU Time Effects

Table 5 provides CPU time statistics. Because CPU time is based on several variables, including program size, input/output, and run-time, it is the most difficult to establish consistent trends. However, Table 5 does show CPU time is less for each technique as you examine the data from left to right. The percentage of reduction is highly variable, however. For example, CPU time from local to global instrumentation is reduced by 35 per cent for program 4, but only 8 per cent for program 7. The worst case results were again obtained by program 3. An interesting anomaly occurred with program 8, where the global instrumentation time was actually less than the uninstrumented time. The explanation for this behavior seems to lie in a DECSYSTEM-10 timing quirk where a better input/output and run-time balance can improve CPU time.

4. Conclusion

In this paper, four types of instrumentation techniques were defined and examined using a small sample of subject programs. Several trends were observable from the data presented and all of them, except possibly storage effects, pointed to global instrumentation as the most efficient instrumentation technique. Although this trend may be true in general, there are several issues that should be examined before one embraces global instrumentation. The first, which was mentioned in section 2.1, is simplicity. Trace instrumentation is the easiest to implement. The second is separate instrumentation. For very large programs it is desirable to instrument only parts of a program. This is easily accomplished with both instrumentation techniques, however, global instrumentation requires some extra bookkeeping that trace instrumentation

Program No.	No Inst.	Trace Inst.	% Incr.	Buff Trace Inst.	% Incr.	Local Inst.	% Incr.	Global Inst.	% Incr.
6	14	--	--	53	279	28	100	20	43
7	11	47	327	--	---	13	18	12	9
4	63	391	520	103	63	100	59	65	3
2	3.7	19.89	438	---	--	---	--	4.46	21
3	.19	21.47	11210	---	--	---	--	.44	132
8	65	2806	4217	536	725	245	277	64	-2

Table 5. Instrumentation CPU Time Effects

does not require. The programming environment also plays a key role. A batch environment with sufficient storage space and available idle time can easily accomodate trace instrumentation. However, an interactive environment with tight schedules and a requirement for quick turnaround would probably prefer the use of global instrumentation. Finally, the language being instrumented is also an issue. FORTRAN is considered a "flat" language that has different execution characteristics than more "contoured" languages such as Pascal or ALGOL. Global referencing in these languages can have expensive run-time implementations depending on efficiency trade-offs made in the compiler.

5. References

[Adri81] W. R. Adrion, M. A. Branstad, and J. C. Cherniavsky, "Validation, Verification, and Testing of Computer Software", NBS Special Publication 500-75, February 1981.

[Houg81] R. Houghton, "Features of Software Development Tools", NBS Special Publication 500-74, February 1981.

[Houg82] R. C. Houghton, Jr., "Software Development Tools", NBS Special Publication 500-88, March 1982.

[Lyon74] G. Lyon and R. B. Stillman, "A FORTRAN Analyzer", NBS Technical Note 849, October 1974.

[Mill77] E. F. Miller, Jr., "Program Testing: Art Meets Theory", Computer, July 1977.

CPEUG82

"Improving Organizational Productivity"

UNIX Performance Analysis

PERFORMANCE PREDICTION IN A UNIX ENVIRONMENT

Lawrence W. Dowdy, Lindsey E. Stephens, and Alfredo Perez-Davila

Computer Science Department
Vanderbilt University
Nashville, Tennessee 37235

Computer system performance prediction addresses the question, "How and by how much will a certain hardware or software change affect the performance of a given computer system?" The difficulties in being able to accurately predict this performance stem from: (a) not having measurement data on the needed parameters, and (b) not knowing how the parameters change with respect to each other as the system changes.

This paper describes a technique to predict the performance of a system as the number of users increases. This involves predicting the user response time and the user throughput, as a function of the number of users. The thrashing point, where throughput and response times severely deteriorate due to program swapping overhead, is also predicted.

The performance prediction technique is validated on a Perkin-Elmer 3220 running UNIX. To handle the data measurement and parameter interdependency problems, a performance monitor and a synthetic workload are constructed.

Key words: Performance prediction; queuing theory; UNIX; validation.

1. Motivation

In capacity planning, the system manager is faced with the problem of determining when to perform a system upgrade and by how much the system should be upgraded. For budgetary reasons, the manager needs to know this information well in advance of when the system will be reconfigured. The manager's goal is to maintain a system at a desired performance level (e.g., a given response time or throughput threshold) under increasing workload demands. For these reasons, the manager needs to be able to accurately predict the performance of the current system as its workload increases in order to determine the point in time at which the performance is unacceptable and a system upgrade becomes necessary.

This performance prediction task is nontrivial. A scenario for performing this task is as follows. First, an abstract model of the system is constructed. The model parameters (e.g., device loadings, device service rates) are then obtained. The model is solved (e.g., analytically, via simulation) to obtain performance metrics such as user response times and throughputs. These metrics are then validated against the measured performance metrics. Once validated, the model parameters are altered to reflect the workload increase. The model is resolved and the metrics obtained are used as the performance prediction at the increased workload.

The most difficult steps in this scenario are the obtaining and the altering of the parameters. As a result, relatively few performance prediction studies and techniques have been reported [1, 4, 5, 6, 7, 10, 12, 13].

In this paper, a technique is given for obtaining the predicted throughput versus workload curve and the predicted response time versus workload curve. The presentation of the technique is facilitated by describing an explicit example. The example environment and the model are described in Section 2. The model parameterization is described in Section 3. Section 4 presents the prediction technique which is then validated in Section 5. Conclusions are given in Section 6.

SYNBEN was run with the above assumptions with five virtual terminals running. From IOUmon the following base model parameters were observed.

$$\mu_{CPU} = \frac{\#\ of\ CPU\ transactions}{CPU\ busy\ time} = 19.5\ transactions/second$$

$$\mu_{SD} = \frac{\sum system\ file\ transactions}{\sum system\ disk\ busy\ time} = 54.0\ transactions/second$$

$$\mu_{UD} = \frac{\sum user\ file\ transactions}{user\ disk\ busy\ time} = 54.0\ transactions/second$$

$$\mu_{term} = \frac{1}{mean\ think\ time} = 0.06536\ transactions/second$$

$$V_{CPU} = \frac{\#\ of\ CPU\ transactions}{monitoring\ period} = 8.3450\ transactions/second$$

$$V_{SD} = \frac{\sum system\ file\ transactions}{monitoring\ period} = 0.7194\ transactions/second$$

$$V_{UD} = \frac{\sum user\ file\ transactions}{monitoring\ period} = 7.3512\ transactions/second$$

$$V_{term} = \frac{\sum terminal\ responses}{monitoring\ period} = 0.2744\ transactions/second$$

$$DMP = 5$$

Since the visit ratios (i.e., V's) are relative to each other, we choose to normalize them relative to the monitoring interval so that they correspond to the actual throughputs. The parameterized system model is shown in Figure 2.

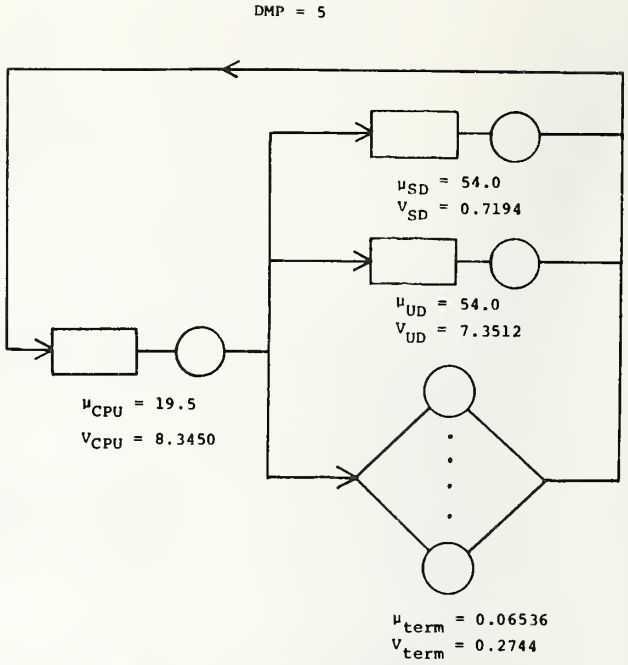


Figure 2: Parameterized System Model

3.4 Model Validation

The parameterized model was solved using mean value analysis techniques [11]. The model reported a throughput of 0.2751 responses per second. The observed throughput was 0.2744 responses per second for an error of less than 0.5%. This validated model formed the basis for performance predictions at increased workloads.

4. Predictions

Once a system model has been constructed, parameterized, and validated, performance prediction of the "high-level" parameters (e.g., throughputs, response times) reduces to the prediction of the "low-level" parameters (e.g., μ's, V's, DMP). If the low level parameters are accurately predicted, and the underlying queuing theoretical assumptions (e.g., work conserving, product form) are valid, then the predicted high-level performance metrics will be accurate because of the provably correct functional relationships between the high and low-level parameters.

4.1 Methodology

A methodology for predicting the low-level parameters is stated and then particularized for the PE-3220 example.

1. *Identify the control variable.* This variable will be directly affected as the system changes. If the scheduling algorithm at the CPU is being changed, μ_{CPU} would be the control variable. If files were being shifted from one device to another, the device visit ratios would be the control variables.

2. Environment and Model

The example environment is a Perkin-Elmer 3220 with 960K bytes of primary memory, running version 7 of UNIX. A central server queuing network model describing the system is shown in Figure 1.

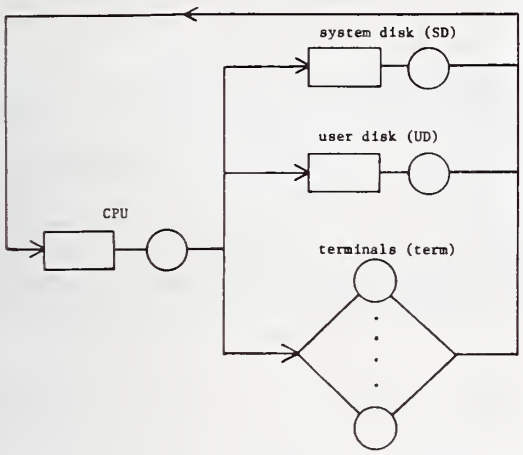


Figure 1: System Model

Standard queuing theoretical assumptions are made [3]. The CPU is a single processor modeled as a load independent, processor-shared server. Each disk has its own selector channel and each is modeled as a load independent first-come-first-serve server. The file system is configured so that all system file activity (e.g., swapping, accounting, library calls) is routed to the system disk, while all user related file activity is routed to the user disk. The terminals are modeled as an infinite server.

3. Model Parameterization

The parameters needed to characterize the system model are:

- the mean service rate for each server: μ_{CPU}, μ_{SD}, μ_{UD}, μ_{term},
- the relative visit counts for each server: V_{CPU}, V_{SD}, V_{UD}, V_{term}, and
- the degree of multiprogramming: DMP.

Each parameter is workload dependent. In order to run controlled experiments for validation purposes, a constant workload is needed.

3.1 Workload

To provide a constant workload, a synthetic benchmark, SYNBEN, is constructed. SYNBEN is a program written in C which places a representative workload on the system. Input parameters to SYNBEN include:

- the number of "virtual" terminals to be represented,
- the think time distribution for each terminal,
- the size of each terminal's program,
- the distribution for the CPU requirements for each terminal's requests, and
- the distribution for the user disk requirements for each terminal's requests.

The operation of SYNBEN is as follows. The parameters are read from the user. The appropriate number of virtual terminal jobs are created. Each virtual terminal job uses a random amount of CPU time, based upon the given CPU service time distribution. Each virtual terminal job selects whether or not a user disk I/O is to be performed, and, if so, selects a random number of blocks to transfer. If the job does not choose to do I/O, it returns to the virtual terminal for a random think time. In this case, information consisting of the virtual terminal number, a time stamp, and the think time is recorded, and the virtual terminal job goes to sleep for its allotted think time. The job then wakes up and continues as before.

Synthetic workloads have been proposed previously [2, 8]. They provide an alternative to traditional benchmarks while being representative, easy to use, and flexible.

3.2 System Monitor

In order to obtain the necessary parameters (μ's, V's, and DMP) for the system model, an instrumentation of UNIX monitor, IOUmon, is constructed. IOUmon is written in C and has "hooks" within the UNIX operating system code. The parameters currently measured by the IOUmon include:

- the busy times of the CPU and disks,
- the number of transactions (i.e., blocks) which are transferred to each file system (e.g., swapping, system overhead files, user files), and
- the distribution of the degree of multiprogramming.

IOUmon's tables and code are kept in primary memory and consume less than 3K bytes. The overhead incurred by running IOUmon is less than 0.5% CPU time.

3.3 Parameterized Model

To predict performance at various workload levels, a specific benchmark is needed upon which to base the predictions. Typical user behavior is used to obtain the following SYNBEN parameters.

- Terminal think times come from a truncated normal distribution with a mean of 15.3 seconds [9].
- The mean program size is 64K bytes.
- The user (not system) CPU time required between successive physical I/Os is exponentially distributed with a mean of 280ms.
- The number of user disk requests required per terminal request is geometrically distributed with a mean of 4 requests.
- Each user disk request requires an exponentially distributed number of blocks to be transferred with a mean of 7 blocks.

2. *Identify those parameters which are independent of the control variable.* These variables will not change when the control variable does. If another CPU is being added, μ_{CPU} would be the control variable, but the user disk requirement per visit $(1/\mu_{UD})$ would remain unchanged.

3. *Identify those parameters which are directly or indirectly dependent upon the control variable.* It must be realized that in actual systems, the parameters represented in queuing network models (e.g., μ's, V's, DMP) are not independent. However, formulating the functional relationships between these dependent variables and the control variable is, in general, difficult. For example, if another CPU is added, throughput at the CPU will be increased, which may affect the degree of multiprogramming, which, in turn, affects the visit ratio to the paging/swapping device [4]. To predict these parametric interdependencies, experimentation and the derivation of empirical formulae may be required.

4.2 Example Predictions

This methodology is applied to the PE-3220 example. The goal is to predict user response time and throughput as the workload increases.

1. *Control variable.* One measure of workload is the average number of active terminals. In the system model this number is the degree of multiprogramming (i.e., DMP is the control variable).

2. *Independent variables.* The mean device speeds $(\mu_{CPU}, \mu_{SD}, \mu_{UD}, \mu_{term})$ are assumed to be invariant with respect to the DMP. This is not to say that the time a request spends at a device and its queue is independent of the DMP, but that the average amount of actual service required by a request from a server is independent from the number of requests circulating within the system. These mean device speeds are assumed invariant at the values 19.5, 54.0, 54.0, and 0.06536, respectively. These values come from the parameterized system model with five active terminals (i.e., $DMP = 5$).

3. *Dependent variables.* As the number of active terminals increases, so does swapping behavior. Therefore, the relative visit counts, V's, are dependent upon the control variable.

V_{UD} is assumed to be related to V_{term} as follows:

$$V_{UD} = C \times V_{term} \quad \text{for a constant } C$$

This states that, regardless of the number of other active terminals and regardless of the amount of swapping, the number of visits to the user disk per terminal request is invariant. C, found from the parameterized system model, is 26.79.

V_{CPU} is assumed to be related to V_{SD}, V_{UD}, and V_{term} as follows:

$$V_{CPU} = V_{SD} + V_{UD} + V_{term}$$

This is consistent with the central server model assumption.

To find the relationship between the number of active terminals and V_{SD} we first find a relationship between the number of active terminals and the swapping rate. This is done empirically. Data points were collected using a variety of program sizes with differing numbers of active terminals. Results are illustrated in Figure 3.

[Note: A similar curve relating swapping to DMP has previously been reported in [6].]

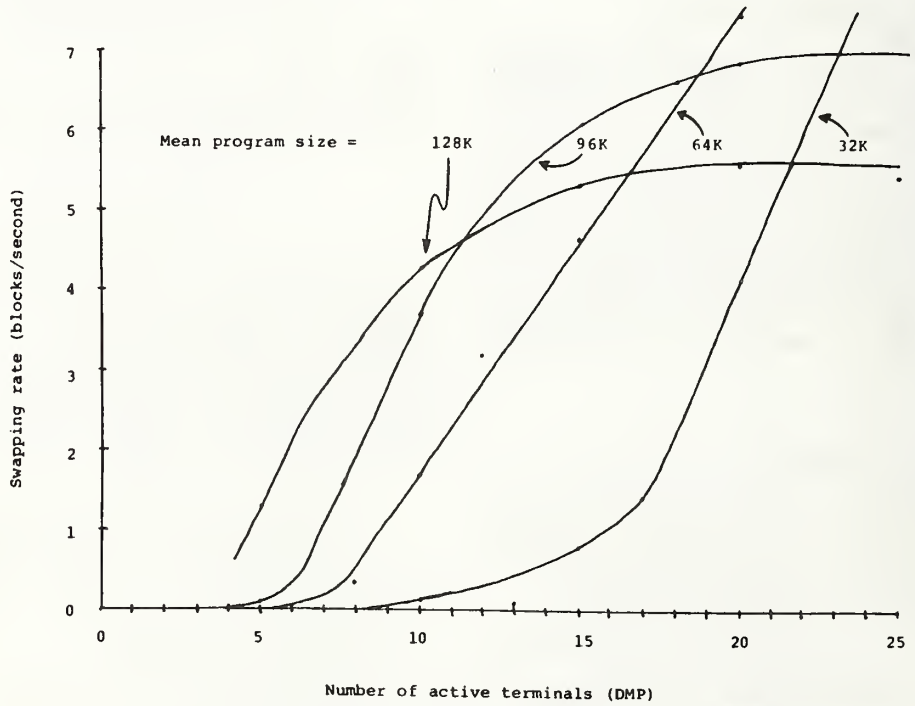


Figure 3: Swapping Rate as a Function of Program Size and Number of Active Terminals

The files on the system disk include both the swapping files and system overhead files. The access rate to the system overhead files is assumed to be linear with the number of active terminals. Combining the system overhead rate and the swapping rate for 64K jobs, the visit rate to the system disk is predicted using the empirical formula:

$$V_{SD} = max[0.6378, 1.07 \times DMP - 6.6]$$

It remains to find V_{term}. However, we notice that the units of V_{SD} is transactions per second. This is the estimated throughput for the system disk, whereas V_{CPU}, V_{UD}, and V_{term} are relative to each other. Therefore, an iteration technique is used. An estimate is made for the value of V_{term} (i.e., the terminal throughput). Using V_{term}, V_{UD} and V_{CPU} are calculated. The model is then solved and if the throughput of the system disk does not match V_{SD}, a new V_{term} is used. Setting V_{term} to the terminal throughput of the previous iteration is a good heuristic. Convergence of this iteration technique is monotonic and occurs typically within 3 to 4 iterations.

5. Validation of the Prediction Methodology

Using the prediction methodology given in the previous section, throughput and response time predictions were made when the active number of terminals were varied between 1 and 20. Mean value analysis techniques [11] were used in solving the queuing network models. For the validation, SYNBEN was run 7 times with 1, 5, 8, 10, 12, 15, and 20 terminals active. Throughputs and response times were measured using IOUmon and the recorded data by SYNBEN. The results are shown in Table 1, Figure 4, and Figure 5.

Table 1: Predicted versus Actual Comparisons

DMP	observed response	predicted response	error	observed throughput	predicted throughput	error
1	1.48	2.72	83.8%	.0594	.0555	6.6%
5	2.40	2.84	18.3%	.2744	.2756	0.4%
8	4.27	4.30	0.7%	.4044	.4082	0.9%
10	6.54	6.83	4.4%	.4511	.4519	0.2%
12	11.26	11.44	1.6%	.4456	.4487	0.7%
15	28.41	25.23	7.7%	.3283	.3612	10.0%
20	42.84	102.90	140.2%	.3394	.1692	50.1%

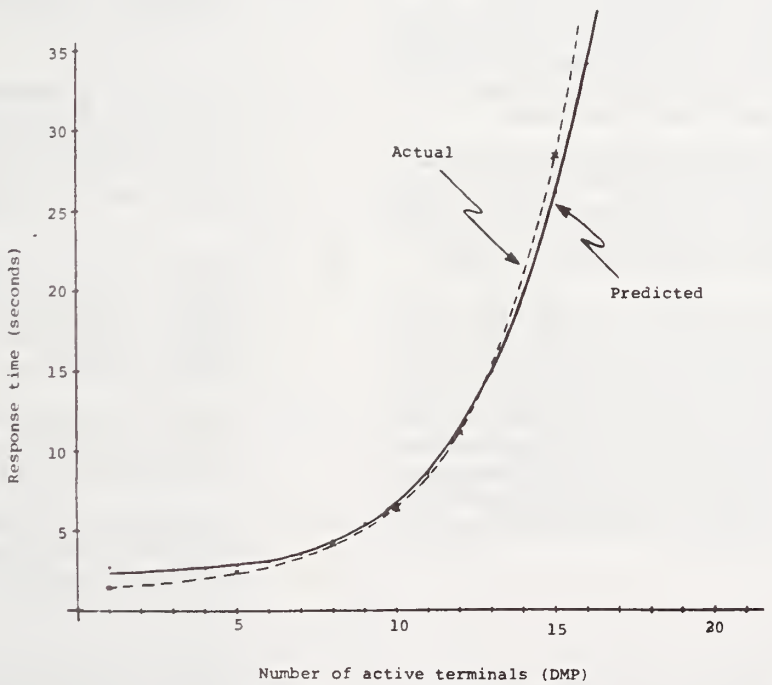


Figure 4: Predicted versus Actual Response Time

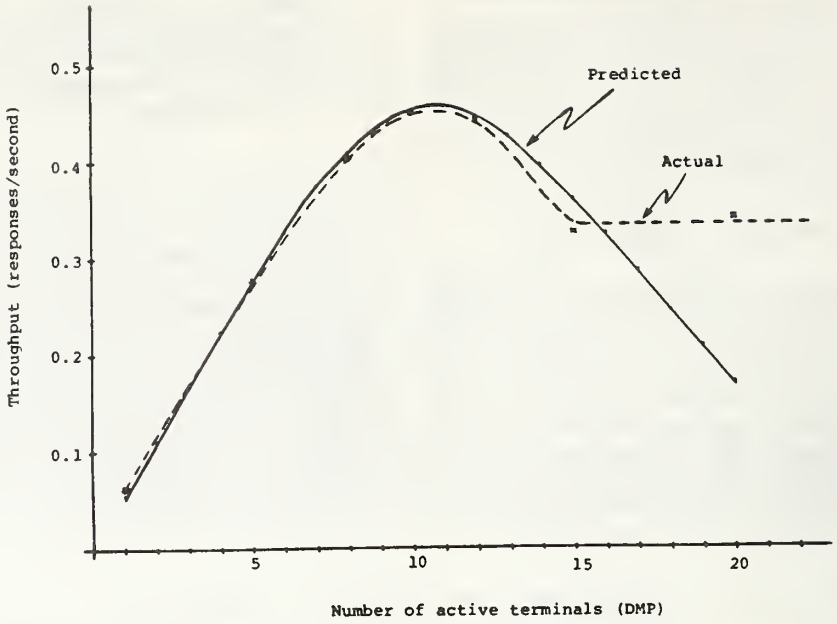


Figure 5: Predicted versus Actual Throughput

6. Conclusions

The comparison between the actual and predicted performance measures shown in Table 1, Figure 4, and Figure 5 is good. Larger errors are observed when the number of active terminals is greater than 15. In these cases, the predictions were overly pessimistic. UNIX has tunable parameters which do not allow the system to deteriorate too severely under heavy loads. These parameters include the minimum times which must pass before a job can be swapped in and swapped out. These parameters were not included in the prediction models. As a result, throughput actually flattens out under heavy loading while the model predicts it to continue to deteriorate (see Figure 5). Similarly, increased errors are observed in the predicted response times under heavy loads.

The point at which throughput begins to deteriorate is with 11 active terminals (i.e., $DMP = 11$). This is the point above which the amount of swapping begins to have a detrimental effect upon the throughput of terminal responses. The model accurately predicts this point.

The presented technique and its validation demonstrate the usefulness of performance prediction techniques. However, caution must be exercised. The modeled system must be well understood by the modeler. This includes understanding and validating parameter interdependencies and empirical formulae. The need and usefulness of performance monitors and workload generators is recognized. Similar studies in other environments are needed to transform performance prediction from an art into a science.

This research was supported in part by the University Research Council of Vanderbilt University and by National Science Foundation Grant no. MCS-8203594.

References

1. Bard, Y., The VM/370 Performance Predictor, *Computing Surveys 10,* 3, September 1978, pp. 333-342.

2. Bashioum, D.L., Benchmarking Interactive Systems: Calibrating the Model, *Performance Evaluation Review 9,* 2, Summer 1980, pp. 35-41.

3. *Computing Surveys, Special Issue: Queueing Network Models of Computer System Performance 10,* 3, September 1978.

4. Dowdy, L.W., Agrawala, A.K., Gordon, K.D., and Tripathi, S.K., Computer Performance Prediction via Analytical Modeling – an Experiment, *Proc. of the Conf. on Simu., Meas., and Modeling of Computer Systems,* August 1979, pp. 13-18.

5. Dowdy, L.W., and Budd, R.M., File Placement Using Predictive Queuing Models, (accepted to appear). Presented at Applied Probability – Computer Science, The Interface, Boca Raton, Florida, January 1981.

6. Dowdy, L.W., and Breitenlohner, H.J., A Model of Univac 1100/42 Swapping, *Performance Evaluation Review 10,* 3, September 1981, pp. 36-47.

7. Diethelm, M.A., An Empirical Evaluation of Analytic Models for Computer System Performance Prediction, *Computer Performance,* North-Holland, 1977, pp. 139-160.

8. Gordon, K.D., On the Construction of Representative Test Workloads, Ph.D. Dissertation, Dept. of Comp. Sci., Univ. of Maryland, College Park, Maryland, 1981.

9. Kresin, H.S., Characterization of Think and Response Time, M.S. Thesis, Dept. of Comp. Sci., Univ. of Maryland, College Park, Maryland, 1980.

10. Lo, T.L., Computer Capacity Planning Using Queueing Network Models, *Performance Evaluation Review 9*, 2, Summer 1980, pp. 145-152.

11. Reiser, M. and Lavenberg, S.S., Mean Value Analysis of Closed Multichain Queueing Networks, *Journal of the ACM 27*, 2, April 1980, pp. 313-322.

12. Rose, C.A., A 'Calibration-Prediction' Technique for Estimating Computer Performance, *Proc. NCC*, 1977, pp. 813-818.

13. Seam, P.H., Modeling Considerations for Predicting Performance of CICS/VS Systems, *IBM System Journal 19*, 1, 1980, pp. 68-80.

"Improving Organizational Productivity"

UNIVAC Performance Analysis

SESSION OVERVIEW

UNIVAC PERFORMANCE ANALYSIS

John C. Kelly

Datametrics Systems Corporation
Fairfax, Virginia 22031

This session explores three aspects of performance evaluation as it applies to UNIVAC 1100 computer systems. The first paper by Hajare looks in detail at the I/O component of the system. Hajare has used a hardware monitor to measure I/O activity across controllers and devices. This paper provides an excellent reference for anyone attempting to optimize the I/O load on a UNIVAC system. The article also provides several interesting insights into the use of a hardware monitor on UNIVAC computers.

The second article by Tibbs and Kelly looks at UNIVAC systems from a much broader view. The authors applied analytic modeling using BEST/1 and simulation modeling using SLAM to size and evaluate a replacement for a large UNIVAC installation. The paper presents numerous insights into the model building process and points out some of the unique aspects of modeling UNIVAC 1100 computer systems.

The third paper by Bays and Voegeli describes how the statistical package P-STAT was used to analyze workload data from the Master Log File. In conjunction with several driver programs, P-STAT was used to identify high resource programs and prepare frequency distributions for basic workload characterization. Considering the lack of analysis programs in UNIVAC environment, most UNIVAC users should find this approach very interesting.

A STUDY OF DISK I/O ON A UNIVAC SYSTEM IN THE SHUTTLE MISSION SIMULATOR COMPUTER COMPLEX*

Ankur R. Hajare

MITRE
Houston, Texas
77058

The Univac 1100/46 in the Shuttle Mission Simulator Computer Complex (SMSCC) had two strings of disk drives with two disk controllers on each string. A Tesdata MS-88D hardware monitor was used to obtain measures of disk usage and disk controller usage. The data collected showed that, under Exec Level 36, the load was balanced between the two controllers on a string. The 8450 fixed disk with the swap file (which was not always the same disk) was used much more than any of the others and it showed a daily variation similar to terminal usage. The I/O activity at the 8450 disks was extremely unbalanced when an empty disk was introduced into the system or when a large amount of space was freed on one disk. This imbalance was caused by all temporary files and all newly cataloged files being placed on the disk that was least full, and it resulted in very degraded terminal response.

Key words: Disk I/O; hardware monitoring; performance measurement; Shuttle Mission Simulator; Univac.

1.0 INTRODUCTION

1.1 Background

The Shuttle Mission Simulator (SMS) located in Building 5 of the National Aeronautics and Space Administration (NASA) Johnson Space Center (JSC) is used to train astronauts for Space Shuttle missions. SMS consists of two simulator bases supported by the Shuttle Mission Simulator Computer Complex (SMSCC). At the time the data for this study were collected the SMSCC contained two large Univac mainframes (one U1100/46 and one U1100/44) and several smaller computers that operated in a dedicated, real-time environment. The U1100/44, which has four Command Arithmetic Units (CAUs) and two Input Output Access Units (IOAUs), is used exclusively for real-time flight simulation. The U1100/46, which has been reconfigured, had six CAUs and three IOAUs. It was used for simulation as well as for software development and associated functions. This study was restricted to the Univac 1100/46 until the establishment of the Guidance and Navigation Simulator (GNS) Computer Complex which has another Univac 1100/44 that is now used for all development work.

1.2 Scope of Work

MITRE's activity in the SMSCC includes computer performance measurement and analysis*. A Tesdata Model MS-88D hardware monitor is one of the tools used in performing this task. Initially, the use of this hardware monitor consisted of measuring CAU and I/O channel activity. There had been no work in the past in measuring disk I/O activity. A study of disk I/O on the U1100/46 was undertaken to provide insight into some current or potential problems. The main issues addressed in this study were load sharing between disk controllers and load sharing between disks.

The study is still in progress, and this paper reports the phase of the study through December 1981. At that time the reconfiguration of the SMSCC and the establishment of the GNS resulted in a disruption of the I/O study until the hardware monitoring equipment was reinstalled.

---

* This work was supported by NASA Johnson Space Center under contract number F19628-82-C-0001 T1612J.

2.0 HARDWARE CONFIGURATION

The Univac disk configuration under study consists of two strings of disk units. The larger string contains four 8433 disk units and six 8450 disk units. This string, which is referred to as the "long string", has two 5046 disk controller units (CUs). The other string, which is called the "short string", contains eight 8433 disk units with two 5039 CUs. Figure 2-1 illustrates these disk subsystems which were part of the Univac 1100/46 (6x3) in the SMSCC when the study was begun but are now a part of the Univac 1100/44 (4x2) in the GNSCC. The functional characteristics and modes of operation of the above mentioned pieces of hardware are described in this section. Additional information on the Univac disks can be found in references [6] through [10].

2.1 8433 Disk Unit

The 8433 disk unit is a random access, moving-head disk unit featuring a removable and interchangeable disk pack which consists of twelve disks on one vertical shaft or spindle. Nineteen surfaces are used for data recording, and they are serviced by one moving actuator mechanism which contains one read/write head for each surface.

Rotational Position Sensing (RPS) is an option provided on the 8433 disk unit for more efficient use of a Control Unit (CU) and I/O channel. Without RPS, the CU would be interrupted by the disk unit when the cylinder and head addressing was complete. The CU would then wait until the required sector is at the head. But with RPS, the CU is not interrupted by the disk unit until the specified sector is reached, thereby reducing the time during which the CU is busy.

At the time the disk subsystem is installed, a hard-wired physical address is assigned to each disk unit by a Sperry Univac customer engineer. This address identifies the physical position the disk unit occupies relative to the CU. The physical address consists of six bits and uses a 3-of-6 code that permits only 8 valid addresses. When used with a CU that can control 16 disk units, one more bit is added to the physical address of the disk units.

However, the processor identifies a disk unit by a logical address. The operator control panel, on the disk unit, includes a removable module select plug which establishes the logical address for the unit. With the sixteen disk unit feature added to the CU, the module select plugs are number 0-F (hexadecimal). The module select plugs are removable and interchangeable, thus permitting a physical disk unit to assume different logical addresses at different times. Because of this feature, a disk pack can be kept on a specific logical unit even if a disk drive fails. This is done by moving both the disk pack and the module select plug to a drive that is functioning.

All 8433 disk units in the SMSCC are equipped with the optional dual access feature which allows the physical attachment of two CUs to an 8433 disk. This dual-access feature allows an active 8433 disk to be interrogated by both CUs as to status and availability. A priority circuit in the drive provides a tie-breaking function if two CUs attempt to select the same drive at the same time.

Three modes of operation are possible using the Mode Selection switch on the operator's panel of the 8433 disk unit:

- access to CU #1 and CU #2 permitting dynamic operation from both CUs,
- access to CU #1 only, and
- access to CU #2 only.

2.2 8450 Disk Unit

The 8450 Disk Unit is a fixed disk media, random access storage device. Each disk unit has a single head/disk assembly (HDA) that contains a disk stack of eight disks. The HDA is a module that is enclosed in a plastic and metal case. It is not operator-removeable but can be removed and replaced by a Sperry Univac customer engineer in case of problems.

Fifteen disk surfaces are used for data recording. They are serviced by a single accessor mechanism with two movable read/write heads on each of the 15 disk surfaces. These two heads are designated as head A and head B. Each of the 15 surfaces has two distinct concentric data areas, one accessed by head A and the other accessed by head B.

The accessor mechanism can assume 560 positions, thus enabling each head to access 560 tracks. The set of 30 tracks that can be accessed in one position of the accessor mechanism is referred to as one cylinder even though physically it consists of two cylinders. Each cylinder thus contains 30 tracks numbered 0 through 29 with heads A accessing the even numbered tracks and heads B accessing the odd numbered tracks.

Like the 8433 disk units, 8450 disk units have distinct logical and physical addresses. However, on the 8450 disk units, both logical and physical addresses are determined by jumper connections inside the disk unit. Therefore, the logical address of an 8450 disk unit cannot be changed by an operator as on an 8433 disk unit. Operator changeability of logical address is of no use on an 8450 disk drive since the disk pack cannot be moved by an operator to another drive.

All 8450 disk units in the SMSCC have the dual access feature by means of which they can communicate with either of two CUs. A disk unit

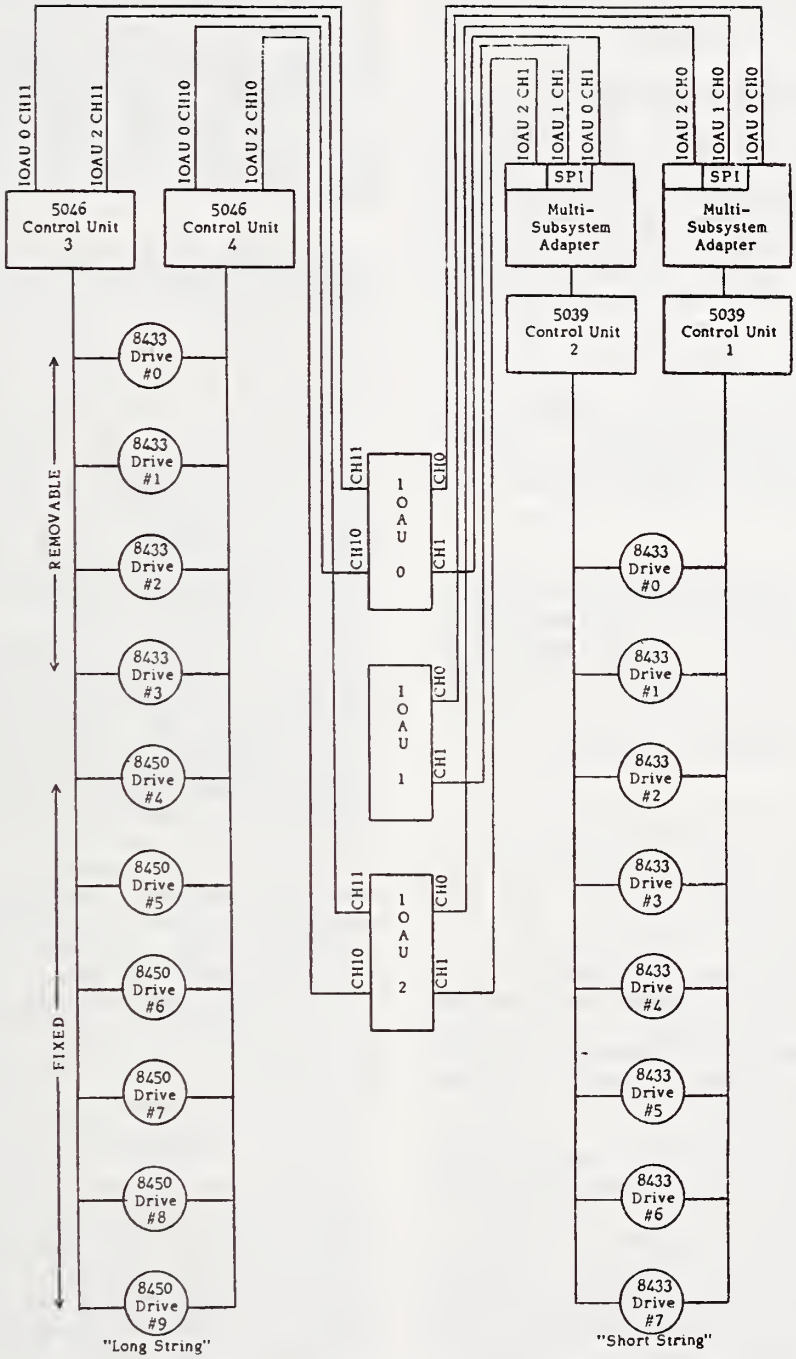


Figure 2-1. Univac 1100/46 (6x3) Disk Subsystems Configuration

can only be selected by one of the two CUs at any given time. A priority circuit in the disk unit provides a tie-breaking function should two CUs attempt to select the same disk unit at the same time. Dual access 8450 disk units have a Mode Selection Switch that permits three modes of operation just as on dual access 8433 disk units. RPS is a standard feature in 8450 disk units.

2.3 Control Units

The control unit (CU) or disk controller in a disk subsystem controls all the functions for direct access storage operations. In operation, the CU accepts selection from one of the channels to which it is connected, establishes connection to the requested disk unit, decodes commands from the channel, and controls the transfer of data to or from the disk unit. The Univac 1100/46 in the SMSCC has two kinds of CUs. The long string has two 5046 CUs and the short string has two 5039 CUs.

A basic 5039 CU controls up to eight 8433 disks. The optional 16 Drive Expansion feature allows control of up to eight additional 8433 disks. A 5039 CU cannot control 8450 disk drives.

A basic 5046 CU can control up to sixteen 8450 disk units. Mutually exclusive options permit a 5046 CU to control either an additional sixteen 8450 disk units or sixteen 8433 disk units. The two 5046 CUs in the SMSCC are configured to control the four 8433 disk units on the long string. Hence, they are limited to sixteen 8450 disk units plus sixteen 8433 disk units.

A 5046 CU has a dual 36-bit channel interface which allows the CU to be connected to two separate word I/O channels. However, the CU can only communicate with one I/O channel at a time and data transfers can only occur over one I/O channel interface at a time. The two 5046 CUs in the SMSCC were each connected to IOAU 0 and to IOAU 2 but not to IOAU 1. Each of the two I/O channel interfaces in the CU can be individually disabled by means of the CHANNEL ENABLE switch on the CU operator's panel and also from the System Partitioning Unit (SPU) panel.

A 5039 CU by itself can only be connected to one I/O channel. However, it can be interfaced to additional channels via a Multi-Subsystem Adapter (MSA) and a Shared Peripheral Interface (SPI). Up to three SPIs may be added, thereby providing as many as four access paths to a 5039 CU. As shown in Figure 2-1, the two 5039 CUs on the short string are each connected to three I/O channels.

2.4 Other Hardware

The other hardware associated with the disk subsystems consists of Multi-Subsystem Adapters (MSAs), Shared Peripheral Interfaces (SPIs), I/O channels and Input-Output Access Units (IOAUs).

An MSA is a programmable device that interfaces byte oriented peripherals to the 36-bit word oriented processor complex of Univac 1100 Series computer systems. The 8433 disk units and 5039 CUs are 8-bit byte oriented devices. Hence, an MSA is required between 5039 CUs and I/O channels. Although each MSA is capable of handling up to 8 CUs, each 5039 CU on the short string has its own MSA to achieve better performance. A basic MSA can be connected to only one I/O channel; however, up to three more channels can be accomodated by the inclusion of an SPI for each additional channel. As shown in Figure 2-1, each MSA on the SMS Univac 1100/46 (6x3) has two SPIs and is connected to three I/O channels.

The 8450 disk units and the 5046 CUs are 36-bit word oriented devices. Hence, they do not require an MSA.

All I/O operations on the Univac 1100/46 (6x3) are performed by one of the three Input-Output Access Units (IOAUs). The IOAU provides control and data paths between main storage and peripheral subsystems and operates under the direct control of the CAUs. Each IOAU interfaces with all of primary storage and all of extended storage via one access path to each Primary Storage Unit (PSU) and dual access paths to each Extended Storage Unit (ESU).

An IOAU receives commands from the control section of either of the two Command/Arithmetic Units (CAUs) to which it is connected. When the IOAU receives its instructions from the CAU's control section, it performs the necessary data transfers independent of control and does not interfere with the execution of instructions in the CAU, except possibly for memory contention. Each IOAU can accomodate up to twenty-four channels. However, the SMS Univac 1100/46 (6x3) had sixteen channels on each of its three IOAUs. All three IOAUs have one channel connected to each of the two 5039 CUs. Two of the three, IOAU 0 and IOAU 2, had one channel connected to each of the two 5046 CUs. IOAU 1 was not connected to the 5046 CUs. Thus, ten channels were used for disk I/O.

3.0 DISK SUBSYSTEM OPERATION

The Univac computer systems in the SMSCC run under the Sperry Univac Series 1100 Executive System (Exec). This vendor-supplied standard operating system has been locally modified for use in the SMSCC. Exec Level 33 was used during the earlier phase of the disk I/O study. But on 26 June 1981 Exec Level 36 was put in operation in the SMSCC. This section describes the operation of the disk subsystems under the Exec. This description applies to both the levels mentioned above. The local modifications to the Exec are not in the I/O area.

3.1 Fixed and Removable Disk Packs

Before a disk pack is used under the Exec it must be "prepped". A disk pack may be prepped as "fixed" or "removable". Fixed packs are used by the system as permanent online mass storage. The set of fixed packs is treated as one pool of mass storage. Removable packs are user assigned and may be mounted and dismounted as required.

A file may be placed on a specific removable pack by specifying a pack-id when it is created. If a file is created without specifying a pack-id, the Exec places it in the fixed disk pool. The Exec decides which specific fixed pack will contain the file, and it may even span a pack boundary.

The six 8450 disk units in the SMSCC have been prepped as fixed. All the 8433 disk units are used for removable packs except for the one which is used for the Exec pack.

3.2 Exec Pack

A system in which the Exec resides on a disk subsystem is known as a disk-resident system (DRS). The pack on which the Exec resides is known as the system pack or the DRS boot pack. The DRS boot pack must be specifically prepped as a DRS pack. On the SMS/GNS Univac systems, which are all DRSs, drive #0 (logical address) on the short string is used for the system pack. DRS packs are not compatible between 5046 and 5039 CUs. Hence the Exec pack cannot be put on any of the 8433 disk drives in the long string. However, it can reside on any of the eight drives in the short string since any of them can be assigned logical address #0 by means of the module select plug.

When there is a hardware problem with the disk drive that has the system pack, it is moved to another drive. The operators attempt to keep one 8433 disk unit as a spare for the system pack in order to reduce the impact of a hardware problem with the drive that has the Exec pack.

A disk pack, including the Exec pack can be moved from one 8433 drive to another in the same string without a keyin at the operator's console and without a system re-boot provided it stays at the same logical address. The process consists of the following steps. First, the module select plug is removed. Then, the drive is stopped and the pack is removed. Next, the pack is mounted on another drive which does not have a module select plug at this time. This drive is started and when it is up to speed the original module select plug is inserted.

3.3 File Organization

For files which are to be retained beyond run termination, entries in a master file directory (MFD), containing the identification and characteristics of each file, are constructed and maintained by the system. The process of entering a file in the MFD is called cataloging, and files having entries in the MFD are called cataloged files.

As part of the MFD, the Exec maintains tables specifying the location of the various granules of storage space allocated to a given file name. This table allows a file to be non-contiguous and also permits it to occupy space on more than one pack.

3.4 Operating System Boot Loading

During an initial boot, the Executive attempts to read a "label record" from each disk unit in an "up" or "suspended" state. Those units that contain packs prepped as fixed are used as fixed mass storage. Once acquired and initialized, these packs cannot be dismounted without a reboot.

On an initial boot, SELECT JUMP switch 13 must be set. Switch 13 causes all on-line fixed mass storage to be initialized. All files cataloged to fixed mass storage that have a tape backup are then reloaded from tape. This process takes about eight hours in the SMSCC and it requires exclusive use of the system.

During a recovery boot, the Exec attempts to acquire as many fixed drives as specified during the system's generation from the mounted packs prepped as fixed. The Exec attempts to obtain fixed packs in the following order:

1) recover those fixed packs that were in use prior to this recovery bootstrap, and

2) initialize any other fixed packs that are mounted.

3.5 Rollout and Rollback

Depending upon the amount of available mass storage, the degree of use given to cataloging files on mass storage, and the manner in which files are assigned, the Exec may need to obtain additional space on mass storage by rolling out cataloged files to magnetic tape. This feature is provided automatically by the Exec. The points at which rollout is turned on and off are system generation parameters.

Rollout occurs when a request for allocation reduces the available mass storage below a threshold specified at system generation. The rollout routine utilizes a file's activity and reference age as part of the criteria for file rollout eligibility. If a current backup of a file selected for rollout exists on a SECURE tape, the file is not written to tape, it is just registered in the MFD as unloaded.

A request to assign a rolled out file causes the Exec to request mounting of the proper

magnetic tape (unless it is already mounted) and to automatically return the file back to mass storage using the SECURE processor.

Because of the rollout/rollback feature, a file can move from one physical location to another. This makes it difficult to track a file's usage by means of a hardware monitor.

3.6 Parallelism and Path Selection

The Univac disk subsystems configuration in the SMSCC contains more equipment than the minimum necessary to access all the disk units. This redundancy improves both system availability and system throughput by permitting parallel operations.

Every disk unit in the SMSCC has the dual-access feature that allows it to be accessed by two controllers. All disks can therefore be accessed even when one of the controllers on each string is not operational. With both CUs on a string in operation, two disks within a string can be accessed simultaneously.

Each 5039 CU can be accessed by all three IOAUs and each 5046 CU can be accessed by two of the three IOAUs. This means that single channel failures do not hamper system availability although they may degrade performance. Even a complete failure of an IOAU does not prohibit access to any of the disks, and the short string can be accessed even with two IOAUs down. The real-time flight simulation does not require the long string. Hence, it can run with just one IOAU in operation.

Because of the redundancy in the disk subsystems, each disk can be accessed via multiple paths. Path selection is done entirely within the Exec with no direct operator or programmer control. Selecting a path consists of selecting an IOAU and selecting a CU. This is equivalent to selecting a channel, since each channel specifies a unique path to a disk.

4.0 MEASUREMENT PROCEDURE

A Tesdata MS-88D hardware monitor was used to study disk I/O. There are two general modes of use of the SMSCC Tesdata system: on-going routine measurement and special studies. These two modes of use have different operational characteristics and are performed for different reasons.

The routine measurement is carried out automatically and does not require the presence of a Tesdata operator. The monitored signals are automatically reduced to five minute sums or averages and stored in Performance Data Files (PDFs) on the Tesdata disk.

Special studies, on the other hand, require exclusive use of the Tesdata system. They are performed in short blocks of time during which routine monitoring is suspended under operator control. Special studies also require removing and re-connecting concentrator cables.

The measurements performed for the disk I/O study were made a part of the routine measurement in order to obtain data on an on-going basis. Thus, data has been collected on an almost continuous basis since it was initiated. However, gaps exist because of four reasons. First, power outages and Tesdata system maintenance time result in gaps in the data collection. Second, special studies necessitate bringing down the routine measurement. Third, a lack of disk space occasionally prevented the automatic generation of new PDFs at the beginning of a month. Finally, sensor tips have occasionally been removed or placed at the wrong location, resulting in invalid data.

The Tesdata MS-88D is limited to eight "collectors", each of which can be used to monitor sixteen signals. Since the disk I/O study had to co-exist with other routine measurements, initially only two collectors could be used for measuring disk I/O. This meant a limit of thirty-two signals. Another restriction was a limit of two concentrators or a total of eighteen sensor assemblies.

Finding the desired electrical signals for measurement has been a major problem throughout the study. Tesdata provides a sensor point library, but it does not adequately cover Univac equipment. A second source of information for sensor points was Tesdata users at other Univac installations. However, these users did not have exactly the same kind of equipment as in the SMSCC. The third source of sensor point information was Univac maintenance personnel. The search for sensor points was a trial and error process during which differences between the documentation and the installed equipment were discovered.

On the 8450 disk units, the signals chosen were "Mod Select A" and "Mod Select". "Mod Select A" indicates that the drive has been selected by controller A. "Mod Select" indicates that the drive has been selected by one of the two controllers. These signals were monitored from March 1981 through September 1981.

An attempt was made to obtain similar measurements on the 8433 disk units. However, the physical placement of the probes inside the cabinet caused hardware problems which resulted in the probes being removed. Since measurements were not being performed, the sensor assemblies were removed when the equipment was moved in May 1981. But after Exec Level 36 was installed on 26 June 1981, the search for the causes of poor terminal response time generated an interest in Exec pack I/O activity. At that time a pair of extra length probe tips was made so that a sensor could be installed in a 8433 cabinet without causing hardware problems. These probes were installed on the 8433 disk unit on the short string with physical address 2, which at the time

had logical address 0 and was used for the Exec pack. The operators were then instructed not to move the Exec pack to another drive unless necessary. Hardware problems with the disk drive made it necessary to move the Exec pack to another drive on two occasions. The signals monitored on this drive were "Mod Select A" and "Mod Select B", which indicate that the drive has been selected by the respective controller. Most of the data obtained from this drive is for the Exec pack, but some of it is for removable packs.

The routine measurement was reconfigured in July, 1981, and this resulted in another collector being made available for the disk I/O study. This collector was used to monitor the disk controllers. An attempt was made to find signals at the CUs that showed the address of the device that was selected. However, it was found that the CUs do not have an accessible register that holds this data. The device address is only accessible at a general purpose register which the CU microprogram uses for various other purposes as well.

The signal monitored on each of the two 5039 CUs was "Trap Selected A", which represents CU busy. Another signal was monitored on one of the 5039 CUs for a few days. That signal was "Read or Write Gate" which indicated that a read or a write operation was in progress. Seven signals were monitored on 5046 #2. Only four of those seven signals were monitored on 5046 #1 because of capacity limitations of the Tesdata system.

5.0 ANALYSIS OF HARDWARE MONITOR DATA

5.1 Load Sharing Between 5046 Controller Units

When the disk I/O study began, a major point of interest was the load sharing between the two controllers in each string. It was suspected that the load was significantly unbalanced, i.e., the two controllers had an unequal amount of activity.

Attempts to measure this at the 5046 controllers themselves were not successful. Signals that represented "controller busy" could not be monitored for reasons which included the physical inaccessibility of the relevant signals. The load balance between the 5046 controllers was, therefore, measured at individual disks. The signals monitored at the 8450 disk units were "Mod Select" and "Mod Select A". "Mod Select" is on whenever the disk is selected by either controller. "Mod Select A" is on whenever the disk is selected by controller A. Therefore, the disk is selected by controller B when "Mod Select" is on but "Mod Select A" is off. The total load at each controller was not obtained with this method since only the six 8450 disk units in the long string were monitored. The four 8433 disk units which are also in the long string were not monitored because of the capacity limitations of the hardware monitor.

The data showed a marked difference between the period before 26 June 1981 and the period after from 26 June 1981. On 26 June a new operating system, Exec Level 36, was put in operation on the U1100/46 (6x3). Prior to 26 June 1981 Exec Level 33 was used on that system.

With the old operating system there was a significant unbalance in the load sharing between the two controllers. The data for the period 12 March 1981 to 25 June 1981 showed that, on the average, controller A had 31.91% of total activity to all 8450 disk units. The daily average for all 8450 disks for that period varies between 26.15% and 49.55% of the activity coming from controller A. The daily average for individual disks shows a variation between 21.15% and 59.36% for "% A". Some of this variation may be due to one of the two controllers being turned off or "downed" for brief periods for maintenance purposes.

On a few occasions, one of the two I/O channels connected to the controller was disabled because of hardware problems either with the IOAU channel or with the channel interface in the controller. This also contributed to an imbalance between the two controllers. Both these situations, when they occur for brief periods, cannot be detected in the data collected by the hardware monitor. But such brief occurences, even if they occurred predominantly on one of the two controllers, do not significantly affect averages computed over a period of many days.

The new operating system was installed after this imbalance was discovered but before a decision was made regarding a course of action. The data collected after the installation of the new operating system revealed a sharp deline in the imbalance. The data subsequent to 25 June 1981 shows that, on the average, controller A had 47.37% of the total activity to all 8450 disk units. The daily average for all 8450 disk units under Level 36 varies between 46.47% and 50.49% of the activity coming from controller A. The daily average for individual disks varies between 43.35% and 53.58%.

Table 5-1 shows the difference in the load sharing under the two operating systems. The load unbalance under Level 36 is not considered significant enough to warrant any further action.

5.2 Load Sharing Between 5039 Controller Units

A signal called "Trap Selected A" was monitored on the two 5039 CUs for the purpose of studying the load sharing between them. This signal is on whenever the controller has selected a disk unit. This signal was monitored for the period from 3 August 1981 with gaps as explained before. Hence, this measurement was only performed under Level 36. The data indicate that, on the average, controller A contributed to 48.87% of the total activity on the short string. The daily average for percent A for the period of observation varies between 43.81% and 53.97%.

Some of this variation can be attributed to one of the two controllers being downed for brief periods. The data thus show that the load on the short string is balanced between the two controllers under Level 36.

Table 5-1

Load Sharing Between 5046 Disk Controllers

	"% A" Under Exec Level 33	"% A" Under Exec Level 36
Average for the period of measurement	31.91	47.37
Maximum daily average for all 8450 disks	49.55	50.49
Minimum daily average for all 8450 disks	26.15	46.47
Maximum daily average for an individual 8450 disk	59.36	53.58
Minimum daily average for an individual 8450 disk	21.15	43.35

A measure of load balance between the two 5039 controllers was also obtained from the "Mod Select A" and "Mod Select B" signals monitored at one of the eight 8433 disk drives on the short string. The 8433 drive that was monitored had the Exec pack on it when the measurement began but the Exec pack did not stay there throughout the period of observation. The data obtained from the 8433 disk drive also show a load balance between the two 5039 controllers. They are in close agreement with the data obtained by monitoring the two controllers themselves. It, therefore, serves to verify the load balance between the two 5039 controllers under Level 36.

5.3 Load Sharing Between 8450 Disks

All six 8450 disk units in the SMSCC have been prepped as fixed. These six disks plus the 8433 Exec disk pack comprise the fixed disk pool.

The signals monitored at the 8450 disks were "Mod Select" and "Mod Select A". "Mod Select" is on when either of the two controllers has selected the disk. "Mod Select A" is on when controller A has selected the disk. As explained in earlier, a disk must be selected by a controller before any operation can be performed. A disk remains selected during a data transfer operation. In the case of a control command (e.g., head positioning), the disk is selected prior to the command being issued. It is de-selected after the command is issued. The disk, therefore, does not remain selected during the execution of the command (e.g., during head movement). Since the time required for issuing a command is small compared to the time spent in a data transfer operation, "Mod Select" approximates the time spent between the beginning and end of data transfers. However, a data transfer command to an 8450 disk unit contains a 16-bit field that specifies the transfer length in 36-bit words. Thus, it is possible to transfer up to 65K words in a single operation. Such a transfer can cross track and cylinder boundaries. When this happens, track addressing and head repositioning is handled by the disk unit while "Mod Select" is on. In such a case, "Mod Select" exceeds the actual data transfer time. A disk is also busy during head movement and rotational positioning prior to the beginning of a data transfer. "Mod Select", therefore, understates disk busy but it is used here as a relative measure of disk busy for studying load sharing between 8450 disks.

Data were collected from 12 March 1981 through 30 September 1981. Data for all disks were not obtained for all days during this period. There are long gaps in the data collected because probes were removed when a disk drive was malfunctioning and were not properly replaced after repairs.

During the course of the study 5-minute averages of disk usage were plotted using PMS-II. These plots revealed that most of the time one of the 8450 disks was used significantly more than the other disks. This is illustrated in Figure 5-1 which contains data from 13:30 p.m. to 16:45 p.m. on 28 July 1981. The disk that was much busier than the others was not always the same one. Figure 5-1 shows that 8450 #6 is busier than the others, but in Figure 5-2 another disk, 8450 #7, is busier than the others.

The 8450 disk that is much busier than the others is the one with the swap file. This file is used by the system to save images of jobs that have been rolled out. The swap file is automatically created by the system when it is booted. Its placement is determined by an algorithm internal to the system and is based on the amount of empty space on each disk. Hence, it may be placed on different disks after each boot operation. It does not move after that until the system is rebooted.

During the course of the measurements, one of the 8450 disk units had hardware problems which caused it to be removed from operation and then put back into operation after a few weeks. This caused imbalances in the I/O activity across the six 8450 disks that were quite noticeable on the Tesdata GDU. At that time MITRE was also investigating the poor terminal response time on the U1100/46 and, thus, the data obtained from the 8450 disks were studied together with terminal

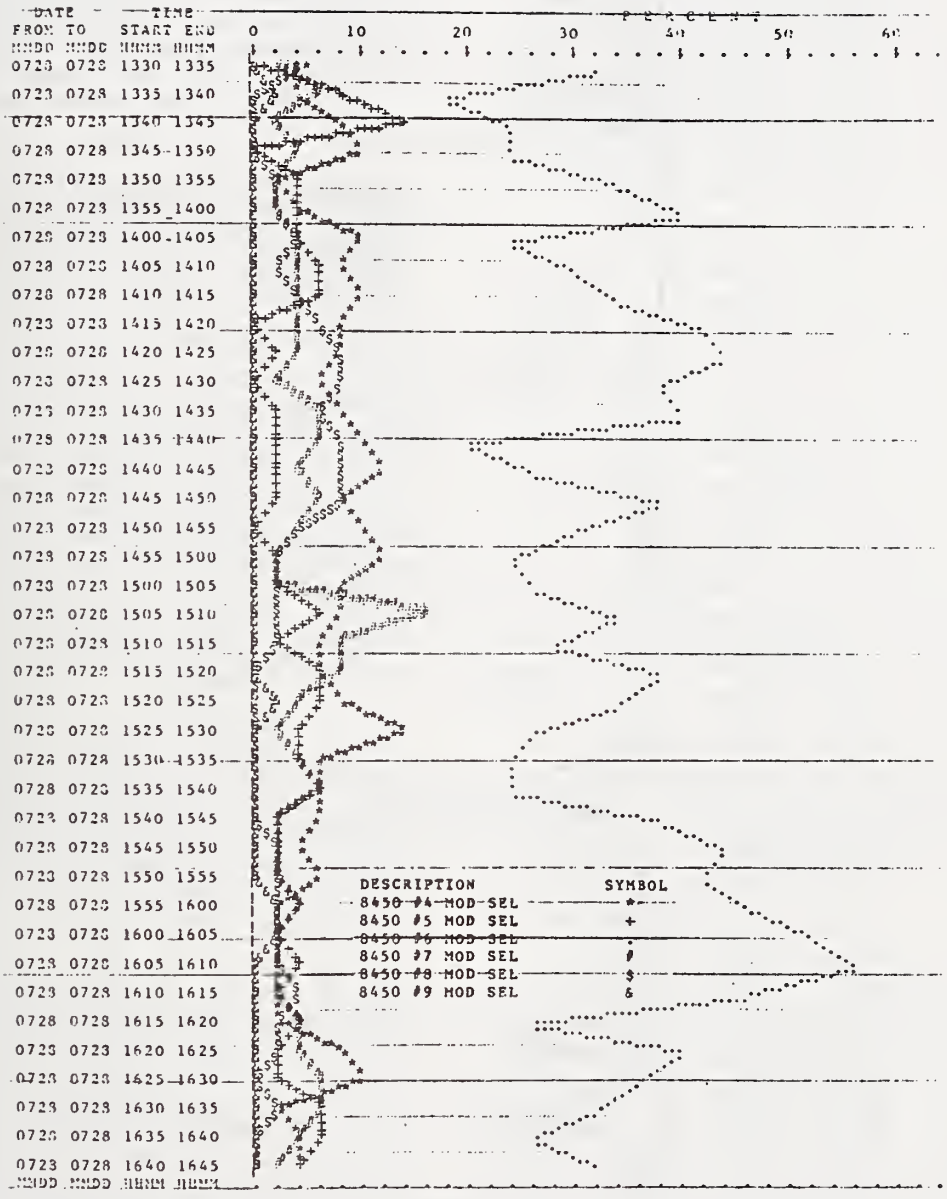


Figure 5-1. 8450 Disk I/O

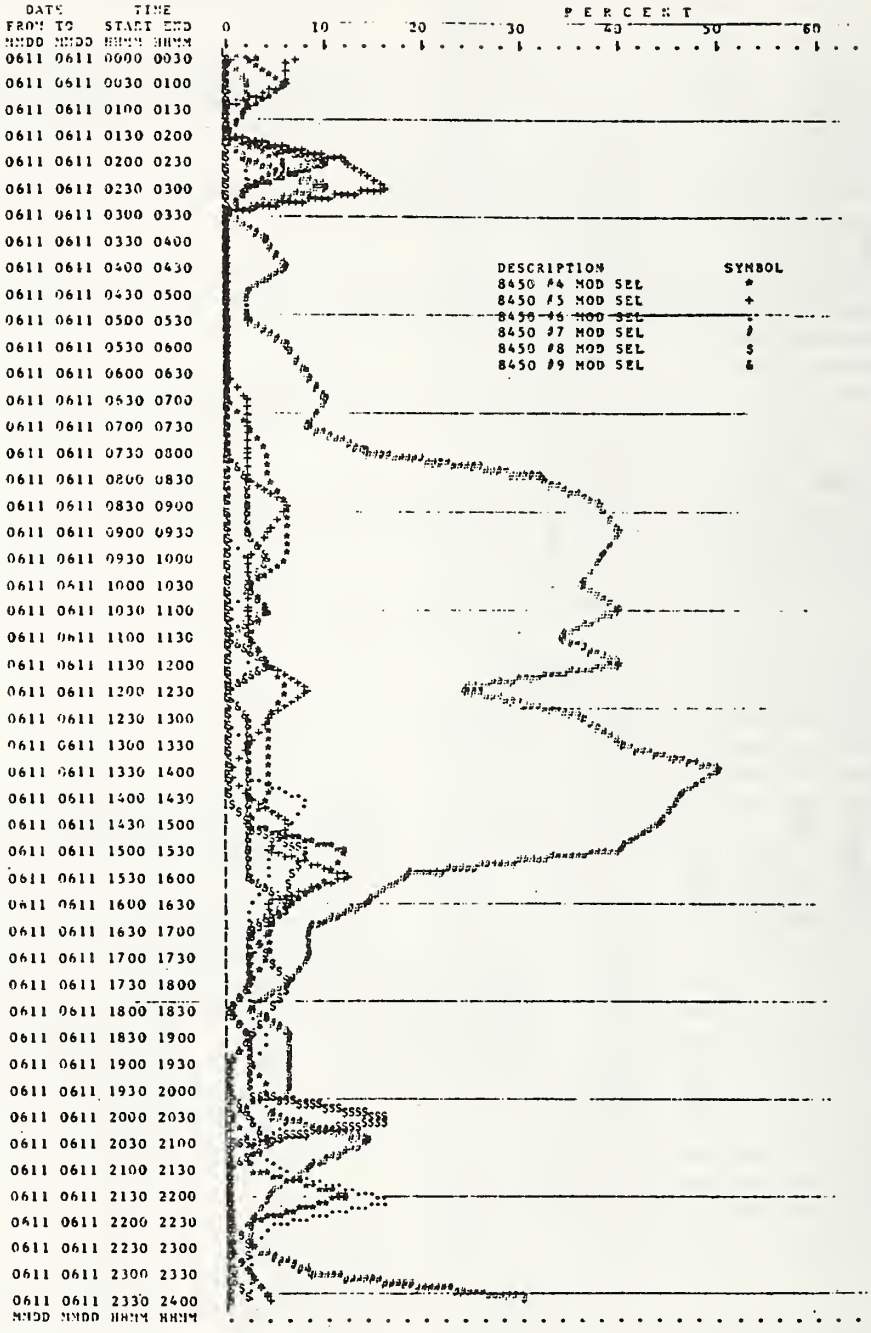


Figure 5-2. 8450 Disk I/O

response time data obtained from a monitor as described in [3]. The changes in terminal response time resulting from hardware problems with a fixed disk are described in [4].

Exec Level 36 places files on fixed disks on the basis of empty space on each disk unit. When a new file is created it is placed on the disk unit with the most empty space. This is done automatically within Exec and cannot be controlled directly by users. When a fixed disk is removed from service, all the files on it are moved to the remaining pool of fixed disks with file placement being done in the manner described above. This results in a good balance in the amount of empty space across disk units and it does not adversely affect the balance of disk activity.

However, when a disk is put back into service after repairs, the balance is greatly affected. Since the disk that has been just put in operation has the most empty space, all new files (including all temporary files) are placed on it. If the system is re-booted at this stage, the swap file is also placed on the disk that has just been put into operation. This results in an inordinately large portion of the fixed disk I/O going to that disk. Access to this disk becomes a bottleneck in the system and terminal response time shows a large increase. Observation of the Tesdata Graphic Display Unit (GDU) under these conditions has shown one 8450 disk being selected as much as 90% of the time in a 5-second interval while each of the other 8450 disks are selected less than 5% of the time. Since data was recorded in PDFs as 5-minute averages, these short peaks are not recorded but can only be observed on the fly at the GDU.

A similar, though less pronounced, situation occurs when a large amount of space on one 8450 disk unit is freed by deletion of files. This has been caused in the past by users and the cause of the resulting bad terminal response time was only discovered after a casual observation of the Tesdata GDU.

Figure 5-2 shows the variation in disk activity on a typical weekday. On that day 8450 #7 was much busier than the other 8450 disks. Access to this disk increases sharply around 8:00 a.m. and remains quite high until a sharp decline around 4:00 p.m. There is a noticeable dip in activity between 12 noon and 1:00 p.m. The variation, therefore, corresponds to prime shift working hours. The difference between 8450 #7 and the other 8450 disks can be attributed to the swap file residing on that disk. Swap file activity is a function of demand users who work during prime shift.

5.4 Exec Pack and Short String Activity

The signals monitored on the short string were quite different from those monitored on the long string. Hence, the performance measures obtained also were different. On the short string, the percentage of time each 5039 controller was busy was recorded in the form of five-minute averages beginning from 3 August 1981.

Data on Exec pack usage was obtained intermittently from 21 August 1981. It could not be obtained on a continual basis as explained earlier. The plots obtained using PMS-II indicated that a large fraction (as high as 90% of a daily average) of the total activity on the short string came from the Exec pack. This can be seen in Figure 5-3, which is a plot of Exec pack activity and 5039 controller activity on a weekday. This figure shows that I/O on the short string is much higher during prime shift than during other periods. Also, the difference between the two plots, which represents I/O to the remaining seven 8433 disks on the short string, is very small outside of prime shift.

The variation of Exec disk I/O activity during a weekday is usually of the form shown in Figure 5-3. It shows a sharp increase around 8:00 a.m. and a sharp drop around 4:00 p.m. with a relatively high level in between except for a dip at lunch time. The variation of 5039 controller busy also shows this form. The two peaks shown in Figure 5-3 (one between 2:00 p.m. and 2:30 p.m. and the other between 3:30 p.m. and 4:00 p.m.) are not typical and are not a daily occurence. But peaks this high (and higher) have been found on occasion and point to bursts of activity on the removable disks on the short string. The peaks in the figure show 70% busy for a thirty-minute average. Since the plot shows the sum of "controller busy" for both controllers, each controller is only about 35% busy (since the load is quite balanced).

The I/O activity on the short string is much higher on weekdays than on weekends, which corresponds to higher usage of the computer system as a whole on weekdays. The data collected also shows that the percentage of short string I/O activity at the Exec disk is higher on weekdays than on weekends.

An additional signal on 5039 #1 was monitored from 8 September 1981 to 14 September 1981. This was "Read or Write Gate" which was "on" during a read or a write operation. It was found that the average value of this signal for the period of observation was 99.5% of the value for "Trap Select A". Also, "Read or Write Gate" was never found to exceed "Trap Select A". "Trap Select A" is "on" during data transfers and while the controller is issuing a command to a disk drive. The measurement, therefore, showed that the time spent in issuing commands to disk drives is small compared to the data transfer time.

6.0 Conclusions

6.1 Load Sharing Between Disk Controllers

When the disk I/O study was begun it was suspected that the load sharing between the two

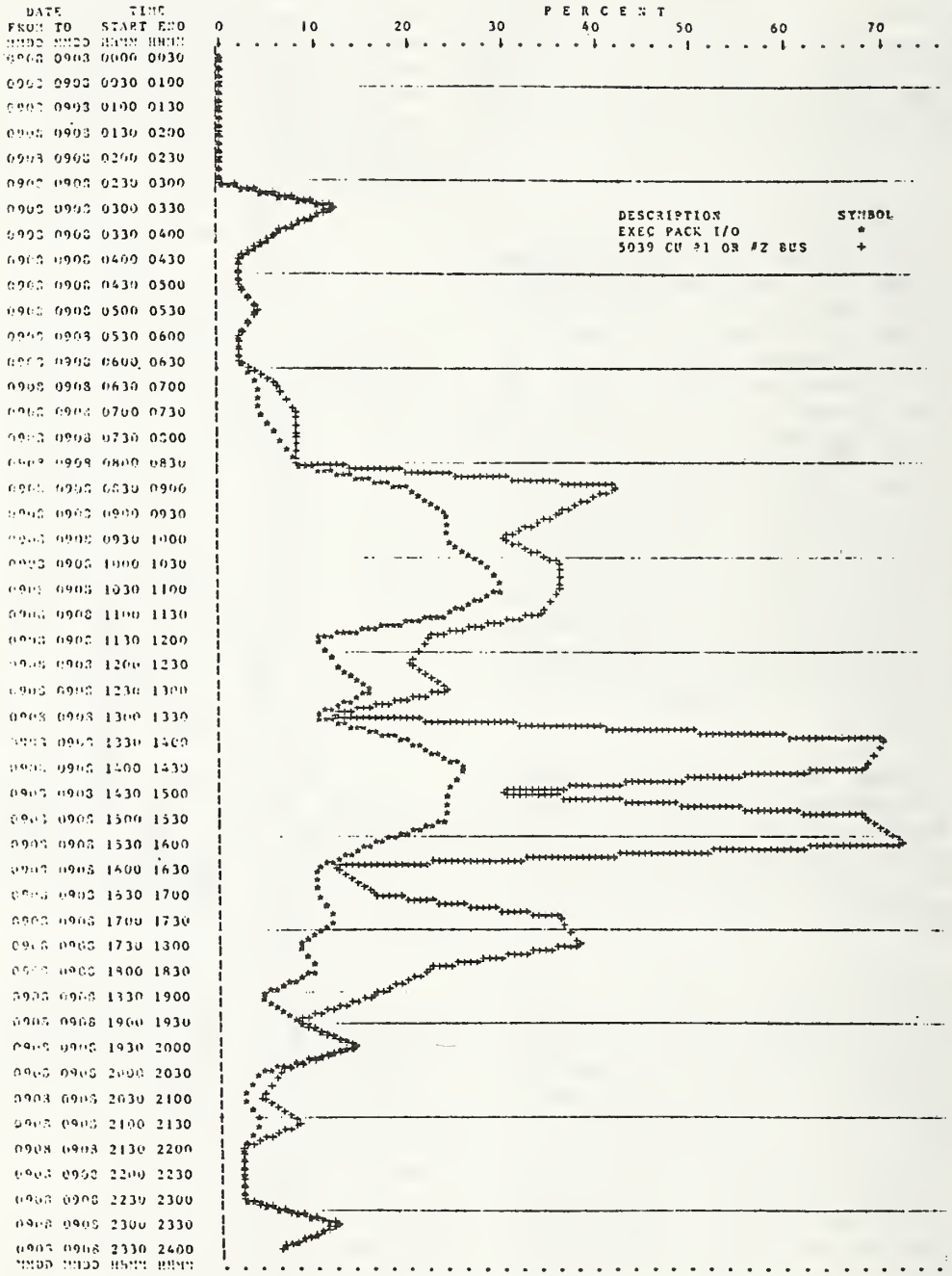


Figure 5-3. Short String and Exec Disk I/O Activity

controllers on each string of disk drives was unbalanced. This study revealed that this load sharing is almost perfectly balanced and is therefore not a source of problems nor a source of performance degradation in the system. A significant unbalance existed under Exec Level 33 but it virtually disappeared after the installation of Exec Level 36 on 26 June 1981.

6.2 Load Sharing Between 8450 Disks

The load sharing between the six 8450 disks was found to be unbalanced with one of the six being much more active than the others. The imbalance was more pronounced during prime shift weekdays. This is due to the swap file residing entirely on one disk. Swap file I/O activity during prime shift constitutes a significant portion of the total I/O to the six 8450 disks and therefore creates a noticeable imbalance.

A method of alleviating this imbalance consists of splitting the swap file into pieces which reside on separate 8450 disks. The placement of the swap file within the pool of fixed disks is handled internally within Exec Level 36 and is not under direct operator control. Splitting the swap file into pieces could not be accomplished without modifying the Exec. One of the other Univac installations that was contacted during the course of the I/O study had done this and thereby achieved an improvement in performance. However, this modification was not made in the SMSCC because Exec Level 37 was forthcoming and it provides the ability to split the swap file across specific devices by means of system generation parameters. Also, modifications to the Exec are usually avoided.

The other problem is a severe imbalance when an empty fixed disk is introduced into the system or when a large amount of space is freed on one of the 8450 fixed disks. This is due to the fact that Exec Level 36 places new files on the fixed disk that has the most empty space. File placement on a specific 8450 fixed disk cannot be accomplished by direct operator or programmer control. The Univac installation referred to earlier had solved this problem, too, by modifying the Exec. They had replaced the algorithm within the Exec for determining the fixed disk on which to place a new file by one that they wrote themselves. The principle behind their file placement algorithm was round-robin placement of new files in the fixed disk pool. As expected, it solved the problem of a severe imbalance in I/O activity.

The algorithm used by Exec Level 36 attempts to balance the percentage of free space on all the fixed disks with no regard to balancing the I/O activity across the fixed disk pool. The round-robin file placement algorithm, on the other hand, makes no attempt to correct any imbalances in the percentage of free space on the disks.

The implementation of this file placement algorithm in the SMSCC was considered but rejected because local modifications to the Exec are difficult to implement, maintain and transfer to new levels. Such modifications, therefore, are not made unless necessary. Besides, Exec Level 37 was forthcoming and it contains a file placement algorithm that spreads new and temporary files across the fixed disk pool.

6.3 Further Work

Data collected over a period of three months indicated a load balance between the two 5046 CUs on the long strings. Further monitoring of the load sharing between the two CUs appeared unnecessary. When the Tesdata routine measurement was reconfigured on 1 October 1981, the probes that had been used for monitoring "Mod Select A" were set up to measure head motion. The measurement of "Mod Select A" was continued on one 8450 disk in order to detect any significant changes from the balanced situation.

Shortly thereafter, the reconfiguration of the Univac systems in the SMS/GNS complex was begun. In the ensuing move of equipment, most of the probes used for the I/O study were removed. They were not re-installed immediately because the reconfiguration of the computer complex was performed in phases extending over a period of months.

The distribution of I/O activity across the eight 8433 disks in the short string was not surprising. Relatively low levels of activity on the remainder of the short string were considered desirable in order to avoid contention with the Exec disk, which could possibly harm the realtime flight simulation which is very time-critical. Hence, it was decided not to investigate the alternative of having fixed 8433 disks on the short string for a better load balance between the two strings of disk drives.

While the Exec disk was being monitored, the GDU often displayed values of Exec disk usage that exceeded 80% and occasionally even 90%. The GDU displays the average over the previous five-second interval whereas five-minute averages are stored in the PDFs. Since the high values observed at the GDU were short lived, such high values were not found in the PDF.

Values of Exec disk usage exceeding 90% were a cause of concern to NASA, and it was considered desirable to record these in order to investigate whether simulation crashes coincided with high Exec disk usage. This was accomplished with the use of Tesdata's Real-Time Exception (RTE) Processor. The almost daily occurence of several instances of Exec disk usage exceeding 90% suggested a need for detailed monitoring of the disk unit with the EXEC pack. Detailed monitoring of all disks is not possible because of the limited capacity of the Tesdata system; however, one disk unit can be monitored in detail. The EXEC pack was selected for detailed

monitoring because it is a read only pack and the placement of files on it can be controlled.

A new Tesdata MS-88D has recently been acquired for the Guidance and Navigation Simulator (GNS) in Building 35. The routine measurement procedure on this Tesdata system is presently being set up whereas the routine measurement on the Building 5 Tesdata system is being completely modified in the I/O area. The current plans include monitoring Exec disk activity and 5039 disk controller busy on both the Univac 1100/44 (4x2) systems in the SMSCC. The same measurements will also be performed on the GNS Univac 1100/44 (4x2). The six 8450 disks and the two 5046 disk controllers are now on the latter system in Building 35. Hence, the data transfer and head motion measurements on the six 8450 disks will be performed with the new Tesdata in Building 35. A detailed measurement of the Exec disk in Building 35 is also planned.

LIST OF ABBREVIATIONS

CAU	Command Arithmetic Unit
Ch	Channel
CU	Control Unit
DRS	Disk Resident System
Exec	Sperry Univac Series 1100 Executive System
GDU	Graphic Display Unit
GNS	Guidance and Navigation Simulator
GNSCC	Guidance and Navigation Simulator Computer Complex
HDA	Head/Disk Assembly
IBM	International Business Machines
I/O	Input/Output
IOAU	Input/Output Access Unit
JSC	Johnson Space Center
K	Kilo
MAI	Multiple Access Interface
MFD	Master File Directory
MMA	Multi-Module Access
MRS	Measurement Recording Software
MSA	Multi-Subsystem Adapter
NASA	National Aeronautics and Space Administration
PDF	Performance Data File
PMS	Performance Measurement Software
PSR	Processor State Register
RPS	Rotational Position Sensing
RTE	Real-Time Exception
SMS	Shuttle Mission Simulator
SMSCC	Shuttle Mission Simulator Computer Complex
SPI	Shared Peripheral Interface
SPU	System Partitioning Unit

REFERENCES

1. Hajare, A. R., An Interim Report on a Study of Disk Usage in the SMSCC, WP-6241, The MITRE Corporation, 1982.

2. Hajare, A. R., Use of a Hardware Monitor in the Shuttle Mission Simulator Computer Complex (SMSCC), WP-6229, The MITRE Corporation, 1981.

3. Gregor, P. J., Terminal Response Time in the Shuttle Mission Simulator Computer Complex (SMSCC), MTR-4740, MITRE Corporation, 1981.

4. Dias, Lakshmi A., et al., SMS/GNS FY 82 Computer System Plan, Volume 1: Review of FY 81, MTR-4747, MITRE Corporation, 1981.

5. General User Documentation (10 volumes), Tesdata Systems Corporation, 1979.

6. Sperry Univac 1100/40 Systems - Hardware System Description, UP-8216, Rev. 1, Sperry-Univac Division, Sperry Rand Corporation, 1977.

7. Sperry Univac 1100 Series Operating System - System Description, UP-8225, Rev. 4, Sperry-Univac Division, Sperry Rand Corporation, 1979.

8. Sperry Univac Series 1100 Executive System Level 36R2 Operator Reference, UP-7928.3, Sperry-Univac Division, Sperry Rand Corporation, 1979.

9. Sperry Univac 1100 Series 8450/5046 Disk Subsystem Programmer Reference, UP-8618, Sperry-Univac Division, Sperry Rand Corporation, 1978.

10. Sperry Univac 1100 Series 8405, 8430, 8433/5039 Disc Subsystems Programmer Reference, UP-8324, Rev. 1, Sperry-Univac Division, Sperry Rand Corporation, 1978.

The Application of Analytic and Simulation Models to Size a Large Computer System

Richard W. Tibbs

Martin Marietta Corporation
Denver Aerospace Division MS 8491
P.O. Box 179
Denver, Colorado 80201

John C. Kelly

Datametrics Systems Corporation
9940 Main St. Suite 201
Fairfax, Va. 22031

Martin Marietta Denver Aerospace, with the aid of Datametrics System Corporation, recently performed a comprehensive study of the performance and capacity of a government computer system and several short and long range alternatives. Performance measurement data were collected periodically from the current systems. These data were reduced and analyzed using the Statistical Analysis System, SAS, and were used as input to the modeling packages BEST/1 and SLAM. Models in each package were evaluated and predictions were used to propose one of several alternative systems.

Keywords: Modeling; performance evaluation; simulation; Univac systems.

1. Introduction

1.1. Overview

The systems under study in this report are a Univac 1100/40 (3x1)* and two long range replacement options, a Univac 1100/90 and an IBM 3081-k. The 1100/40 (3x1) was upgraded an 1100/80 (3x3) as a short term performance solution during the study. Several opportunities to measure the performance of the 1100/43, and later the 1100/83, were used to obtain accurate performance measurements of the system.

The system workload consists of a high volume transaction processing environment with critical Deadline Batch jobs imposing periodic peak loading. The workload analysis was bsed on three sources of data available within the Univac Exec-8 operating system: Log Analysis (LA), Software Instrumentation Package (SIP) and I/OTRACE. Non-linear regression statistics were used to forecast the growth of the critical Deadline Batch processing requirements.

Once the system workload was defined a baseline model was created and validated using actual performance measurement data. Sixteen alternative configurations were modeled to predict the performance of the 1100/83 upgrade until replacement by a long range alternative. Forecasts of the long range alternatives were produced as well, and results were presented graphically

* The terminology for configurations used in this report is: (nxm), meaning n processors and m I/O units.

using the SASGRAPH* capability of the Statistical Analysis System (SAS)*.

1.2. Modeling Objectives

The project was conducted in two phases. The first phase was involved proposal preparation and consisted of preliminary BEST/1** models to establish the adequacy of either the Univac 1100/90 or the IBM 3081-k as a satisfactory long range alternative to the Univac 1100/80. The second phase was more detailed and used a refined BEST/1 model in parallel with a simulation model written in SLAM*. Since the results from the second phase supercede those from the first phase, this paper focuses only on the more detailed results.

During the second phase, a detailed parametric analysis was performed. The sensitivities of the system to the following parameters were studied:

(1) Increased transaction arrival rate.

(2) Decreased transaction processing requirements.

(3) Increased data to be processed by critical deadline batch.

(4) Decreased CPU processing required by deadline batch due to a redesign .

(5) Decreased I/O processing requirements by all workloads due to newer and more efficient data base management system releases.

2. Approach

2.1. Analytic vs. Simulation Modeling

The modeling approach was based heavily upon performance data. In addition, the question of whether to use analytic queuing theory or discrete event methodology (simulation) was examined closely.

Current system utilization information was available from the Univac performance monitors, and initially the level of abstraction of BEST/1 seemed to be too high (not enough detail) to capture all of the effects of operating system peculiarities and architectural differences between target systems. On the other hand, a discrete event simulation built in SLAM was felt to be to cumbersome, although capable of lower levels of abstraction (more detail). From experience with SLAM we also knew that execution time of model runs would be slow due to the greater computational complexity of a simulation model and the fact that SLAM was hosted on a VAX 11/780 whereas BEST/1 was available on an IBM 3081.

Finally, it was decided that both BEST/1 and SLAM would be used and compared with each other during the course of the project. A summary of our reasoning is:

(1) We felt we needed the faster turnaround of a tool like BEST/1 running on a large mainframe in order to respond quickly to design questions and the need for analysis. we felt that initially, the greater detail of a SLAM model was not necessary.

(2) Using two independent approaches would provide a high degree of confidence in the model results.

(3) BEST/1 could be implemented easily to provide results in the early phases of the proposal effort. As more detailed results were needed later in the project, the SlAM model would be available for use.

2.2. BEST/1

BEST/1 allows the following elements to be modeled explicitly:

(1) Servers and queues:

.... CPUs,

.... Disk, drum and other devices,

.... Channels,

.... Memory (a passive server)

(2) Workloads:

.... Time sharing (TS),

.... Transaction processing (TP),

.... Batch processing (BP).

Time sharing (TS) workloads assume that the arrival rate is a function of the service rate, and the arrival pattern is described in terms of operator think time. Tranaction processing (TP) workloads assume that the arrival rate is independent of the service rate, and the arrival pattern is described in terms of the mean arrival rate. Batch processing (BP) workloads assume that a backlog always exists, and that the arrival rate is equal to the service rate.

Two advanced features of BEST/1 that proved beneficial were the ability to model:

(1) RPS disk activity;

---

* SAS and SASGRAPH are trademarks of SAS Institute Inc.
** BEST/1 is a trademark of BGS Systems Inc.
*** SLAM is a trademark of Pritsker and Associates Inc.

(2) Priority levels;

.... We implemented 5 priority levels to correspond to the Univac user types exec, real time, tip, deadline, batch.

The following system attributes were modeled implicitly by adjusting the model input data: data base efficiency, memory fragmentation, paging, and cache disk operation. Additional information on BEST/1 may be found in the BEST/1 manual [1] .

2.3. SLAM

The Simulation Language for Alternative Modeling (SLAM) is a general purpose simulation language designed to model discrete event Monte Carlo type simulations. SLAM also offers the features of network modelling methods and continuous event capabilities in an integrated framework. Detailed information on SLAM is available in the SLAM users guide [2] .

To aid in validataion and to provide a baseline for future enhancements the initial SLAM model was developed at the same degree of detail as a BEST/1 model. Once confidence at this level was achieved, additional details could be introduced. We defined a generalized processor model (GPM) as a tool to model all of our processor alternatives in SLAM.

The events chosen to be explicitly modeled in the GPM using the discrete event capability of SLAM, and which closely approximate the level of detail of BEST/1 were:

(1) Arrival;

(2) Request memory;

(3) Request CPU;

(4) CPU quantum exceeded;

(5) Request I/O device;

(6) I/O complete;

(7) Processing complete;

(8) Swap out (priority preemption);

(9) Memory quantum exceeded;

The workload types available in the GPM, following the terminology and assumptions of BEST/1, were chosen as:

---

[1] BGS Systems Inc.
BEST/1 Users Guide, BGS Systems, 1979.

[2] Pritsker, A. A. B. and Pegden, C. D.
Introduction to Simulation and SLAM, Halsted Press, N. Y. 1979.

(1) Transaction processing;

(2) Batch processing;

The GPM was implmented in SLAM by the Central Software Engineering Facility (CSEF) Model Shop, an independent simulation group at Martin Marietta Denver Aerospace.

3. Target Systems

3.1. Univac 1100/40 and 1100/80 Baseline Systems

The baseline systems from which extensive performance measurements were taken, are shown in figure 1. During the course of the modeling effort, an 1100/80 system was installed side by side with the 1100/40, and eventually replaced it. In addition to the increased CPU power, differences between the 1100/40 and the 1100/80 included:

(1) Doubling the existing memory from 1.5 Mwords to 3.0 Mwords.

(2) One more Amperif cache disk subsystem and a total of 11 new model 8434 300 Mbyte disks added to the cache subsystems.

(3) Two new 8433 disks added to the conventional disk subsystems.

(4) Two more IOUs added to the system.

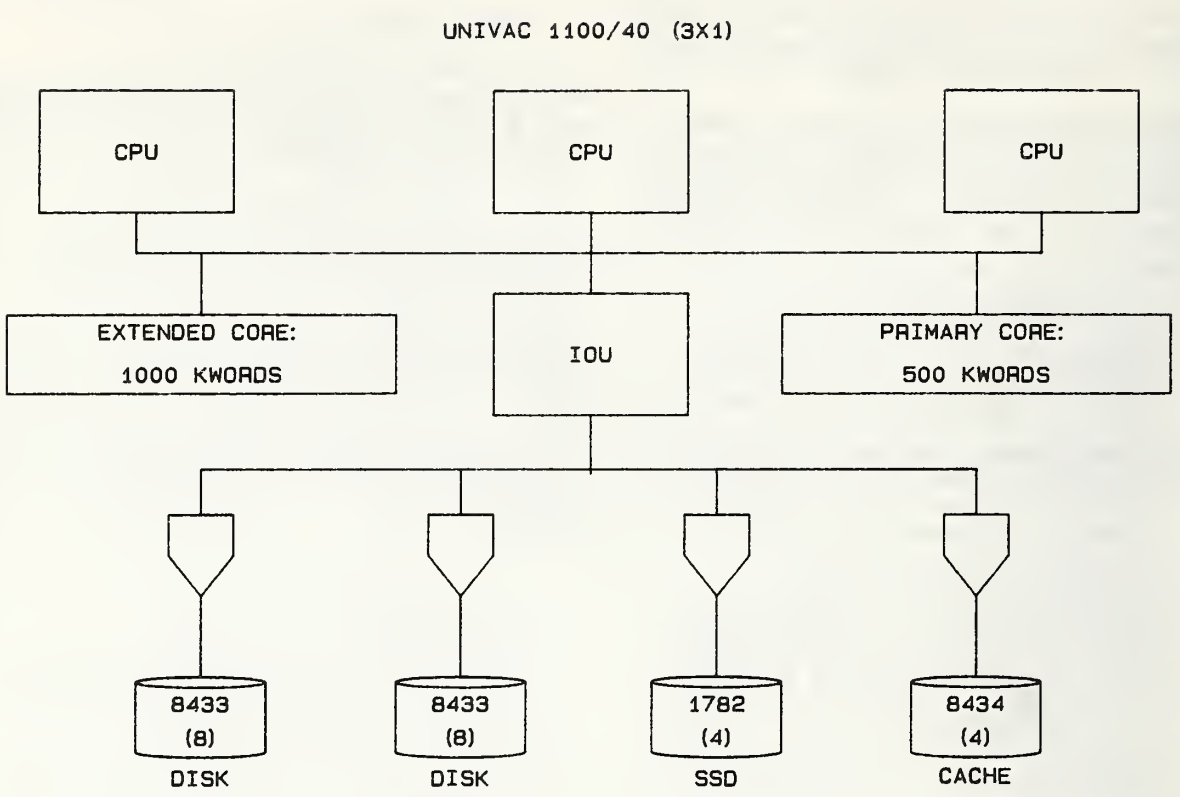
The addition of a new system (the 1100/80) under the same loading as the original baseline system (the 1100/40) offered a unique opportunity to validate the BEST/1 models using two sets of measurements.

A baseline model of the 1100/40 system was built and validated. When performance data was available from the 1100/80 system, the 1100/40 model was modified to reflect the 1100/80 hardware features (no significant software changes were involved). This model predicted the 1100/80 performance within 10% of actual measurements.

3.2. Univac 1100/90 and IBM 3081-K Replacement Systems

The long range replacement systems are shown in figures 2 and 3. The figures show a variable number of processors (or processor pairs) because we modeled a range of multiprocessor configurations for each system, ranging from one to three or four processors. The purpose of this was to establish minimum configurations as well as more generous ones capable of satisfying response time requirements under projected loading.

A



B

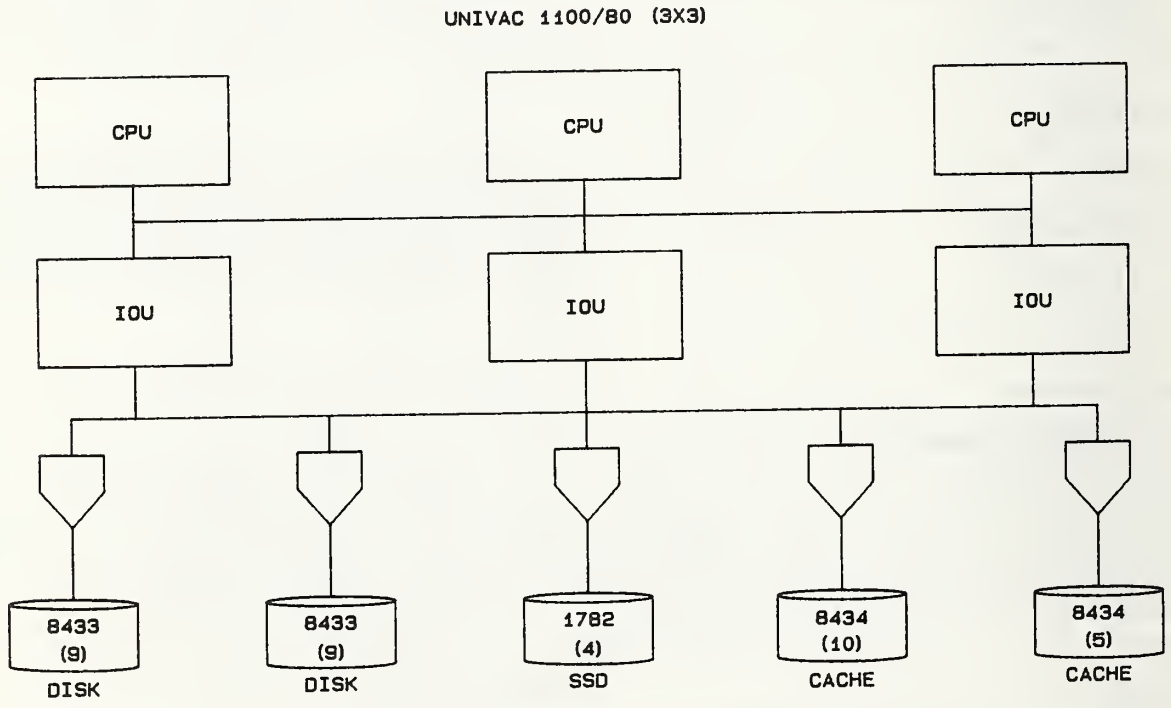


Figure 1. Baseline Systems

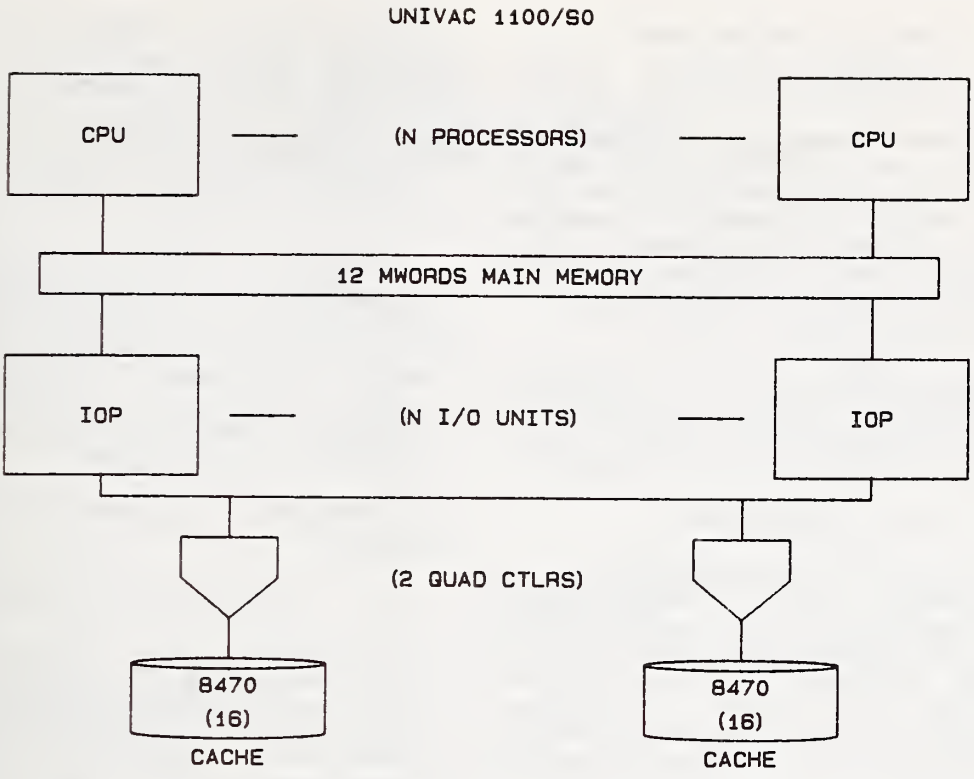


Figure 2. Univac 1100/90 Replacement

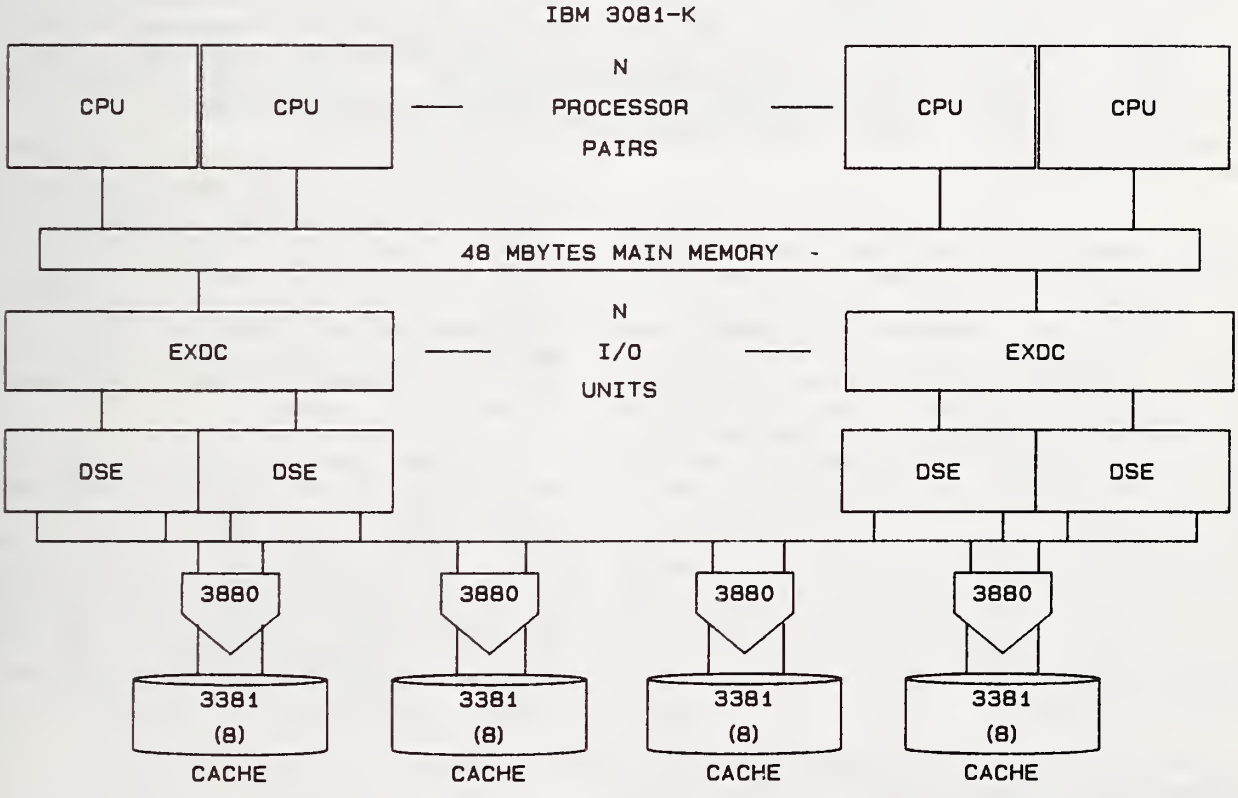


Figure 3. IBM 3081-K Replacement

3.3. Modeling Workload Growth

3.3.1. Transaction Loading

An independent model of the operational environment was used to predict future transaction volumes. This model was based on staff loading and operational scenarios and generated estimates of future transaction volumes as output. The Operations model is outside the scope of this report, but its output was used as input to our BEST/1 and SLAM processor models.

The Operations model predicted a worst-case load of approximately 300 transactions per minute for the short term time frame, and 400 transactions per minute in the long term time frame.

3.3.2. Batch Loading

Two types of batch workload existed in the Univac environment, Batch and Deadline Batch. There was no response time requirement given for Batch processing; however Deadline Batch did have a response time requirement. We will concern ourselves primarily with Deadline Batch loading, discussed in the next section, and be content to merely report the available processing capacity for Batch workloads.

3.3.3. Deadline Batch Loading Analysis

Non-linear regression statistics were used to fit a second order (quadratic) and a third order (cubic) approximation to sets of CPU and I/O resource utilization data collected on the Deadline Batch programs. These approximations were used to plot Deadline Batch CPU seconds and I/O seconds as functions of the number of data items to be processed.

We acquired our resource usage data from Log Analysis output for individual Deadline Batch runs. The amount of data items to be processed by each Deadline Batch program is well known by the time of day in our customers environment, and could be correlated exactly with resource usage statistics in the Log Analysis output. Each data point consisted of three values, the number of data items to be processed by the Deadline Batch programs, the amount of CPU seconds used, and the amount of I/O seconds used. The data is summarized in figure 4-a and b, where the quadratic regression curves appear just above the cubic curves. A linear regression was also performed as a reference. The linear approximation appears as a solid line. Figure 4-a shows CPU sups (CPU seconds) , while figure 4-b shows I/O sups (I/O seconds) graphed as functions of the number of data items processed.

The quadratic fit was chosen as the better estimator because the cubic fit was found to possess a second derivative sign change near the projected number of items to be processed. This can be seen most clearly in figure 4-a, where the cubic approximation to Deadline Batch CPU requirements visibly begins to decline near the required number of data items for the long term time frame (1100 data items). With this anomaly, the cubic approximation, although of higher degree, estimated less resource utilization than the quadratic case. We therefore chose the quadratic fit as the more accurate, if perhaps more pessimistic, estimator.

The purpose of this loading analysis was to demonstrate the non-linearity of the Deadline Batch resource usage. The non-linearity pointed out the folly of assuming linearity in the resource requirements of future Deadline Batch loads, and motivated an investigation of the efficiency of the computational methods used in the Deadline Batch programs.

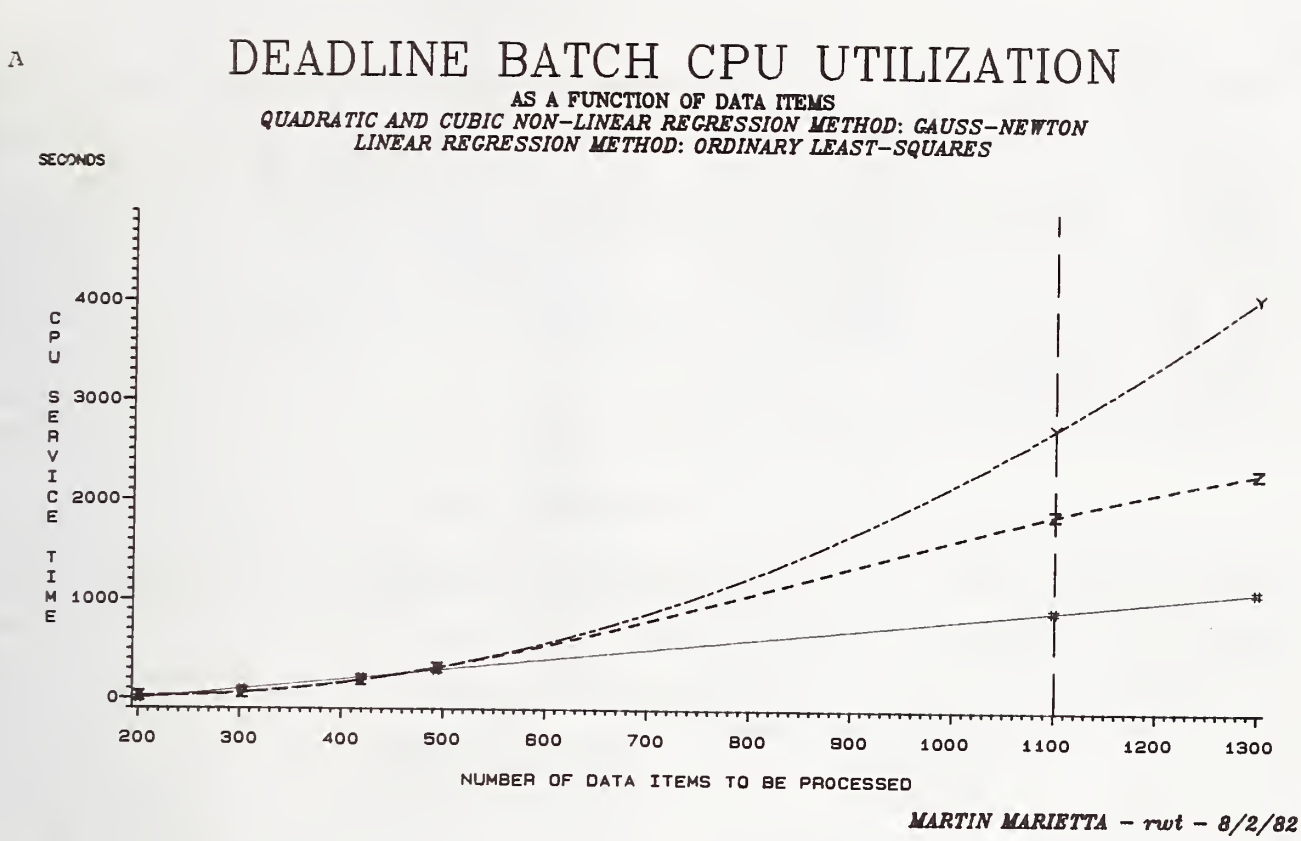
4. Performance Measurement and Input Data Preparation

4.1. Schedule of SIP, LA, and I/OTRACE Data

The acquisition of performance measurement data throughout the time span of the project followed the schedule shown in table 1. The data was collected in an inconsistent manner, as the comments column indicates. This was a detrimental factor because although there were ten collection dates (enough for a reasonably sized sample bucket), the same type of data was not available on each date. Thus no time dependent statistics such as trend analyses could be reliably generated. For a steady state measurement of instantaneous system characteristics, we were able to use the September 28, 1981 sample data, which operational personnel identified as representative of a peak loading period.

Insofar as averaging the performance measurements over a large period of time tends to destroy the relationships between loading and response times, the presence of data from many sample points was only useful for examining different load mixes and for trend analysis. For this reason we were content to rely on our most complete sample point, the 28th of September 1981, for performance and model validation data for the 1100/40 system.

The later sample dates in 1982 were all measurements of the 1100/80 system, which was installed late in 1981 and became operational in January 1982. Particularly, the data from the 5th of January 1982 (a complete set of LA, SIP, and I/OTRACE measurements) was used to revalidate the BEST/1 models. The results of the comparison of 1100/80 model predictions and 1100/80 performance measurement are discussed in a later section.



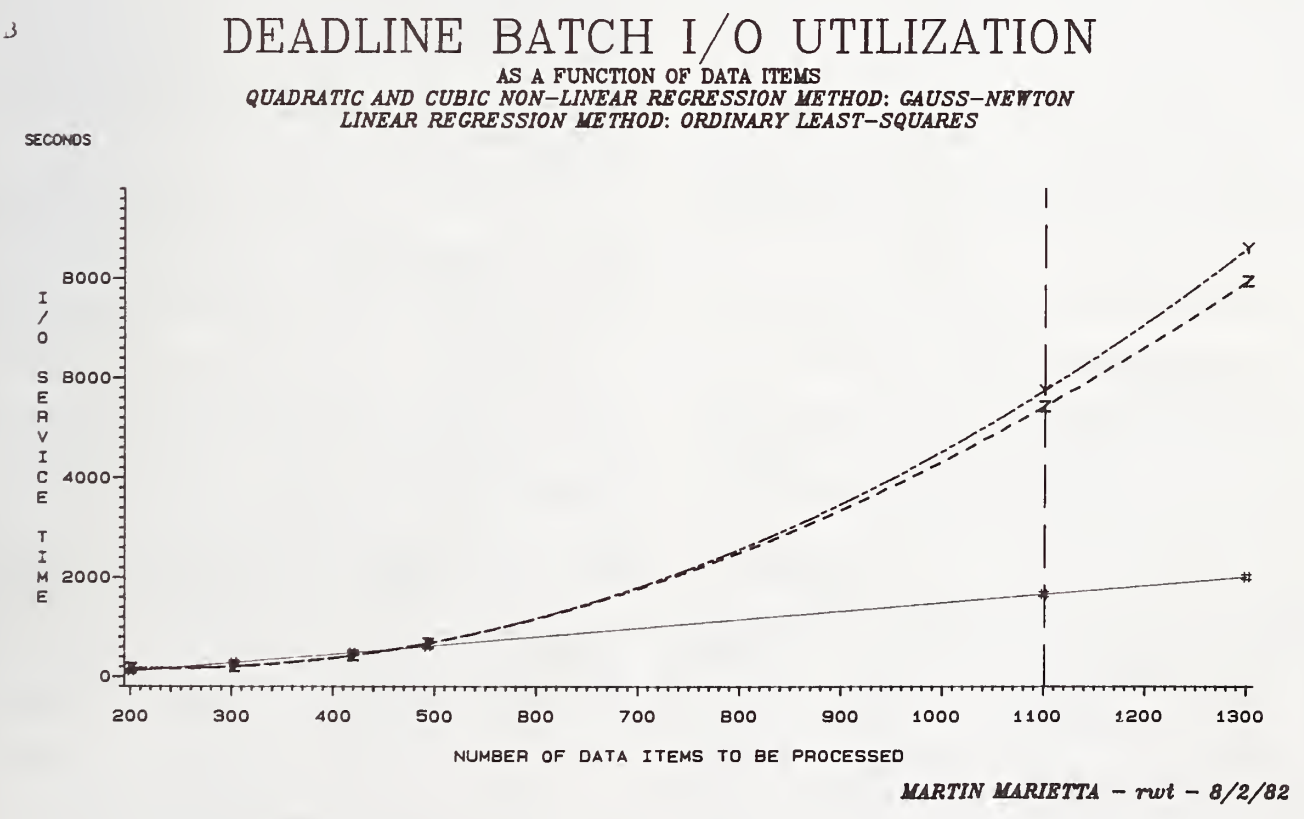


Figure 4. Deadline Batch Resource Utilization

Table 1. Measurement Data Schedule

date	SIP	LA	I/OTRACE	comments
16 Mar 81			14:30-15:00	some fields blank, no tape
12 May 81		13:14-10:57		one large report, no tape
21 Sep 81	01:01-23:00	00:00-23:00	22:30-23:23	1/2 hr. SIP reports, I/OTRACE by file
28 Sep 81	01:00-23:00	00:00-24:00	09:15-09:50	same as above
17 Nov 81		00:19-06:45		
16 Dec 81		07:00-07:30		only 8 most used TIPs reported
05 Jan 82	00:38-03:44	00:20-02:11	00:38-03:44	I/OTRACE by runid
26 Jan 82		01:52-23:49		1/2 hr. LA reports
08 Feb 82	07:00-16:00			1/2 hr. SIP reports
22 Feb 82	01:00-23:44	00:00-24:00	08:15-11:43	1/2 hr. SIP reports, I/OTRACE by file

Table 2. Workload Characterization

workload characteristic		performance measurement source	comment
workload type -BP,TP,TS		Exec priority levels	
priority		Exec priority levels	
arrival rate	TIP (TP) Deadline (BP) Batch (BP)	exec PI$NIT frequency run summary report sys idle activity report	n/a for BEST/1 since no arrival for BP loads. Used for GPM.
MPL	TIP Deadline Batch	SIP memory util. report LA run summary report LA graph of runs active	
Response Time	TIP Deadline Batch	LA TIP response time dist. report LA run summary report LA run summary report	
Throughput	TIP Deadline Batch	n/a LA run summary report SIP sys idle activity report	
CPU time	(all workloads)	SIP CAU report	
I/O time		I/OTRACE+S/W	

Table 4. Exec and Real Time Apportionment Results

workload	CPU seconds per hour user class	Exec	R/T	Total sec/hour	Arrivals/ hour	CPU sec/ arrival
TIP	6772.56	2689.08	96.48	9558.12	7442	1.284
fraction of total	.71	.28	.01			
Deadline	150.40	185.68	0.00	336.08	.86	390.791
fraction of total	.45	.55	.00			
Batch	354.60	437.68	0.00	792.28	48.8	16.235
fraction of total	.45	.55	.00			

Table 1 contains entries showing the time span over which SIP, LA, or I/OTRACE statistics were collected, or contains blank entries where no data for a given monitor was collected. The times are expressed in military format. The comments field indicates that in some instances the data was not available on tape, making processing sometimes tedious, if not impossible, since it could only be done by hand. The inconsistency of the data collection process shown here illustrates the need for data collection planning to be done far in advance of a performance analysis effort.

4.2. Data Reduction and Transformation Procedure

The objectives of the data reduction effort, applied to the performance data, were to produce a set of input parameters to populate BEST/1 and SLAM/GPM models. As discussed earlier, the workloads, defined by priority classes for out models were:

(1) TIP

(2) Batch

(3) Deadline Batch

The real-time priority workload was very small (less than 1% of total CPU utilization) and was factored into the TIP load since the primary real time program in the Univac environment, CMS, is invoked continually to handle TIP communications processing. The Exec priority levels (Exec0,1,3) were apportioned among the TIP Batch and Deadline Batch workloads by the relative proportion of I/O SUPS accrued by each workload. These methods of dealing with real time and Exec loading were gleaned from two technical notes from BGS Systems Inc. [3] , [4]
The basic assumption of this apportionment technique is that Exec CPU usage is dominated by I/O interrupt processing for the rest of the user types.

For each of the workloads the data necessary to populate the BEST/1 and SLAM models is summarized in table 2. The table shows for each performance parameter or group of performance parameters which software monitor and what report was used to derive the parameter. Noted in the performance measurement source column are some details concerning the derivation of the parameter from the source.

Sources of the arrival rates for the workloads were among the most difficult to find. For example, the number of times that the exec routine PI$NIT is called is good gauge of TIP arrival rate. This value is available in the SIP Exec summary report, a table of all exec routines and their frequency of invocation. The arrival rates of Deadline Batch and Batch workloads, although unnecessary for BEST/1, were required for the SLAM/GPM. In the case of Deadline, the run summary report from LA was sufficient, but for Batch a more direct tabulation appeared in the SIP system idle activity report. This report gives the number of runs opened and number of runs in backlog, for the period of the report. We had five 1/2 hour reports, and were able to calculate the average growth of the backlog. This number, subtracted from the average number of runs opened gave us an accurate throughput figure (see next section).

I/O time per device for each arrival in each user class was extracted from I/OTRACE data using special data reduction software. The normal I/OTRACE processor, in the case of the 1100/40 system, does not sort the raw data according to runid, which can be used to distinguish between user classes, nor does the PAR processor in the case of the 1100/80. Furthermore, because our I/OTRACE data was collected for only a half hour, no Deadline Batch runs were captured by coincidence.

We resorted to an apportionment method to determine the I/O seconds per device per Deadline Batch run. We knew that I/OSUPS were an estimate of I/O time devised for accounting purposes, and that they were not reliable as absolute measures of I/O seconds. The method by which I/OSUPS are calculated does make them a reliable relative measure, however, and we used the ratio of I/OSUPS taken from the LA run summary reports and applied this ratio multiplicatively to the Batch I/O seconds from I/OTRACE to produce Deadline Batch I/O seconds.

4.2.1. Arrival Rate, CPU Utilization, and Batch Throughput Calculations

In addition to the characteristics in the previous section, for TIP, Batch and Deadline Batch, table 3 shows the I/O SUP figures used to apportion Exec CPU usage among these workloads. I/O SUPS were derived form the Log Analyzer output of 28 September 1981, while Exec CPU seconds were derived from SIP collected on the same date.

The results of the Exec apportionment added to the CPU utilization of TIP, Deadline Batch, and Batch are shown in table 4 along with arrivals per hour for each workload used to convert CPU seconds per hour to CPU seconds per arrival. All real time CPU utilization was attributed to TIP. The sources of all figures, with the exception of Deadline Batch and Batch arrival rates were derived from SIP output on the 28th of September 1981. The Deadline arrival rate was determined simply by tallying the deadline runs over a period of time in the run summary report. Batch arrival was determined from the

---

[3] BGS Systems Inc.
Technical Note #10, BGS Systems, 1981.
[4] BGS Systems Inc.
Technical Note #15, BGS Systems, 1981.

Table 3. Apportionment of Exec CPU Usage among TIP, Batch, Deadline

work-load	I/O SUPS/hr.	fraction of total	apportioned exec CPU seconds
TIP	4206.57	.811	2689.08
Deadline	293.15	.056	185.68
Batch	687.51	.132	437.68
Total			3312.44

system idle activity report in SIP, as mentioned earlier. The calculations performed to extract Batch arrival rate and throughput are summarized in table 5.

A report duration of 1/2 hour with an average size of 24.4 Batch arrivals implies 48.8 arrivals per hour of which 5 arrivals are backloged. This in turn implies a throughput of 43.8 per hour. The trend in the backlog grows slowly during the peak period over which the SIP data was collected, which is to be expected since TIP programs at higher priority will always preempt batch processing.

Table 5. Batch Arrival and Throughput

Report Number	Total runs opened	Avg runs in backlog	Backlog trend
1	33	11.9	
		+1.8	
2	22	13.7	
		+10.2	
3	36	24.9	
		-4.2	
4	26	20.7	
		+3.7	
5	5	24.4	
averages	24.4		+2.5

4.2.2. Memory Utilization Calculations

The memory use of the system is calculated here in terms of the number of active transactions (or jobs) occupying memory. This number is called the Multiprogramming Limit (MPL) in BEST/1 terminology, and we adopted this terminology for the SLAM/GPM as well. The TIP MPL calculations are shown in table 6. BGS technical note 15 [4] gives a simple formula for calculation of the MAX-MPL parameter for a TIP workload as (Average Total Memory Used)/(Average Program Size) + 1. This value, calculated from the figures in the table, is approximately 13.

Memory use of Deadline Batch was implied from the regular nature of this workload's arrival (about one per hour) and the throughput calculated from LA run summary information (greater than one per hour). With this information we determined that the attained multiprogramming limit (ATT-MPL in BEST/1 terminology) was approximately one.

Batch processing memory use was obtained from the Batch graph of runs active, a line printer histogram in the LA output. This histogram showed the fluctuations of the number of Batch Jobs active over time, so values for the attained multiprogramming distribution (ATT-MPD in BEST/1 terminology) could be obtained. The calculations for these values are shown in table 7.

The average of the ATT-MPD (the equivalent of ATT-MPL) was used in the SLAM GPM since a distribution feature was not implemented. This value, computed as the time-weighted sum of the values in the table, is 1.92.

4.2.3. I/O Utilization Calculations

The source of our I/O device service times was the output of the I/OTRACE monitor, summarized by a specially written program to extract I/O utilization to each user type and the Exec. The I/O configuration of the baseline 1100/40 system consisted of two conventional subsystems

Table 6. TIP Memory Utilization

Average Total Memory Used by TIP Programs (Kwords)	Average Resident TIP Program Size (Kwords)
380.18	31.96

Table 7. Batch Memory Utilization Distribution

Level of Batch runs active	Minutes Spent at that level	Fraction of total
0	0	0
1	19.78	.39
2	15.20	.30
3	15.85	.31
.4	0	0
total	50.83	1.0

of Univac model 8433 disks with a total of 14 disks operating on the date of measurement, September 28, 1981. An Amperif cache disk subsystem with 4 Amperif 8434 drives and an Amperif Solid State Drum (SSD) subsystem with 4 SSD units were also present. Table 8 shows the average device service time in milliseconds per arrival per device for each workload across all subsystems.

The distribution of the disk activity within each subsystem is shown in figure 5 and in figure 6 for TIP and Batch respectively. A distribution for Deadline Batch was not available and simple averages were used instead.

These distributions show the lack of balance present in the disk usage; essentially a data base design problem. By assigning values derived according to these distributions to the device service time array in BEST/1, the baseline model captured the unbalanced nature of I/O use and its associated effects on disk queuing and transaction response times.

Table 8. I/O Device Service Time by User Type

Subsystem	TIP	Deadline	Batch
	(msecs / arrival / device)		
Disk 1	12.32	3817.40	627.28
Disk 2	4.31	3817.40	131.58
Cache	31.82	56716.81	432.85
SSD	18.53	0	310.99

4.3. BEST/1 Input Array

The BEST/1 input array is shown in figure 7, representing the culmination of the results presented in the preceding sections. The SLAM/GPM was populated with essentially identical information, but the format and length of the actual SLAM data structure prohibits its presentation here.

5. Model Results

5.1. Validation of Baseline Models

Baseline model results compared very favorably with baseline system measurements, instilling a good level of confidence in our models ability to accurately predict performance. Figure 8 shows a plot of the response time distribution predicted by BEST/1 superimposed on a plot of the observed response time extracted from Log Analyzer output on the September 28 1981 sample date.

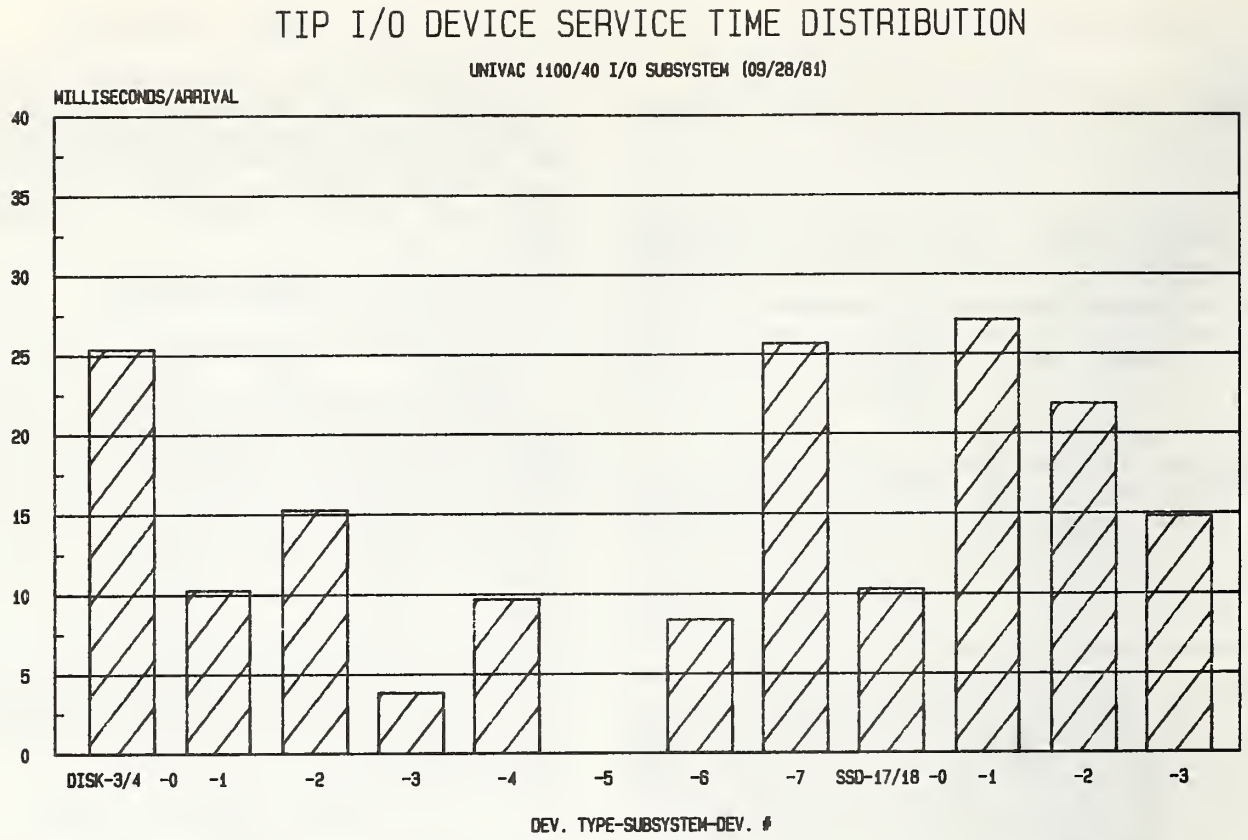
The baseline models were tuned by comparison with the response time distribution information so that 95th percentile response time was approximately matched.

5.2. Model Prediction Results

The results of the parametric analysis (see introduction) performed upon the various models is summarized in the following sub-sections. The parameterization is illustrated in table 9 as a matrix of percentages of change applied to the Deadline Batch CPU time requirement, all I/O service time requirements, and TIP CPU time requirements. The sensitivity of the system performance to these three parameters was of highest interest to design decisions being made at the time, and therefore defined the extent of the parameterization.

Table 9. Parameterization Values

Value	Percentage Reduction		
TIP CPU seconds	0.%	-10.%	-20.%
Deadline CPU	0.%	-20.%	-40.%
All I/O seconds	0.%	-45.%	-90.%



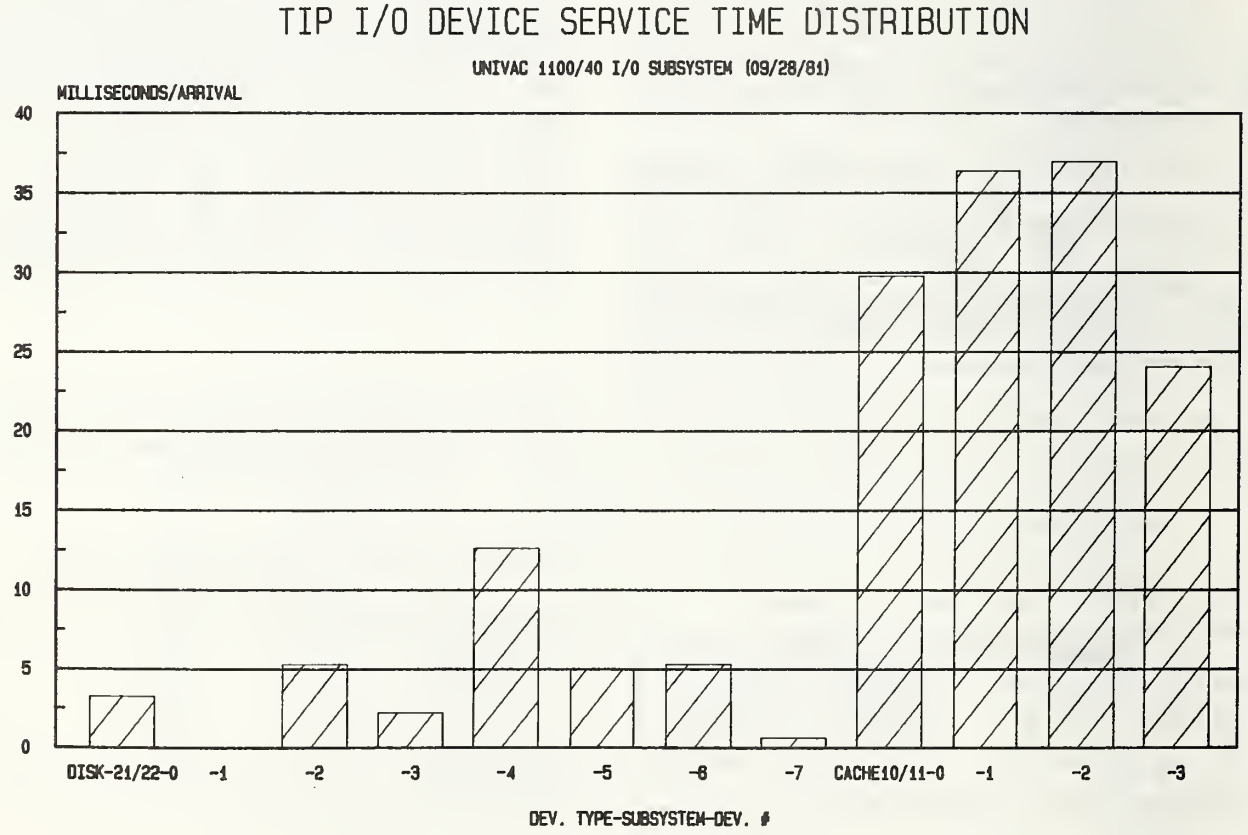
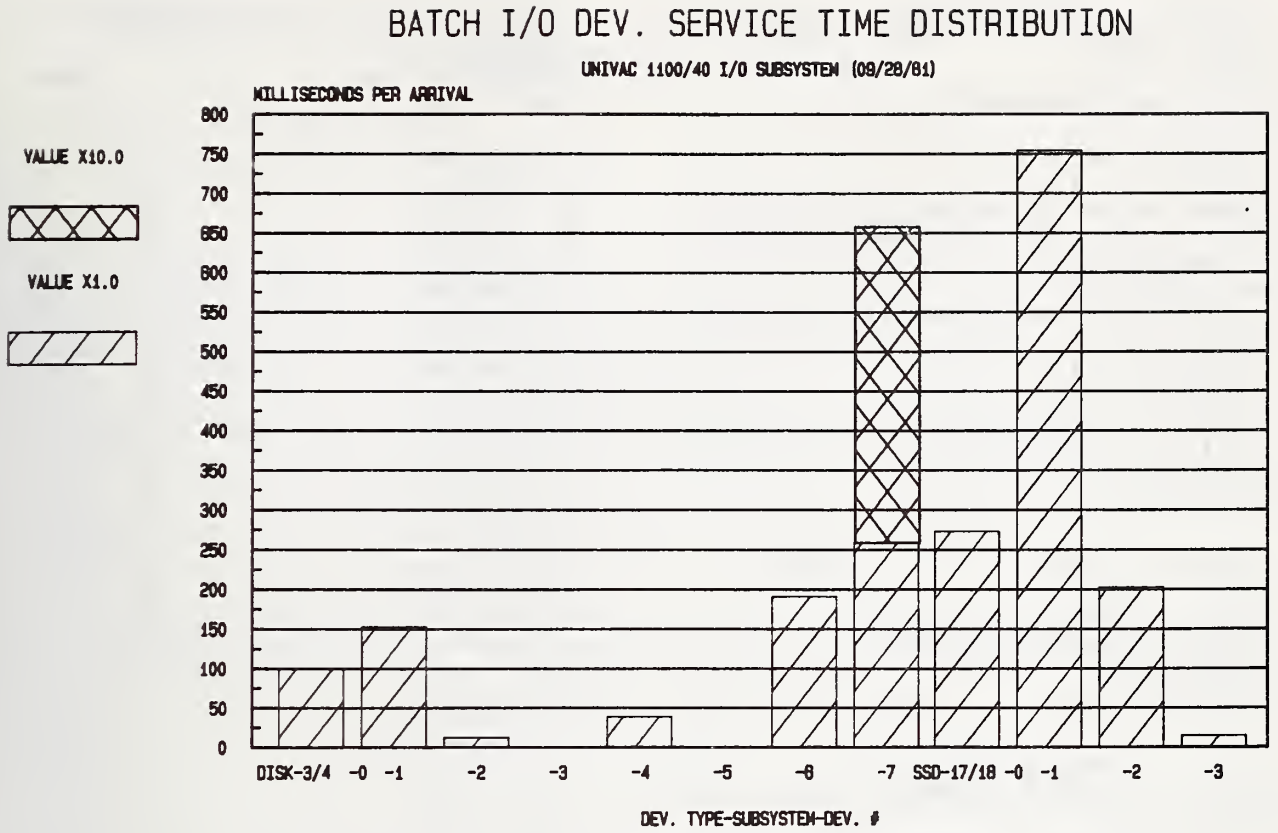


Figure 5. TIP I/O Device Service Time Distribution



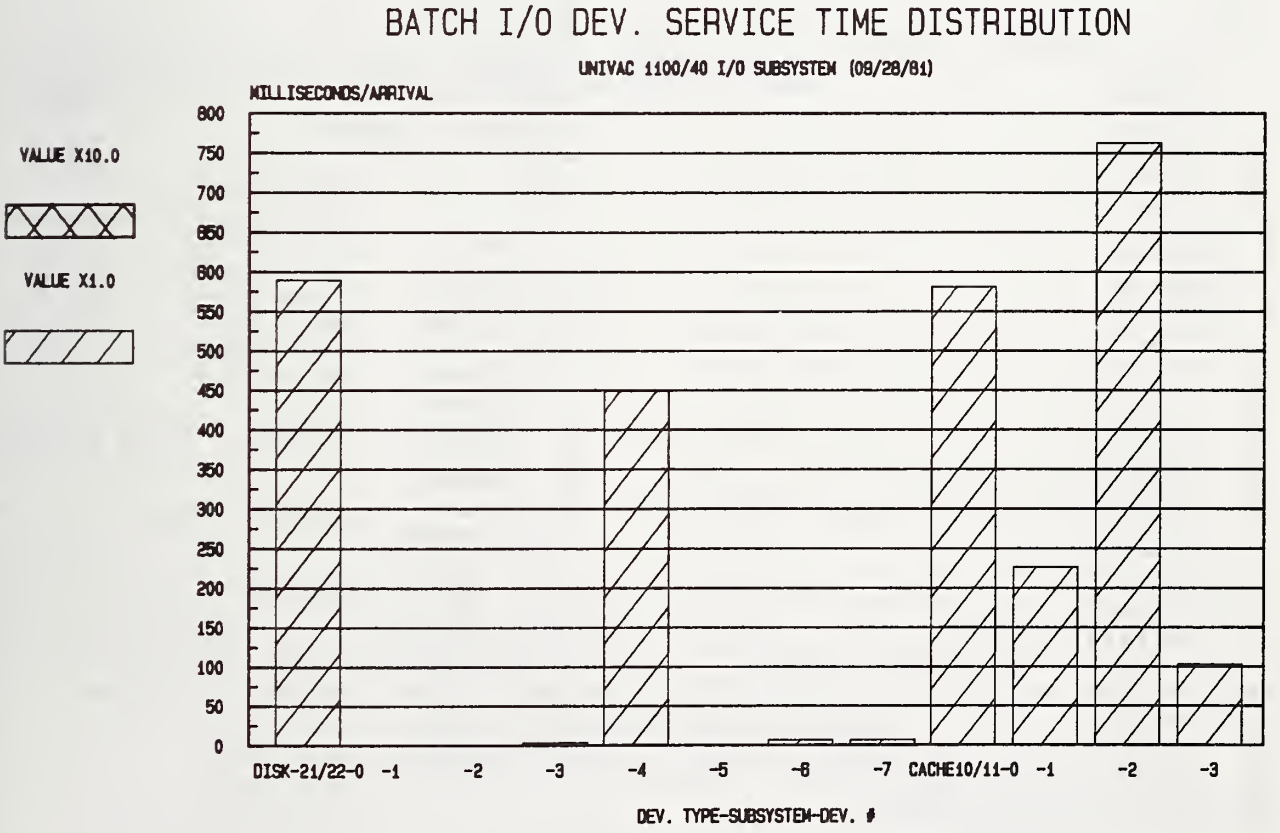
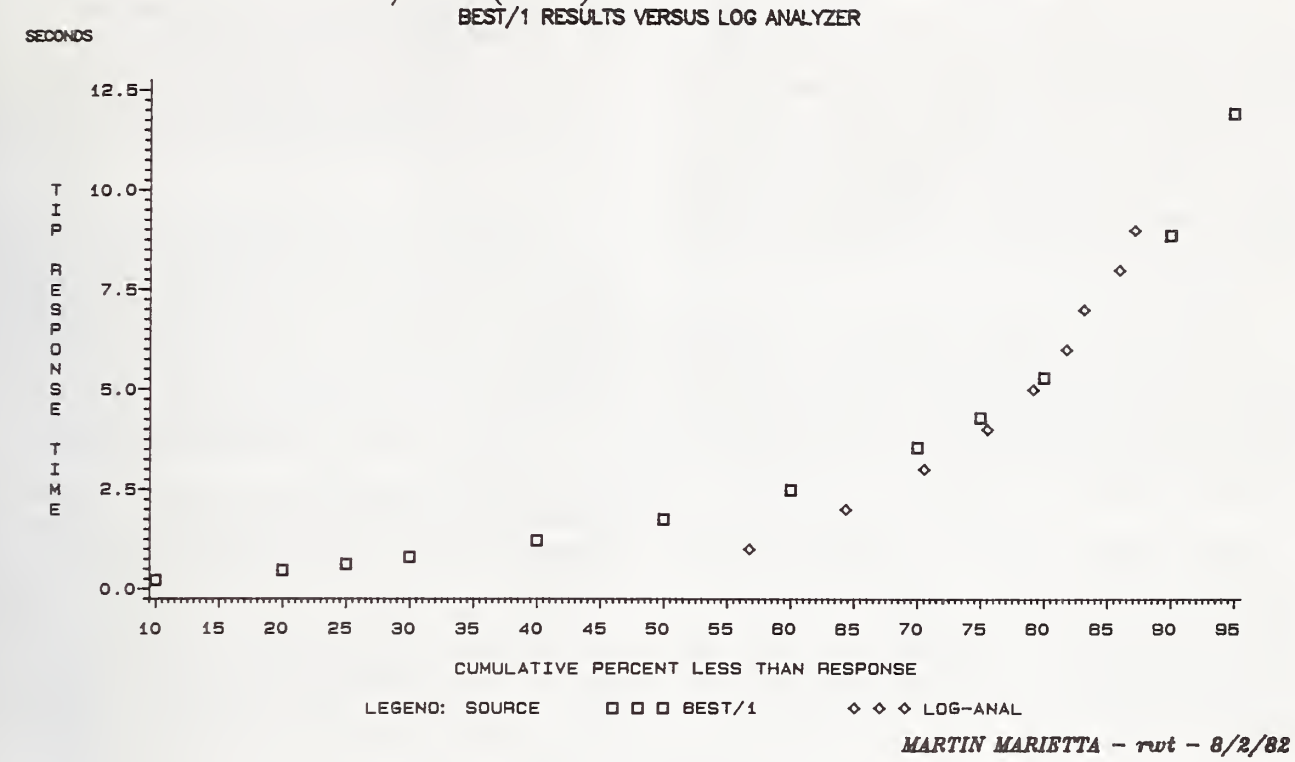


Figure 6. Batch I/O Device Service Time Distribution

```
LCHANNEL 1 LABEL = 'DISK    CHANNEL  '
PCHAN = 2.0
CHTIME = 3.4
SERVER 28,29,49,50,51 LCH-ID = 1
STR-ID = 1
REVTIME = 4.0
DEVTIME = 6.5
SERVER 4,5,6,7,8,9,10,11 LCH-ID = 1
STR-ID = 2
REVTIME = 8.3
DEVTIME = 13.8
SERVER 12,13,14,15,16,17,18,19 LCH-ID = 1
STR-ID = 3
REVTIME = 8.3
DEVTIME = 14.6
SERVER 20,21,22,23,24,25,26,27 LCH-ID = 1
STR-ID = 4
REVTIME = 8.
DEVTIME = 15.

LCHANNEL 2 LABEL = 'SSD CHANNEL'
PCHAN = 2.00
CHTIME = 7.31
SERVER 2,3,41,42 LCH-ID = 2
STR-ID = 1
REVTIME = 0.
DEVTIME = 7.32

SERVER 1 SPEED = 3
WKLOAD 1 LABEL = 'CURRENT DEADLINE BATCH
TYPE = BP
ARRIVAL = 1.0
ATT-MPL = 1.0
TIME

1 = 390791
2 = 00000.0
3 = 00000.0
41 = 00000.0
42 = 00001.0
4 = 7634.80
5 = 7634.80
6 = 7634.80
7 = 7634.80
8 = 7634.80
9 = 7634.80
10 = 7634.80
11 = 7634.80
12 = 7634.80
13 = 7634.80
14 = 7634.80
15 = 7634.80
16 = 7634.80
17 = 7634.80
18 = 7634.80
19 = 7634.80
28 = 56716.0
29 = 56716.0
49 = 56716.0
50 = 56716.0
51 = 0000.0
35 = 223.0
30 = 5698.0
31 = 288.0

WKLOAD 2 LABEL = 'REAL CURRENT BATCH'
TYPE = BP
ARRIVAL = 48.8
TIME

1 = 16235
2 = 272.1
3 = 753.09
41 = 202.46
42 = 16.32
4 = 98.690
5 = 152.56
6 = 12.550
7 = 000.00
8 = 39.680
9 = 000.00
10 = 189.67
11 = 4528.3
12 = 590.98
13 = 000.00
14 = 000.00
15 = 2.5500
16 = 448.28
17 = 000.00
18 = 5.5300
19 = 5.3200
28 = 580.22
29 = 227.34
49 = 761.49
50 = 162.34
51 = 000.00
35 = 27.0
30 = 694.00
31 = 35.0

WKLOAD 3 LABEL = 'REAL CURRENT TIP
TYPE = TP
ARRIVAL = 7442.0
MAX-MPL = 13
TIME

1 = 1284
2 = 10.31
3 = 27.20
41 = 21.80
42 = 14.82
4 = 25.41
5 = 10.24
6 = 15.27
7 = 03.89
8 = 09.66
9 = 00.00
10 = 08.38
11 = 25.74
12 = 03.24
13 = 00.00
14 = 05.36
15 = 02.21
16 = 12.69
17 = 05.00
18 = 05.31
28 = 29.84
29 = 36.37
49 = 37.00
50 = 24.08
51 = 00.00
35 = 01.00
30 = 29.00
31 = 01.00

WKLOAD 1 PDIST 2 = 0.45
         PDIST 5 = 0.55

WKLOAD 2 PDIST 1 = 0.45
         PDIST 5 = 0.55

WKLOAD 3 PDIST 3 = 0.71
         PDIST 4 = 0.01
         PDIST 5 = 0.28

WKLOAD 2 ATT-MPD

1 = 0.39
2 = 0.30
3 = 0.31
```

Figure 7. Best/1 Input Array

A BASELINE 1100/40 (3X1) RESPONSE TIME DISTRIBUTION



B BASELINE 1100/80 (3X3) RESPONSE TIME DISTRIBUTION

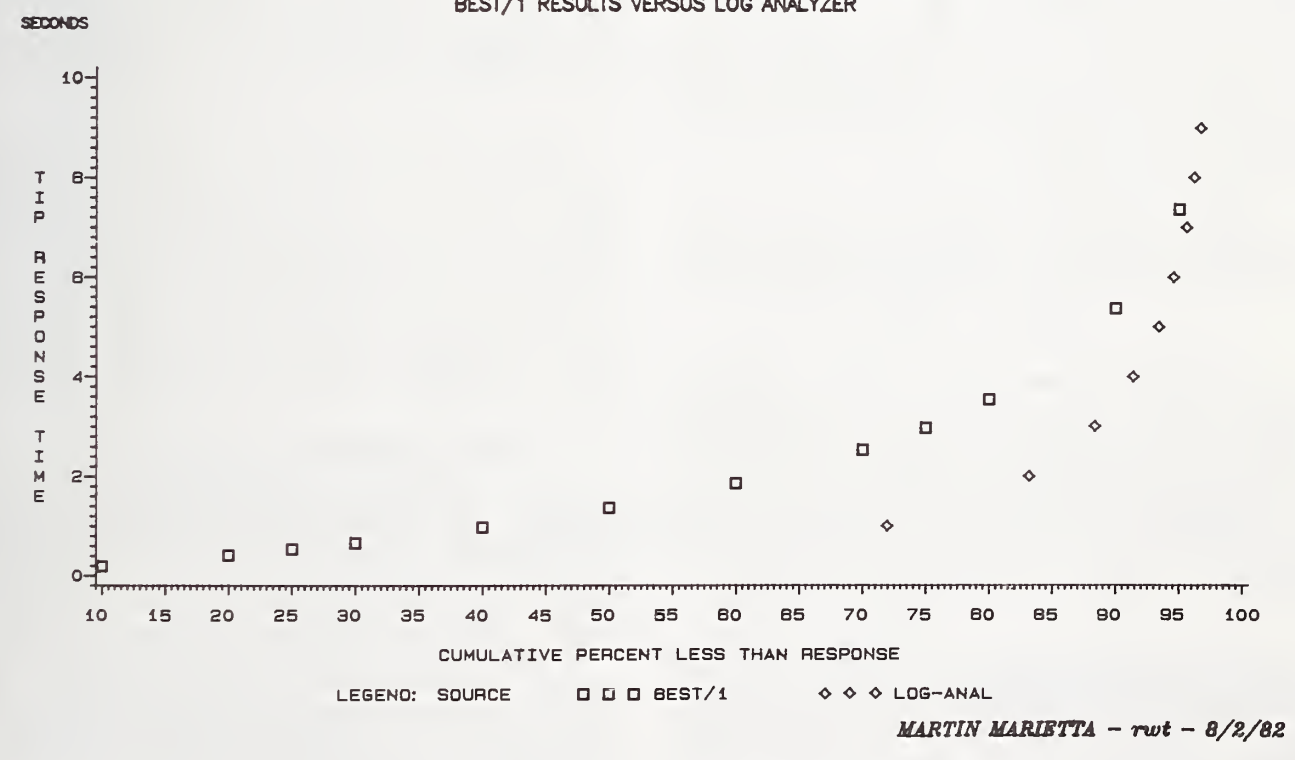


Figure 8. Baseline Model Results

5.2.1. Univac 1100/90 Performance

The curves shown in figure 9 a through c show that the single processor 1100/90 is insufficient to satisfy the required response time of three seconds for TIP transactions, even with a 20% reduction in TIP CPU service time. Similarly , two of the three required Deadline Batch loads, shown in figure 9-b and c, exceed the required response time of 25 minutes. Performance is still marginal at a reduction in Deadline Batch CPU service time of 20%.

In this model TIP CPU service time was the only parameter of high sensitivity, as could be expected of a single processor system. Reductions in I/O service times for all workloads did not significantly affect performance characteristics. The case run at 45% reduction in I/O service time is shown here alone for brevity.

When a second processor was added, the performance picture brightened considerably as shown in figure 10. This set of results reflects no reduction in Deadline Batch CPU time or in I/O service time. The plot of response time for the case of no reduction in TIP CPU time behaves well at the transaction rate requirement point for all loads.

Figure 11 shows a reduced case of 20% reduction in Deadline Batch CPU time and 45% reduction in all I/O service times. The sensitivity to these parameters is only noticeable in Deadline Batch response time, but is not large. The sensitivity to TIP CPU service time is much more evident in the case of TIP response time (figure 11a), where I/O service time sensitivity is low. By contrast, the sensitivity of Deadline Batch to I/O service time reductions is reasonably large as can be seen by comparing figures 10-b and c with figures 11-b and c.

These results established the two-processor Univac 1100/90 (2x2) as a candidate configuration for long term replacement of the baseline system.

5.2.2. IBM 3081-K Performance

The IBM 3081-k was chosen as an alternate configuration to study. The results that follow are for a two-processor system (2x1). The 3081 is packaged in processor pairs (dyadic processors), and configurations with odd numbers of processors are unrealizable. Therefore, the next largest configuration is a four processor system, driving costs past program constraints. Our study was therefore limited to the two processor system.

Figure 12 shows our results for the IBM 3081-K (2x1) with unreduced Deadline Batch CPU time and unreduced I/O service times. The results show a marginality in the TIP transaction response time (figure 12-a) and in Deadline Batch processing (figure 12-c).

Figure 13 show the results for the case of 40% reduction in Deadline Batch CPU service time and 45% reduction in all I/O service times. TIP transaction response time has been improved very slightly, but is still marginal. The improvement in Deadline Batch performance is the only marked difference, as comparison of figures 12-c and 13-c will show.

The overall conclusion drawn here is that the IBM 3081-K (2x1) configuration is only marginally adequate to satisfy project requirements, even at substantial improvements in the efficiency of DBMS direct I/O, Deadline Batch CPU service time and to a lesser extent TIP CPU service time.

5.2.3. Univac 1100/80 Model Results

For the interim time period between the present and the final installation of the selected configuration (either alternative), the performance of the 1100/80 (3x3) was studied. The initial replacement to the baseline Univac 1100/40 (3x1) was an 1100/80 (3x3) three processor system with three IOUs. The results in figure 14 show the familiar trend for TIP transaction response time - no sensitivity to I/O, great sensitivity to CPU service time/capacity.

Figure 14-a shows TIP response time for the case of 45% reduction in I/O service time, while figure 14-b shows the case for 90% reduction. In each case only a 20% reduction in TIP CPU service time can make the system marginally capable of handling the required 300 transactions/minute at peak loading. It is clear that the system will become CPU bound at this point, as model output shows a CPU utilization of greater than 90%.

The immediate solution to our short term performance problem was to add another CPU to the 1100/80 configuration. Transforming the 1100/80 (3x3) into a (4x4) solved TIP transaction response time problems, but did not remove the marginality of Deadline Batch performance until a 20% reduction in Deadline Batch CPU service time and a 45% reduction in I/O service time was made. Figure 15 shows the unreduced cases and figure 16 shows those with reduced service times.

5.2.4. Offload Processor Results

As the preceding section indicates, the addition of a fourth processor to the 1100/80 system did not completely solve the performance problems. The notion of adding yet another processor dedicated to Deadline Batch processing alone, thus "offloading" it from the 1100/80 (4x4) was studied. Since four processors is a maximum configuration for the 1100/80, the offload processor would be interfaced by means of a hyperchannel. Deadline Batch functions would be performed and a mass data base update would be transmitted to the 1100/80 (4x4) mainframe over the hyperchannel.

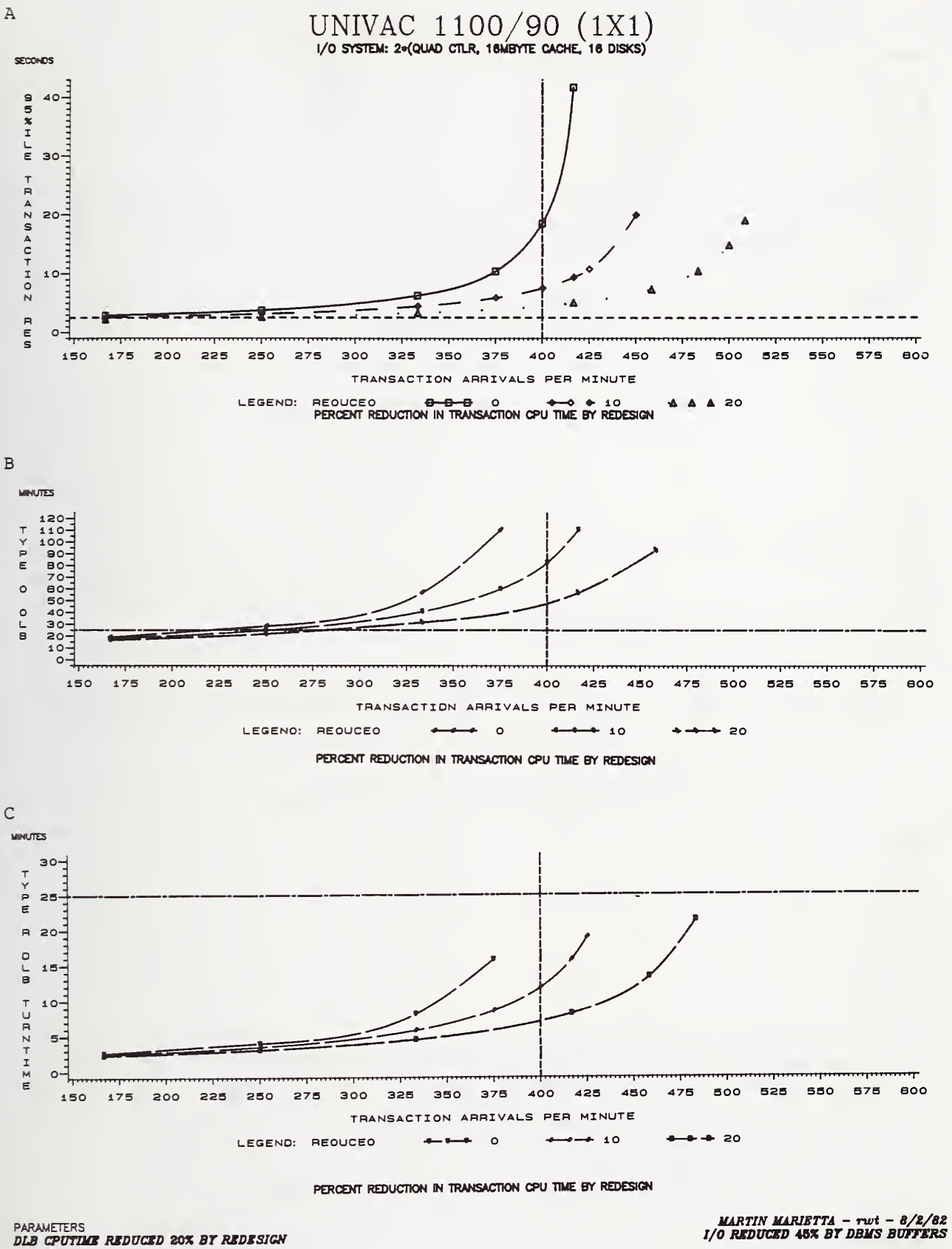


Figure 9. Univac Single Processor Response Time

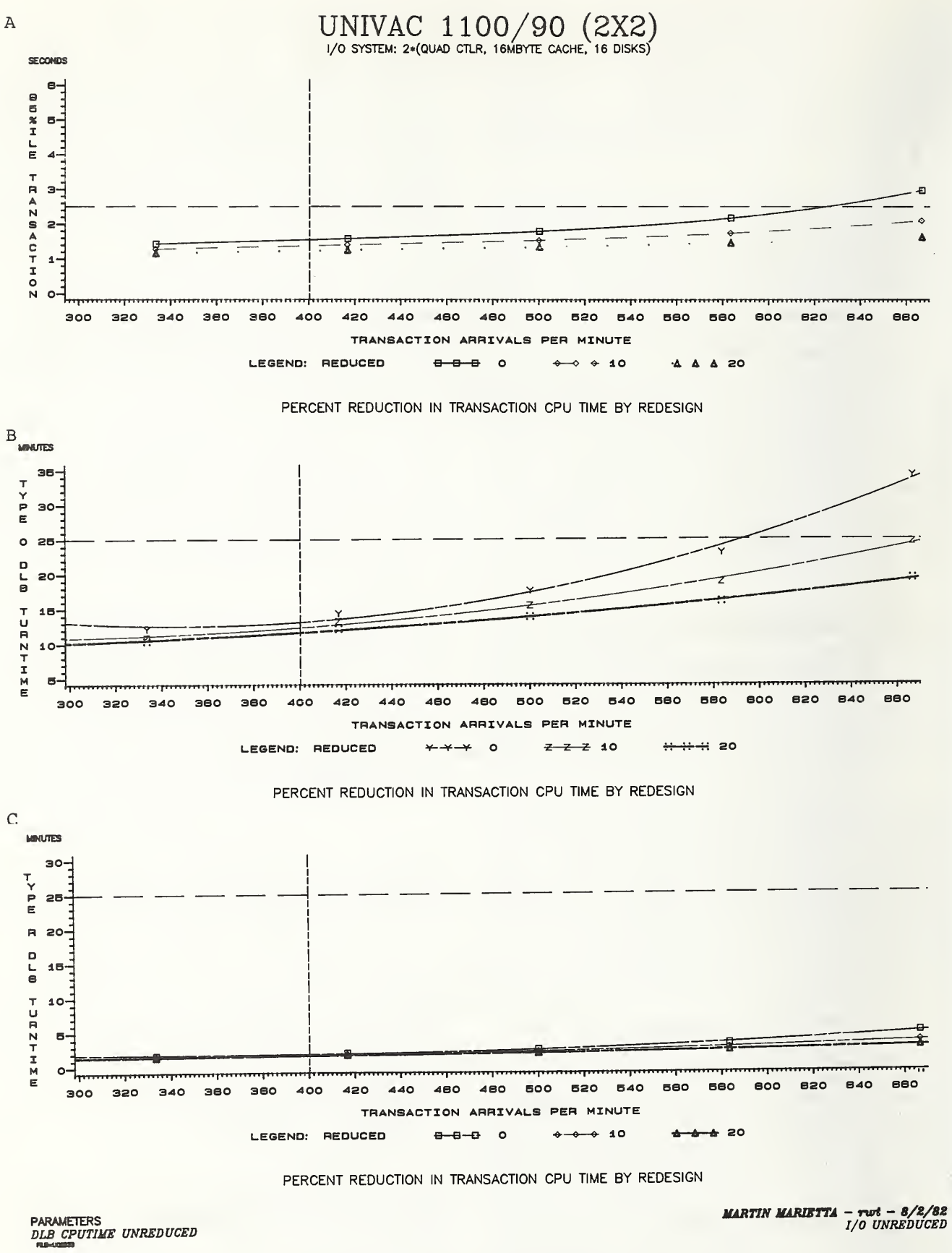
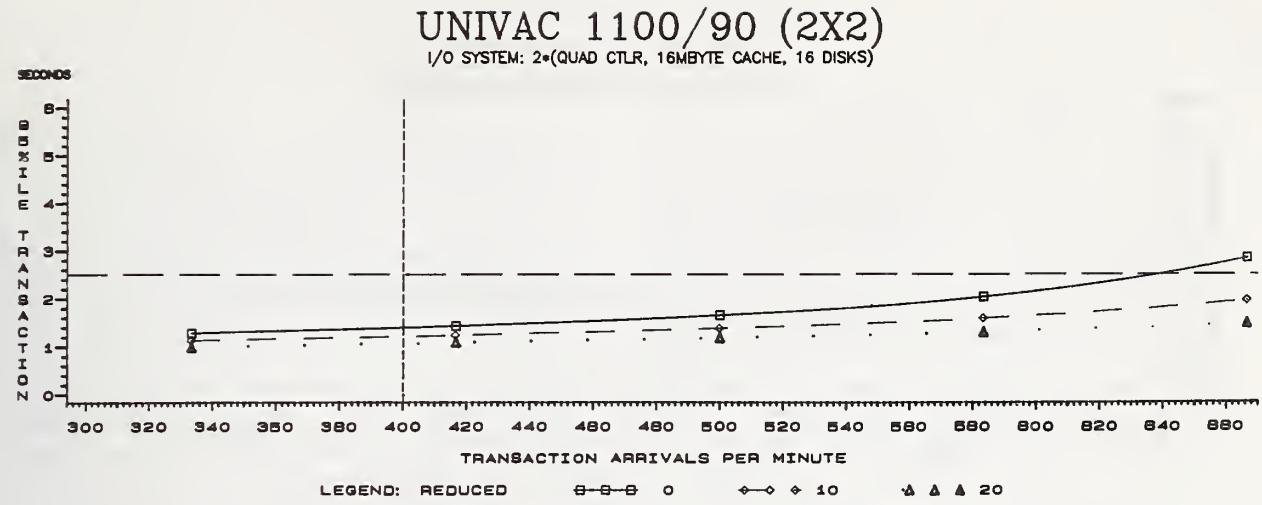


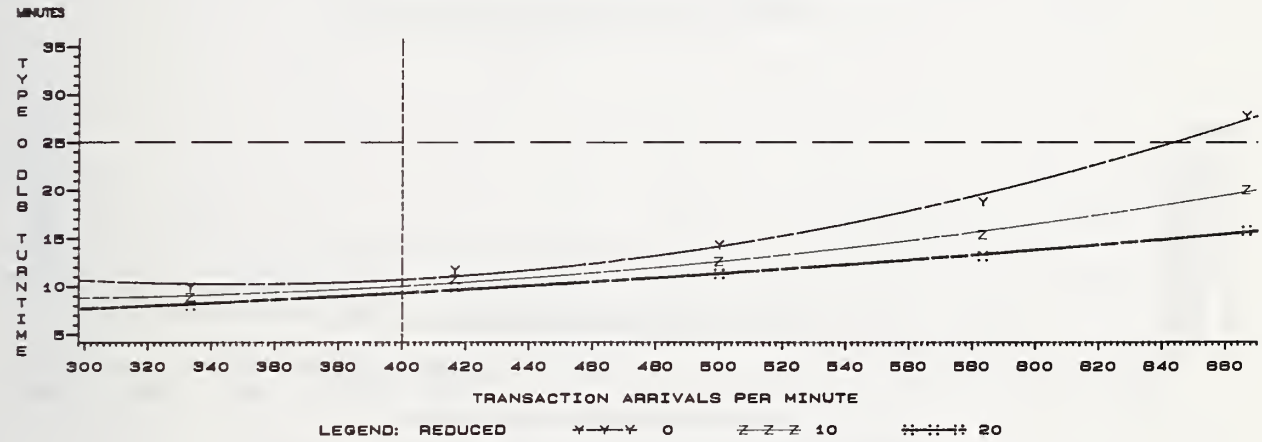
Figure 10. Univac Dual Processor Response Times

A



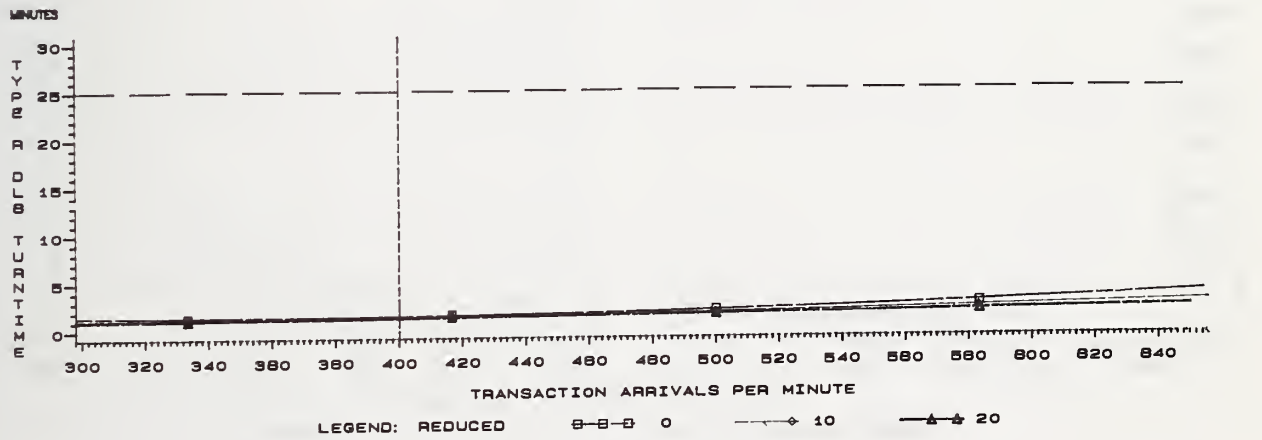
PERCENT REDUCTION IN TRANSACTION CPU TIME BY REDESIGN

B



PERCENT REDUCTION IN TRANSACTION CPU TIME BY REDESIGN

C



PERCENT REDUCTION IN TRANSACTION CPU TIME BY REDESIGN

PARAMETERS
DLB CPUTIME REDUCED 20% BY REDESIGN

MARTIN MARIETTA – rwt – 8/2/82
I/O REDUCED 45% BY DBMS BUFFERS

Figure 11. Univac Dual Processor Response Times (reduced cases)

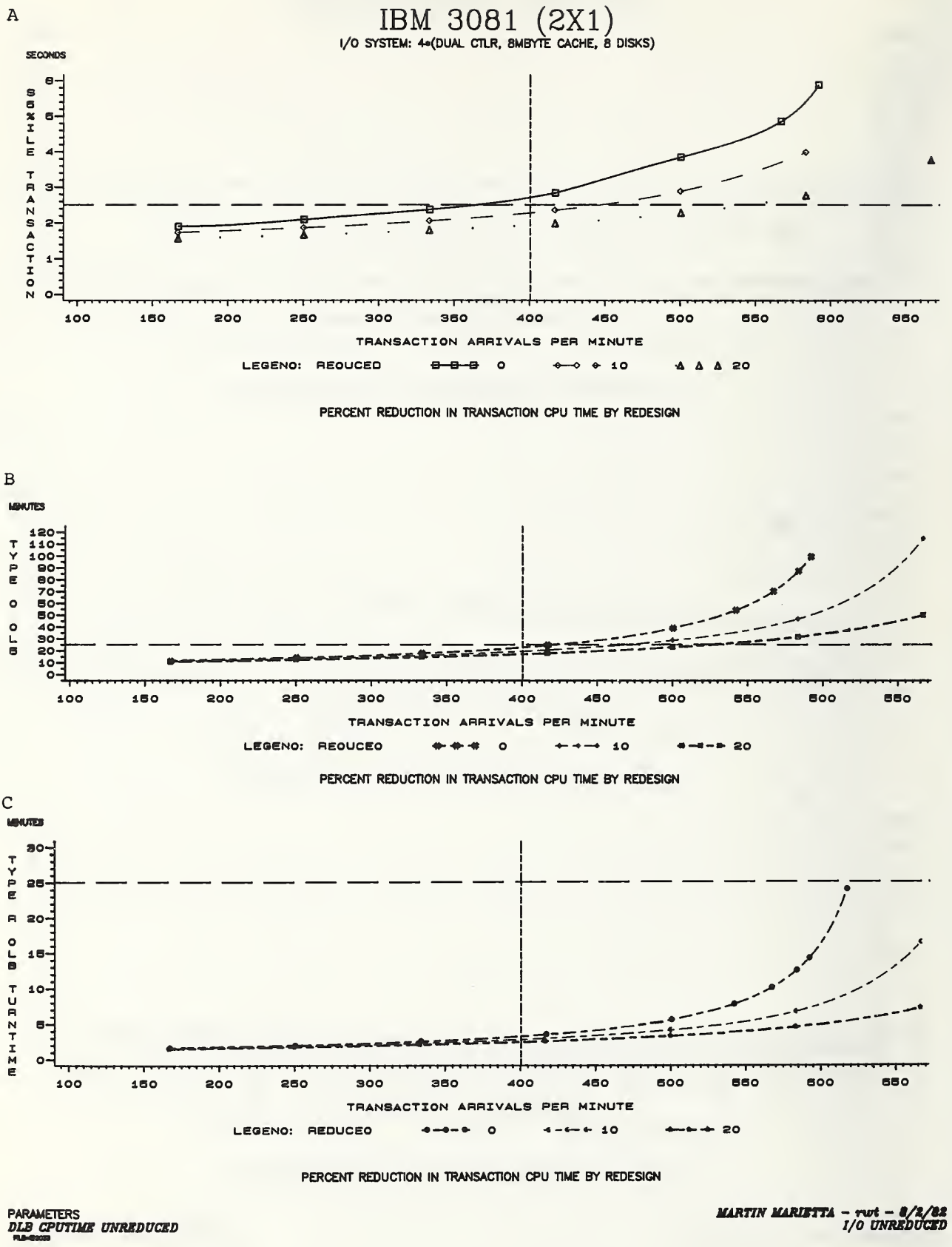


Figure 12. IBM 3081-K Dual Processor Response Times

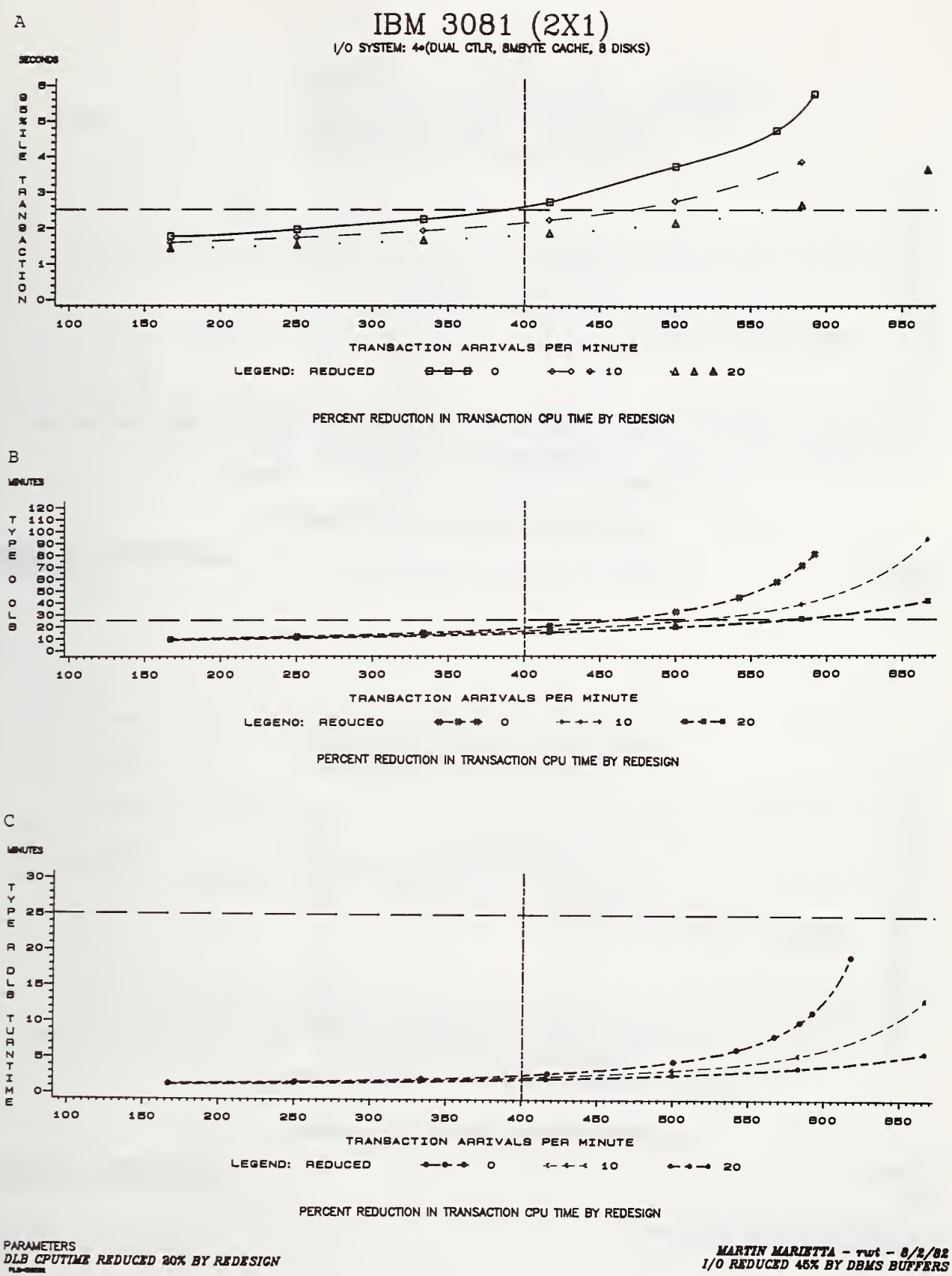
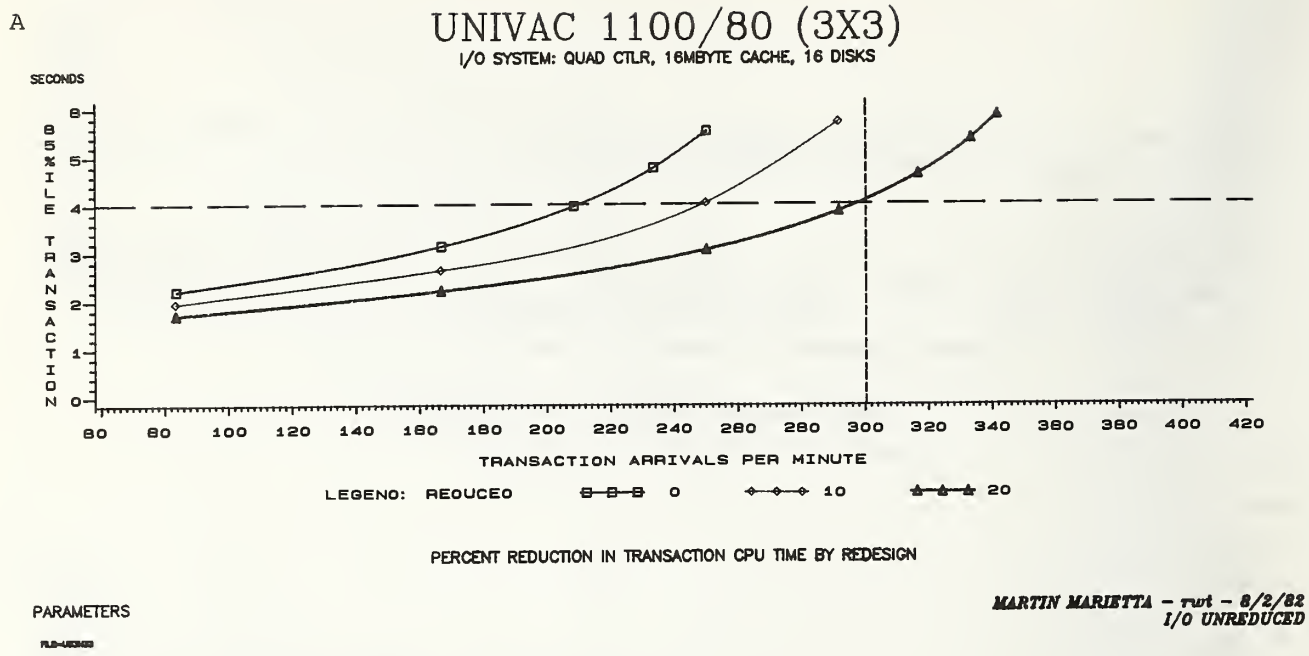


Figure 13. IBM Dual Processor Response Times (reduced cases)



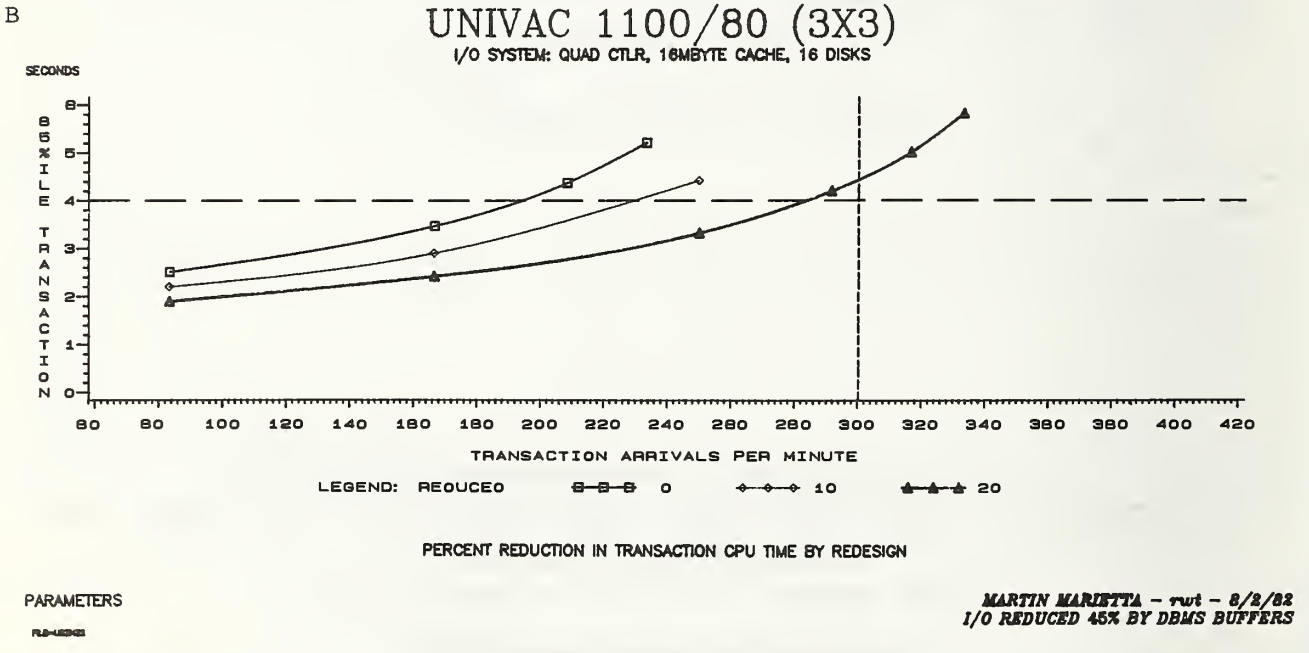
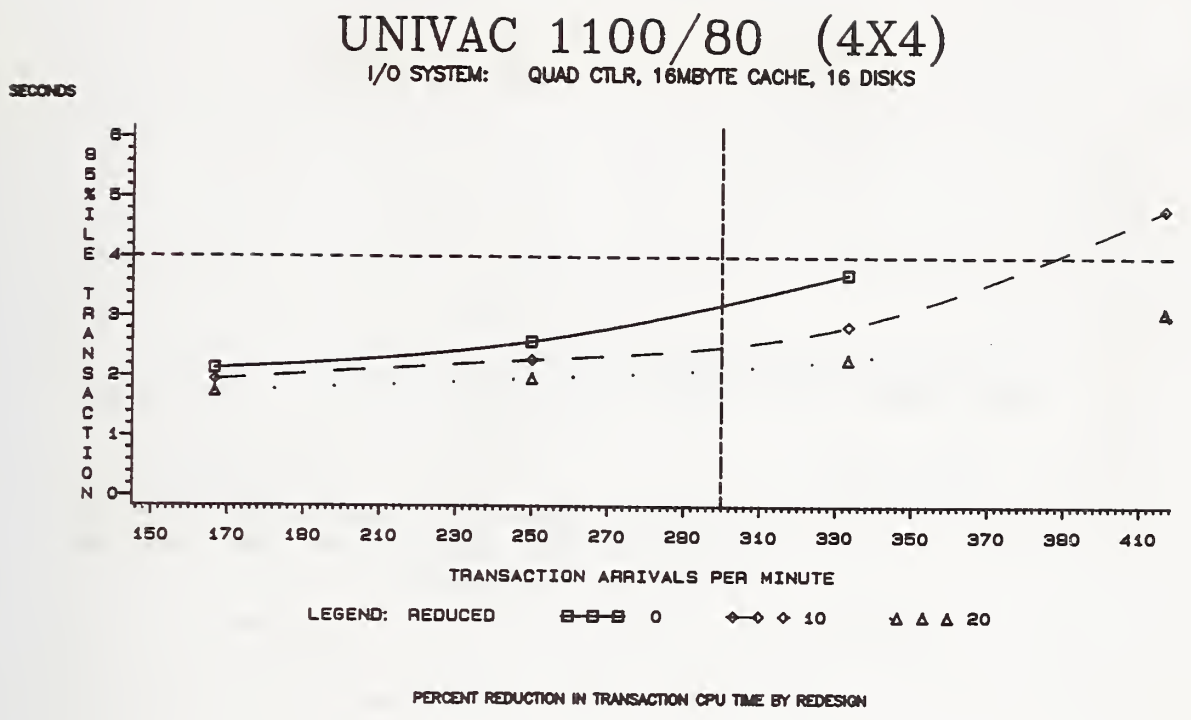
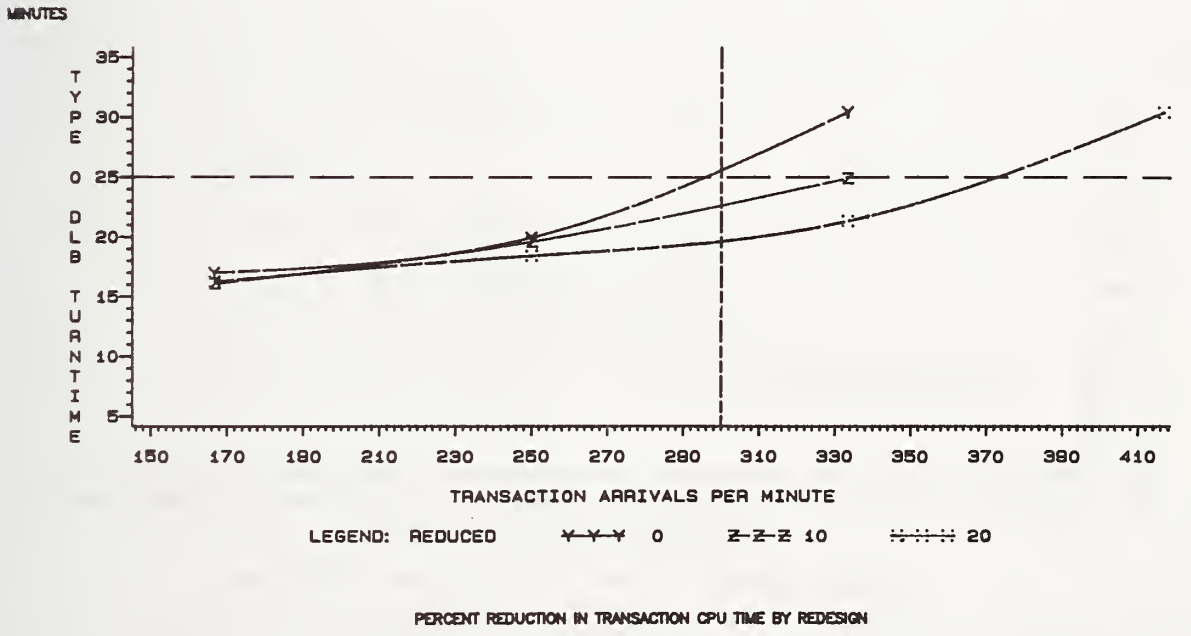


Figure 14. Univac 1100/80 Response Times (reduced cases)

A



B



PARAMETERS
DLB CPUTIME UNREDUCED

MARTIN MARIETTA – rwt – 8/2/82
I/O UNREDUCED

Figure 15. Univac 1100/80 Four Processor Response Times

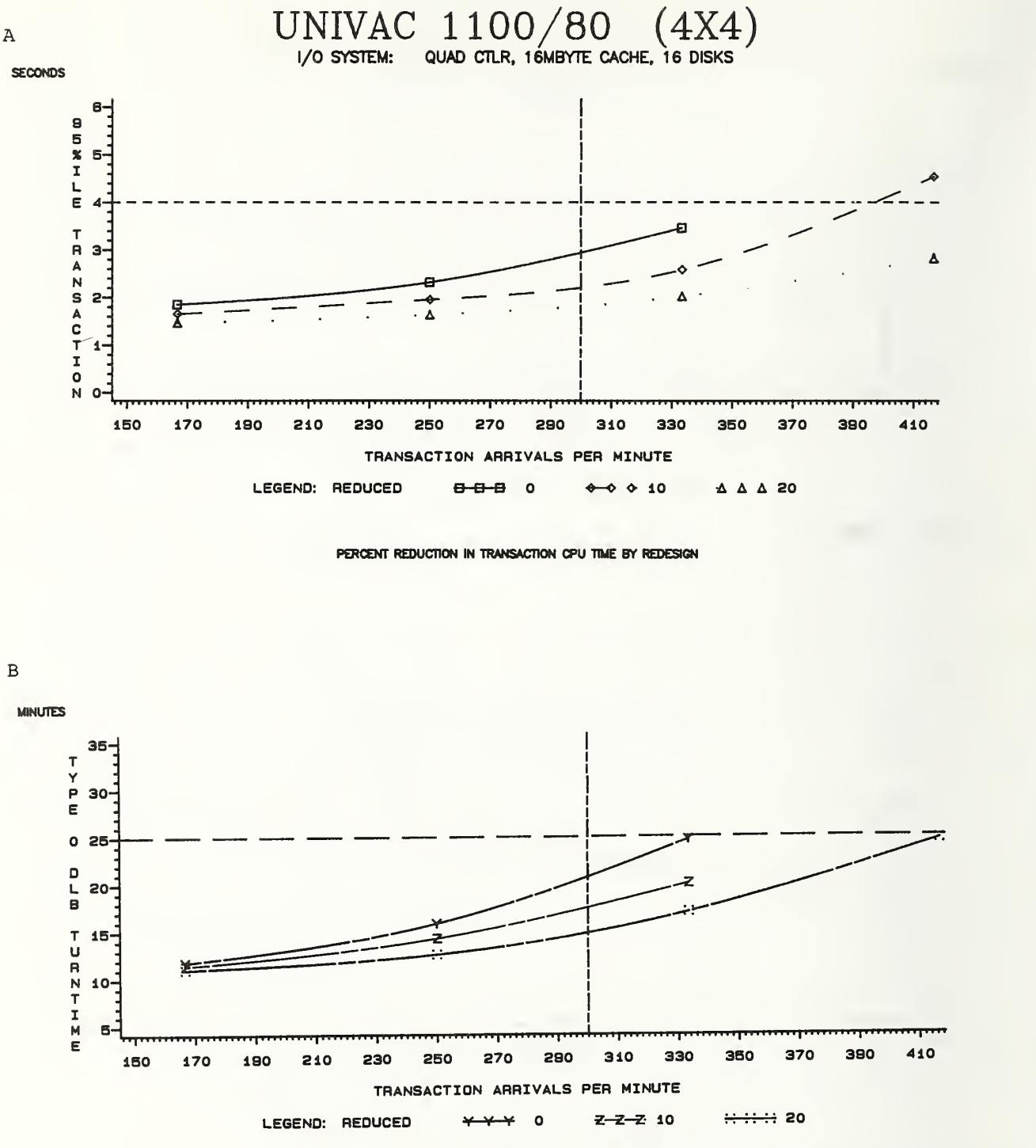


Figure 16. Univac 1100/80 Four Processor Response Times (reduced cases)

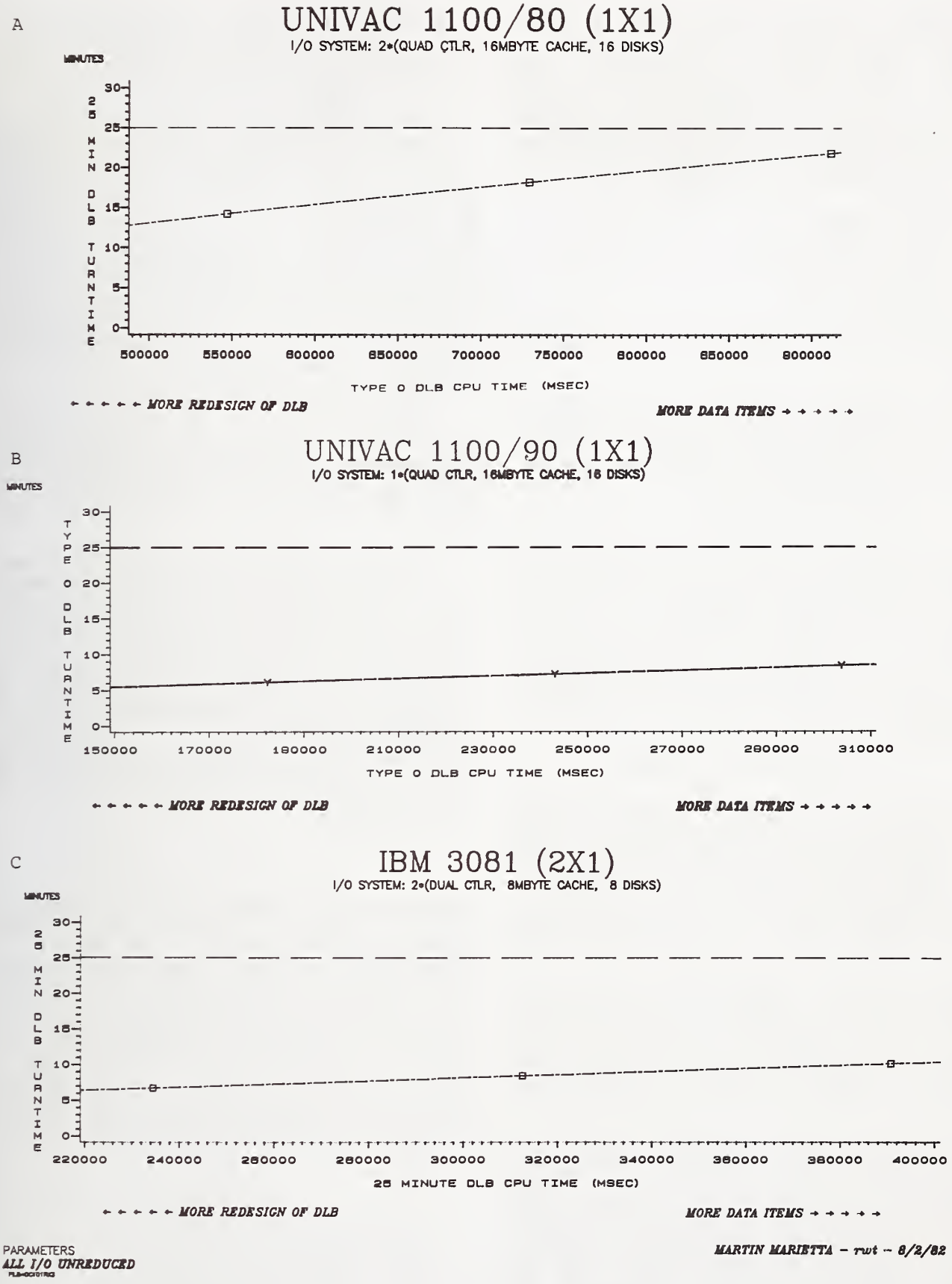


Figure 17. Offload Processor Response Times

Results of putting a single Deadline Batch load on a single 1100/80, and a dual IBM 3081-K, offload processor were so encouraging that increasing the amount of data processed by the Deadline batch function became feasible. Figure 17 shows graphs of Deadline Batch response time versus Deadline Batch CPU service time, a measure of the number of items processed. Each of the three candidate offload processors:

(1) Univac 1100/80 (1x1);

(2) Univac 1100/90 (1x1);

(3) IBM 3081-K (2x1)

are graphed in figures 17-a, b, and c respectively.

The inclusion of the single processor 1100/90 was realistic since this single CPU configuration was announced to be available within the short term time frame, although the multiple processor configurations would not be available until our long term time frame.

Deadline Batch CPU time was selected as a rough measure of the amount of data being processed by the Deadline Batch jobs. The center point on each graph represents the required amount of data to be processed per job. The point to the left represents a 25% reduction in this amount, while the point to the right represents a 25% increase. Alternatively, these points can also be interpreted as increased or decreased efficiency of the Deadline Batch function.

6. Conclusions and Epilogue

The foregoing model results led us to conclude that the Univac 1100/90 (2x2) was the most cost effective option available to satisfy program requirements. To provide an alternate vendor, the IBM 3081-K (2x1) was proposed as a second choice, with the caveat that performance could be marginal without substantial increases in code efficiency.

For the short term performance of the 1100/80 system, we concluded that an offload processor was necessary unless requirements for the Deadline Batch response time were relaxed. Our efforts to measure 1100/80 performance are continuing in order to improve the accuracy and level of detail of our short term predictions.

The choice of modeling and simulation tools produced an interesting comparison between BEST/1 and SLAM, and furthermore, between analytic and simulation methodologies. Our conclusions based upon this experience are that both approaches and tools are capable of providing adequate accuracy and flexibility for the sizing and prediction of computer system performance. Addressing the questions of adequacy and accuracy of BEST/1 and in particular queuing theory, our experience generally corroborates the widely publicized expectation that queuing theory approximations can provide as good or better accuracy as simulation modeling with significant advantages in computational efficiency and model turnaround.

The performance of the model tools BEST/1 and SLAM can be compared in this case study since we built the GPM in SLAM at approximately the same level of detail as BEST/1. While no detailed record of computer resource usage by the tools was kept, we observed a rough order of magnitude in the CPU time required for equivalent runs in BEST/1 and SLAM. It must be kept in mind, however, that BEST/1 was running on an IBM 3081 whereas SLAM was running on a VAX 11/780. Assuming that the IBM 3081 is roughly five times as fast as VAX 11/780, (a figure supported by MIPS ratings of the two machines), BEST/1 exhibited a 2:1 speed advantage over SLAM.

The implementation of the GPM in SLAM took approximately four man-months, slightly longer than expected. The developement time required for SLAM simulations was a primary factor in our choice of BEST/1 as a tool for preliminary models.

Finally, it must be stressed that an issue of much greater importance than which methodology to use is the proper interpretation and preparation of the model input data. Collecting the necessary input data and putting it in a form understandable to the models was a significant undertaking. The software monitors available on the Univac 1100 series and their corresponding data reduction software were not designed with modeling in mind. It has been said many times that a model is only as good as its input and the truth of this maxim has certainly been borne out by our experience.

The staff of the Central Software Engineering Facility (CSEF) Model Shop built the GPM in SLAM according to our requirements. The project and subcontractor personnel without whose support this paper would not have been possible include, but are not limited to: Scott Gilles, Jim Elliot and the Martin Marietta Data Systems group, Ron Jacobsen, Gary Sandler, for technical support; Pam Clark and Kevin Goodrich for keeping me awake; and Lyle McElhaney for single-handedly debugging the UNIX (UNIX is a trademark of Bell Laboratories) formatter macros which produced this paper.

References

1 BGS Systems Inc.,
BEST/1 Users Guide, BGS Systems, 1979.

2 Pritsker, A. A. B. and Pegden, C. D.,
Introduction to Simulation and SLAM, Halsted Press, N. Y. 1979.

3 BGS Systems Inc.,
Technical Note #10, BGS Systems, 1981.

4 BGS Systems Inc.,
Technical Note #15, BGS Systems, 1981.

A UNIVAC WORKLOAD CHARACTERIZATION SYSTEM

Walter N. Bays
Dawn L. Voegeli

MITRE
1812 Space Park Drive
Houston, TX 77058

Analytic modeling, simulation, and benchmarking are major tools for computer performance evaluation. All require an accurate characterization of the system workload. A statistical analysis package has proven useful for workload characterization and other analyses of system accounting. This paper describes the use of P-STAT for workload characterization on UNIVAC systems.

Key words: Capacity planning; job accounting; resource management; statistical analysis; workload characterization

1. Introduction

The Workload Characterization System (WCS) was developed for NASA's Johnson Space Center (JSC) by MITRE. The Central Computing Facility (CCF) provides a full range of computational services to the engineering, scientific, flight operations, and program management organizations at JSC. The CCF is composed of two UNIVAC 1100/81 unit processors (1181-0, and 1181-5), one 1100/82 multiprocessor (1182-2), and one 1110 multiprocessor (1110-6).

The CCF processes a varied workload from a large number of user groups. In most cases these groups are responsible for development and maintenance of their own application programs. In this environment it is important to determine which programs consume which resources, and to estimate how resource consumption might change with changing requirements of the various user groups [5,6,7,8].

This paper describes the capabilities of WCS, and some of its uses. Section 2 describes the structure of WCS. Section 3 discusses the use of WCS, specifically its use in selecting candidate programs for application tuning. Appendix A contains sample standard and ad hoc reports. They appear for purposes of illustration only and are not representative of the CCF workload.

1.1 UNIVAC Accounting

A UNIVAC run (job) comprises the execution of some number of programs interspersed with job control statements. Runs are either batch or demand (interactive). The UNIVAC 1100 operating system (EXEC) collects extensive data on each run and program, and on general EXEC actions, for use in resource accounting, charge-back, error analysis, et cetera [10]. Accounting is based on the Standard Unit of Processing (SUP), which is a measure of the serial processing time of a program. That is, the SUP time of a program is approximately the length of time that program would require to complete processing if computation and I/O were done serially with no overlap and if no other programs were active in the system [4].

Total SUPs are the sum of CPU SUPs, I/O SUPs, and CC/ER SUPs. CPU SUPs are the total CPU processing time. The SUP charge for a single I/O operation is the number of words transferred divided by the transfer rate of the I/O device used plus the average rotational latency time of the device. I/O SUPs are the sum over all I/O operations. CC/ER SUPs are an estimated time to process control statements and executive requests. CBSUPs, or core-block SUPs, are not a component of total SUPs. Instead they are an accumulation over time of the current memory size of the program times the SUPs consumed. CBSUPs divided by SUPs is the average memory size expressed as the number of 512 word core-blocks.

WCS reads the UNIVAC accounting records. Accounting records are generated on run initiation, run termination, program initiation, program termination, at time of file assigns and frees, and at system recovery after crash. Many other records are generated which are not used by WCS. Two condensed files are generated by WCS containing variables listed in Section 2.5.2.

2. WCS Structure

2.1 Overview

The purpose of WCS is to give insight into the workload being executed on CCF computer systems, and to produce a workload model suitable for use in a benchmark or other type of system model [1,3]. Other potential applications include capacity planning, identification of candidate programs for application tuning, and ad hoc studies. For these purposes, a representative sample of accounting data is required. WCS could also be used for resource accounting by defining additional reports.

WCS comprises a number of FORTRAN and P-STAT routines for extraction of workload data and analysis of that data (see Figure 1). P-STAT is a proprietary program for data manipulation, statistical analysis, and report generation [2]. It executes on a wide variety of computer systems, including UNIVAC. WCS provides a set of standard reports. It also provides the capability to retrieve and process accounting information on an ad hoc basis. This capability is called Interactive Workload Query.

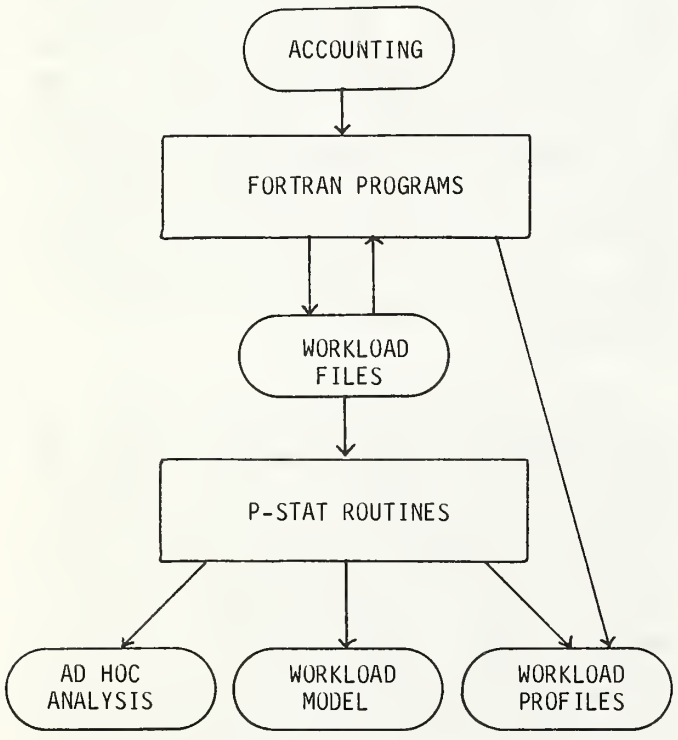


Figure 1. WCS Overview

The WCS FORTRAN programs are CONDENSE, PSORT, and PROGDETAIL. CONDENSE reads the accounting file and creates two condensed workload files: one for run records, and one for program records. These workload files may be efficiently sorted and merged with PSORT, an interface to the UNIVAC sort package. These workload files are then loaded into P-STAT for analysis and reporting.

2.2 Standard Reports

The WCS reports are produced as required for the purpose of generating a workload model, and to document the current workload characteristics. Six reports are currently defined in WCS. Each report is generated by an ADD element (job control language procedure) which may be used directly, or may be combined and edited by the user. The following Sections describe these reports and their generation. Sample reports are included in Appendix A.

2.2.1 Program Profile Report

The Program Profile Report, shown in Figure A-1, lists by program name, the total, average, and standard deviation of several important program characteristics. It shows the relative usage of system processors and major applications. The manner in which a program is used can be tracked, for instance, to detect growth in average memory size. Candidate programs for application tuning can be selected based on relative usage, and compared with known characteristics from previous tuning tasks.

As explained in Section 2.4, a "name label" file defines the significant programs to be included in the Program Profile Report. All other (miscellaneous) programs are reported together. Each program listed in the name label file has an entry in this report for batch mode executions, and an entry for demand mode executions. Each entry has a line for the total of all program executions, a line for the average of each execution, and a line for the standard deviation.

2.2.2 Program Detail Report

The Program Detail Report, shown in Figure A-2, lists the relative resource consumption of every program executed. It is used to identify new applications which use significant resources. Each program is listed with its part of the total number of executions, its part of the total SUPs, and its part of the total CBSUPs. This listing is repeated sorted by each of these four columns.

2.2.3 Workload Model Report

The Workload Model Report, shown in Figure A-3, is used in conjunction with the Program Profile Report to develop a model of the CCF workload for benchmarking, and analytic or simulation modeling. It shows the

characteristics of miscellaneous programs by dividing them into a number of similar categories.

This report divides miscellaneous batch programs by size and the ratio of I/O SUPs to total SUPs (IOPART). Programs from system runs are listed separtately. There is a listing for each size range 0-10K, 10-20K, 20-40K, and over 40K, and for each IOPART range 0-.2, .2-.4, .4-.6, .6-.8, and .8-1.0.

Miscellaneous demand programs are divided by size and SUPs. Programs which have greater voluntary delay time than elapsed time have multiple parallel activities and are listed separately. The same size ranges are listed for demand as for batch programs. The SUP ranges in seconds are 0-2, 2-4, 4-8, 8-16, and over 16.

2.2.4 Frequency Profile Report

The Frequency Profile Report, shown in Figure A-4, gives frequency distributions of size, SUPs, ratio of I/O SUPs to total SUPs, and elapsed time according to categories of all programs, batch programs, and demand programs. Ten equal categories below the mean, and ten equal categories above the mean are listed with the number of programs, and the percent and the cumulative percent of programs. The mean, variance, and standard deviation are listed; but the mean is unweighted (by SUPs).

2.2.5 Account Profile Report

The Account Profile Report, shown in Figure A-5, gives the total and average of several important workload measures by account (NASA branch) and operations shift for batch and demand programs. It shows the workload characteristics by user groups, and the quality of service each group is receiving.

2.2.6 Terminal Profile Report

The Terminal Profile Report, shown in Figure A-6, gives the averages of several important interactive job characteristics by terminal. It shows the relative utilization of terminals, the characteristics of the work run from each terminal, and the response time received.

Certain errors are unavoidable in this report. The number of transactions of a demand run, the number of discrete user requests for service, is not measured by the EXEC. "Cards in", which is measured, is only an approximate measure of the number of transactions. Voluntary delay may be reported greater than elapsed time for multiple activity programs. Since response time is calculated as elapsed time minus voluntary delay time, divided by cards in, response time for runs with multiple activity programs may be calculated as negative. These runs are ignored in the terminal averages.

2.3 CCF Conventions

Some of the WCS variables are available because of CCF standards. These variables are mailbox number (BOX), project number (PROJ), and badge number (BADGE). If the standards are not adhered to by the user, the corresponding variables are zero.

UNIVAC run IDs are up to six characters long. In the CCF, the first three are usually a mailbox number to aid in distribution of print output. The last three characters are arbitrary. The mailbox number implies the organization (NASA or contractor) and the physical location of the person submitting the run. The run IDs of a number of special system runs, such as the ROLBAK file management run and the TALONS tape labeling system run, are recognized by WCS and given specific four digit codes.

In the CCF, the four digit JSC project numbers are used for UNIVAC user IDs. Special system project names are given specific five digit codes by WCS.

UNIVAC file qualifiers (logon directories) are twelve characters. In the CCF a qualifier is the NASA branch mail code, followed by a dash, and the user's badge number.

2.4 Label Files

The major deficiency of P-STAT for WCS is that it cannot directly handle character data. This deficiency was overcome by using P-STAT's capability to print character labels for given numeric values in reports. P-STAT uses a label file to define character values for particular values for a variable. The label files are read both by CONDENSE for encoding character values as numeric values, and by P-STAT for decoding them.

For example, suppose the possible values of the variable NAME are 0 through 61. These may be printed as "MISC", "ABS", "ABSOL", etc. using the "name label" file:

```
NAME
    (  0)   MISC
    (  1)   ABS
    (  2)   ABSOL
       ...
    ( 61)   XAXIS
```

The "name label" file defines the applications which are reported separately. All other programs are reported with the name MISC. One of the most difficult decisions to make in workload characterization is which programs are "significant" applications requiring separate consideration by name, and which may be considered "miscellaneous". This decision is made using the Program Detail Report based on the resources consumed by the application, and is reflected in the contents of the name label file.

For some uses of WCS, only system processors and major applications will be listed separately. For other uses of WCS, a separate listing may be desired for each distinct program name. A label file defining each program name may be generated in conjunction with the Program Detail Report.

The condensed workload file contains both the numeric encoding of the program name, and the twelve character (ASCII) name itself. The PROGDETAIL program uses the character name to report on each distinct program executed. PSORT can then be used to relabel the condensed file according to the new label file.

2.5 Interactive Query

2.5.1 General

Interactive workload query is the execution at a demand terminal of commands to retrieve information from a workload file, perform statistical analyses on it, and report the results. It gives the analyst the capability to conduct ad hoc investigation into the use and behavior of the computer system. It is also used to formulate and test theories on the workload, to find problems with the system, and to develop new WCS reports.

It is not possible to anticipate every question that will be of interest and provide a report to answer it. Indeed, the answer to one question will likely suggest another, and the analyst will proceed by "interacting" with WCS. Some of the standard reports may be run first to assist in the formulation of additional queries.

2.5.2 WCS Variables

The following variables are in both the program and run records. All these variables are real numbers, though some may take only integral values. All times and SUP times are given in seconds.

ACCT — Code for account (NASA division) encoded using the "account label" file.

BOX — JSC mailbox number.

CAT — Number of assigns and frees of catalogued files. For runs this is the number of job control language assigns and frees. For programs this is the number of assigns and frees from within the program.

DATEO — Date run or program started, in Julian (YYDDD) notation.

DELAY — Voluntary delay time, think time and requested wait time.

ERR — Execution error code. This value is zero if there was no error.

ET — Elapsed time of run or program.

KSEC — Core K-seconds, the product of SIZE and SUP.

RUNSEQ — Run sequence number, used to link run and program records together.

RUNTYP — 1 = batch, 2 = interactive

SIZE — Average memory size over the execution of the run or program in 1000 words.

SUP — Total SUPs (seconds).

TEMP — Number of assigns and frees of temporary files.

TIMEO — Time run or program started in hours, minutes, seconds (HHMMSS) notation.

TIME1 — Time run or program stopped in HHMMSS.

The following variables are in the run records only.

BADGE — JSC badge number of the person executing the run.

BAKLOG — The time a batch run spent in the backlog before beginning execution.

CRDIN — Number of cards (records) input to run.

DATE1 — Date run ended (YYDDD).

DEVICE — Code for the JSC terminal ID of an interactive run.

ESTSUP — User's estimated SUP time of the run.

NPROG — Number of programs executed by the run.

PAGES — Number of pages of output generated by run.

PRI — Priority of run.

PROJ — JSC project number.

TRKSEC — The product of the average number of tracks of mass storage assigned to the run as temporary files and as expansion of catalogued files, times SUP.

The following variables are in the program records only.

CCER — CC/ER SUPs (seconds).

CPU — CPU SUPs (seconds).

IO — I/O SUPs (seconds).

IOPART — The part of total SUPs due to I/O. Calculated as IO/SUP.

NAME The code for the program name, encoded using the "name label" file.

SHIFT Operations shift during which the program started.

TMRT Time spent in real-time mode.

VER Code for the program version, encoded using the "version label" file.

3. Use for Application Tuning

The requirements for workload characterization depend, of course, on the goal of the characterization. Some typical goals are analytic modeling, benchmarks, capacity planning, and system performance evaluation. This Section discusses a less common goal: the characterization of the workload to aid in choosing candidates for application software tuning.

3.1 General

Application software tuning is the process of improving the efficiency of major resource-consuming application programs [9]. Objectives of tuning are determined by those resources. The most common objectives are to reduce memory utilization, SUPs, and improve quality of service. The following examples demonstrate how to use WCS to choose tuning candidates for those objectives.

The use of WCS is the first step in identifying a candidate program to be tuned. These candidates should be reviewed by an analyst in order to estimate the viability of tuning the programs. Tuning should only be undertaken when a significant reduction in consumption of resources or improvement in quality of service is expected. After a number of candidates have been identified with WCS, the analyst then proceeds, through user contact, code inspection, and experimentation, to choose one or more candidates and evaluate the feasibility of tuning those candidates.

Other factors, which cannot be quantified, should be considered along with the expected benefits to decide if the program should be tuned. Some of these factors are information from the user regarding the application and its use, quality of service the program receives, the time of day the program is executed, the organizational priority of the user group (if applicable) and the type of run (demand or batch).

3.2 SUP Utilization

SUPs are a measure of system utilization. At the CCF, resource accounting is based on total SUPs, so there is no differentiation between CPU, I/O, and CC/ER SUPs. Therefore, the benefit of reducing SUP utilization is quantifiable and the method of choosing tuning candidates is more straightforward.

Reducing the SUP consumption of a program makes those SUPs available for other programs. SUPs are equated to a dollar value. A program is worth tuning if the expected dollar gain of SUPs saved is greater than the cost of the analyst to tune it over the life cycle of the application. This implies a program is only worth tuning if it uses at least a certain percent of the total machine SUPs.

3.2.1 Methods

The Program Detail Report lists all programs in order of SUP utilization. Programs that have already been tuned, system processors, and other proprietary programs are eliminated from consideration. The name '@@@@@@@@@@@@' is given by the EXEC to any program not explicitly named by the user. It should not be considered for tuning because it represents many applications and involves development work.

It is possible that different user groups run entirely different programs with the same name. Also, different "versions" of the same program may have different names. These are identified with Interactive Workload Query along with user contacts.

The availability of the Program Profile Report enables the list to be reduced based on the number of times a program has executed, the size of the program, and the service factor (see Section 3.4) of the program for both demand and batch runs.

Users of programs are identified using Interactive Workload Query, and can be contacted for detailed information on the program. Based on this information, the code may then be inspected.

3.2.2 Example

The Program Detail Report in Figure A-2 lists programs in order of decreasing SUP utilization. Using the methods discussed above, there are 28 potential SUP tuning candidates in this example. The excerpted Program Profile in Figure A-1 supplies additional information on potential candidates. The Interactive Workload Query is then used to investigate these candidates.

Each potential candidate should be investigated beginning with the program having the greatest part of total SUPs. Potential candidates are summarized by ACCT, BADGE, and BOX to identify users of the programs. Some of these candidates are shown in Figure A-7. Several programs named CHAIN are observed. There is a strong indication these programs are different versions of the same program. Each version has the same users (BADGE), same project number (PROJ), and similar characteristics. Figure A-8

lists the resources used for all versions of CHAIN. The assumption that CHAIN is a single program should be verified when the user is contacted.

3.3 Memory Size

Memory becomes a scarce resource as more runs enter a system, particularly if those runs are large and require substantial processing time. More swapping occurs and core wait times increase as these runs compete for memory. The result is more overhead and less efficient systems performance.

If application tuning reduces the size of a large program with high elapsed time and high SUPs, an improvement in overall system performance may be expected.

If resource accounting were done using CBSUPs, they would determine the feasibility of the tuning task. The CCF does not charge for CBSUPs so the benefit of reducing CBSUPs is not quantifiable and the method relies more on judgement.

3.3.1 Methods

The size at which a program is considered a candidate for tuning is workload dependent. The Frequency Profile can be used to aid the analyst in determining this "cutoff size". In this example, the largest 1% of programs executed are considered.

Of those programs being considered, candidates should be chosen that, when executed, run long enough to significantly affect the system. SUPs and elapsed time are primarily considered in this regard. Another measure of interest may be the time of day that the program was executed. For instance, if it usually executed during prime shift it would have a greater impact on the system than if it were executed during non-prime shift.

By these methods one or more tuning candidates are chosen. After a preliminary analysis of these programs, similar to that done for SUP utilization, one or more are picked to be tuned, according to the available tuning resources.

Using tuning techniques such as those described in [9], program size may be decreased. For example, dynamic arrays may be used instead of static arrays, allowing the program to grow and shrink in size as needed.

3.3.2 Example

The Frequency Profile of program size in Figure A-4 aids the analyst in picking the cutoff size for the desired percentile. Frequency profiles are also produced for SUP, IOPART, and ET. For this example, the cutoff size chosen is 66K, to correspond to the top 1% of executions. Eighty two program executions are above this size for this example. Each potential candidate that exceeds the cutoff size should be investigated.

The Program Detail Report in Figure A-2 lists programs in order of decreasing CBSUPs. Using the feasibility criteria described above, there are 18 potential memory tuning candidates in this example.

3.4 Quality of Service

Quality of service is usually measured by response time and turnaround time. Response time of a transaction is "the elapsed time from the last keystroke of an operator input at an interactive remote device until the first printable character of the resulting [system] response appears at the user's device". Turnaround time is "the time interval between the initiation of a user workload demand and the successful completion of that demand" [11]. From UNIVAC accounting data it is possible to measure the turnaround time of runs and of program executions (elapsed time). The average response time over a run can be estimated, but the response time of a program execution cannot.

Another important quality of service measure for a program execution is the service factor: the ratio of elapsed time to SUPs. Theoretically, this number is the "stretch-out" factor of the turnaround time to what the program would have experienced on a "single string" machine. For demand programs, the service factor may be adjusted by subtracting voluntary delay from elapsed time before dividing by SUPs.

Better turnaround time will enable the user to use his time more efficiently. In particular, with demand programs, the time spent waiting for a response at a terminal should be minimized. With prime shift batch programs the time spent waiting for completion of a program should be minimized.

The benefit for reducing turnaround time is not quantifiable, but is reflected in user satisfaction.

3.4.1 Methods

The Program Profile Report can be used to identify programs with a high service factor. If the voluntary delay associated with the program is low then that program is a candidate for further inspection. Other measures of interest are the time of day the program is executed, number of file assignments and frees, and number of I/O accesses. Some possible reasons for high service factor are large memory size, inefficient segmentation or banking, file conflicts, and tape mount delays. Development executions are not considered as candidates because there is not yet a final program to tune. It is important to pick service goals when choosing candidates. For the purpose of this presentation a service goal of an

average of 10 seconds for response time over a run is used.

If the Terminal Profile report indicates long response time for a given terminal, considering average memory size, the programs executed from that terminal may be examined. This may result in identification of a candidate program for application tuning to reduce response time.

3.4.2 Example

Eight terminals receiving response time over 10 seconds, despite being small runs, were identified in Figure A-6. Figure A-9 lists the runs on these 8 terminals with the SUPs per transaction calculated. Run 142 is identified as a multi-activity run because of the negative response time. Run 418 has an acceptable service factor. Run 375 generates a high number of pages. Runs 103, 107, and 219 meet the goal response time. Run 238 is a large run. Figure A-10 lists additional variables for these runs. Runs 167, 219, and 370 are development work. All other runs meet goal response times except runs 272 and 372 . Figure A-11 lists the programs executed by runs 272 and 372. A file conflict could exist.

This work was supported by NASA Johnson Space Center under contract T-1612J.

References

[1] Bays, W., and Kincy, W., Universal Skeleton for Benchmarking Batch and Interactive Workloads: UNISKEL BM, Proceedings of the Computer Performance Evaluation Users Group 17th Meeting, November 1981.

[2] Buhler, S., and Buhler, R., P-STAT 78 User's Manual, P-STAT Inc., February 1981.

[3] Fiske, R., Kincy, W., and Brazile, R., EXEC-8 Performance Evaluation System, MITRE, MTR-4575 Revision 1, June 1975.

[4] Kelly, J., UNIVAC 1100 Performance, Datametrics Systems Corporation, 1980.

[5] Linde, S., and Preston, S., CPE's New Panacea: The SAS-Based PMS, Proceedings of the 7th Annual SAS Users Group International Conference, February 1982.

[6] Morris, J., A Performance Management System Using SAS, Proceedings of the Computer Measurement Group X International Conference, December 1979.

[7] Preston, S., and Erb, D., Implementation of System and Resource Management Analysis of IBM/MVS, The Fourth Annual International Conference on Computer Capacity Management, April 1982.

[8] Rainbolt, M., On the Generation of a Demand and Batch Workload Model, MITRE, MTR-4561, August 1973.

[9] Richards, F., and Williams, E., Application Software Tuning, USE Fall Conference, October 1981.

[10] UNIVAC Operatings System Installation Reference, UP-8486.

[11] Wyrick, T., and Self, C. Use of Remote Terminal Emulation in Federal ADP System Procurements, FEDSIM, Draft Working Paper NA-018-025-GSA, March 1979.

APPENDIX A

FIGURE A-1

PROGRAM PROFILE REPORT

MITRE WORKLDAD CHARACTERIZATIDN SYSTEM / PROGRAM PRDFILE PAGE 1.

NAME	RUNTYP	ROW	NUMBER	SIZE	IDPART	SERVICE FACTDR	SUP	CPU	CCER	ID	ET
@@@@@@@@@@@@	DEMAND	TDTAL	48	--	--	--	1535.80	263.45	143.94	1128.41	8867.0
--	--	AVERAGE	--	22.81	0.73	5.77	32.00	5.49	3.00	23.51	184.7
--	--	SDEV	--	9.63	0.27	110.64	69.45	13.80	4.59	51.89	183.5
ASM	BATCH	TDTAL	4	--	--	--	17.43	0.93	13.83	2.67	491.0
--	--	AVERAGE	--	27.33	0.15	28.18	4.36	0.23	3.46	0.67	122.7
--	--	SDEV	--	0.33	0.05	10.10	0.57	0.19	0.07	0.31	31.0
ASM	DEMAND	TDTAL	2	--	--	--	4.23	0.07	3.76	0.41	67.0
--	--	AVERAGE	--	16.90	0.10	15.84	2.11	0.03	1.88	0.20	33.5
--	--	SDEV	--	0.00	0.06	12.71	2.08	0.05	1.77	0.27	19.1
CHAIN1	BATCH	TDTAL	12	--	--	--	28.12	0.00	16.23	11.88	66.0
--	--	AVERAGE	--	16.98	0.42	2.35	2.34	0.00	1.35	0.99	5.5
--	--	SOEV	--	0.20	0.00	2.25	0.01	0.00	0.00	0.01	5.2
CHAIN1	DEMAND	TDTAL	13	--	--	--	26.66	0.07	14.89	11.70	71.0
--	--	AVERAGE	--	16.20	0.44	2.66	2.05	0.01	1.15	0.90	5.5
--	--	SDEV	--	0.58	0.21	3.06	0.57	0.01	0.50	0.10	4.1
CHAIN2	BATCH	TOTAL	12	--	--	--	577.65	69.27	62.22	446.16	4812.0
--	--	AVERAGE	--	62.39	0.77	8.33	48.14	5.77	5.18	37.18	401.0
--	--	SDEV	--	1.08	0.11	11.49	32.92	3.97	3.47	26.06	375.6
CHAIN2	DEMAND	TDTAL	9	--	--	--	388.32	92.16	33.92	262.24	1192.0
--	--	AVERAGE	--	60.76	0.68	3.07	43.15	10.24	3.77	29.14	132.4
--	--	SDEV	--	2.15	0.18	0.87	22.88	10.30	1.94	17.84	87.6
CHAIN3	BATCH	TDTAL	17	--	--	--	566.56	376.05	32.51	158.01	6106.0
--	--	AVERAGE	--	62.48	0.28	10.78	33.33	22.12	1.91	9.29	359.2
--	--	SDEV	--	0.03	0.06	13.91	16.57	12.76	0.31	3.76	419.4
CHAIN3	DEMAND	TDTAL	2	--	--	--	39.81	8.85	8.22	22.74	65.0
--	--	AVERAGE	--	64.14	0.57	1.63	19.90	4.43	4.11	11.37	32.5
--	--	SDEV	--	0.04	0.11	0.05	5.32	0.09	0.26	5.15	7.8
FURPUR	BATCH	TDTAL	513	--	--	--	7909.12	25.82	3251.28	4632.02	50355.0
--	--	AVERAGE	--	14.38	0.59	6.37	15.42	0.05	6.34	9.03	98.2
--	--	SDEV	--	0.94	0.28	48.30	35.56	0.21	12.93	31.85	221.2
FURPUR	DEMAND	TDTAL	1505	--	--	--	15646.23	60.70	3956.29	11629.24	117308.0
--	--	AVERAGE	--	13.13	0.74	7.50	10.40	0.04	2.63	7.73	77.9
--	--	SDEV	--	0.76	0.30	95.48	33.29	0.17	5.42	31.44	144.1
HDPE	BATCH	TDTAL	5	--	--	--	1213.27	421.27	101.04	690.96	4690.0
--	--	AVERAGE	--	67.50	0.57	3.87	242.65	84.25	20.21	138.19	938.0
--	--	SDEV	--	22.84	0.13	0.93	536.48	188.40	40.85	307.23	2081.2
HOPE	DEMANO	TOTAL	6	--	--	--	16.22	0.00	11.66	4.56	42.0
--	--	AVERAGE	--	16.53	0.28	2.59	2.70	0.00	1.94	0.76	7.0
--	--	SDEV	--	0.23	0.00	2.10	0.01	0.00	0.01	0.01	5.7
MAP	BATCH	TDTAL	44	--	--	--	3020.35	303.31	935.64	1781.40	12562.0
--	--	AVERAGE	--	43.16	0.59	4.16	68.64	6.89	21.26	40.49	285.5
--	--	SDEV	--	10.45	0.09	2.77	39.12	4.26	11.64	25.19	281.1
MAP	OEMANO	TDTAL	161	--	--	--	4090.56	365.42	1408.01	2317.13	17974.0
--	--	AVERAGE	--	30.86	0.57	4.39	25.41	2.27	8.75	14.39	111.6
--	--	SOEV	--	6.72	0.12	17.39	28.87	3.16	6.99	19.98	108.8
SVDS2	DEMAND	TOTAL	6	--	--	--	201.59	37.98	35.42	128.19	1542.0
--	--	AVERAGE	--	64.92	0.64	7.65	33.60	6.33	5.90	21.37	257.0
--	--	SDEV	--	0.50	0.08	2.90	11.65	4.24	2.55	5.06	103.0
SVOS3	BATCH	TOTAL	6	--	--	--	5106.73	3865.60	284.17	956.95	75549.0
--	--	AVERAGE	--	112.32	0.19	14.79	851.12	644.27	47.36	159.49	12591.5
--	--	SDEV	--	5.57	0.28	39.63	1105.00	1033.36	18.36	59.64	7209.8

FIGURE A-2

PROGRAM DETAIL REPORT

MITRE WDRKLDAD CHARACTERIZATIDN SYSTEM
1181-O 24 JULY 1981
LISTED BY SUPS

NUMBER DF PRDGRAMS 6389.
SUP SECDNDS 107418.040
SUP SEC * K WDRDS 4301904.3

NAME	PART DF TDTAL NUMBER	PART DF TDTAL SUPS	PART DF TOTAL CBSUPS
FURPUR	.3159	.2193	.0742
ED	.1537	.0762	.0181
MAP	.0321	.0662	.0597
SVDS1	.0019	.0580	.1255
SVDS3	.0009	.0475	.1333
DTT	.0020	.0448	.0620
CCTXQT	.0009	.0307	.0311
FDR	.1011	.0287	.0270
AMDP	.0002	.0224	.0209
QDINPT	.0055	.0196	.0332
SVDS	.0005	.0187	.0333
ABS	.0064	.0173	.0208
CHECK	.0006	.0156	.0080
GRTLS	.0008	.0154	.0196
@@@@@@@@@@@@	.0075	.0143	.0081
SSFS	.0006	.0134	.0195
SAGE	.0016	.0132	.0130
ELT	.1215	.0122	.0028
HDPE	.0017	.0114	.0191
TRWPLT-32	.0013	.0094	.0139
SBMIN	.0003	.0094	.0127
CHAIN2	.0033	.0090	.0139
SVDSIN	.0008	.0089	.0067
CDPIBM	.0005	.0077	.0035
PLDT	.0025	.0077	.0123
EDABS	.0003	.0068	.0096
EMAP	.0002	.0068	.0103
CHAIN4	.0027	.0063	.0094
SHAPER	.0002	.0063	.0101
ACTOO7	.0002	.0062	.0080
PSTAT	.0009	.0061	.0086
SECURE	.0006	.0059	.0040
SVDS5	.0002	.0058	.0083
DAP	.0002	.0057	.0080
CHAIN3	.0030	.0056	.0088
SVDS14	.0002	.0056	.0102
TABDIS	.0009	.0054	.0038
A	.0003	.0054	.0020
EMAP2	.0002	.0047	.0078
B	.0002	.0046	.0038
TAPELABEL	.0285	.0041	.0005
PLDTTAB	.0013	.0039	.0023

MITRE WORKLDAD CHARACTERIZATION SYSTEM
1181-O 24 JULY 1981
LISTED BY CBSUPS

NUMBER DF PRDGRAMS 6389.
SUP SECDNDS 107418.040
SUP SEC * K WORDS 4301904.3

NAME	PART DF TOTAL NUMBER	PART DF TOTAL SUPS	PART DF TOTAL CBSUPS
SVDS3	.0009	.0475	.1333
SVDS1	.0019	.0580	.1255
FURPUR	.3159	.2193	.0742
OTT	.0020	.0448	.0620
MAP	.0321	.0662	.0597
SVDS	.0005	.0187	.0333
DDINPT	.0055	.0196	.0332
CCTXQT	.0009	.0307	.0311
FOR	.1011	.0287	.0270
AMDP	.0002	.0224	.0209
ABS	.0064	.0173	.0208
GRTLS	.0008	.0154	.0196
SSFS	.0006	.0134	.0195
HOPE	.0017	.0114	.0191
ED	.1537	.0762	.0181
TRWPLT-32	.0013	.0094	.0139
CHAIN2	.0033	.0090	.0139
SAGE	.0016	.0132	.0130
SBMIN	.0003	.0094	.0127
PLOT	.0025	.0077	.0123
EMAP	.0002	.0068	.0103
SVDS14	.0002	.0056	.0102
SHAPER	.0002	.0063	.0101
EOABS	.0003	.0068	.0096
CHAIN4	.0027	.0063	.0094
CHAIN3	.0030	.0056	.0088
PSTAT	.0009	.0061	.0086
SVOS5	.0002	.0058	.0083
@@@@@@@@@@@@	.0075	.0143	.0081
DAP	.0002	.0057	.0080
CHECK	.0006	.0156	.0080
ACTOO7	.0002	.0062	.0080
EMAP2	.0002	.0047	.0078
SVDSIN	.0008	.0089	.0067
LAND6D	.0002	.0032	.0059

FIGURE A-3

WORKLOAD MODEL REPORT

MISC.BATCH AFTER PROCESSING OUPLICATES

NAME	SIZE RANGE	IOPART RANGE	NUMBER	SIZE	SUP	CPU	CCER	IO	ET	DELAY
.299ACS	--	--	172	22.5	5.7	0.7	1.1	3.9	38.2	13.2
.790PR	--	--	10	5.6	98.8	4.8	93.4	0.6	10032.7	10744.0
PLTOMP	--	--	2	11.6	4.2	0.1	3.5	0.6	7346.0	7265.2
--	0 - 10 K	0 - .2	237	5.7	2.0	0.0	1.8	0.3	66.4	0.0
--	0 - 10 K	.2 - .4	146	8.6	1.6	0.0	1.2	0.4	3.3	0.3
--	0 - 10 K	.4 - .6	19	8.7	3.0	0.1	1.4	1.5	10.2	0.0
--	0 - 10 K	.6 - .8	7	9.1	4.2	0.1	1.3	2.8	38.3	3.1
--	0 - 10 K	.8 - 1.0	58	6.7	4.7	0.0	0.5	4.2	33.6	0.1
--	10 - 20 K	0 - .2	431	15.7	9.4	0.3	7.9	1.2	40.3	0.0
--	10 - 20 K	.2 - .4	275	15.8	14.6	3.8	7.2	3.5	57.7	0.0
--	10 - 20 K	.4 - .6	103	16.0	12.2	2.9	2.8	6.5	44.4	0.0
--	10 - 20 K	.6 - .8	82	13.3	9.7	1.2	1.7	6.8	52.9	0.1
--	10 - 20 K	.8 - 1.0	121	14.9	60.1	0.5	4.1	55.5	202.6	0.1
--	20 - 40 K	0 - .2	18	29.5	113.4	98.8	5.8	8.8	559.8	0.
--	20 - 40 K	.2 - .4	21	36.3	8.3	2.5	3.4	2.4	92.4	0.0
--	20 - 40 K	.4 - .6	294	33.6	9.5	2.1	2.5	4.9	149.6	0.0
--	20 - 40 K	.6 - .8	53	29.4	29.7	3.7	6.2	19.8	115.1	0.0
--	20 - 40 K	.8 - 1.0	6	33.7	186.3	10.5	18.2	157.6	441.5	0.1
--	OVER 40 K	0 - .2	44	93.1	944.1	827.3	36.0	80.8	10484.9	0.3
--	OVER 40 K	.2 - .4	54	67.8	118.5	64.2	21.1	33.2	2798.6	0.4
--	OVER 40 K	.4 - .6	81	60.2	175.8	61.5	27.9	86.4	1173.4	0.2
--	OVER 40 K	.6 - .8	55	47.8	114.3	20.5	10.7	83.2	971.1	0.0
--	OVER 40 K	.8 - 1.0	11	45.9	668.6	14.4	40.3	613.9	824.7	0.1

MISC.OEMAND AFTER PROCESSING OUPLICATES

NAME	SIZE RANGE	SUP RANGE	NUMBER	SIZE	SUP	CPU	CCER	IO	ET	DELAY
MULT.ACTIVITY	--	--	31	40.3	68.4	3.0	38.7	26.7	456.6	679.1
--	0 - 10 K	0 - 2 SEC	1047	7.8	1.2	0.0	0.8	0.4	32.2	24.3
--	0 - 10 K	2 - 4 SEC	457	8.0	2.7	0.1	1.7	0.9	122.9	98.3
--	0 - 10 K	4 - 8 SEC	306	8.9	5.4	0.2	3.4	1.8	305.4	277.2
--	0 - 10 K	8 - 16 SEC	103	8.9	11.0	0.7	5.1	5.3	995.3	919.0
--	0 - 10 K	OVER 16 SEC	45	9.1	37.0	6.2	7.6	23.3	2919.4	2684.5
--	10 - 20 K	0 - 2 SEC	1254	12.7	1.1	0.0	0.8	0.3	34.6	18.5
--	10 - 20 K	2 - 4 SEC	426	12.6	2.8	0.1	1.9	0.8	58.4	42.8
--	10 - 20 K	4 - 8 SEC	403	13.6	5.6	0.4	2.8	2.4	55.4	34.5
--	10 - 20 K	8 - 16 SEC	212	14.4	10.3	0.9	4.4	5.0	74.3	43.1
--	10 - 20 K	OVER 16 SEC	233	13.4	66.6	4.3	9.4	53.0	271.3	106.6
--	20 - 40 K	0 - 2 SEC	262	37.2	1.1	0.0-	0.7	0.4	6.9	2.6
--	20 - 40 K	2 - 4 SEC	386	35.9	3.0	0.3	1.1	1.6	19.0	4.4
--	20 - 40 K	4 - 8 SEC	215	35.4	5.5	0.8	1.7	3.1	19.3	3.6
--	20 - 40 K	8 - 16 SEC	151	27.5	11.0	1.3	4.1	5.6	27.0	1.8
--	20 - 40 K	OVER 16 SEC	134	30.0	87.5	10.2	29.7	47.6	238.2	19.0
--	OVER 40 K	0 - 2 SEC	3	57.6	1.9	0.0	0.0	1.8	56.3	34.9
--	OVER 40 K	2 - 4 SEC	21	56.5	3.0	0.1	0.7	2.1	29.7	20.8
--	OVER 40 K	4 - 8 SEC	154	63.8	5.7	0.5	1.4	3.8	32.6	11.7
--	OVER 40 K	8 - 16 SEC	43	61.2	10.3	2.0	3.1	5.1	109.3	54.1
--	OVER 40 K	OVER 16 SEC	102	55.6	190.1	65.3	28.6	96.3	867.9	272.5

FIGURE A-4

FREQUENCY PROFILE REPORT

MITRE WORKLOAD CHARACTERIZATION SYSTEM / FREQUENCY PROFILE

VARIABLE 2, SIZE

		ALL			BATCH			DEMAND		
LOW	HIGH	N	PCT	CUM	N	PCT	CUM	N	PCT	CUM
1.54	2.61	28	0	0	1	0	0	27	1	1
3.03	3.62	254	4	4	102	5	5	152	3	4
4.10	4.65	82	1	6	38	2	7	44	1	5
4.91	5.71	77	1	7	52	3	10	25	1	6
6.14	6.66	43	1	8	29	1	11	14	0	6
7.17	7.68	72	1	9	53	3	14	19	0	6
8.19	9.05	983	15	24	426	22	35	557	13	19
9.09	10.13	675	11	35	72	4	39	603	14	33
10.15	11.19	226	4	38	65	3	42	161	4	36
11.23	12.28	133	2	40	47	2	45	86	2	38
12.29	13.36	1452	23	63	174	9	54	1278	29	67
13.36	14.37	418	7	70	160	8	62	258	6	73
14.45	15.41	241	4	73	159	8	70	82	2	75
15.72	16.53	81	1	75	55	3	72	26	1	76
16.73	17.64	57	1	75	26	1	74	31	1	76
17.87	33.57	431	7	82	72	4	77	359	8	84
33.99	49.71	856	13	96	287	14	92	569	13	97
50.07	65.54	198	3	99	117	6	98	81	2	99
66.38	80.92	67	1	100	32	2	99	35	1	100
87.15	94.21	8	0	100	5	0	100	3	0	100
102.95	106.06	5	0	100	5	0	100			
118.55	118.55	1	0	100	1	0	100			
260.10	260.10	1	0	100				1	0	100
MISSING DATA 1			0.			0.			0.	
MISSING DATA 2			0.			0.			0.	
MISSING DATA 3			0.			0.			0.	
GOOD N			6389.			1978.			4411.	
MEAN			17.6609			19.5675			16.8059	
VARIANCE			211.7307			301.1689			169.3220	
S.D.			14.5510			17.3542			13.0124	

MITRE WORKLOAD CHARACTERIZATION SYSTEM / FREQUENCY PROFILE

VARIABLE 3, SUP

		ALL			BATCH			DEMAND		
LOW	HIGH	N	PCT	CUM	N	PCT	CUM	N	PCT	CUM
.0508	1.1674	1259	20	20	431	22	22	828	19	19
1.1686	2.2856	1641	26	45	537	27	49	1104	25	44
2.2862	3.4014	828	13	58	225	11	60	603	14	57
3.4068	4.5176	430	7	65	107	5	66	323	7	65
4.5230	5.6328	359	6	71	89	4	70	270	6	71
5.6506	6.7380	299	5	75	89	4	75	210	5	76
6.7564	7.8662	152	2	78	19	1	76	133	3	79
7.8740	8.9756	139	2	80	32	2	77	107	2	81
9.0160	10.0998	93	1	81	25	1	79	68	2	83
10.1082	11.2036	80	1	83	11	1	79	69	2	84
11.2452	12.3336	78	1	84	14	1	80	64	1	86
12.3770	13.4552	71	1	85	14	1	81	57	1	87
13.4606	14.5434	49	1	86	18	1	81	31	1	88
14.6480	15.6886	27	0	86	13	1	82	14	0	88
15.6992	16.8118	35	1	87	8	0	82	27	1	89
16.8310	209.4292	767	12	99	305	15	98	462	10	99
217.3740	393.3298	39	1	99	19	1	99	20	0	100
406.9988	557.4022	19	0	100	7	0	99	12	0	100
598.3640	761.8736	14	0	100	7	0	100	7	0	100
891.3162	954.0808	2	0	100				2	0	100
984.4178	984.4178	1	0	100	1	0	100			
1202.3346	1339.1040	2	0	100	2	0	100			
1857.8888	1857.8888	1	0	100	1	0	100			
1948.9998	2129.7172	2	0	100	2	0	100			
2405.4118	2405.4118	1	0	100	1	0	100			
2911.0068	2911.0068	1	0	100	1	0	100			
MISSING DATA 1			0.			0.			0.	
MISSING DATA 2			0.			0.			0.	
MISSING DATA 3			0.			0.			0.	
GOOD N			6389.			1978.			4411.	
MEAN			16.8133			25.6291			12.8600	
VARIANCE			7476.4481			18588.9903			2445.9072	
S.D.			86.4665			136.3414			49.4561	

FIGURE A-5

ACCOUNT PROFILE REPORT

WDRKLDAD CHARACTERIZATIDN SYSTEM / 1181-O APRIL 82 / ACCDUNT PRDFILE PAGE 1, FILE RESULT

SHIFT	RUNTYP	RDW	NUMBER	SIZE	IDPART	SERVICE FACTOR	SUP	CPU	CCER	ID	ET
----------ACCT = FM1----------											
PRIME	DEMAND	TOTAL	57	--	--	--	459.82	9.24	280.05	170.52	5497.0
--	--	AVERAGE	--	17.40	0.37	11.95	8.07	0.16	4.91	2.99	96.4
NDN.PRIME	OEMAND	TDTAL	102	--	--	--	1013.19	20.72	112.12	880.35	2541.0
--	--	AVERAGE	--	13.41	0.87	2.51	9.93	0.20	1.10	8.63	24.9
DVER.NIGHT	DEMAND	TDTAL	15	--	--	--	529.95	4.43	57.58	467.94	993.0
--	--	AVERAGE	--	26.84	0.88	1.87	35.33	0.30	3.84	31.20	66.2
----------ACCT = FM2----------											
PRIME	BATCH	TOTAL	62	--	--	--	58.22	1.60	33.95	22.67	1754.0
--	--	AVERAGE	--	12.15	0.39	30.13	0.94	0.03	0.55	0.37	28.3
PRIME	DEMAND	TDTAL	222	--	--	--	3749.52	632.35	576.40	2540.77	38398.0
--	--	AVERAGE	--	24.31	0.68	10.24	16.89	2.85	2.60	11.44	173.0
NDN.PRIME	DEMAND	TDTAL	70	--	--	--	1707.35	1180.31	161.53	365.50	12361.0
--	--	AVERAGE	--	48.52	0.21	7.24	24.39	16.86	2.31	5.22	176.6
----------ACCT = FM4----------											
PRIME	BATCH	TDTAL	221	--	--	--	11032.82	6208.87	1779.61	3044.35	80295.0
--	--	AVERAGE	--	58.90	0.28	7.28	49.92	28.09	8.05	13.78	363.3
PRIME	DEMAND	TDTAL	655	--	--	--	8053.09	1521.82	2293.19	4238.08	104833.0
--	--	AVERAGE	--	25.34	0.53	13.02	12.29	2.32	3.50	6.47	160.1
NON.PRIME	DEMAND	TOTAL	172	--	--	--	2430.69	395.94	556.76	1477.99	20989.0
--	--	AVERAGE	--	27.47	0.61	8.63	14.13	2.30	3.24	8.59	122.0
DVER.NIGHT	BATCH	TDTAL	116	--	--	--	6672.43	3705.67	1373.03	1593.73	41179.0
--	--	AVERAGE	--	62.58	0.24	6.17	57.52	31.95	11.84	13.74	355.0
OVER.NIGHT	DEMAND	TDTAL	27	--	--	--	194.31	5.61	102.88	85.81	1760.0
--	--	AVERAGE	--	26.80	0.44	9.06	7.20	0.21	3.81	3.18	65.2
----------ACCT = FM5----------											
PRIME	BATCH	TOTAL	570	--	--	--	8764.60	1798.40	1955.03	5011.17	114803.0
--	--	AVERAGE	--	37.35	0.57	13.10	15.38	3.16	3.43	8.79	201.4
PRIME	DEMAND	TDTAL	239	--	--	--	9208.94	2712.63	1039.55	5456.75	76740.0
--	--	AVERAGE	--	46.42	0.59	8.33	38.53	11.35	4.35	22.83	321.1
NON.PRIME	BATCH	TOTAL	56	--	--	--	2298.42	1253.98	333.17	711.28	18856.0
--	--	AVERAGE	--	71.91	0.31	8.20	41.04	22.39	5.95	12.70	336.7
NDN.PRIME	DEMANO	TOTAL	14	--	--	--	50.60	2.65	32.91	15.05	1674.0
--	--	AVERAGE	--	16.94	0.30	33.08	3.61	0.19	2.35	1.07	119.6
OVER.NIGHT	BATCH	TOTAL	69	--	--	--	4023.17	2699.75	425.73	897.69	30395.0
--	--	AVERAGE	--	100.17	0.22	7.55	58.31	39.13	6.17	13.01	440.5
DVER.NIGHT	DEMAND	TDTAL	3	--	--	--	474.33	243.78	26.37	204.18	3445.0
--	--	AVERAGE	--	54.98	0.43	7.26	158.11	81.26	8.79	68.06	1148.3
----------ACCT = FM8----------											
PRIME	BATCH	TOTAL	3	--	--	--	16.56	0.17	7.08	9.31	349.0
--	--	AVERAGE	--	17.33	0.56	21.08	5.52	0.06	2.36	3.10	116.3
PRIME	DEMAND	TDTAL	1119	--	--	--	11546.22	964.33	2927.08	7654.82	137186.0
--	--	AVERAGE	--	23.99	0.66	11.88	10.32	0.86	2.62	6.84	122.6
NON.PRIME	BATCH	TDTAL	19	--	--	--	1388.49	437.54	130.05	820.90	6181.0
--	--	AVERAGE	--	61.51	0.59	4.45	73.08	23.03	6.84	43.21	325.3
NON.PRIME	DEMAND	TDTAL	103	--	--	--	1436.78	95.17	311.79	1029.82	11230.0
--	--	AVERAGE	--	41.03	0.72	7.82	13.95	0.92	3.03	10.00	109.0
OVER.NIGHT	BATCH	TOTAL	186	--	--	--	3461.45	906.06	442.71	2112.68	18841.0
--	--	AVERAGE	--	46.50	0.61	5.44	18.61	4.87	2.38	11.36	101.3
OVER.NIGHT	OEMANO	TOTAL	8	--	--	--	34.39	1.38	16.47	16.54	169.0
--	--	AVERAGE	--	15.14	0.48	4.91	4.30	0.17	2.06	2.07	21.1
----------ACCT = DVHD----------											
PRIME	BATCH	TOTAL	42	--	--	--	2994.71	365.56	1363.64	1265.50	16178.0
--	--	AVERAGE	--	24.94	0.42	5.40	71.30	8.70	32.47	30.13	385.2
PRIME	DEMAND	TOTAL	87	--	--	--	697.36	71.55	301.98	323.83	45093.0
--	--	AVERAGE	--	16.88	0.46	64.66	8.02	0.82	3.47	3.72	518.3
NON.PRIME	BATCH	TDTAL	6	--	--	--	416.14	3.65	24.76	387.73	808.0
--	--	AVERAGE	--	10.33	0.93	1.94	69.36	0.61	4.13	64.62	134.7
NON.PRIME	OEMAND	TDTAL	29	--	--	--	118.09	11.28	57.89	48.92	2592.0

FIGURE A-6

TERMINAL PROFILE REPORT

MITRE WORKLOAD CHARACTERIZATION SYSTEM / TERMINAL PROFILE
SORTED AFTER PROCESSING DUPLICATES

DEVICE	ET	NUMBER	SIZE	SUP MIN	PAGES	CROIN	THINK TIME	RESPONSE TIME
170	34061	25	23.7	29.5	396	3893	5.5	3.2
171	32363	15	24.8	58.7	1762	13872	1.2	1.1
270	24026	9	23.4	72.8	1249	876	12.2	15.2
271	24170	10	14.7	40.8	1282	1237	10.6	9.0
272	30610	9	31.4	46.7	562	1982	8.6	6.8
273	28272	8	34.7	23.7	488	3122	5.8	3.3
670	4924	6	23.0	13.6	26	190	15.3	10.6
1070	24774	12	20.1	37.7	593	1983	7.5	5.0
1071	28034	19	22.0	13.8	410	1304	15.6	5.9
2072	2091	3	10.5	1.1	57	62	13.8	19.9
2073	18605	4	22.6	26.9	1124	959	8.7	10.7
2770	29353	16	17.2	25.0	349	2016	9.5	5.1
2771	26459	10	22.3	28.2	206	1539	11.7	5.5
10025	2327	1	49.5	0.6	8	76	16.5	14.1
10026	30347	19	20.7	26.7	726	9865	2.1	1.0
10029	5771	3	24.0	7.4	33	317	9.2	9.0
10030	11674	2	19.4	11.2	106	3884	1.8	1.2
10035	6129	2	13.4	21.8	61	448	2.4	11.3
10038	30192	19	13.2	27.0	261	1890	10.7	5.3
10039	13139	2	35.9	10.2	160	408	18.8	13.4
10061	7363	2	33.0	12.0	30	305	14.2	9.9
10063	2375	2	9.3	0.6	16	509	3.4	1.2
10064	7055	3	14.2	2.6	32	252	23.0	5.0
10077	30402	15	21.6	29.3	598	4468	4.4	2.4
10092	16806	4	32.4	18.0	409	5448	1.8	1.3
10101	587	1	11.4	1.2	2	15	22.6	16.6
10103	1229	2	7.5	1.9	72	101	4.4	7.8
10105	19662	9	9.8	4.9	87	1029	15.5	3.6
10112	26744	18	21.0	28.8	278	1219	14.0	8.0
10120	3329	1	18.2	5.7	38	294	5.0	6.3
10121	14774	4	19.1	21.0	378	1465	6.4	3.7
10132	16977	4	15.0	11.8	304	1389	8.0	4.3
10134	543	1	16.5	0.4	2	24	15.3	7.3
10142	29446	6	15.4	23.8	157	1302	16.2	6.4
20013	19866	7	40.0	52.9	214	1754	4.3	7.0
20014	32342	21	49.4	82.4	303	4445	3.3	4.0
20015	1380	1	18.2	2.0	26	102	3.7	9.8
20016	33834	14	26.5	60.2	1741	2553	7.2	6.1
20023	27875	5	43.5	41.5	186	2243	4.8	7.6
20024	20590	7	29.9	15.1	141	1336	7.0	8.4
20034	26165	14	31.4	32.8	499	2538	6.3	4.0
20040	30619	16	33.7	37.7	440	4463	3.7	3.1

FIGURE A-7

USERS OF SELECTED APPLICATIONS

ACCT	BADGE	BOX	PROJ	NUMBER	SIZE	SUP	IO	ADJUSTED SERVICE FACTOR
----------NAME = CHAIN1----------								
FM8	1800650	333	8910	6	15.5	10.3	4.8	3.8
FM8	1800108	334	8910	6	16.9	14.0	5.9	3.0
FM8	1800483	332	8910	6	17.1	14.1	6.0	1.7
FM8	1800808	332	8910	7	16.6	16.3	6.9	2.0
----------NAME = CHAIN2----------								
FM8	1800650	333	8910	2	57.3	106.2	41.8	2.3
FM8	1800483	332	8910	6	63.3	192.2	140.6	9.9
FM8	1800808	332	8910	7	62.0	282.1	220.4	3.3
FM8	1800108	334	8910	6	62.0	385.4	305.6	7.5
----------NAME = CHAIN3----------								
FM8	1800650	333	8910	2	64.1	39.8	22.7	1.6
FM8	1800483	332	8910	9	62.5	230.4	61.7	6.0
FM8	1800108	334	8910	8	62.5	336.1	96.3	14.0
----------NAME = CHAIN4----------								
FM8	1800483	332	8910	9	59.9	285.0	104.7	5.2
FM8	1800108	334	8910	8	59.9	393.4	195.1	6.4
----------NAME = CHAIN5----------								
FM8	1800483	332	8910	3	63.8	20.1	13.1	5.4
FM8	1800108	334	8910	2	63.5	35.3	23.1	7.3
FM8	1800650	333	8910	2	62.6	53.1	21.2	2.5
----------NAME = CHAIN6----------								
FM8	1800650	333	8910	1	80.9	43.3	25.6	2.4

FIGURE A-8

"CHAIN" APPLICATIONS

SUB.STATS. SUMS OF S.CHAIN

AVG SIZE	AVG IOPART	TOTAL SUP	TOTAL CPU	TOTAL CCER	TOTAL IO	TOTAL ET	NUMBER OF EXECUTIONS
----------RUNTYP = BATCH----------							
61.01	0.32	2234.36	1416.76	103.36	714.25	16956	35
----------RUNTYP = DEMAND----------							
65.79	0.48	454.49	199.59	36.50	218.40	1298	9

FIGURE A-9

RUNS ON DEVICES WITH LONG RESPONSE TIME

DEVICE	RUNSEQ	ET	SIZE	SUP	PAGES	CRDIN	THINK TIME	RESPD NSE TIME	SUP PER XACT
670	2	1016	26.8	515.4	11	33	6.6	24.2	15.6
670	11	659	12.7	19.4	2	13	48.4	2.2	1.5
10101	29	587	11.4	72,7	2	15	22.6	16.6	4.8
270	55	1254	17.1	678.7	95	95	2.7	10.5	7.1
270	81	1511	12.3	767.0	2	31	15.8	32.9	24.7
2073	103	429	9.4	24.3	22	19	18.8	3.8	1.3
10035	107	776	10.1	22.4	4	24	28.1	4.2	0.9
270	112	3258	22.6	919.2	29	86	23.9	14.0	10.7
2073	142	1579	15.1	8.3	3	47	52.2	-18.6	0.2
10025	149	2327	49.5	38.9	8	76	16.5	14.1	0.5
2072	167	569	19.7	20.9	7	20	32.8	-4.3	1.0
270	193	3146	8.6	81.5	11	181	9.7	7.7	0.5
670	208	1504	22.1	158.5	3	34	25.9	18.3	4.7
2073	219	3256	20.5	208.8	20	157	14.9	5.9	1.3
270	222	1668	12.2	436.4	3	38	14.5	29.4	11.5
10039	238	6238	49.3	349.6	24	141	27.7	16.5	2.5
270	250	2949	20.8	412.3	64	167	7.8	9.9	2.5
2072	272	983	5.8	31.4	34	30	22.8	9.9	1.0
270	280	2884	9.6	91.6	12	161	12.7	5.2	0.6
270	292	687	6.6	16.2	2	23	20.1	9.7	0.7
670	325	321	12.8	19.6	2	13	16.4	8.3	1.5
670	336	307	6.6	9.1	2	9	13.1	21.0	1.0
10039	370	6901	17.8	259.7	136	267	14.1	11.8	1.0
2072	372	539	7.0	13.4	16	12	14.3	30.6	1.1
2073	375	13341	23.2	1370.3	1079	736	7.7	10.4	1.9
270	386	6669	46.6	965.4	1031	94	18.7	52.3	10.3
10035	418	5353	13.5	1286.4	57	424	0.9	11.7	3.0
670	480	1117	9.1	91.3	6	88	9.7	3.0	1.0

FIGURE A-10

ADDITIONAL RUN VARIABLES

RUNSEQ	DEVICE	ACCT	PRDJ	BOX	BADGE	TIMEO	TIME1	NPRDG	ERR
103	2073	FD8	5816	120	1903084	93904	94613	3	0
107	10035	FM1	1490	1013	0	94634	95930	4	0
142	2073	FD8	2078	780	0	100611	103230	2	0
167	2072	OVHO	5164	800	2618149	105438	110407	3	0
219	2073	FD8	2078	263	0	110108	115524	14	1
238	10039	FD8	5815	740	2922186	104818	123216	20	0
272	2072	FD8	5816	269	1903284	130921	132544	4	0
370	10039	DVHD	5228	740	2914669	133557	153058	34	5
372	2072	FD8	5816	269	1903284	152456	153355	1	0
375	2073	FD8	2078	263	0	115708	153929	166	7
418	10035	FM1	1490	236	0	150032	162945	128	0

FIGURE A-11

PROGRAMS EXECUTED BY TWO SELECTED RUNS

TIMEO	NAME	SIZE	SUP	IDPART	CAT	TEMP	SERVICE FACTOR
----------RUNSEQ = 272----------							
131053	FURPUR	12.8	3.38	0.45	2	0	5.70
131406	FURPUR	12.8	3.37	0.46	2	0	7.15
131559	FURPUR	12.8	3.37	0.46	2	0	4.35
131807	FURPUR	12.8	1.95	0.19	2	0	62.85
----------RUNSEQ = 372----------							
152552	FURPUR	12.8	6.65	0.32	6	0	48.57

CPEUG82

"Improving Organizational Productivity"

IBM Performance Analysis

SESSION OVERVIEW

IBM PERFORMANCE ANALYSIS

Tim Oliver

National Institutes of Health
Bethesda, MD 20205

The IBM Performance Analysis session treats, with regard to both MVS and VM systems, performance evaluation (measurement, analysis, and reporting) and enhancement (system reconfiguration, system modification, and workload scheduling). Specifically the papers include:

- Formulas to convert MVS RMF data into jobclass-level measures suitable for queueing models and performance analysis.
- The design of an MVS jobscheduler that assigns jobclass based upon resource parameters in existing JCL.
- Locations of hooks into MVS to measure breakdowns of CPU and I/O times more accurately than does RMF.
- An approach to finding VM bottlenecks by deriving breakdowns of response time from user state samples.
- Case studies of structural modifications to VM designed to improve performance.
- A description of a system to accumulate, analyze, and report VM performance measurements.

RMF EQUATIONS: Obtaining Job Class Level Results from RMF

Bob Irwin

IKON
250 Highland Street
West Haven, CT 06516

RMF (Resource Management Facility, IBM's large system software monitor) is an IBM software monitor program product which runs at nearly every large scale IBM installation. RMF is the de facto performance tool for performance analysis, problem determination, and capacity planning. Yet, RMF is not adequate for analysis at the job class level, and virtually impossible to use as a source of input for mathematical modeling of the CPU. It is the purpose of this paper to develop a set of RMF equations, which when applied to RMF monitor measurement data, will obtain for the analyst, job class level results for machines which process multiple job classes. The RMF equations represent a productivity aid to the analyst. These equations will amplify the utility of RMF, extend the capabilities of RMF, and offer the analyst more insight into system performance and behavior.

Key words: Job class; software monitor; mathematical modeling; application of basic queueing theory; IBM's RMF; performance/modeling data acquisition.

1. Introduction

This paper first presents equations which obtain system level (TSO, CICS, IMS, MVS, BATCH...) results. Next we obtain fi(r)-that fraction of utilization against device i by job class r. We derive fi(r) in terms of queueing primitives. Then there are three short proofs which show that the utilization, service time, and busy time for a job class, are indeed obtainable from the product of fi(r) and their corresponding system level measurements. Given fi(r) we may now obtain job class level results. Next we obtain fi(r) in terms of RMF fields. We also obtain a bounding condition on which we base an experiment to measure and validate the job class level results obtained from applying the RMF equation of fi(r). We then use fi(r) to compute job class level results from RMF workload activity reports. Finally we derive from RMF multiprogramming levels, response time, total resident and active time, queue length, and queueing time, all at the job class level. We do this using fi(r) and basic queueing relationships. All results have examples of their use within an RMF context, and all RMF equations

are tabularized for quick and easy reference/use. The numerical results are not presented in scientific notation, but instead, are shown as decimal fraction expansions.

RMF (Resource Management Facility) is an IBM program product which performs performance monitoring of large scale IBM mainframes. It is the principle IBM performance product and runs at virtually every large scale installation.

1.1 Design of Paper

The design of this paper is from the top down. We start developing non-job class specific relationships and proceed downward to lower levels (job class specific) of detail in our measurement analysis. The format of the results is presented as follows: first a general equation, then the equation in terms of RMF fields, and lastly, one or more examples from RMF. The results are collected in two tables for your use and reference. All equations are given in terms of RMF field notation, hence the values may be directly introduced into the equation from the RMF reports. We first develop preliminary results on a non-job class specific level. The remainder of the paper concentrates on developing results on a job class level.

1.2 The Modeling Process

Mathematical modeling is composed of two phases, the model building phase and the model execution phase. In this paper we focus on the execution phase. In the building phase we employ intermediate, and when necessary, advanced mathematical techniques. The resulting model is a system of equation(s), generally differential equations, which describe the time variant behavior of some physical system, in this case a physical computing machine. The solutions to these differential equations (functions) are the equations used in basic and intermediate queueing theory. Other, more advanced results, do not admit exact solutions and thus approximation methods are employed. The model characterizes the behavior of the physical system in terms of its salient operational properties. In the instance of the computing system, we select only those properties which are characterized by performance, as its salient operational properties.

The testing/execution phase of the model requires a solution to a set of problems which differ from the model building task. These problems may be divided into three phases; the first phase is obtaining and validating the required input parameters to the model; the second phase is developing experiments which test the validity of the model; and the third phase is interpreting the model results in terms of the physical system under study. In particular, this paper focuses on the first phase: obtaining and validating model input.

1.3 Developing Input

There are two methods for developing input to mathematical models. One is by data collection, the other is by data derivation. Data collection is slow, inflexible, and non extensible, but accurate. Data derivation is fast, flexible, and extensible but, for the most part, not as accurate as data collection. In data collection we physically measure the required data. In data derivation we construct relationships (equations) which obtain the required data from gross level measurements we already have.

Typically the model exits; therefore, the data collection and development of techniques for data collection is both the most time intensive and critical task in the modeling effort. Job class level modeling data is not often captured in software monitors, and rarely in hardware monitors. The result is that data reduction programs must be built (and maintained) to capture this job class level data from log records (RMF and SMF records).

What is data derivation? Data derivation identifies the components of existing gross level measurements, and develops relationships between these data and its components. These relationships are equations which define the desired job class level data in terms of the components of existing gross system level measurement data. Once this is done, these relationships may be used to factor out the job class level data from <u>any</u> software or hardware monitor reports. In this case we use the IBM RMF software monitor.

The accuracy of these relationships and hence, the required modeling input, depends on the logical consistency of the relationships, and the integrity of

the data. If the equations are logically consistent and the data is incomplete or inaccurate then the results will be invalid. If the equations do not reflect the relationships between the existing data, then the desired results, again, will be invalid. Validation of accuracy depends on the experimental ingenuity of the analyst. Accuracy is important, however, it should not grow into an issue unto itself. The usefulness or utility of the results is at issue. Questions of accuracy are relevant but they must not dominate a study.

1.4 First Order Approximation

The reader is cautioned that the equations below represent a first order approximation to the model input parameters they represent. This is the result of the inability of RMF to monitor, in certain instances, the system measurements which the equations represent. The degree of approximation varies with respect to different equations; however, the instances where these appproximations are significant will be noted in the body of the paper.

Thus the use of these equations in a modeling effort represent a first order approximate modeling effort. If this first order level is of value (meets your needs) then the effort is complete. At the next level, data reduction programs must be built to monitor and report multi class level measurements which RMF, presently, does not monitor. As stated above, this is a time intensive effort, and should only be performed after first order modeling efforts prove unsatisfactory.

1.5 Notation

All queueing equations are given in terms of operational analysis notation (see "Definitions", Table 2. below) and not the Greek notation of queueing theory. In operational analysis notation the capital letter signifies the measurement parameter, and the subscript "i" signifies the device being measured. In this paper "i" is always understood to signify the CPU device. To introduce the concept of job class, a parenthetical subscript "(r)" , where r=job class, is incorporated into this notation. See Table 2. below. For example the notation Ui(r) is read: the utilization of device i by job class r. An instance being, Ucpu(TSO3). The device and job class subscripts are self defining, except for the case where the device subscript is "system", and the class subscript is "all". "System" signifies all devices, and "all" signifies all job classes incorporated into a single job class image. For example, consider: MPLcpu(TSO1) and MPLsys(ALL).

There are times when the class names "TSO" or "BATCH" are used. In these instances the class names contain their subclasses and their subclasses' associated measurements. The subclasses of TSO may be TSO1, TSO2 and TSO3. The subclasses of BATCH may be BATCH1 and BATCH2.

When the context is clear, the subscripts "i", "r", or both, may be understood to simplify notation, or, if the result is general, the "i" and "r" will be retained and will signify i=CPU, r=any job class.

The RMF notation, with the assistance of the "RMF Field Name Table" (see Table 1.) is self defining. The notation directly relates, for reference, to the RMF field names in the RMF Workload Activity Report printout. For your own computations the string "RMF" may be dropped from the paper's notation without loss of uniqueness. In the notation please note that the hyphen between the string "RMF" and some field name of the RMF report is one name: the hyphen does not represent a subtraction operation. Thus, "RMF-ET" is read, RMF Ended Transactions, and not RMF minus Ended Transactions. This should not present a source of confusion. The RMF notation also incorporates job class and, when required for clarity, device subscripts. Job classes correspond to measurements from the RMF Performance Group Reports, and the specific job class named "ALL", corresponds to measurements from the System Summary Report For example consider: RMF-IOC(ALL) and RMF-IOC(TSO1).

The notation for footnotes and references is as follows: "|n|" denotes a footnote, and "<n>" denotes a reference, where "n" is the reference or footnote number.

2. RMF System Level Results

Initially our goal is to define, in terms of RMF notation, the ten parame-

ters in Table 2. below, named "Definitions". These performance parameters do not differentiate between job classes, subclasses or devices. All results are in terms of the CPU for a conglomerate single job class image. It is assumed that all user defined RMF service coefficients are greater than zero. This effort is preliminary, it sets the stage for deriving model input parameters and output results at the job class level. All numerical results may be verified by using RMF report printouts; see figures 1, 2, 3, 4, and 5 at the end of this paper.

Table 1. RMF Field Names

1. AAR - Average Absorption Rate
2. ATSR - Average Transaction Service Rate
3. ISPS - Interval Service Per Second
4. ET - Ended Transactions
5. IOC - I/O service units
6. INTVL- Interval
7. CPU - CPU service units
8. SRB - SRB service units
9. SDC - Service Definition Coefficient
10. UTIL - Utilization (1-WAIT STATE)

Table 2. Definitions

1. T - Measurement interval in seconds
2. Bi(r) - Total Busy time of device i
3. Ci(r) - Number of Completions at device i = number of job service requests at device i
4. Co - Number of job Completions at the system
5. Ui(r) - Utilization of device i
6. Si(r) - Mean Service time per request at device i
7. Xi(r) - Mean Departure rate per request (throughput) at device i
8. Vi(r) - Number of Visits jobs make to device i
9. So - Mean Service time per job
10. Xo - Mean Departure rate per job

Unless noted otherwise the subscript "i" in this paper is always understood to mean the CPU device. In terms of RMF fields these definitions follow directly from basic queueing equations as follows: |1|

T = RMF-INTVL coverted to seconds.
= 3599 sec

Ui(ALL) = RMF-UTIL
= (1.00-(RMF-WAIT-STATE))
= Bi(ALL)/T
= 1.00-0.2009
= 0.7903

Bi(ALL) = Ui(ALL)*T
= RMF-UTIL*RMF-INTVL
= (0.7903)(3599 sec)
= 2844.29 sec

Si(ALL) = Bi(ALL)/Ci(ALL)

$$= \frac{\text{(RMF-UTIL*RMF-INTVL*RMF-IOC-SDC)}}{\text{RMF-IOC(ALL)}}$$

= 2844.29 sec/842292 req
= 0.00337 sec/req

Xi(ALL) = Ci(ALL)/T

$$= \frac{\text{RMF-IOC(ALL)}}{\text{RMF-IOC-SDC*RMF-INTVL}}$$

= 842292 req/3599 sec
= 234.035 req/sec

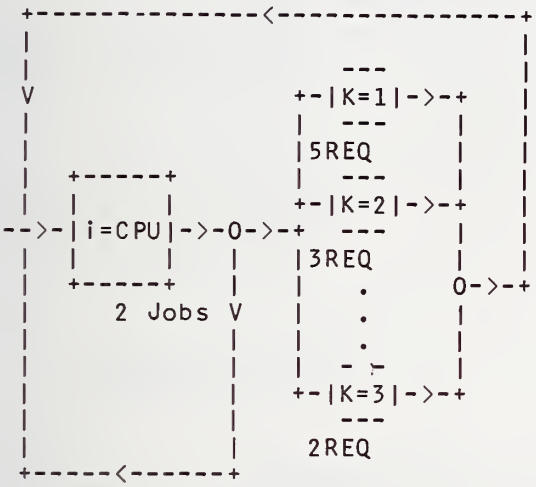
Ci(r) is defined below. "req" = requests

A request is a unit of service demanded from some device by a job associated with some class of jobs. A job is a unit of work which makes one or more requests in its journey through the system.

|1| See reference <1> for a source of these basic queueing equations. To validate results see RMF reports at the end of this paper.

2.1 What is Ci and how do we find it?

Ci is the sum of Cij for all i,j, and Cij is the "number of times a job requests sevice at device j immediately after completing a service request at device i." <1>
Graphically,



Cij = (5+3+2)REQ = 10REQ
for i = CPU, j = I/O device 1,2,3

In shorthand, for i = CPU,
j = I/O device 1,2,3

$$Ci = \sum_{j=1}^{K} Cij = (5+3+2)\ req = 10\ req.$$

From the figure we see that two jobs made ten service requests at K I/O devices after completing service at device i=CPU. Hence, Ci may be interpreted as the total number of requests of all jobs made at device i. Suppose we now divide Ci by the number of job completions. Job completions we denote as Co, where the subscript "o" suggests work which leaves the system and does not cycle in the network.

Thus Ci/Co = 10req/2jobs = 5req/job

We call this ratio the visit ratio. It denotes the average number of requests (visits) a job makes at device i.

Hence Vi = Ci/Co

Vi is a building block for service time, Si, and response time, Ri.

We have described what Ci is but how do we find it from RMF? By the above network figure we see that, |2|

Ci = RMF-IOC(ALL)/RMF-IOC-SDC <2>
Co = RMF-ET(ALL)

$$\text{and } Vi(ALL) = \frac{\text{RMF-IOC(ALL)/RMF-IOC-SDC}}{\text{RMF-ET(ALL)}}$$

$$= \frac{\text{RMF-IOC(ALL)}}{\text{RMF-IOC-SDC*RMF-ET(ALL)}}$$

For example, taking Si(ALL) above, |3|

Si(ALL) = 2844.29sec/(4211460/5)
= 2844.29sec/842292req
= 0.003376sec/req

And

Vi(ALL) = 842292req/9817job
= 85.80req/job

Hence, mean service time per job at the CPU is

Vi(ALL)*Si(ALL) =

(85.80req/job)*(0.003376sec/req)

= 0.2897sec/job

But

So = Bi/Co

|2| Here Rose, see <2>, defines Ci(r)=I/O OPi(r) for some job class r. In words, Ci equals the number of I/O operations initiated by device i=CPU. The method used in computing Ci(r) in this paper parallels Rose's method. Please note that the IOC does not include paging or swapping I/O counts. An IOC corresponds to one EXCP or JES spool read/write, and does not include EXCP's executed on behalf of the problem program in the system key, viz., paging/swapping I/O. The interested reader is directed to the section "Selecting Service Definition Coefficients" in the IBM Initialization and Tuning Guide. Therefore, this measurement and all other measurements which depend on this value are approximations.

|3| Since we are not concerned with job classes in these computations, all values are taken from the RMF System Summary of the Workload Activity Report. See figures at the end of this paper.

$$= \frac{(RMF\text{-}UTIL * RMF\text{-}INTVL)}{RMF\text{-}ET(ALL)}$$

$$= (0.7903 * 3599\ sec)/9817 job$$
$$= 0.289731 sec/job$$

Here we have derived the same result (correct to the ten thousand's place) for So by the conventional queueing equation (So = Bi/Co), and also, by the use of the visit ratio Vi which incorporates our construction of Ci. Hence, we have shown a method for validating Ci and Vi by comparing measurements which depend on Ci and Vi with a known measurement. The same procedure will be used for Xo.

Lastly, the system throughput is

$$Xo = Co/T$$
$$= 9817 job/3599 sec$$
$$= 2.73 job/sec$$

Below we will show that the same results for Xo may be derived with the use of Vi = Ci/Co.

The user is again warned (see footnote |2| above) that the value for "C", completion count, is approximate. Hence all subsequent equations which depend on C are also approximations.

We have now set the groundwork for the next level of detail, the multi class level. For the totality of job classes at the CPU, all relevant modeling parameters can be derived in terms of RMF fields. In summary we have the following reference list which comprises Table 3.

Table 3.
Summary Results
System Level Measurements

1. $T = RMF\text{-}INTVL$

2. $Ci(ALL) = RMF\text{-}IOC(ALL)/RMF\text{-}IOC\text{-}SDC$

3. $Co = RMF\text{-}ET(ALL)$

4. $Ui(ALL) = Bi(ALL)/T$
$= (RMF\text{-}WAIT\text{-}STATE)\text{-}1.00$
$= RMF\text{-}UTIL$

5. $Bi(ALL) = Ui(ALL) * T$
$= RMF\text{-}UTIL * RMF\text{-}INTVL$

6. $Si(ALL) = Bi(ALL)/Ci(ALL)$

$$= \frac{RMF\text{-}UTIL * RMF\text{-}INTVL * RMF\text{-}IOC\text{-}SDC}{RMF\text{-}IOC(ALL)}$$

7. $Xi(ALL) = Ci(ALL)/T$

$$= \frac{RMF\text{-}IOC(ALL)}{RMF\text{-}IOC\text{-}SDC * RMF\text{-}INTVL}$$

8. $Xo = Co/T$

$$= \frac{RMF\text{-}ET(ALL)}{RMF\text{-}INTVL}$$

9. $So = Bi(ALL)/Co$

$$= \frac{(RMF\text{-}UTIL * RMF\text{-}INTVL)}{RMF\text{-}ET(ALL)}$$

10. $Vi(ALL) = Ci(ALL)/Co$

$$= \frac{RMF\text{-}IOC(ALL)}{RMF\text{-}ET(ALL) * RMF\text{-}IOC\text{-}SDC}$$

3. Job Class Level Results

We move our attention from the System Summary Report of RMF to the Performance Group Period Report. Now the objective is to derive, on a job class basis, the CPU results which were derived on a system basis. In addition, we will derive other results which were not derived at the system level. Again, all results are for the CPU device only. Several of these job class parameters follow directly, others require a transformation to a job class level by an auxiliary value, denoted fi(r). Furthermore, the following additional parameters at the job class level will be derived: response time, queue length, multiprogramming levels, and total resident and active time.

In the following notation it is understood that i=CPU. The job class formulas will be derived and examples given. The following is a direct result of the system level formulas. The interested reader may validate these results by figures 1, 2, 3, 4, and 5 at the end of this paper.

$$Ci(TSO1) = RMF\text{-}IOC(TSO1)/RMF\text{-}IOC\text{-}SDC$$
$$= 316990 req/5$$
$$= 63398 req$$

$$Co(TSO1) = RMF\text{-}ET(TSO1)$$
$$= 8825 job$$

$$Vi(TOS1) = Ci(r)/Co$$
$$= 7.1839 req/job$$

Similarly for TSO2, TSO3, TSO-ALL

```
Vcpu(TSO2)            = 118.4565 req/job
Vcpu(TSO3)            = 707.8656 req/job
Vcpu(TSO-ALL)         = 23.0512 req/job
Vcpu(ALL)             = 85.7980 req/job
Vcpu(domain1=batch)   = 2867.87 req/job
```

Two observations to be made here are 1), the increasing values in Vi meets our expectations, and 2), Vi(r) averages across job classes or subclasses have large variances which render them uncharacteristic of the mean = Vi(r=ALL). Continuing, we look at throughput on the class level.

$$Xi(TSO1) = \frac{RMF\text{-}IOC(TSO1)/SDC}{RMF\text{-}INTLV}$$

$$= (316990/5)/3599$$
$$= 17.61 req/sec$$

$$Xo(TSO1) = RMF\text{-}ET(TSO1)/RMF\text{-}INTLV$$
$$= 8825 job/3599 sec$$
$$= 2.45 job/sec$$

Equivalently,

$$Xo(TSO1) = Xi(TSO1)/Vi(TSO1)$$

$$= \frac{17.61 req/sec}{7.1839 req/job}$$

$$= 2.45 \text{ job/sec}$$

Observe that,

$Vi(r)=Xi(r)/Xo(r)$ implies $Vi(r)=Ci/Co$,

hence $Xo(r)=Xi(r)/Vi(r)$
and $Xi(r)=Xo(r)*Vi(r)$.

Note the importance of the parameter Vi(r). The dimension of the result may be obtained by treating the units as algebraic quantites, thusly,

$$Xi(r) = Xo(r)*Vi(r)$$

$$= (job/sec) * (req/job)$$

$$= req/sec \text{ (job units cancel)}$$

No matter how complicated the expression is you may treat the units as algebraic quantities, simplify, and obtain their dimension. I will carry units in most cases for the reader's verification. If you are not confident of the dimension of a result then simplify the expression to units only.

3.1 fi(r)

The measurements Ui(r), Si(r), and Bi(r) require the computation of fi(r) |4| . fi(r) is that "fraction of utilization against device i by job class r" <3>. It is shown below that Ui(r), Si(r) are all functions of Bi(r). We wish to find some coefficient of Bi which will transform Bi into Bi(r). We call this transformational coefficient fi(r). When fi(r) is found we can compute Bi(r). Ui(r) and Si(r) are then found as functions of Bi(r).

In what follows we derive a relationship between fi(r) and device time. We know

$$\sum_{r=1}^{n} Bi(r) = Bi$$

Multiplying by one

$$Bi/Bi \sum_{r=1}^{n} Bi(r) = Bi/Bi^2$$

$$\sum_{r=1}^{n} \frac{Bi(r)}{Bi} * Bi = Bi$$

$$\sum_{r=1}^{n} fi(r)*Bi = Bi$$

$$fi(1)Bi + fi(2)Bi + \ldots + fi(n)Bi = Bi$$

$$Bi(1) + Bi(2) + \ldots + Bi(n) = Bi$$

But this is

$$\sum_{r=1}^{n} Bi(r) = Bi$$

Then

$$fi(r) = \frac{Bi(r)}{Bi}$$

|4| The notation "fi(r)" comes from the paper by Rose cited earlier. Rose uses a SAS data analysis program to compute fi(r) from SMF log records.

If during some measurement interval a device is busy for 10 seconds, and for two of the 10 seconds class r jobs used the device, then

fi(r) = Bi(r)/Bi = 2sec/10sec

= 1/5 = 0.20 OR 20%.

One interpretation of this result is: The rth job class represents 20% utilization of the ith device for that measurement interval. Another interpretation of this result is: The rth job class represents 20% of the busy time of the ith device. The author prefers the second interpretation, the reader may choose the one he feels most comfortable with. Hence, fi(r) is that fraction of the ith device busy time consumed by class r jobs during some measurement interval. fi(r) is intrinsically important, moreover, it is essential in the computation of U, S, and B on a job class level.

Using our definition of fi(r) we now prove that the relationship just derived for fi(r) will transform system level measurements into job class level measurements. The three short proofs are as follows.

```
Ui(r) = fi(r)*Ui
      = (Bi(r)/Bi)(Bi/T)
      = Bi(r)/T
      = Ui(r)
```

```
Si(r) = Bi(r)/Ci(r)
      = (fi(r)*Ui*T)/Ci(r)
      = (fi(r)*Bi)/Ci(r)
      = Bi(r)/Ci(r)
      = Si(r)
```

```
Bi(r) = fi(r)*Bi
      = (Bi(r)/Bi)(Bi)
      = Bi(r)
```

Which is what we wish to show for U, S, and B.

Then, for example, suppose we have computed fi=cpu(r=TSO1).

```
Scpu(TSO1)

      = Bcpu(TSO1)/Ccpu(TSO1)
      = (fcpu(TSO1)*Ucpu*T)/Ccpu(TSO1)
      = (Ucpu(TSO1)*T)/Ccpu(TSO1)
      = Bcpu(TSO1)/Ccpu(TSO1)
      = Scpu(TSO1)
```

which is what we wish to find.

Hence the importance of fi(r). Therefore, coefficient fi(r) transforms system level measurements into job class level measurements. This observation is true for any software or hardware monitor measurement data. Given the system level measurements of RMF, the job class level measurements can be obtained by fi(r): that fraction of busy time of device i consumed by job class r. We will see computations using fi(r), however, first we must derive fi(r) in terms of RMF fields.

3.1.1 fi(r) in terms of RMF notation

This is a dual issue. First we must develop a relationship between the derivation of fi(r) and RMF field notation, and second, we must create an experiment to verify the utility of the RMF equation which represents fi(r).

There are several actions we can take to increase our confidence in the RMF formula which represents fi(r). We can simply use the RMF formula to compute job class performance measures and validate them for their reasonableness. We will do this, but there are other actions we can take which are less qualitative, (not by much) and will serve to further increase our confidence in the utility of the RMF formula for fi(r). First, fi(r) can be bounded between known RMF measurement values, next the sum of the fi(r) must equal one, and the sum of the Bi(r) and Ui(r) should equal the RMF software monitor data for the same performance variables. These results are collected into a verification experiment and shown in Table 4. below. Let us take a closer look at these verification checks.

We wish to bound fi(r) between a high and a low, a maximum and a minimum. The boundary values must be known (measured) quantites. Once we have developed the boundary condition then we can proceed to develop the relationship between fi(r) and RMF field names, and then execute a validation experiment.

In particular, we know that for every job class = r, the Bi(r) for some measurement interval must not exceed the busy time for the system = Ui*T = Bi. Bi then, is chosen as our maximum for some measurement interval, and it is an easily obtained quantity from RMF. We also know that the TCB+SRB time for each "r" job class, for some measurement interval, must also not exceed the busy time for the system. This is also a

known quantity which is easily obtained from RMF. We take this to be our minimum value. This minimum value represents problem program time and a component of TCB/SRB time. The remaining component of TCB/SRB time which is not captured is noted in the Performance Notebook. The interested reader is directed to the chapter, "Investigating the Use of Processor Time" in the Performance Notebook for the details. Therefore, the computed value of fi(r)*Bi for any job class will be, at its worst, as good as the typical computation of TCB+SRB time for the same job class (performance groups); and will be, at its best, very close to the actual value of Bi(r). fi(r) is bound on the low side by TCB+SRB time, and on the high side by the busy time of the system (CPU).

To give this notion precision we state the bounding condition described above.

For all r, i=CPU

$$(TCB+SRB)i(r) <= (fi(r) * Bi) <= Bi$$

Therefore, for any job class, (fi(r) * Bi) computes to a value which is at its worse, as good as a direct RMF computation of TCB+SRB time for that r job class.

This restriction guarantees that each component, fi(r) * Bi, for all r, be bounded.

The formulation of fi(r) in terms of RMF fields follows thusly,

From the IBM Performance Notebook <4>

$$TCB\text{-}TIME(r)=\frac{RMF\text{-}CPU(r)*SRM\text{-}POWER\text{-}FACTOR}{RMF\text{-}CPU\text{-}SDC}$$

$$SRB\text{-}TIME(r)=\frac{RMF\text{-}SRB(r)*SRM\text{-}POWER\text{-}FACTOR}{RMF\text{-}SRB\text{-}SDC}$$

where SRM power factor equals seconds of task execution time per service unit = 0.0078 sec/unit for an IBM 3033.

By the definition of fi(r) and simplifying we have

$$fi(r) = \frac{RMF\text{-}CPU(r)}{RMF\text{-}CPU\text{-}SDC} + \frac{RMF\text{-}SRB(r)}{RMF\text{-}SRB\text{-}SDC}$$

$$\frac{}{}$$

$$\frac{RMF\text{-}CPU(ALL)}{RMF\text{-}CPU\text{-}SDC} + \frac{RMF\text{-}SRB(ALL)}{RMF\text{-}SRB\text{-}SDC}$$

further simplification yields

$$fi(r) = \frac{((RMF\text{-}CPU(r)*RMF\text{-}SRB\text{-}SDC) + (RMF\text{-}SRB(r)*RMF\text{-}CPU\text{-}SDC))}{((RMF\text{-}CPU(ALL)*RMF\text{-}SRB\text{-}SDC) + (RMF\text{-}SRB(ALL)*RMF\text{-}CPU\text{-}SDC))}$$

This above formula is used in Table 4. below to compute Bi(r) and Ui(r), and upon reflection is, as it should be, Bi(r)/Bi, the definition of fi(r). The SRM power factor drops out of the formula. It is a pure time ratio that we are interested in. The user is warned that the formula for fi(r), and all equations which incorporate it, are approximate. RMF does not capture RCT time and the time denoted as "other" in the Performance Notebook. The reader is referred to the IBM Performance Notebook and the discussions of TCB time, in particular the chapter of the Notebook named, "Investigating the Uses of Processor Time".

If your installation does not have RMF extensions you can not capture SRB time. Therefore, drop all references to SRB in the above equation. This, of course, will result in an attendant loss of accuracy. This loss may or may not be important. The analysis must be carried out and an evaluation made to answer this question. If future RMF monitors are able to capture greater amounts of service time (TCB+SRB time) then the above equation will compute to a greater degree of accuracy.

Given the definition of fi(r) it is clear that the sum of each fraction of device busy time consumed by each job class must equal unity. And if the sum of the fi(r)'s is unity then the product

of the fi(r)'s with Bi and Ui will be about equal to the actual values for Bi and Ui directly obtained from RMF. The verification table show these checks to be as we expect them to be.

The results from the verification experiment are given below in Table 4. Due to length (number of performance groups and periods) the table is based on job class = r = domain. Of particular note in Table 4 is fi(r) and its associated measurements for TSO, BATCH and CICS. In performing computations of this nature it is necessary that the event representing job class be non overlapping in time, i.e., the intersection of all such job classes be null. If this condition is not adhered to then the summing process will duplicate time occuring in the intersection of the two events. The results of this overlap would manifest itself in values for the sum of the Bi(r)'s and Ui(r)'s which are greater than the corresponding RMF values, namely, Bi and Ui.

For all r = domain it was found that

$$(TCB+SRB)i(r) \leq fi(r)*Bi \leq Bi$$

meets the bounding condition.

And

$$\sum_{r=1}^{51} fi(r) = 0.9999133 \doteq 1.0$$

And

$$\sum_{r=1}^{51} fi(r)*Bi = 2844.02\text{sec} = 2844.29 \quad \text{RMF's Bi}$$

And

Table 4 fi(r) Verification Experiment

r=Domain	fi(r)	Bi(r)	(TCB+SRB)i(r)	Ui(r)
0=SYS	0.04439	126.25	85.89	0.035081
1=BATCH	0.28356	806.53	548.72	0.22409
2=TSO1,2	0.25147	715.25	468.58	0.19873
3=TSO	0.06334	180.16	122.56	0.05005
4	0.036065	102.58	69.79	0.02850
5-9	0	0	0	0
10=BATCH	0.10224	290.80	197.85	0.08080
11-19	0	0	0	0
20	0.02269	64.52	43.91	0.01793
21-29	0	0	0	0
30=CICS	0.18610	529.32	360.21	0.14707
31-39	0	0	0	0
40	0.003853	10.96	7.45	0.003045
41-50	0	0	0	0
51	0.0062053	17.64	12.01	0.004904
Totals	0.9999133	2844.02	1916.97	0.79020

$$\sum_{r=1}^{51} f_i(r) * U_i = 0.79023$$

$$= 0.79030 \quad \text{RMF's } U_i$$

The strongest condition which this experiment meets is the bounding condition. If any one of the job classes did not meet this condition the RFM formula for fi(r) would be cast into doubt. The other three conditions are accurate, but not strong checks for the RMF formula of fi(r). They do show, however, that each of our chosen job classes (RMF domains) are disjoint sets.

3.2 Results

Now we may proceed in calculating Bi, Si, and Ui at the class level, ie.,

Bi(r)
Si(r)
Ui(r)

mediated by $f_i(r) = B_i(r)/B_i$.

Again, looking at r = class = TSO, i = device = CPU, find Si(r), Bi(r), and Ui(r).

$$S_i(TSO3) = f_i(r)*(B_i/C_i(r))$$

$$= (0.06334)(0.0427458\text{sec/req})$$

$$= 0.002707\text{sec/req}$$

$$B_i(TSO3) = f_i(r)*B_i$$

$$= (0.06334)(2844.29\text{sec})$$

$$= 180\text{sec}$$

$$U_i(TSO3) = f_i(r)*U_i$$

$$= (0.06334)(0.7903)$$

$$= .05 \text{ or } 5\%$$

Similar computations are made for other job classes and job subclasses. Below we give these results in terms of RMF equations. The equations are not complex but they are long, the reader is refered to Table 5., "Summary Results for Job Class Level Measurements" for such RMF equations. The reader is invited to compute other job class results and check them against his intuition. Table 4. also represents a number of job class results the reader may validate. In the following results we use the short form queueing equation or, in some cases, the numerical result only. For the same class = TSO3, we now compute service time per job or interaction.

$S_i(TSO3) = 0.002707\text{sec/job}$
$V_i(TSO3) = 707.8656\text{req/job}$

Where TSO job=Interaction

and service per job for r=TSO3 is

$$V_i(TSO3) * S_i(TSO3) = (708\text{req/sec} * 0.002707\text{sec/req})$$

$$= 1.9166\text{sec/job}$$

If we knew the queueing time Qcpu(TSO3) for TSO3 at the CPU we could compute the response time at the CPU for TSO3, for a request, and for a job.

$R_i(r) = S_i(r)+Q_i(r)$

Qi(r) will be computed, but only after Ri(r), the response time at device = i for job class = r is derived and computed. RMF data yields Ri(r) with less manipulations than Qi(r), thus we strive to derive Ri(r) first. The field RMF-ATT(TSO3) has a value of one minute and thirteen seconds. This is not the response time at the CPU, this is the response time at the system; the sum of the response times at each device the job visits in its journey through the system. In shorthand, for r = TSO3, we have

$$\text{RMF-ATT}(r) = \sum_{i=1}^{K} V_i(r)*R_i(r).$$

It is of interest to compare the service time per request for TSO1:

$S_i(TSO1) = 0.1275471*0.0448727 \text{ sec/req}$
$= 0.005723 \text{ sec/req}$

with other job classes. If we do this we find that the mean quantum of service per request is similar notwithstanding the job class. In the limiting condition, measurements suggest that Si(r) converges for all r. Therefore, notwithstanding the specific job characteristics, jobs tend to demand a similar quantum of service. This quantum may be interpreted as the inter-I/O request time <5> [5]. If this observation is true then in what way is it helpful? The author suggests that this

condition may be used as another (more accurate, and in terms of RMF data, easier to obtain) means of computing fi(r), and hence, job class level measurements. Suppose that in the limit Si(r)-->I, then we can use Ci to obtain service time. That is, the product of the completion count (Ci) of jobs at some ith device, and the limiting quantum of time (I) spent per completion at that device, is the total service time consumed by all jobs at that device. The limiting parameter is time, and we let time approach the inter-I/O time. We may view the inter-I/O time as the execution time required to process the mean number of instructions between I/O events (interrupts).

Ci(r)*I is the total service time consumed by all r class jobs at the ith device. Ci(ALL)*I is the total service time consumed by all jobs at the ith device. Then

$$\frac{Ci(r)*I}{Ci(ALL)*I}$$

is Bi(r)/Bi(ALL). But this is fi(r).

The derivation goes:

Suppose Lim Si(r)--->I. Then

$$Ci(r)*I = Ci(r)*Si(r)$$
$$Ci(ALL)*I = Ci(ALL)*Si(ALL)$$

and

$$Si(r) = Bi(r)/Ci(r)$$
$$Si(ALL) = Bi(ALL)/Ci(ALL)$$

Simplifying

$$\frac{Ci(r)*I}{Ci(ALL)*I} = \frac{Ci(r)*(Bi(r)/Ci(r))}{Ci(ALL)*(Bi(ALL)/Ci(ALL))}$$

$$= \frac{Bi(r)}{Bi(ALL)}$$

$$= fi(r)$$

which is what we wish to show.

Then fi(r) simply reduces to the formula

$$fi(r) = \frac{Ci(r)}{Ci(ALL)}$$

$$= \frac{RMF\text{-}IOC(r)/RMF\text{-}IOC\text{-}SDC}{RMF\text{-}IOC(ALL)/RMF\text{-}IOC\text{-}SDC}$$

$$= \frac{RMF\text{-}IOC(r)}{RMF\text{-}IOC(ALL)}$$

Please note that the above derivation makes the assumption that Si(r) = (approximates) Si(ALL). This, of course, is tantamount to saying that Lim Si(r)--->I, which is our leading assumption. This is a significant simplification of the first formula for fi(r). The result is somewhat counter-intuitive. It claims that fi(r) is the ratio of I/O operations of the rth job class and all job classes.

Initial analysis shows that this method of computing fi(r) breaks the bounding condition for some job classes, and hence, can not be used. It is the author's feeling that this value for fi(r), viz., fi(r)=Ci(r)/Ci(ALL), offers a more simple and better (less dependencies) solution to fi(r) than the first derivation of fi(r). The problem that fi(r) has in passing the bounding condition may be in RMF's inability to capture I/O (paging, swapping) EXCP's done in the system key. This unhappy state of affairs does not, however, stop us from using the formula fi(r)=Ci(r)/Ci(ALL) for other software/hardware monitor data. Alternatively, for job classes with distinctly different resource demand (workload) characteristics, and service time quantums may have statistically wide variances. This however, the author feels, is not the case, and, in fact, when viewed at a low enough level,

---

|5| Boyse and Warn describe service quantum in terms of inter I/O request time. This article is further referenced in: Arnold O. Allen's, Probability, Statistics, and Queueing Theory W/Computer Science Applications; Kleinrock's, Queueing Systems dual volume I, II; Hisashi Kobayashi's Modeling and Analysis: An Introduction to System Performance Evaluation Methodology; IBM Systems Programming Series, and Ferrari's, Computer System Performance Evaluation. For those analysts who wish to see how the formulas (not the theory) of queueing theory are applied, and what results are obtainable, this article is a good place to start. The system modeled at General Motors by Boyse and Warn was a system which ran only one job class; a uni-class system. However, using the RMF Equations we can still use this model, we merely model each job class independently.

all jobs have statistically similar time dependent resource demands. To further investigate this is not within the bounds of this paper. The decision was made, because the second formulation of fi(r) did not pass the bounding condition for some job classes, to not use it in computing job class level measurements.

4. Multiprogramming Level MPL(r) |6|

Suppose T = 100 sec, and for T measurement interval, Wi = 750 seconds of active and resident time accumulated at device i by all jobs in the system. Then

750 sec/100 sec = 7.5 = MPL = Wi/T.

MPL may be derived from RMF as follows ?

RMF-ISPS =

$$\frac{\text{TOTAL SERVICE ACCUMULATED IN INTERVAL}}{\text{LENGTH OF MEASUREMENT INTERVAL}}$$

RMF-AAR =

$$\frac{\text{TOTAL SERVICE ACCUMULATED IN INTERVAL}}{\text{TOTAL RESIDENT + ACTIVE TIME}}$$

of all jobs, and

RMF-AAR/RMF-ISPS =

$$\frac{\text{TOTAL RESIDENT + ACTIVE TIME OF ALL JOBS}}{\text{LENGTH OF MEASUREMENT INTERVAL}}$$

But this equals Wi/T, hence

$$\text{MPL} = \frac{\text{RMF-AAR}}{\text{RMF-ISPS}}$$

This can be verified directly. MPL for the system should approximate the RMF-IN field. And the MPL for any non-swapable task (performance group) should be equal to one. Performing these computations,

MPL(ALL) = RMF-AAR(ALL)/RMF-ISPS(ALL)
= 21.838
and
RMF-IN = 20.9.

Further

MPL(CICS) = RMF-AAR(CICS)/RMF-ISPS(CICS)
= 223/223
= 1 as predicted.

Continuing. MPL(r), r = TSO, BATCH

MPL(TSO) = 4.3
MPL(BATCH) = 4.85

Similar MPL results are computable for other performance groups.

5. Ri(r) Wi(r) Ni(r) Qi(r)

Lastly we consider Ri(r), Wi(r), Ni(r), and Qi(r) where,

Ri(r) = Response time of class r jobs per request (not per job) at device i.

Wi(r) = Total resident and active time (known as job seconds) accumulated by class r jobs at device i.

Ni(r) = queue length of class r jobs at device i.

Qi(r) = queueing time of class r jobs at device i.

Elementary queueing theory tells us Ri=Wi/Ci. If 10 job requests Ci accumulate 125 job seconds Wi at device i, then the mean response time per request is 125 sec/10 req = 12.5 sec/req = Wi/Ci = Ri seconds per request.

We also have by MPL analysis,

Wi(r)/T = RMF-AAR(r)/RMF-ISPS(r)

$$= \frac{\text{TOT RES AND ACT TIME OF ALL JOBS(r)}}{\text{LENGTH OF MEASUREMENT INTERVAL}}$$

implies

Wi(r) = (RMF-AAR(r)/RMF-ISPS(r)) * T ,

where T=RMF-INTVL

|6| For this computation of MPL I use Mr. Levy's RMF formula. For any modeling effort MPL is used as an input parameter to the model. It is our intent to derive MPL(r) from RMF fields and use this as a building block for capturing Ri(r) and Qi(r) from RMF. See also <6>.

Then

$$R_i(r) = W_i(r)/C_i(r)$$

$$= \frac{(RMF\text{-}AAR(r)/RMF\text{-}ISPS(r))*RMF\text{-}INTVL}{C_i(r)}$$

where

$$C_i = RMF\text{-}IOC(r)/RMF\text{-}IOC\text{-}SDC$$

Then

$$R_i(r) = \frac{(RMF\text{-}AAR(r)/RMF\text{-}ISPS(r))/RMF\text{-}INTVL}{RMF\text{-}IOC(r)/RMF\text{-}IOC\text{-}SDC}$$

and simplfying

$$R_i(r) = \frac{RMF\text{-}AAR(r)*RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTVL}{RMF\text{-}ISPS(r)*RMF\text{-}IOC(r)}$$

By Little's theorem

$N_i = W_i/T$ and $R_i = W_i/C_i$

impliies

$W_i = R_i * C_i$ (It is now clear that W is total active plus resident time.)

substituting in the equation for Ni

$$N_i = (R_i*C_i)/T$$
$$= R_i*X_i$$

The job class level result is

$$N_i(r) = R_i(r) * X_i(r)$$

Then in terms of RMF fields

$$N_i(r) = \left[\frac{RMF\text{-}AAR(r)*RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTVL}{RMF\text{-}ISPS(r)*RMF\text{-}IOC(r)}\right] * \left[\frac{RMF\text{-}IOC(r)}{RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTLV}\right]$$

simplifying

$$N_i(r) = \frac{RMF\text{-}AAR(r)}{RMF\text{-}ISPS(r)}$$

$$= MPL(r)$$ (Which is what we expect to happen.)

We have reached MPL(r) via two routes. One, through Mr. Levy's formulation. The other, we have reached independently, through queueing primitives. In queueing terms MPL is the queue length of in-core jobs waiting execution, this we call the mean CPU queue length.

We also note that response time is related to N by the relationship,

$$R_i(r) = N_i(r)/X_i(r) = MPL(r)/X_i(r)$$

hence the sensitivity of response time to MPL loads.

Now lets apply these results to RMF data. As in past computations, i=CPU and TSO=TSO1, TSO2, and TSO3.

$W_i(BATCH) = 2991.36$ sec
$W_i(TSO) = 7771.23$ sec,

NOTE

Wi calculation must be done for each performance group period independently and the results summed. Do not use the RMF-ALL field of a performance group.

$$R_i(TSO3) = \frac{(192)(5)(3599)sec}{(210)(332697)req}$$

$$= 0.049452 \text{ sec/req}$$

$$V_i(r) * R_i(r)$$
$$= (req/job)(sec/req)$$
$$= sec/job \quad \text{(canceling units)}$$
$$= \text{Response time per job}$$

then

$$V_i(TSO3) * R_i(TSO3)$$
$$= (707.8656 \text{ req/job})(0.049452 \text{ sec/req})$$
$$= 35.0054 \text{ sec/job}$$

In words, the response time of TSO3 at the CPU is 35.0054 seconds per job. The remainder of the response time, RMF-ATT(TSO3), is spent at other devices. This time at other devices has two components, sevice time at the device and queueing time at the device. For TSO3 this time is computable as a difference,

For r = TSO3 and VR = Vcpu(r) * Rcpu(r)

$$\sum_{i=1}^{K} Vi(r)*Ri(r) = VR + \sum_{i=1}^{K-1} Vi(r)*Ri(r)$$

$$\sum_{i=1}^{K-1} Vi(r)*Ri(r) = \sum_{i=1}^{K} Vi(r)*Ri(r) - VR$$

$$\sum_{i=1}^{K-1} Vi(r)*Ri(r) = RMF\text{-}ATT(r) - VR$$

$$\sum_{i=1}^{K-1} Vi(r)*Ri(r) = 77 \text{ sec} - 35 \text{ sec}$$

$$= 38 \text{ sec.}$$

In words, on the average a TSO3 job class job has a 73 sec. response time. 35 of the 73 seconds are spent waiting for or executing on the CPU, and the remaining 38 seconds are spent at other devices in the system.

If we have service time and response time then queueing time for device i is also computable.

Ri(r) = Si(r)+Qi(r)

whence

Qi(r) = Ri(r)-Si(r) per request.

In terms of RMF fields,

$$Qi(r) = \frac{RMF\text{-}AAR(r)*RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTVL}{RMF\text{-}ISPS(r)*RMF\text{-}IOC(r)} - (fi(r)*Ui*T)/Ci(r)$$

Multiplying the equation for Qi(r) by Vi(r) results in the queueing time per job. I leave the simplification in terms of RMF fields to the reader; fi(r), Si, and Vi(r) are all given above in terms of RMF field names.

Let us look at an example of applying the equation for Q to RMF data. We will still focus on job class = r = TSO3. Since the computations for R and S for r = TSO3 are available we will use the form of the equation Qi(r) = Ri(r)-Si(r).

Then

Qi(r) = 35.0054 sec/job - 1.9163 sec/job
= 33.0891 sec/job

Or using the RMF equation for Q above, and multiplying by Vi(r) to obtain the per job value, we have,

$$Qi(r) = \frac{192*5*3599\text{sec}}{210*332697} * \frac{290041*0.1+359*10}{4446063*0.1+6460*10}$$

$$\frac{0.7903*5*3599\text{sec}}{332697\text{req}} * \frac{(332697/5)\text{req}}{94\text{jobs}}$$

= (0.0567249 sec) -
(0.0640096*0.0427459) *
((332697 req/5)/94 job))

= 33.0686 sec/job

We first note that both methods yield consistent results. This value for Q has two interpretations. The first interpretation claims that Q merely confirms our experience and observations, namely, TSO3 jobs wait a lot. TSO jobs (interaction = job) which fall in period three have enlongated response time. And this is perfectly predictable, indeed, the RMF driver parameters are often specified to force this condition. Therefore, the calculation of a high Q comes as no surprise. Alternatively, we may want to claim that this augurs ill. Nearly all the time of TSO3 jobs are spent waiting to execute. At this point the modeling study is secondary. We should 1) verify Qi(r) by computing it for other job classes within TSO, and 2) if "1)" checks out we should immediately determine if this is true for other RMF intervals. If this is true we should find out why and correct the situation. The last step will bring us back in the loop of performing the modeling study. Many problem like this, problems that go undetected and manifest themselves as performance service level problems, are dredged up while performing a modeling study.

These festering performance problems can be more to the issue than the modeling study itself.

Similar results as above are obtainable for disks, I/O paths, controllers, drums, and core. <7> The techniques differ in obtaining these other results, and in most instances, employ significantly more advanced mathematical methods. Before the decision is made to invest the time and people in building and maintaining data reduction programs to capture these results, the other alternative, a first order approximate analytic approach, should also be considered. As your needs, requirements, and operating system software changes, then so must the set of data reduction programs change.

6. Summary and Concluding Remarks

The relevant model parameters for modeling the CPU at the job class level have been obtained from RMF. In achieving this goal we have developed several intermediate results (Wi(r) and Qi(r)), and also, the output equations, (Ri(r), Ui(r), Xi(r)) by which the model is verified. All such equations, both primary and intermediary, are given in terms of RMF field names. A summary of these results are given below in Table 5. As in past formulas it is understood that i = CPU, and not any device.

6.1 Remarks

Using relatively simple methods we were able to quickly obtain, from RMF, job class level input/output parameters to CPU models of multi class machines. Similar but less tractable methods yield results for I/O paths, storage devices, controllers, and first level store. In this paper we concentrated on the CPU and collecting only those CPU measurements which are useful in mathematical modeling and performance/capacity analysis. These same simple methods may be applied to any data, from software or hardware monitors, to collect similar results. The results themselves will be accurate in proportion to the integrity of the data. The value of such work lies as much in the insight into system behavior as in the results themselves - perhaps more so in the insight.

The above tables of RMF equations may be used by the analyst for performance analysis, capacity planning, and problem determination. For ease of use the table equation's notation is keyed to the appropriate RMF field names. The user is warned that these formulas require field testing to fully bound their capabilities and restrictions. The author invites comments from the users of the RMF equations. The equations are not complex but they are long, therefore, it is advised that they be programmed for a card programmable calculator (like the Texas Instruments TI-59) or an office based micro/mini computer.

I would like to thank Bob Meister for his patience and repeated editing of this paper, and for the use of his DEC computer-interfaced IBM Selectric typewriter, which was used in conjunction with RUNOFF to generate this paper.

References

<1> Buzen and Denning, The Operational Analysis of Queueing Network Models, ACM Computing Surveys, Vol. 10 No. 3, September 1978 p. 234 -235.

<2> C. A. Rose, A measurement Proceedure For Queueing Networks Of Computing Systems, ACM Computing Surveys, Sept 78. For Ci in terms of I/O OPs see this paper.

<3> Allan I. Levy, An Introduction to Pratical Operational Analysis: An MVS Perspective, CMG XI Proeedings, Boston 1980, p. 208-214. Also from IBM supplied formulas.

<4> IBM OS/VS2 MVS Performance Notebook, GC28-0886-0.

<5> Boyse and Warn, A Straightforward Model for Computer Performance Prediction, ACM Computing Surveys Vol. 7, No. 2 June 1975.

<6> IBiD. <3>.

<7> Y. Bard, A Model of Shared DASD and Multipathing, IBM Cambridge Scientific Center, Communications of the Associations of Computing Machinery, Vol. 23 No. 10, Oct 80. Also see, same author, Task Queueing in Auxiliary Storage Devices with Rotational Position Sensing, IBM Cambridge Scientific Center Report G320-2070, MAR 75.

Table 5. Summary Results for Job Class Level Measurements

1. $$fi(r) = \frac{Bi(r)}{Bi}$$

$$= \frac{(RMF\text{-}CPU(r)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(r)*RMF\text{-}CPU\text{-}SDC)}{(RMF\text{-}CPU(ALL)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(ALL)*RMF\text{-}CPU\text{-}SDC)}$$

2. $$Ci(r) = \frac{RMF\text{-}IOC(r)}{RMF\text{-}IOC\text{-}SDC}$$

3. $$Co(r) = RMF\text{-}ET(r)$$

4. $$Vi(r) = \frac{Ci(r)}{Co}$$

$$= \frac{RMF\text{-}IOC(r)}{RMF\text{-}IOC\text{-}SDC*RMF\text{-}ET(r)}$$

5. $$Xi(r) = \frac{Ci(r)}{T}$$

$$= \frac{RMF\text{-}IOC(r)}{RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTVL}$$

6. $$Si(r) = \frac{fi(r)*Ui*T}{Ci(r)}$$

$$= \frac{(RMF\text{-}CPU(r)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(r)*RMF\text{-}CPU\text{-}SDC)}{(RMF\text{-}CPU(ALL)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(ALL)*RMF\text{-}CPU\text{-}SDC)}$$

$$* (RMF\text{-}UTIL*RMF\text{-}INTLV)$$

$$/ (RMF\text{-}IOC(r)/RMF\text{-}IOC\text{-}SDC)$$

7. $$Bi(r) = fi(r)*Bi$$

$$= \frac{(RMF\text{-}CPU(r)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(r)*RMF\text{-}CPU\text{-}SDC)}{(RMF\text{-}CPU(ALL)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(ALL)*RMF\text{-}CPU\text{-}SDC)}$$

$$* (RMF\text{-}UTIL*RMF\text{-}INTVL)$$

8. $$Ui(r) = fi(r)*Ui$$

$$= \frac{(RMF\text{-}CPU(r)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(r)*RMF\text{-}CPU\text{-}SDC)}{(RMF\text{-}CPU(ALL)*RMF\text{-}SRB\text{-}SDC)+(RMF\text{-}SRB(ALL)*RMF\text{-}CPU\text{-}SDC)} * RMF\text{-}UTIL$$

9. $$MPL(r) = \frac{Wi(r)}{T}$$

$$= \frac{RMF\text{-}AAR(r)}{RMF\text{-}ISPS(r)}$$

10. $$Ni(r) = MPL(r) = Xi(r)*Ri(r)$$

11. $$Wi(r) = Ni(r)*T$$

$$= \frac{RMF\text{-}AAR(r)*RMF\text{-}INTVL}{RMF\text{-}ISPS(r)}$$

12. $$Ri(r) = \frac{Wi(r)}{Ci(r)}$$

$$= \frac{RMF\text{-}AAR(r)*RMF\text{-}INTVL*RMF\text{-}IOC\text{-}SDC}{RMF\text{-}ISPS(r)*RMF\text{-}IOC(r)}$$

13. $$Qi(r) = Ri(r)-Si(r)$$

$$= \frac{RMF\text{-}AAR(r)*RMF\text{-}IOC\text{-}SDC*RMF\text{-}INTVL}{RMF\text{-}ISPS(r)*RMF\text{-}IOC(r)} - ((fi(r)*Ui*T)/Ci(r))$$

NOTE

For X, S, R, and Q, the above equations compute the "per request" values. To compute the "per job" values we must form the product of these equations with the visit ratio, V.

SERVICE LEVEL MANAGEMENT THROUGH WORKLOAD SCHEDULING

David G. Halbig

U.S. Senate Computer Center
Washington, D.C. 20510

The author describes techniques for managing batch workloads in the IBM environment using controls other than the MVS System Resource Manager (SRM). Problems of DSNAME ENQUEUE conflict management, tape and disk space over-allocation, and resource-sensitive job scheduling are addressed. Results of implementing workload scheduling at the author's installation are presented.

Keywords: MVS SRM; workload scheduling; batch; service levels; resource-sensitive job scheduling; DSNAME ENQUEUE conflict management; SMF exits.

1. Introduction

The purpose of this paper is to describe the motivations for, and the implementation of, a batch workload service level management system. The ability (and inability) of the MVS System Resource Manager (SRM) to manage batch workloads is discussed, as is the list of requirements for a batch workload manager. The implementation of an alternative to the SRM is presented in 4 parts: interfaces, scheduler methodology, user reaction, and quantitative results.

2. MVS System Resource Manager

In the IBM MVS environment, the System Resource Manager (SRM) is given the responsibility of controlling service levels (turnaround time, response time) and system load (CPU busy, page-in rate, etc.) /1/. In the case of batch workloads, the SRM controls only jobs which have already been initiated. The primary function used by the SRM to exercise control is swapping, or removing a job (physically or logically) from main memory, thus denying access to certain system resources (main memory, CPU cycles) /2/. Other resources already assigned to a swapped-out job, however, are not released for use by other work in the system.

Such resources include tape drives, temporary (scratch) disk space, data sets, and the initiator in which the job is running. If these resources are in short supply, or have been over-committed by large individual resource holders, swapping has the effect of delaying other work not yet swapped out. Additionally, the swap function against batch work requires significant system resources, dictating a minimum of batch swap activity /3,4/. Finally, the SRM is a reactive manager, taking action only after a bottleneck or potential service level problem has been detected. Taken by itself, this characteristic is not consequential, but if workloads consisting of long as well as short running jobs are considered, problems arise. Specifically, if a long-running job is is permitted to initiate and only then is swapped out, the initiator is unavailable to other short-running jobs. Admittedly, production workloads dictate some long-running jobs must initiated during prime shift, but the SRM has no

means to avoid their initiation in anticipation of subsequent delays for other batch work.

The SRM, then, has three shortcomings as a batch workload manager:

1) The SRM scope of control excludes resources whose overcommitment has an adverse effect on batch job throughput (tape drives, DASD scratch space, batch initiators, data set names). In the case of each of these resources, the SRM can detect an overcommitment and react to it (detected wait swap), but cannot explicitly obtain the resource from the resource holder nor release other, already-obtained, resources from the resource requestor;

2) The SRM cannot anticipate problems and solve them through avoidance (preventative management), but rather waits for undesireable conditions before taking action (reactive management);

3) The primary function of control for the SRM is swapping, which can aggravate the problem the SRM is attempting to solve (e.g. I/O contention).

3. Workload Scheduling Requirements

The objectives of the SRM, namely meeting of site-defined service levels and optimization of system resources under the service level definition, are nonetheless valid. The problem in the case of batch workload management involves the methods used to meet the objectives. The above shortcomings of the SRM led to an investigation of other methods to manage batch workloads.

The first several design constraints were derivatives of the SRM investigation, namely:

1) The scope of control for a workload manager must be expanded to competently handle tape drive, temporary DASD space, and DSNAME ENQUEUE conflicts problems;

2) The workload manager control function must be preventative rather than reactive, attempting more to avoid workload management problems rather than reacting to them once they occur;

3) The control function must not itself be a source of significant overhead or service level degradation.

Another design constraint addressed the lack of performance incentives available to the user community. Repeated investigations of specific batch job (and total system) performance problems uncovered inappropriate use of some system resource (e.g. small blocking factors, small files on tape rather than disk, testing with too much data, gross overallocation of temporary disk files, illegal specification of the PERFORM= keyword, repetative OPENs of a file during a job, etc). In each case, no amount of SYSTEM-level tuning would have had much impact; the solution involved instead the correction of an inappropriate user practice. Incentives were missing which would gradually correct the bad practices without requiring constant vigilance of and negotiation with the user community.

The final design constraint was to avoid as much future maintenance as possible. Thus, direct modification of IBM routines was not an option.

3.1 Interfaces

The reactive nature of the SRM implied it was gaining control too late in the life cycle of a job. To be effective in the batch environment required the ability to PREVENT initiation of work based on a set of rules. JES job classes and initiator job class strings were well-suited to this function, but only if the job classes could be explicitly associated with the resources to be managed (i.e.

tape drives, disk space, etc.). If JES job classes were used, the information available to the workload manager would be limited to only what was known or could be discovered about jobs on the JES input queue.

The MVS JES2 architecture, unlike its predecessor MVT HASP, provides significant information about jobs awaiting execution. Under MVS JES2, the Job Control Language (JCL) of a batch job is (usually) expanded and converted into Internal Text immediately after being submitted to the system. Internal Text, a parsed-string language, is then written to SPOOL to await the next phase of the job, namely initiation. Most importantly, the Internal Text contains the expansion from JCL Procedure Libraries, virtually guaranteeing the completeness of necessary information about the job. Finally, since Internal Text is an integral part of NJE, as well as MVS JES2, architecture, the stability of any function based on it is assured /5/.

The design constraint which required no direct modifications to IBM code necessarily forced any control function into SMF (System Management Facility) exits. The exits are IBM-authorized points of departure from the MVS and JES environments /6/. The exits used by the workload scheduler include IEFUJV, IEFUJI, IEFUSI, IEFU83, IEFUSO, IEFUTL, and IEFACTRT. It is significant to note IBM's sensitivity to changes in the SMF interfaces as shown by the remarkable stability of these interfaces.

In fact, the workload scheduling functions eventually implemented as SMF exits required virtually no coding changes from MVS 3.7 up to MVS/SP1.3. Because of the massive design changes in Enqueue management with MVS/SP1.3, however, the workload scheduler did require significant rework for the MVS/SP1.3 environment.

3.2 Scheduler Methodology

The workload scheduler is based on early and accurate knowledge of the maximum resources a batch job will use. JCL and its derivative, Internal Text, contain significant information on a batch job's estimated transit time in an initiator, as well as the potential impact the job will have on other, already initiated, work. Specifically, the TIME= keyword on the job card defines the maximum amount of CPU time the job may use, as the REGION= and VIRT= keywords define the maximum amount of memory the job mayuse, and whether the memory is virtual or real.

Similarly, individual DD statements define the maximum concurrent number of tape drives and amount of temporary disk space a job will use. The PERFORM= keyword defines the rules of competition between the job and other already-initiated jobs for CPU time, memory, and I/O.

In all, some 12 characteristics of a batch job's future behavior are available from the JCL. These characteristics comprise the "signature" of the batch job. The scheduler ASSIGNS a job class based on the signature using a lookup table of resource limits by class (see Figure 1). The JES initiator class strings are then structured to meet two objectives:

1) Favor low-resource use jobs over high-resource use jobs by placement of the class in the initiator string; furthermore, the initiator strings are modified automatically or by the operator several times during the day to permit or deny the initiation of very high resource use jobs.

2) Prevent overcommitting certain resources, accomplished by distribution of classes among the initiators. An example will clarify this: assume there are only two types of work in the system: those which required a maximum of two tape drives per step, and those which required none (non-setup). Assume the non-setup work was assigned to class A and the other work was assigned to class B. If the initiators were constructed as shown below, it would not be possible to initiate

CLASS LIMITS	A	B	C	D	E	F	G	H	I	J	K	L	Z	T	P	W	X	Y	Z
CPU TIME (MIN)	.5	2	10	60	.5	2	10	60	.5	2	10	60	60	480	480	300	300	300	--
6250/1600 BPI	0	0	0	0	1	1	1	1	2	2	2	2	7	1	1	0	1	2	--
1600/800 BPI	0	0	0	0	1	1	1	1	1	1	1	1	2	1	1	0	1	1	--
K-LINES	10	10	100	1000	10	100	1000	1000	10	100	1000	1000	1000	100	100	1000	1000	1000	--
REAL*	(25)	(50)	(200)	(500)	(25)	(200)	(200)	(500)	(25)	(25)	(400)	(400)	(400)	(25)	(25)	(500)	(500)	(400)	--
DASD	12K	24K	98K	240K	12K	98K	98K	240K	12K	12K	197K	197K	197K	12K	12K	240K	240K	197K	--
VIO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	--
DASD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	--
PERFORM=	2	2	2	2	2	2	2	2	2	2	2	2	2	3	4	2	2	2	--
PRTY=	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	--
DUA	0	0	0	0	0	0	0	0	0	0	0	0	0	99	99	0	0	0	--
T/P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	--
U/R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	--
REGION=	800 K	1500 K	1500 K	1500 K	800 K	1500 K	1500 K	1500 K	800 K	1500 K	1500 K	1500 K	4000 K	4000 K	4000 K	1500 K	1500 K	1500 K	-- --
AUTO-HOLD	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO	YES	NO	NO	NO	NO	NO	YES
TURN-AROUND TARGET**	30 MIN	60 MIN	2 HR	24 HR	30 MIN	60 MIN	2 HR	24 HR	45 MIN	90 MIN	3 HRS	24 HRS	48 HRS	N/A	N/A	8 HR	48 HR	48 HR	N/A --

* Given as the number of 1K-blocks (e.g. Class A = 12,000 1K-blocks) and parenthetically as the number of equivalent 3350 cylinders (e.g. Class A = (25)).

** "Turnaround" is defined as the time between when a job is read in and when the last page of output is printed.

Figure 1. Job Scheduler Resource Limit Table

more than three class B jobs concurrently, and thus not use more than six tape drives concurrently (2 X 3).

I1 = A
I2 = AB
I3 = AB
I4 = AB

The concept for temporary disk space and (roughly) real memory overcommitment is similar.

It was noted earlier that the scheduler ASSIGNS the job class; this technique is a departure from similar systems which validate the user's explicit specification of job class. Several considerations forced this design. First, as fine-tuning of the workload progressed, the number of job classes mushroomed (there are currently 19). It was impractical to expect the user community to remember precisely what attributes were associated with what class. Instead, it was found to be sufficient that users understood the OBJECTIVES of the scheduler, and that any tuning effort on their part that could be reflected in the JCL (lower CPU time, smaller disk space requirements, lower concurrent tape drive use) would be rewarded with better turnaround. Second, as the machine room floor changed (number of tape drives, temporary disk packs, available CPU power, new applications) the scheduler resource limit table underwent dramatic change. The administrative difficulties of changing job class specifications on user JCL would have been immense.

Of all the characteristics of future behavior predictable through JCL, DSNAME enqueue conflicts could neither be easily predicted or prevented.

Ultimately, the scheduler was redesigned to permit the conflict occur, but then intercept it and place the job requesting the DSNAME(victim) back on the input queue. Under the current design, if the victim is a batch job and the DSNAME holder (conflictor) is also a batch job, the victim job is held on the input queue until the conflictor ends; the victim job is then released.

If conflictor is a TSO session, an informational message is sent to the TSO user and the victim job is placed back on the input queue; when the TSO session ends (LOGOFF), the victim job is automatically released. If the victim is a system task or TSO session attempting to sign on, normal MVS handling of enqueue conflicts takes control.

To review, the operation of the scheduler has two components. First, jobs are assigned into job classes based on the jobs' resource signatures. The JES initiator class strings are then constructed to favor jobs based on estimated resource use and to prevent overcommitment of certain static resources (tape drives, temporary disk space). Second, DSNAME enqueue conflict management reduces one of the major causes of cross-initiator dependencies. The result is a system which applies queueing theory concepts (multi-server prioritized queueing, reduced covariance among servers) to batch workload management.

3.3 User Reaction

Because the scheduler has been implemented at several installations, it has been possible to uncover patterns of user reaction. If the site is already using some form of resource-sensitive job scheduling, the scheduler is well-received, since it usually expands the scope of control of the previous scheduling system. If, however, the site has initiators dedicated to groups within an organization (as our site had), the resistance can be fierce. The difficult task of describing the cost of ignoring, or finding the wrong culprit for, resource conflicts between users is not to be underestimated. Some techniques have been found to ease the path, nonetheless. Full implementation of the scheduler is best delayed until the workload is at a seasonal ebb. When utilization levels are low, practically any scheduling scheme will work (even user-dedicated initiators). Second, if members of the community are mathematically inclined, the performance differences between a multi-server initiator structure and multiple single-server initiators can be explained /7/. Third, installing the scheduler without at first turning on the class scheduling facility can be helpful.

The resource signature for the job is displayed on the JES Job Log, along with the job class which otherwise would have been assigned (see Figure 2). This method permits users to modify their jobs in anticipation of full implementation. This recommendation emphasizes that major improvements in workloads are often the result of dramatic changes in USER (repeat USER) practices. Fourth, classes explaining the objectives of the scheduler and potential benefits resulting from its implementation reduces the chances for insurrection at the working level.

The sharp reactions to the scheduler normally occur well before its actual implementation. The counter-arguments noted above have been useful only to dull the prospective loss of "personalized" initiators. Of equal interest, however, are the user reactions, both immediately after full implementation, and later, in a steady-state environment.

Needless to say, if the site has no prior experience with resource-sensitive job scheduling, the first two weeks under full implementation are hectic. Users previously too "busy" to attend scheduler workshops or read the implementation newsletters create considerable work for the performance management group. The overwhelming majority of requests involve answering the question "How do I get good turnaround?", and assisting in (sometimes massive but always overdue) changes to JCL and programs to qualify for fast-turnaround job classes. During the initial weeks a dramatic change comes over the relationship between the performance management group and the user community: performance people (usually) needed no longer to seek out tuning targets and engage in onesy-twosy with recalcitrant users. In the case of the author's site, performance advice was sought out readily AND IMPLEMENTED. Progressively better- attended performance workshops attested to the new sensitivity to performance issues.

Shortly after full implementation, however, a major deficiency appeared:

The users had incentive to address long-standing performance problems, but the information necessary to address the problems was lacking or was available only by requesting special reports from the performance group. Missing was information useful in tuning and readily available to the users. As a result, the step termination statistics were greatly expanded to provide most of this information (see Figure 3). These statistics, for example, allowed the users to target high-use files with poor blocking factors or option codes or high open-close counts. Similarly, temporary disk space allocations could be compared with the amount of disk space actually used.

3.4 Quantitative Results

The scheduler was designed to improve batch throughput by several methods:

1) Multi-server initiator structure. Said another way, each initiator (server) is set up to accept multiple classes. This differs from the multiple single-server initiator structure associated with "personalized" initiators (single class per initiator). As utilization levels increase (increased levels of initiator busy), the turnaround time differences between the two structures widens markedly /8/.

2) Reduced covariance or dependencies between initiated jobs. Said another way, interference of one job with another is minimized as much as possible. Preventing DSNAME enqueue conflicts and tape and disk space allocation recovery from blocking initiators (servers) are examples of this.

3) Prioritizing work on the input queue based on estimated initiator transit time (service time). Jobs with low estimated initiator transit times are

```
                                    J E S 2   J O B   L O G

06.30.30 JOB   233  IEF196I IEF237I 100  ALLOCATED TO SYS00003
06.30.30 JOB   233  IEF196I IEF237I 1C2  ALLOCATED TO SYS00004
06.30.31 JOB   233                      LOAD NEW TAPE       750-CP-LT-P            PRTY=00
                   JOBNAME  *  PG# CPU-TIME KCORE KLNES 6250 1600 1K-OASD 1K-VID TP DU UR
                   CPDGHHDZ L    2  1:00:00   640    10    2    0       0      0  0  0  0
06.31.27 JOB   233  IEF677I WARNING MESSAGE(S) FOR JOB CPDGHHOZ ISSUED
06.31.27 JOB   233  $HASP373 CPDGHHDZ STARTED - INIT 14 - CLASS L - SYS D168
06.31.27 JOB   233  IEF403I CPDGHHDZ - STARTED - TIME=06.31.27
06.31.30 JOB   233 *IEF233A M 381,EXMC61,,CPDGHHDZ,STEP1,DZ.UGUIDE.TEXT
06.41.50 JOB   233  IEC502E RK 381,EXMC61,SL,CPDGHHDZ,STEP3,EXMC61.F5179.R012
06.41.50 JOB   233 *IEC501A M 381,EXMC62,SL,6250 BPI,CPDGHHDZ,STEP3,EXMC61.F5179.R012
06.44.19 JOB   233 *IEA000A 381,INT REQ,D3,0200,,,EXMC62,CPDGHHDZ
                   IEA000I 381,,,,402200060040200000080000008EEA03D7D1708F1F010000,,
07.02.37 JOB   233  IEF234E R 381,EXMC62,PVT,CPDGHHDZ,STEP4
07.02.37 JOB   233 *IEF233A M 382,EXMC61,,CPDGHHDZ,STEP4,EXMC61.F5179.R012
07.02.37 JOB   233 *IEF233A M 381,PRIVAT,SL,CPDGHHDZ,STEP4,SI.INFDMVS.BACKUP.G0019V00
07.04.22 JOB   233  IECTMS9 381,905338,CPDGHHDZ,SYSUT2  ,99000,S.BACKUP.G0019V00
07.04.24 JOB   233  IEC705I TAPE ON 381,905338,SL,6250 BPI,CPDGHHDZ,STEP4,SI.INFDMVS.BACKUP.G0019V00
07.08.51 JOB   233  IEC502E RK 382,EXMC61,SL,CPDGHHDZ,STEP4,EXMC61.F5179.R012
07.08.51 JOB   233 *IEC501A M 382,EXMC62,SL,6250 BPI,CPDGHHDZ,STEP4,EXMC61.F5179.R012
07.11.53 JOB   233  IEC502E K 381,905338,SL,CPDGHHDZ,STEP4,SI.INFDMVS.BACKUP.G0019V00
07.11.53 JOB   233 *IEC501A M 381,PRIVAT,SL,6250 BPI,CPDGHHDZ,STEP4,SI.INFDMVS.BACKUP.G0019V00
07.12.14 JOB   233 *IEA000A 381,INT REQ,D3,0200,,,,CPDGHHDZ
                   IEA000I 381,,,,402000060040200000080000008EEA03D7D1708303010000,,
07.12.35 JOB   233  IECTMS9 381,907511,CPDGHHDZ,SYSUT2  ,99000,S.BACKUP.G0019V00
07.12.37 JOB   233  IEC705I TAPE DN 381,907511,SL,6250 BPI,CPDGHHDZ,STEP4,SI.INFDMVS.BACKUP.G0019V00
07.15.25 JOB   233  IEF234E K 381,907511,PVT,CPDGHHDZ,STEP4
07.15.30 JOB   233  IEF234E R 382,EXMC62,PVT,CPDGHHDZ,STEP5
07.15.30 JOB   233 *IEF233A M 383,EXMC61,,CPDGHHDZ,STEP5,DZ.CICS.MESSAGES
07.18.19 JOB   233  IEC140I TAPE    ,EXMC61 START OF DATA SET NOT ON VOLUME
07.18.19 JOB   233  IEC502E RK 383,EXMC61,SL,CPDGHHDZ,STEP5
07.18.19 JOB   233 *IEC501A M 383,EXMC62,SL,6250 BPI,CPDGHHDZ,STEP5,DZ.CICS.MESSAGES
07.20.48 JOB   233  IEF234E R 383,EXMC62,PVT,CPDGHHDZ,STEP6
07.20.49 JOB   233 *IEF233A M 38B,EXMC61,,CPDGHHDZ,STEP6,DZ.VTAM.MESSAGES
07.24.54 JOB   233  IEC140I TAPE    ,EXMC61 START OF DATA SET NOT ON VOLUME
07.24.55 JOB   233  IEC502E RK 38B,EXMC61,SL,CPDGHHDZ,STEP6
07.24.55 JOB   233 *IEC501A M 38B,EXMC62,SL,6250 BPI,CPDGHHDZ,STEP6,DZ.VTAM.MESSAGES
07.27.09 JOB   233  IEF234E K 38B,EXMC62,PVT,CPDGHHOZ
07.27.09 JOB   233  IEF471E FDLLDWING VOLUMES NO LONGER NEEDED BY CPDGHHDZ
                   EXMC61.
07.27.09 JOB   233  IEF404I CPDGHHDZ - ENDED - TIME=07.27.09
07.27.09 JOB   233  $HASP395 CPDGHHDZ ENDED
```

Figure 2. Job Resource Signature

```
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 383  ALLOCATED TO TAPE
IEF237I 1C2  ALLOCATED TO OZVSAM
IEF237I 1C2  ALLOCATED TO SYS00010
IEF237I JES2 ALLOCATED TO SYSIN
IEC140I TAPE     ,EXMC61 START OF DATA SET NOT ON VOLUME
IEF142I CPDGHHOZ STEP5 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS1.OZLIB                                    KEPT
IEF285I   VOL SER NOS= SCCS13.
IEF285I   JES2.JOB00233.S00109                          SYSOUT
IEF285I   OZ.CICS.MESSAGES                              KEPT
IEF285I   VOL SER NOS= EXMC61,EXMC62.
IEF285I   SIV6.BLGOZS.CICS.MESSAGES.CLUSTER             KEPT
IEF285I   VOL SER NOS= SCCPO2.
IEF285I   USERCAT.VSCCPO2                               KEPT
IEF285I   VOL SER NOS= SCCPO2.
IEF285I   JES2.JOB00233.SIO102                          SYSIN
IEF373I STEP /STEP5   / START 82208.0715
IEF374I STEP /STEP5   / STOP  82208.0720 CPU    0MIN 06.56SEC SRB    0MIN 00.15SEC VIRT   272K SYS   240K
*********************************************************************************************************
* USED     (V)   272K * ...PAGE-INS RECLAIMS * TCB TIME          6.56 * TCB/SRB  43.7:1  * STEP NO      5        *
* LSQA+SWA (V)   240K * PRV      6        3 * SRB TIME           .15 * RES/TCB  48.3:1  * STEP NAME  STEP5       *
* AVE W/S SIZE   383K * CSA     18        3 * RES TIME       5:16.89 * ATV/RES   1.0:1  * CONO CODE  0000        *
* SWAP CNT         1  * VIO      0        0 * ALLOC TIME        2.36 * PFM GRP       2  * PGM=       BLGOZC      *
*                     * SWAP    28      N/A * UNRC'O TIME        .52 * PRTY   DISP(009) * S/U'S       17,472     *
**UCB.DEVICE....EXCP'S***UCB.DEVICE....EXCP'S***UCB.DEVICE....EXCP'S***UCB.DEVICE....EXCP'S***UCB.OEVICE....EXCP'S**
* 123 3350          2 * 383 3400         115 * 1C2 3350          0 *                                          *
* JES/DUMMY       N/A * 1C2 3350         140 * JES/DUMMY       N/A *                                          *
*********************************************NON-VSAM(126/006/000)*********************************************
* DDNAME   OEVICE VOLSER OPNTYP OPNCNT DSORG OPTCD RECFM LRECL BLKSIZE  BUFNO BLOCK-CNT EXTS TRKALC TRKUSD TRKRLS  *
* STEPLIB  3350   SCCS13 INPUT       1  PO          U          0  19,069      0         2     1     30    N/A      0  *
* TAPE     3400   EXMC62 INPUT       1  PS          VB    12996  13,000      0       115  N/A    N/A    N/A    N/A  *
*********************************************VSAM(126/006/000)*********************************************
* DDNAME   CPNT STAT OPCT  LEVELS EXTENTS RECOROS OELETES INSERTS UPOATES    GETS UNUS-CI SPLT-CI SPLT-CA  BLKCNT  *
* OZVSAM   IOX  CURR    1       0       1       0       0       0       0       0 131,072       0       0       0  *
* OZVSAM   IOX  CHNG    1       2       0       4       0       0       0       0  16,384       0       0      18  *
* OZVSAM   OATA CURR    1       0       1       0       0       0       0   04745,600       0       0       0  *
* OZVSAM   DATA CHNG    1       0       0   6,874       0       0       0   01298,432       0       0     122  *
*********************************************************************************************************

IEF236I ALLOC. FOR CPOGHHOZ STEP6
IEF237I 123  ALLOCATEO TO STEPLIB
IEF237I JES2 ALLOCATEO TO SYSPRINT
IEF237I 38B  ALLOCATED TO TAPE
IEF237I 1C2  ALLOCATED TO OZVSAM
IEF237I 1C2  ALLOCATED TO SYS00012
IEF237I JES2 ALLOCATED TO SYSIN
IEC140I TAPE     ,EXMC61 START OF DATA SET NOT ON VOLUME
IEF142I CPDGHHOZ STEP6 - STEP WAS EXECUTEO - CONO CODE 0000
IEF285I   SYS1.OZLIB                                    KEPT
IEF285I   VOL SER NOS= SCCS13.
IEF285I   JES2.JOB00233.S00110                          SYSOUT
IEF285I   OZ.VTAM.MESSAGES                              KEPT
IEF285I   VOL SER NOS= EXMC61,EXMC62.
IEF285I   SIV6.BLGOZS.VTAM.MESSAGES.CLUSTER             KEPT
IEF285I   VOL SER NOS= SCCPO2.
IEF285I   USERCAT.VSCCPO2                               KEPT
IEF285I   VOL SER NOS= SCCPO2.
IEF285I   JES2.JOB00233.SIO103                          SYSIN
IEF373I STEP /STEP6   / START 82208.0720
IEF374I STEP /STEP6   / STOP  82208.0727 CPU    0MIN 03.97SEC SRB    0MIN 00.11SEC VIRT   272K SYS   240K
*********************************************************************************************************
* USEO     (V)   272K * ...PAGE-INS RECLAIMS * TCB TIME          3.97 * TCB/SRB  36.0:1  * STEP NO      6        *
* LSQA+SWA (V)   240K * PRV     28       11 * SRB TIME           .11 * RES/TCB  95.4:1  * STEP NAME  STEP6       *
* AVE W/S SIZE   328K * CSA     66        3 * RES TIME       6:19.04 * ATV/RES   1.0:1  * CONO CODE  0000        *
* SWAP CNT         1  * VIO      0        0 * ALLOC TIME        1.28 * PFM GRP       2  * PGM=       BLGOZC      *
*                     * SWAP    11      N/A * UNRC'D TIME        .77 * PRTY   OISP(009) * S/U'S       10,415     *

**UCB.OEVICE....EXCP'S***UCB.OEVICE....EXCP'S***UCB.OEVICE....EXCP'S***UCB.DEVICE....EXCP'S***UCB.OEVICE....EXCP'S**
* 123 3350          2 * 38B 3400         74 * 2C2 3350          0 *                                          *
* JES/OUMMY       N/A * 2C2 3350         92 * JES/OUMMY       N/A *                                          *
*********************************************NON-VSAM(126/006/000)*********************************************
* OONAME   OEVICE VOLSER OPNTYP OPNCNT DSORG OPTCO RECFM LRECL BLKSIZE  BUFNO BLOCK-CNT EXTS TRKALC TRKUSO TRKRLS  *
* STEPLIB  3350   SCCS13 INPUT       1  PO          U          0  19,069      0         2     1     30    N/A      0  *
* TAPE     3400   EXMC62 INPUT       1  PS          VB    12996  13,000      0        74  N/A    N/A    N/A    N/A  *
*********************************************VSAM(126/006/000)*********************************************
* OONAME   CPNT STAT OPCT  LEVELS EXTENTS RECOROS OELETES INSERTS UPDATES    GETS UNUS-CI SPLT-CI SPLT-CA  BLKCNT  *
* OZVSAM   IDX  CURR    1       0       1       0       0       0       0       0 131,072       0       0       0  *
* OZVSAM   IDX  CHNG    1       2       0       3       0       0       0       0  12,288       0       0      11  *
* OZVSAM   DATA CURR    1       0       1       0       0       0       0   04745,600       0       0       0  *
* OZVSAM   OATA CHNG    1       0       0   4,013       0       0       0       0 839,680       0       0      81  *
*********************************************************************************************************

IEF375I  JOB /CPOGHHOZ/ START 82208.0631
IEF376I  JOB /CPDGHHOZ/ STOP  82208.0727 CPU    24MIN 56.33SEC SRB    0MIN 26.13SEC
*********************************************************************************************************
* TCB TIME     24:56.33 * START TIME   6:31:27.33 * JOB NAME CPOGHHOZ PROGRAMMER LOAO NEW TAPE        U.S. SENATE  *
* SRB TIME        26.13 * ENO TIME     7:27:09.37 * RLS LEVL 03.8E    ACCT NO    750-CP-LT-P       COMPUTER CENTER*
* ACTV TIME    55:08.80 * ELP'SO TIME    55:42.04 * SYSTEM   0168     JOB LOG    06:30:29.01*07/27/82.208 - L -     *
*********************************************************************************************************
```

Figure 3. Job Step Termination Statistics Display

favored over jobs with higher estimated transit times by the relative position of the associated job class on the initiator strings.

4) Promoting good user practices which result in lower initiator transit times. Examples of this include reblocking of files to reduce I/O delays and program changes to reduce CPU consumption.

Attempts at reducing the dependencies between initiated jobs proved quite successful. As shown in figures 4 and 5 below, contention due to DSNAME enqueue conflict was virtually eliminated, and time spent in allocation recovery was sharply reduced (week 18 is when the DSNAME conflict manager and scheduler were introduced).

Aggregate turnaround time was improved significantly during the year following full implementation, as figure 6 attests (the scheduler was fully implemented in May 1980). No attempt has been made to compute the percentage improvement attributable to each facit of the scheduler (multi-server model, reduced initiator service time covariance, prioritized input queue, improved user practices).

An unexpected side effect of implementing the scheduler was the improved consistency of turnaround time. As shown in figures 7 through 10, the turnaround times for class A-type work (low resource use) and class D-type work (high resource use) showed a lower dispersion after scheduler implementation than before. This characteristic had a positive impact not only on the queueing theory aspects of the system (improved throughput), but also made the rewards system for users more predictable.

4. Conclusions

The scheduler system described above asserts the efficacy of proper controls over batch workloads. In the MVS environment in particular, alternatives to vendor-provided control systems may be particularly effective. For those sites with major batch workloads, the scheduler can provide significant benefit.

References

/1/ "OS/VS2 MVS System Programming Library: Initialization and Tuning Guide", No. GC28-1029, IBM Corporation, Data Processing Division, White Plains, New York, page 5-1.

/2/ Ibid, page 5-4.

/3/ R. Schardt, "An MVS Tuning Perspective", No. GG22-9023, IBM Corporation, Washington Systems Center, Gaithersburg, Maryland, Appendix A.

/4/ T. Beretvas, "Performance Tuning in MVS", IBM Systems Journal 17, No. 3, 290-313 (1978).

/5/ R. Simpson and G. Phillips, "Network Job Entry Facility for JES2",IBM Systems Journal 17, No. 3, 221-240 (1978).

/6/ "OS/VS2 MVS System Programming Library: System Management Facilities(SMF)", No. GC28-1030, IBM Corporation, Data Processing Division, White Plains, New York, pages 4-1 through 4-42.

/7/ J. Martin, "Systems Analysis for Data Transmission", Prentice-Hall, Englewood Cliffs, N.J. (1972), pp 413-480.

/8/ Ibid, pp 413-480.

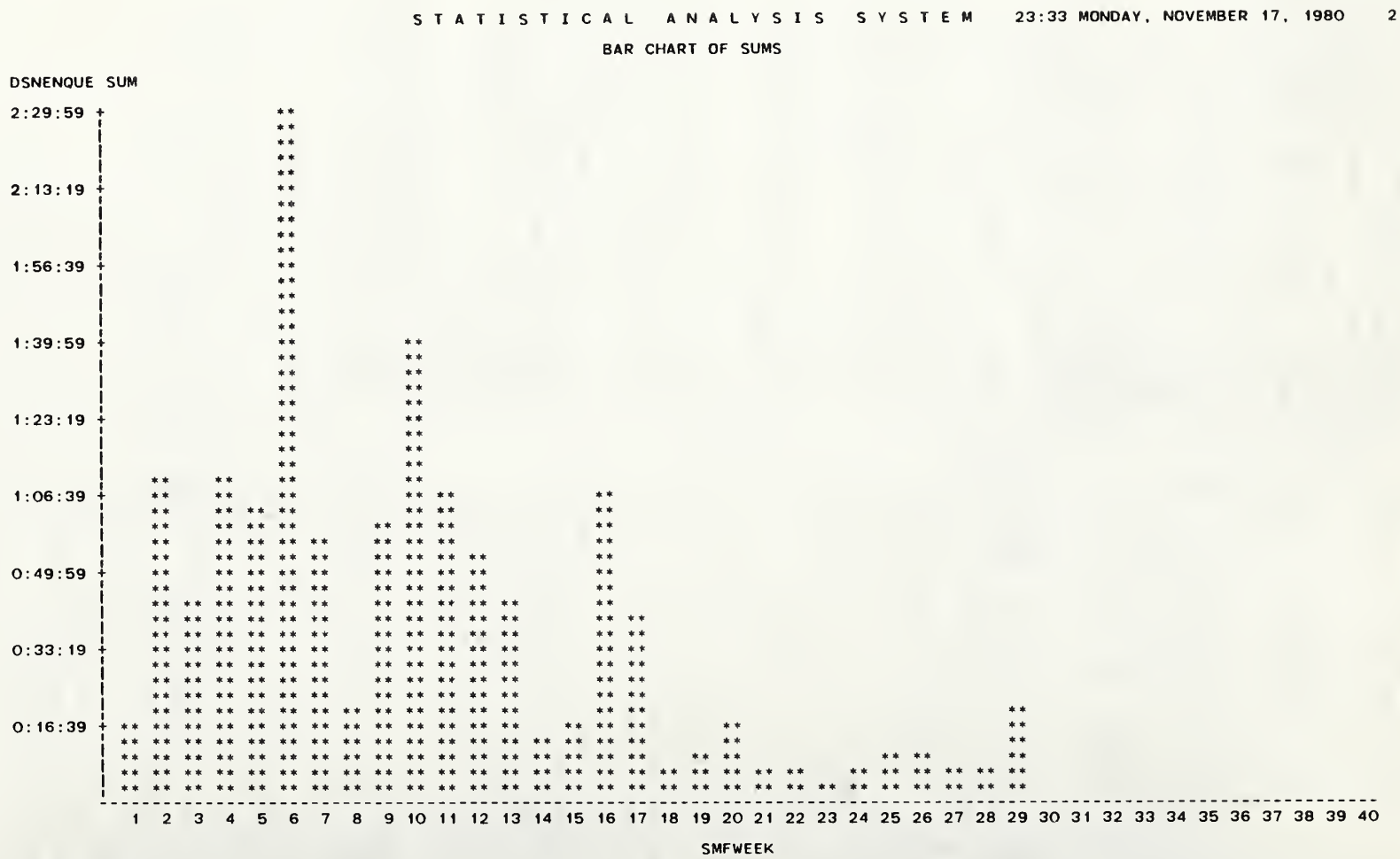


Figure 4. Average Initiator Hours Lost Per Week Due to DSNAME Enqueue Conflicts (Uncorrected for SMF Noise)

Figure 5. Average Initiator Hours Lost Per Week Due to Allocation Recovery

T U R N A R O U N D T I M E T R E N D

D E C E M B E R 1 9 8 0

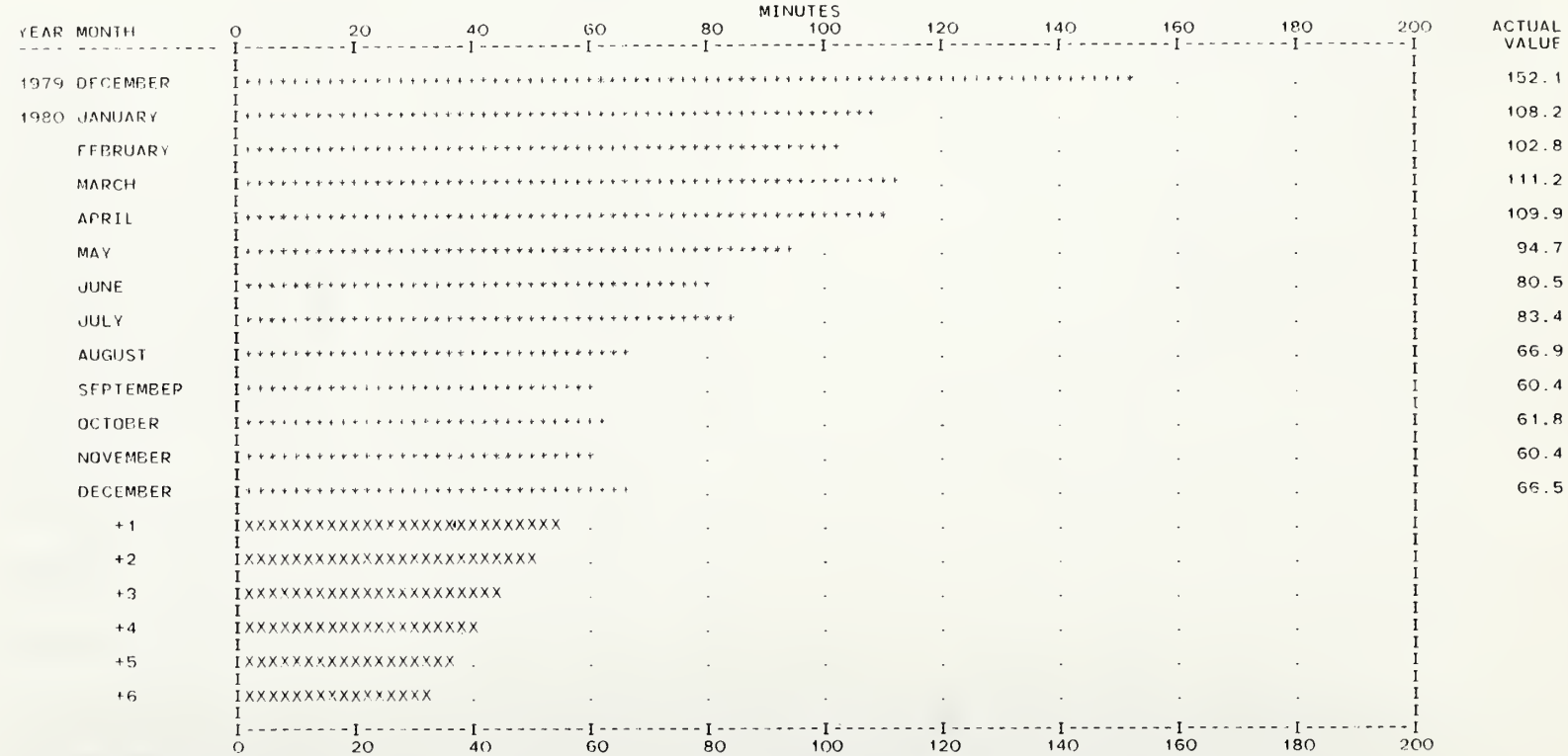


Figure 6. Average Batch Job Turnaround, by Month

JOBCLASS=A

FREQUENCY BAR CHART

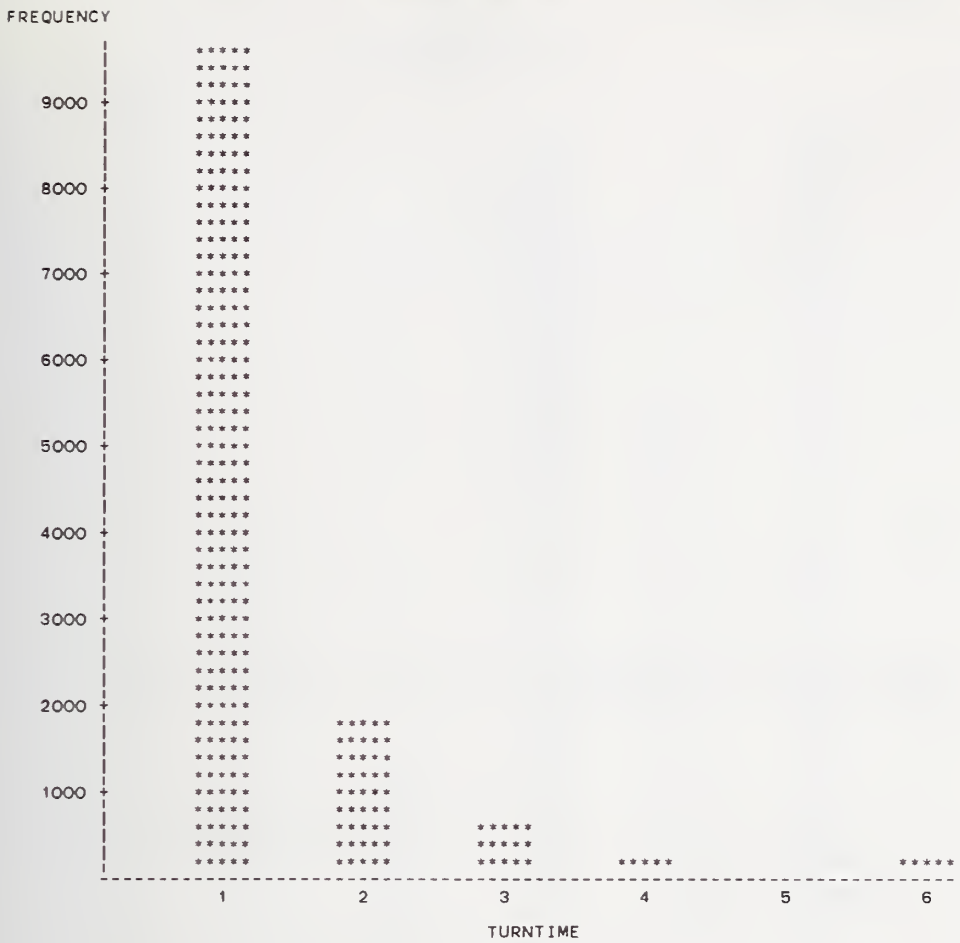


Figure 7. Turnaround Time Distribution for Class A (Low Resource Estimate) Work - Schedules (Reconstructed)

JOBCLASS=A

FREQUENCY BAR CHART

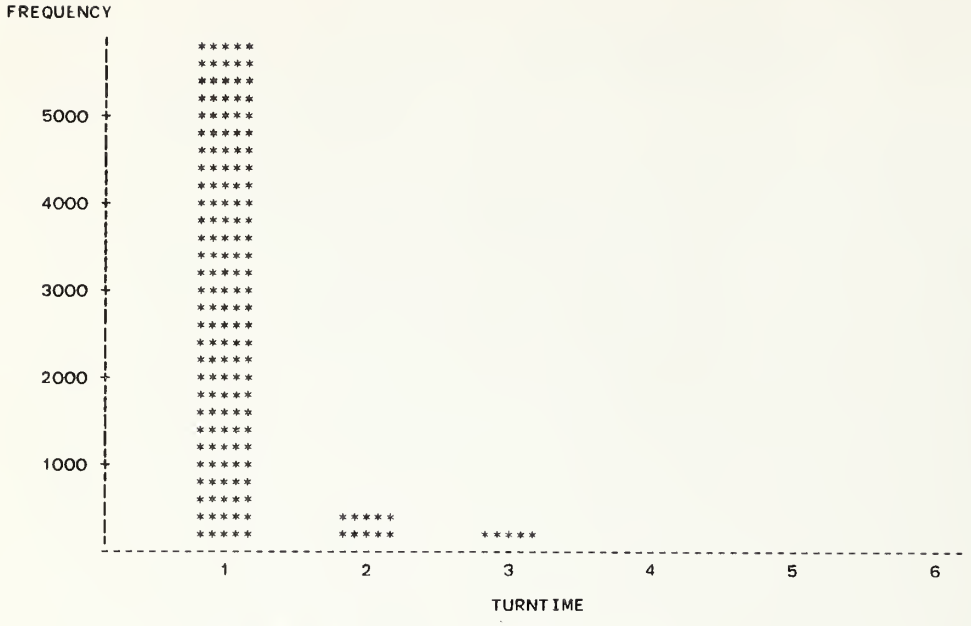


Figure 8. Turnaround Time Distribution for Class A Work - After Implementation of Scheduler

JOBCLASS=D

FREQUENCY BAR CHART

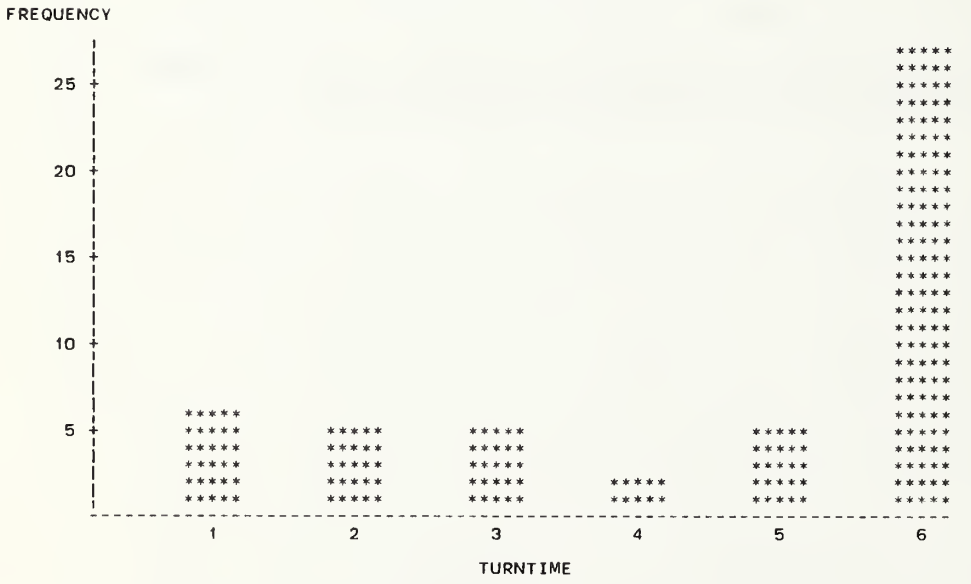


Figure 9. Turnaround Time Distribution for Class D Work - After Implementation of Scheduler

JOBCLASS=D

FREQUENCY BAR CHART

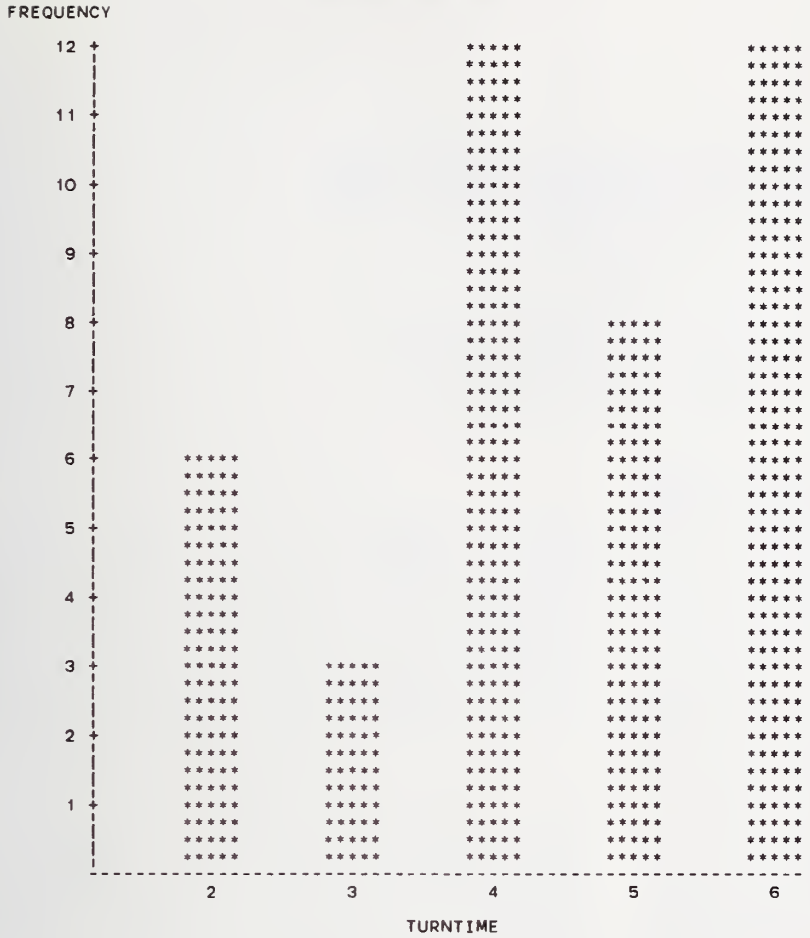


Figure 10. Turnaround Time Distribution for Class D (High Resource Estimate) Work - Pre - Scheduler

EVENT DRIVEN MEASUREMENTS OF MVS THAT IMPROVE CONFIGURATION TUNING AND MODELING

Glen F. Chatfield

Duquesne Systems Inc.
622 - Two Allegheny Center
Pittsburgh, PA. 15212

1. Objectives

The objectives of this paper are the following: 1) to review current requirements for additional MVS performance measurement data, 2) to identify those measurements that would significantly simplify tuning and modeling efforts, and 3) to describe new event driven measurement techniques for obtaining this useful data.

2. INTRODUCTION

System tuning can be divided into three basic categories as follows: hardware configuration tuning including volume placement; data tuning which consists of dataset placement, blocking and file organization; and software tuning which involves such diverse functions as setting SRM parameters, optimizing BLDL lists and even modifying the operating system as a last resort. The essence of configuration tuning is to improve the organization of hardware components and volumes to acheive installation defined service objectives. Rapid growth and the expanded use of shared DASD have made this task increasingly difficult in recent years.

3. CPU MEASUREMENT PROBLEMS

The CPU is the single most critical system resource, and as such, it is extremely important to have accurate and complete CPU measurement data. Unfortunately, many analysts rely on SMF or RMF for CPU measurement data without fully realizing their inherent inaccuracies. The accuracy of SMF and RMF was discussed in an article by J. C. Cooper [8] which appeared in the 1980 IBM Systems Journal, volume 19, number 1. In that article, titled "A Capacity Planning Methodology", Cooper stated, "SMF does not record all the CPU time expended on behalf of a particular job. Its purpose is to be consistent rather than complete, in order to satisfy requirements of charge-back systems. In addition, the portion of time captured by SMF varies with the workload type and the specific SMF implementation for the various SCPs."

Let's examine Cooper's statements more closely. To begin with, it is clear that SMF's inaccuracies are the result of its design purpose and are not simply the consequence of an oversight or an unfixed bug. Incidentally, the same holds true for RMF, since it uses SMF CPU data. Cooper's last statement is even more disturbing. The fact that SMF/RMF's accuracy is different for each workload makes its data very difficult to use. Before an analyst can begin to use raw SMF/RMF CPU data he must try to determine the correct capture ratios that should be applied to each measurement. Since the accuracy varies, this process is not simple like calibrating an instrument. Typically the analyst needs to massage the data with complicated numerical techniques. His problem is analogous to engineers and scientists being forced to use elastic measuring tapes, the result of which would destroy those disiplines as we know them today.

A reasonable question at this point would be, "Just how inaccurate are SMF and RMF?". If their error is only a few percent, one could easily live with that for most CPE work. Unfortunately, this is not the case. Page 33 of Cooper's article lists some typical capture ra-

tios. TSO trivial, for example, has a capture ratio of only .36. TSO program development is only slightly better at reasons why TSO is so difficult to tune and forcast is that the analyst can only measure a fraction of its impact on the systems. Moreover, this problem is not unique to TSO either. The capture ratio for batch testing is listed as only .60.

There are three major causes of SMF/RMF's inaccuracy. The first is that SMF does not measure and allocate I/O interrupt CPU time to the task that requested the I/O. This problem distorts nearly all SMF/TCB CPU measurments. SMF's second major omission is that it does not account for initiation/termination CPU time. This time can be quite significant for batch testing which typically contains many short jobs and steps. SMF's third flaw is that it does not specifically measure Auxiliary Storage Manager CPU time. Consequently, the CPU time required to support paging and swapping can not be easily identified and allocated to particular workloads. In the final analysis as Cooper stated, SMF/RMF CPU measurement was designed for consistent job accounting not for complete system measurement.

Another CPU measurement that is missing from SMF and RMF is the CPU visit count. This parameter, which is required by many configuration modeling tools, must be maintained separately for each workload and, in reality should be expressed as a function of CPU speed and workload processing rate. Approximating a workload's CPU visit count by equating it to its total EXCP count can be very inaccurate. In order for this approximation to be valid the following assumptions must be true: 1) the task must be I/O bound, 2) it must not overlap I/O requests, 3) it must not use QSAM, VSAM, VIO or other access methods that satisfy more than one EXCP request with a single SIO, 4) its I/O to system data areas such as VTOC's, catalogs, and directories must be negligible because these I/O's are not counted by SMF, 5) it must not wait a significant number of times for indirect I/O such as database calls, SPOOL reads or writes, program loads and paging, and 6) it must not issue many ENQs or wait on many timer interrupts. Recognizing the significance of the previous assumptions, it is clear that the analyst needs a better measure.

4. CPU MEASUREMENT SOLUTIONS

The best solution to SMF and RMF CPU measurement problems is obtained with event-driven measurement software. An event-driven software monitor, such as the QCM Performance Monitor, can accurately measure 100% of the CPU's activity because it receives control each time "ownership" of the CPU changes. At each of these change events the monitor computes the elapsed time (using the system time-of-day clock) since the last event and updates the appropriate workload measurement control block (WMCB). The ownership change events in MVS are TCB and SRB dispatches, I/O interrupts, external interrupts, page faults, and non-dispatcher exits from the I/O, external, and program check interrupt handlers. Control is passed to the monitor via dynamically inserted "hooks" at the required ownership change points. At these points the monitor determines which workload class is acquiring ownership of the CPU and saves a pointer to its WMCB. This pointer is then used by the monitor to locate the proper WMCB at the time of the next ownership change event. One or more special WMCBs can be used to accumulate times which do not explicitly belong to any of the workload classes defined by the user of the monitor. Examples of special workloads are the master scheduler, external interrupt processing, started tasks such as JES and the auxiliary storage manager. Since every ownership change event is intercepted by the monitor, all CPU activity is accurately attributed to the proper WMCB. (No "capture ratios" [8] are required.)

To provide CPU visit count functions, the monitor must intercept and count WAITs. A workload's voluntary CPU visit count is equal to its WAIT count, not its EXCP, SIO or dispatch count. Its WAIT count will be a function of the speed of the CPU it executes on and the rate at which the workload is given CPU service. With this in mind, a workload's voluntary CPU visit rate, as a function of its CPU service rate, can be empirically determined from different observations measured by the monitor. The workload's involuntary CPU visit rate will be a function of its page fault rate and the rate at which it is preempted from the CPU by higher priority workloads, factors that are functions of the multiprogrammed mix and not the workload itself.

5. I/O MEASUREMENT DIFFICULTIES

Moving away from the CPU, data paths are the next major components of the configuration. IBM 370 systems have three types of data paths, selector, byte multiplexor and block multiplexor. Because the the vast majority of configuration tuning problems are with the DASD I/O subsystem, discussions in this paper will be limited to block multiplexor data paths (the phrase RPS data path is synonymous). An RPS data path consists of a block multiplexor channel, a control unit and a head-of-string (called simply string for the remainder of this paper). S. E. Friesenborg[12] gives two excellent examples of path measurement problems in the IBM Technical Bulletin GG22-9217-00 titled "DASD Path and Device Contention Considerations".

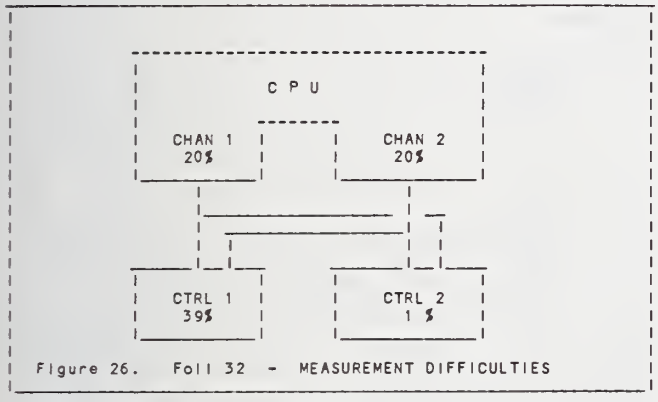


Figure 26. Foil 32 - MEASUREMENT DIFFICULTIES

"We are concerned with path busy, not Channel busy. If an RMF report showed a perfectly balanced situation of 20% utilization on each Channel, you may start congratulating yourselves on a great job...

When in reality the worst possible kind of imbalance may exist. The visual represents two Channels connected to two Control Units through a two-channel switch arrangement. Path busy is not Channel busy. Don't assume it is just because Channels are easy to measure."

The channel balancing algorithms within MVS will automatically balance the channels. However, by doing so, MVS can easily hide significant I/O bottlenecks because RMF does not measures the usage of the other components of the data path, namely the control unit and the string. The previous example illustrates this problem very clearly.

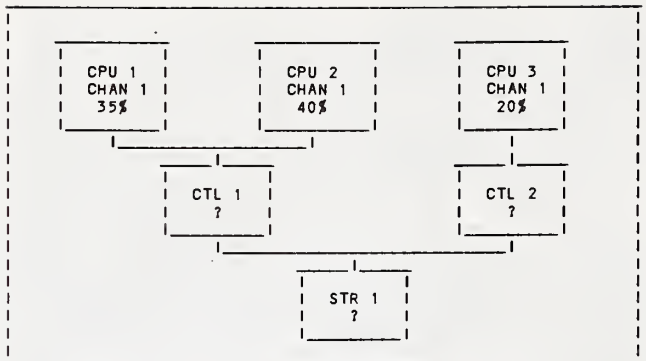


Figure 27. Foil 33 - MORE MEASUREMENT DIFFICULTIES.

Again quoting Friesenborg, "Things can get even worse.

One CPU's view of the world may look pretty good while a total view of the installation is a disaster. If the diagram above represents three systems sharing a single string, the measurement reports from each of the systems would show Channel busy percentages of 35, 40, and 20. The 40 is a bit above the 'universal goodness guideline' of 35%, but that shouldn't be too bad...

Actually, CTL 1 is at least 75% busy. This would cause great concern in most cases, but not this one. The reason that 75% utilization should not concern us is because the String Head is 95% utilized, and the contention is causing each I/O to be elongated by 19 revolutions. That is about three tenths of a second. In this example string head busy is the primary cause of application responsiveness problems."

6. EVENT-DRIVEN_I/O_MEASUREMENTS

It is apparent from the previous examples that the configuration tuner needs additional DASD measurements. Here too, an event-driven approach must be taken to obtain these important I/O measurements. Sampling monitors are only capable of measuring channel busy time (via sampling with the Test Channel instruction) and total device busy time (via sampling of the UCB busy bit).

The key hardware feature of the IBM 370 which can be used for detailed, event-driven DASD measurement is the Program Controlled Interrupt (PCI). When a channel fetches a Channel Command Word (CCW) which has the PCI bit on, it causes a PCI I/O interrupt while continuing to execute the channel program in a com-

pletely normal way. When the PCI interrupt is fielded by the monitor, it indicates that the previous DASD command has finished and the current command is begining. The monitor then performs an elapsed time computation to measure the duration of the pervious DASD function. For example, by turning the PCI bit on in the CCW that follows the initial seek CCW, seek time can be measured. Seek time is the elapsed time from the time of the SIO (the monitor must also intercept SIOs) to the time of the PCI. Also by placing another PCI on the first CCW after a set sector CCW, normal latency and RPS degradation times are measured. The time from the first PCI to the second PCI is divided by the device's rotation time. The integral quotient is the number of revolutions of RPS degradation (which is easily converted back to time) and the remainder is the normal fraction of a revolution of path disconnected latency time.

Seek times, measured by the PCI method described above, can be classified as one of three types, namely, productive seek time, seek degradation time, and shared seek degradation time. The monitor distinguishes among them by maintaining the transaction identifier and the cylinder number for the last I/O operation to each disk spindle. When the next I/O operation is performed, a comparison is made to determine if it is for the same transaction. If so, the seek time is accumulated as productive seek time; otherwise it is accumulated as seek degradation time. In a shared DASD environment an initial test prior to the above test is made to determine if the next operation is to the same cylinder as the last (a "zero cylinder seek"). If so and if a non-zero seek time is actually measured, this time is accumulated as shared seek degradation time since the arm must have been "stolen" by another system.

When format write CCWs are not present in a channel program, path busy time is measured as the elapsed time from the second PCI to the terminating channel end, device end interrupt. This portion of the operation (namely, search and data transfer) requires all the components of the path. When format write CCWs are present, a third PCI is required to measure path busy time. This PCI is placed on the first CCW after a command chained sequence of format write CCWs, and the elapsed time from the second PCI to this third PCI is the measured path busy time. The time from the third PCI to the channel end, device end interrupt is erase time, which can be maintained separately or included as part of total productive device busy time. (For certain DASD models, e.g., 3330 Models 1 and 2 and earlier DASDs, the control unit and head of string - but not the channel - are also busy during format write time, so in this case it is time when only a portion of the path, namely the channel, is free). For selector channel devices, such as tapes, total device busy time is simply measured as the elapsed time from SIO to device end interrupt. Path busy time, for both the channel and control unit, is the elapsed time from SIO to channel end. No PCIs are required.

7. AUXILIARY STORAGE MANAGER MEASUREMENT

Measurements of the auxiliary storage manager (ASM) are maintained by the monitor in a separate WMCB. For all releases of MVS prior to SP 1.3, ASM I/O is recognized at SIO time by its disabled interrupt exit (DIE) address in the IOSB. For SP 1.3, the ASM's IOS driver ID is used. The monitor also accumulates ASM CPU time in the CPU time entry of the ASM's WMCB. This time consists of the time to process page faults interrupts, the time to process ASM I/O interrupts, and the time to execute ASM SRBs. In order to be able to apportion the ASM WMCB values among the various workload classes, if desired, the monitor maintains a count for each workload class of the number of page transfers as a result of demand paging, swapping, and VIO (these values are obtained from the OUSB and OUXB). The ASM I/O and CPU times in its WMCB may then be divided among the classes according to the ratio of the number of page transfers for each class to the system wide total of page transfers. Note that it is not possible to directly assign an ASM I/O operation at SIO time to a particular workload class since a given ASM operation may involve the transfer of pages for multiple classes.

8. TUNING SUMMARY

Thus far, this paper has presented event-driven techniques for the direct measurement of critical MVS configuration tuning data. The parameters chosen to be measured were those that were felt to be most helpful in simplifing and improving the configuration tuning process. In particular, it was shown that it was

possible to obtain the following information:

1. complete CPU measurement,
2. productive seek time,
3. seek degradation time,
4. shared seek degradation time,
5. RPS latency time,
6. RPS degradation time,
7. erase time, and
8. path busy time for channels, control units and devices.

The remainder of the paper will discuss measurement issues related to the application of queueing network models to configuation tuning.

9. MODELING BACKGROUND

The emergence over the last several years of commercially available, "user-friendly" queueing network modeling packages has greatly spurred interest among performance analysts in the application of modeling to configuration tuning problems. These packages make it possible for analysts who would not previously have attempted modeling studies to now be able to do so.

The major effort in using such packages is normally in the preparation of its input data, the measurement of output data for model validation purposes, and the collection of additional performance data to facilitate modification analysis. This is not surprising since, for the most part, modeling theory and measurement methods have evolved independently. In particular, measurement tools have rarely been designed with the input and output requirements of models taken into consideration. When modeling IBM's MVS operating system, for example, few of the measurements of RMF and SMF, IBM's major monitors, can be used directly. The analyst must estimate modeling parameters from RMF and SMF values following methods such as described in [1] and [9], a process which is not only time consuming but can introduce inaccuracies into modeling results. Also, in some cases a meaningful estimation of parameters from known values may not be possible and the analyst is forced to use simpler than desired models.

In general, the measurement problems associated with analytical modeling can be grouped into three general categories. The first such category consists of those problems related to collecting measurement data for the critical processing periods, frequently called windowing. Because there is no specific modeling measurement tool available today, analysts are forced to use data from other sources. As a result it is difficult, if not impossible, to accurately characterize these critical processing periods.

The second major problem area has to do with lack of readily available transaction counters and response time measurements. With the exception of TSO, there is very little relevant transaction data for online systems. To meet the requirements for modeling each sub system must provide this type of measurement data.

The last category includes all of the problems that result from inappropriate, inaccurate or incomplete workload and server measurement. It is this third category of problems that can be solved cleanly with the same event-driven methods already described.

10. WORKLOAD AND SERVER MEASUREMENT REQUIREMENTS

Measurement needs for using queueing network models can be classified into three basic types:

1) Measurement of a system to obtain input values required by modeling algorithms;

2) Measurement of a system to obtain output values produced by modeling algorithms (used to validate a model); and,

3) Measurement of a system to obtain various values which do not correspond directly to model inputs or outputs but rather are used to facilitate modification analysis (i.e., the process of preparing model inputs which represent a hypothetical system and hence cannot actually be measured).

Specific examples of each of these types of values are given in the text that follows.

11. Model Inputs

Major input values required by typical modeling algorithms are [1, 2, 3, 4, 5, 6, 7](not all values are required by all algorithms):

1) The number of workload classes, each consisting of "transactions" (jobs or

steps, TSO transactions, IMS transactions, etc.);

2) For each class - CPU time or the number of CPU visits per transaction, and CPU mean service time per visit; and,

3) For each class and each I/O device - I/O time or the number of successful SIOs per transaction, and mean device service time per SIO inclusive of contention time.

12. Model Outputs

Major outputs produced by modeling algorithms and required for model validation are [1, 2, 3, 4, 5, 6, 7]:

1) Transaction response time per class;

2) Throughput per class for closed models (for open models this value, interpreted as arrival rate, is a model input); and,

3) Server utilizations by class.

13. Modification Analysis

Some DASD measurements useful in the modification analysis process are:

1) Counts of bytes transferred per transaction per class to each device; and,

2) All individual components of device busy time, e.g., normal latency, RPS degradation, format write, search and data transfer, and seek degradation caused by arm stealing by another system or by competing transactions in the same system.

These measurements are helpful, for example, in performing modification analyses which involve moving datasets from one disk to another, reblocking datasets, or moving disks from one string to another.

14. EVENT-DRIVEN WORKLOAD MEASUREMENT

This section will highlight the additional event-driven methods that are particularly helpful to modeling. It will also provide a more detailed description of how the WMCB is used.

15. Workload Classes

Workloads may be classified in different ways in MVS, two examples being by address space (or groups of address spaces) or by performance group period. From the standpoint of the measurement tool, the major constraint in assigning workload classes is that for efficiency's sake, the process of associating units of CPU and I/O activity with their workload classes must be straightforward. This is true of address spaces, for example, since the address space number corresponding to a particular activity is usually quite easy to determine. Corresponding to each workload class, the monitor assigns a WMCB which contains the measurement data for that workload.

16. Transaction Counters

The number of transactions encountered during a measurement interval must be counted in order to obtain the number of CPU and I/O visits per transaction. For batch workloads, the job initiation exit (IEFUJI) increments a counter by one in the appropriate WMCB each time it executes. If desired, step initiations can be counted as well, using IEFUSI. For other types of workloads (e.g., TSO and IMS), hooks at transaction initiation points in these systems are required which update counters in the appropriate WMCBs. (Alternatively for TSO and batch jobs, the monitor can copy the transaction count maintained by MVS in the OUXB control block to the appropriate WMCB).

17. Additional Workload Measurements

The monitor's CPU and I/O timing techniques were described earlier in the tuning section. Many modeling tools, however, require visit counts also. The value which should be used for the total number of CPU visits is the number of real waits issued by the workload. This value should be taken from the empirical CPU visit function described previously. I/O visit counts are simply the number of successful SIOs which, strange as it may seem, is most easily obtained by counting device end interrupts [10]. This technique is simpler than counting executions of the SIO and SIOF instructions with condition code zero since some SIOFs may result in deferred condition code one interrupts (meaning the I/O operation did not start) and hence should not be counted. The monitor can also obtain the number of bytes transferred by a channel

program by scanning the program and accumulating the byte count fields of the CCWs in the program. This technique is described in detail in [11].

In order to maintain all of the above measurements (seek times, RPS degradation, path busy time, event counts, bytes transferred, etc.) by individual device (i.e., I/O server) by workload class, the monitor functions as follows: At the time of an SIO (or SIOF) the monitor receives control (via a dynamically installed hook) and, by examining MVS control blocks, determines which workload class is responsible for the upcoming I/O operation. (For example, the number of an address space responsible for an I/O operation is located in the IOSB which is pointed to by register 2 at the time of an SIO). The monitor then saves a pointer to the WMCB for that workload class in a control block, called a device status block (DSB), corresponding to the device to which the SIO is being issued. At the time of an I/O interrupt (PCI or device end) the monitor uses the interrupting device's address to locate the proper DSB, which in turn points to the proper WMCB. Each WMCB has an entry for each device and the monitor accumulates its various measurements in the proper entry.

18. Model Outputs

In order to derive transaction response times the monitor maintains the total transaction active time for each workload class. This value is available in the OUXB for batch jobs and TSO and is copied to the WMCB. For subsystems such as IMS, hooks at transaction initiation and termination points are required in order to measure it. Transaction active time divided by the number of transactions gives transaction response time in the main system, i.e., delays in a communications network or spooling delays in printing a job's output are not included. Consequently these response times can only be validated against model outputs which also ignore these delays.

Throughput per class is easily computed since the total number of transactions per class and the length of a measurement interval are available. Throughput is only a model output for closed system models. For open models, this value, treated as arrival rate, is actually a model input.

Since total device busy time at each device can be obtained from the WMCBs, device (i.e., server) utilizations by class can be easily derived.

20. SUMMARY AND CONCLUSION

This article has presented methods pertaining to MVS for the direct, efficient measurement by a software monitor of important parameters required for effective configuration tuning and modeling. Event-driven techniques for accurate and complete CPU I/O measurement were described. In particular, for RPS DASDs, which are of critical importance in virtually any performance study of MVS, it was shown that the following measures can be obtained for each workload class and each DASD:

(1) Total number of successful SIOs;

(2) Total productive seek time;

(3) Total seek degradation time;

(4) Total shared seek degradation time;

(5) Total RPS degradation time;

(6) Total normal latency time;

(7) Total path busy time (i.e., search and data transfer time);

(8) Total format write time; and

(9) Total number of bytes transferred.

In addition, it was shown for each workload class that the

(10) Total number of transactions can be maintained.

For modeling studies, the analyst can combine certain of these values to obtain the necessary inputs for a particular modeling tool. For example, mean service time per transaction inclusive of contention time is the sum of items (2) through (8) divided by item (10), and mean service time exclusive of contention time is the sum of items (2), (6), (7), and (8) divided by item (10). Items which are not used directly as inputs are still of value to the analyst because of their importance in performing modification analyses. For example, if the impact of a particular dataset rearrangement is to be modeled, then items (6), (7), and (8) are unchanged and the analyst need only predict the changes in items (2) through (5).

It is important to note the distinction between the DASD measures above and those possible with RMF and SMF. RMF only measures system-wide (i.e., independent of workload class) total device busy time (i.e., inclusive of all contention times). In general, these total times can only be approximately apportioned to workload classes by using ratios of SMF EXCP counts for each class at each device to the total number of EXCPs for the device. This apportionment can be quite inaccurate since EXCP counts do not reflect the true amount of device usage [10]. Detailed values such as RPS degradation times cannot even be estimated accurately from RMF and SMF data.

Finally, the configuration tuning effort can be significantly simplified by the path busy and RPS degradation measurements at the channel, control unit, string and device levels.

21. REFERENCES

1. Rose, C.A., "A measurement procedure for queueing network models of computer systems", Computing Surveys 10, 3 (September 1980).

2. "An integrated approach to capacity planning: A detailed product description of BEST/1 and CAPTURE/MVS", BGS Systems, Waltham, MA.

3. Krzesinski, A. and Teunissen, P., "An introduction to system modelling using the "Stochastic Network Analysis Program", Report No. RW77-05, Department of Computer Science, University of Stellenbrosch, South Africa, June 1977.

4. "CADS (Computer Analysis and Design System): A brief description", Information Research Associates, Austin, TX.

5. "PAWS (Performance Analyst's Workbench System): Introduction and technical summary", Information Research Associates, Austin, TX.

6. Bruell, S.C., and Balbo, G., Computational algorithms for closed queueing networks, North Holland, NY, 1980.

7. Bard, Y., "A model of shared DASD and multipathing", Communications of the ACM, 23, 10 (October 1980).

8. Cooper, J.C., "A capacity planning methodology", IBM Systems Journal 19, 1(1980), pp. 32-33.

9. Levy, A., "Introduction to practical operational analysis: An MVS perspective", CMG Proceedings, Boston, 1980.

10. Conner, W., "An analysis of I/O operation counts in MVS", Proceedings of the ICCCM, Chicago, 1981.

11. Conner, W., "A more accurate software technique for counting number of bytes transferred", Proceedings of the ICCCM, Washington, D.C., 1979.

12. Friesenborg, S. E., "DASD path and device contention considerations", IBM Technical Bulletin, GG22-9217-00.

A NEW APPROACH TO VM PERFORMANCE ANALYSIS

William Tetzlaff

Research Division
Yorktown Heights, New York

Thomas Beretvas

Information System and Technology Group
Poughkeepsie, New York

In the past VM performance analysis centered around resource utilization, and more specifically around CPU utilization. In this paper the concept of state sampling is used to characterize the system responses to interactive users. We will show how to break the response time into its components and consequently how to locate the limiting resource effecting response time.

INTRODUCTION

In the past VM performance analysis centered around resource utilization, and more specifically around CPU utilization. The concept of state sampling in VM/370 was introduced by Tetzlaff [1] in order to characterize the time spent waiting for, and using resources by all users. We extend this technique in order to characterize the system responses to interactive users. Doherty and Kelisky [2] and Thadani [3] have demonstrated the high value of fast response time to the interactive terminal user.

In this paper we will show how to break the response time into its components and consequently how to locate the limiting resource affecting response time and suggest ways that it can then be improved.

The question that was normally asked in the past: "how many users can be supported by the next larger size CPU?". Sometimes another question was asked "how many fast disks are needed for the support of the users?". These questions are still being asked of course, but with the advent of faster CPU's and larger disks the focus has changed. There is an increasing emphasis on fast response times.

The need arose to *understand* the reported "response times". An analysis of the response times had to break the response times into its components, arising from CPU, I/O and paging. Such analysis was not done before, and no existing program had been equipped to provide this capability. In order to provide this function, the VM Monitor records were processed by an internally developed program, and the resulting reports were used to obtain the response time components. In the analysis special emphasis was placed on the hitherto largely neglected paging component of the response time.

Analysis of the data revealed that often the reported response times were not "true" response times, they were only "Q-drop" times. Some transactions really consist of multiple Q-drops, and therefore the true response time may be longer than the reported Q-drop time. Analysis of the trace data revealed ways to reduce the frequency of two kinds of queue drops occuring.

The paper discusses the formulae used for the evaluation of the response times components, specifically, the supervisor and problem program CPU times, I/O and paging response times and CPU wait times.

OVERVIEW OF VM/SP SCHEDULING.

In VM/SP the scheduler moves users between several scheduling categories. The broadest distinction between users is between those "in a queue," "eligible to be added to a queue," and those not in a queue. The users that are in a queue constitute the multiprogramming set. These users are in a single ("run") list that is sorted by an internally assigned priority. The list is searched by the dispatcher, which dispatches the highest priority user possible. If a user has work to process, but there is insufficient main storage, he is not placed in queue, he is given "eligible" status, (i. e. eligible to be placed in queue).

The users in a queue can be classified into three categories. The system attempts to place interactive users in queue 1 ("Q1"), longer running transactions in queue 2 ("Q2"), and very long running transactions in queue 3 ("Q3"). The three queues are not really separate, they represent different status bits in the users' VMBLOK.

Q1.

Q1 is intended for interactive users. A virtual machine is placed in Q1, whenever an I/O operation is completed on a Teleprocessing Device. For most purposes this means the console of a CMS user is placed in Q1.

The purpose of placing users in Q1, is to identify the interactive users on the system, in order to give them better service.

There is a side effect of this way of determining which virtual machines represent interactive users. Certain service machines, RSCS (VNET) or Passthrough do many I/O operations to Teleprocessing Devices. If the machine is not in Q1, it is immediately placed there as a result of these operations, even though it is not an "interactive user".

If a virtual machine completes its work and goes into an idle state, it is removed from Q1. This is referred to as a voluntary drop from Q1. If the virtual machine has not completed its work after a specific quantity of CPU time has been consumed (known as a time slice), then the machine is involuntarily dropped from Q1. The user then becomes *eligible* to be placed in Q2. If no other users are waiting to be placed in Q2, then the user will actually be placed in Q2, otherwise he may languish in "eligible" state until his turn comes. The time slice that is used to define the limit of Q1 is inversely proportional to the processing power of the CPU that is running VM/SP.

Q2.

Q2 users are given repeated time slices in Q2, with possible intervening times of being eligible to enter into Q2. After eight time slices in Q2 a machine may be placed in Q3.

Q3.

Virtual machines are placed in Q3 if the system decides that the working set of the machine is so large that it would be more profitable to move the machine in and out of queue less frequently. Q3 users stay in queue for eight time slices, and then are placed on the eligible list. They are then allowed to stay in the eligible state proportionately longer than Q2 users. Thus, the number of transitions in and out of queue is reduced by a factor of eight, without affecting the long term service rate.

DATA COLLECTION AND REDUCTION.

All of the data that is used in our analysis of VM/SP response time is available through the VM/Monitor [4] which is a standard part of the VM/SP system control program. VM/Monitor collects system performance and resource utilization data by means of sampling and trace techniques, and writes the data collected on tape or to a disk (spool) file. Some of the captured events -- terminal input, for example -- are caused by activities of system users. Other events correspond to the use of such resources as individual Direct Access Storage Device (DASD) accesses. Still other events, such as moving a user from one scheduling queue to another, are caused when scheduling decisions are made. Sampling is initiated by the expiration of an interval of time. The time-driven events cause information about each individual user, also about DASD and tape device utilization to be written periodically.

VM/Monitor data are reduced by a local data reduction program [1] written at Yorktown for IBM internal use only.

THE YORKTOWN VM SYSTEM.

Unless otherwise noted, all of the examples of data come from the Computing Center at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. The VM/370

systems are used interactively, primarily from display terminals in offices. but also in terminal rooms, and in some cases from home.

See Reference 2 for more information on the computing environment at Yorktown. The Yorktown VM systems contain many local modifications. Some of these affect interactive performance, others affect the collection of data.

For the purpose of this study it was desired to bring the number of Q-drops more in line with the number of interactive transactions. Consequently the system was modified to match the Q1-drops of users with trivial transactions. Service machines like RSCS (VNET,CJN) and PVM (Passthrough VM) were found to generate large numbers of drops from Q1. A VM/SP feature that circumvents some of the queue drop processing was generalized in order to fully eliminate as many queue drops as possible. The new feature allows a particular userid(s) to be excluded from part of the queue drop processing. The Yorktown system was modified so that when one of these users goes into PSW wait he is not immediately dropped from queue. At the time that they would have been dropped a .3 second timer interrupt is set. If the user is still inactive at the end of the .3 second interval he is dropped from queue. These machines normally become active before the time expires, and thus stay in queue. These changes lowered the Q1 drop count, and increased the reported response times, however the response time as seen by the user remained the same.

TYPES OF RESPONSE TIMES.

The value of fast response time is now well known, how to provide it to a user community is less well known. We have developed techniques for characterizing interactive response time in order to find the limiting resources, and this can lead to the enrichment of the limiting resource, so that the system limitation is removed.

There are three different ways to measure and report VM response times. Because the three different methods produce measures that significantly differ in magnitude, it is necessary to distinguish among them.

TERMINAL TRANSACTION RELATED RESPONSE TIME.

If both scheduler trace data and a trace of user inputs and system outputs is collected, it is possible to measure the time from terminal input to drop from Q1. This measure is the one that is closest to the magnitude seen by the user, because it includes data transmission time. The disadvantage of using this definition of response time is the quantity of data that must be written to the monitor and processed later. Our analysis did not require the delays associated with terminal interaction.

RESOURCE MANAGER (Q1) RESPONSE TIMES.

The Resource Manager provides the average time spent by users waiting to get into Q1 and the average time in Q1. On most systems the time waiting in the eligible list is zero, thus the Q1 response time is the same as time spent in Q1. Q1 response data represent a mix of at least two very different kinds of transactions. The truly interactive transactions use only about ten percent of the CPU time slice, and do few if any I/O operations. The longer running transactions that are started in Q1, but are later dropped from Q1 and placed in Q2, are very different. Because they are removed when they consume the time slice of CPU time, they all use a full time slice of CPU time. In addition they usually do tens of I/O operations. Analysis of all Q1 transactions thus represents an averaging of two quite different kinds of work.

TRIVIAL TRANSACTION (Q1 VOLUNTARY DROP) RESPONSE TIMES.

Scheduler trace data allows response time measurement for the transactions that complete before the Q1 time slice has been consumed. Thus, the transactions that ultimately end up in Q2 are excluded from the response time calculation. The primary goal of this analysis is to understand how these ("trivial") transactions spend their time.

AVAILABLE REPORTS.

All of the data used in this analysis comes from VM/Monitor. VM/Monitor allows selective collection of data on a tape or as a spool file (disk). This is done through keyword selection of the classes of monitor data to be collected. It is necessary to run the monitor with the PERFORM and USER classes specified in order to do basic Q1 analysis. In order to measure trivial responses, meaning those due to "Q1 voluntary drop", and estimate its primary components, it is necessary to collect SCHEDULE class data. Analysis of I/O devices requires DASTAP class data.

STATE SAMPLES OF USERS.

At regular intervals, the VM/Monitor records state data about each user. By examining the data, it is possible to classify each user into one of several run states that are useful for analysis. The states are summarized as follows:

```
IDLE-------User keying, thinking, or absent.

RUNNABLE---User in storage and ready to use the CPU.

RUNNING----User using the CPU.

I/OWAIT----User waiting for completion of I/O.

PAGEWAIT---User waiting for the completion of a page-in.

ELIGIBLE---User waiting for his turn in storage.

DEFERRED---User waiting for the system lock to become free.

INST SIM---Instruction Simulation,  a form of CPU wait.

PSW WAIT---Service Machine waiting for slow speed TP I/O.
```

Analysis of state data helps locate bottlenecks in the system, and allows analysts to measure the effect of bottlenecks on the users.

The state information makes it possible to determine whether a particular Virtual Machine is actually using the CPU. The IDLE state indicates that the system is waiting for a response from the user by showing that a user's virtual machine is idle. All the other states represent active states of the virtual machine.

The state information available through the monitor is only sufficient to indicate that a user cannot run again until an I/O activity completes. Thus, it is not possible to distinguish between the state of waiting for the physical device and that of using the device. Similarly for paging, it is not possible to distinguish between the state of waiting for a page read to be started and the state of having a DASD device reading it into main memory.

A user in the ELIGIBLE state has been temporarily removed from the dispatchable set because there is insufficient main storage for all active users. For more information on VM/370 states and the transitions among states, the reader is referred to Reference 5.

Once all of the state data snapshots are available, they can be used in a number of ways by postprocessing them, as it is done at Yorktown. One way to use the state data is to count all the states; calculate percentages of occurrences of each state, and tabulate them. By aggregating the data for all users, the relative time spent by all users or by an average user in each state is observable. The percentages of all states provide a picture of how the logged-on time of the users is spent. It is expected that most of the logged-on time on a time-sharing system is spent in the IDLE or user-response state. Even when a user is working intensely at the terminal, the user response time -- which includes keying time and thinking time -- dominates the elapsed time. The number of terminals, placement of terminals, local habits and any forced log-off procedures also influence the idle time.

The system-wide summary of user states gives insight into the relative importance of I/O, CPU paging, and main storage. As the state samples also contain an indication of what queue a user is in, it is possible to characterize the states of Q1, Q2 and Q3 users separately. In the user-state summary in Table 1, I/O wait is the dominant state of the system for all queues.

Table 1 Summary of user states

REPORT SELECTION NAME: USER_STATE_SUMMARY_BY_TOD

USER STATUS NAME	COUNT STATUS NOT IN A QUEUE	COUNT STATUS QUEUE Q1	COUNT STATUS QUEUE Q2	COUNT STATUS QUEUE Q3
?-IDLE	953	0	26	32
I/O WAIT	0	193	241	17
IDLE	17258	0	0	0
PSW WAIT	0	72	5	0
DEFER FUNC	0	3	3	1
PAGEWAIT	0	71	29	3
RUNNING	0	11	8	8
INST SIM	0	27	9	0
RUNNABLE	0	2	6	3
ELIGIBLE	0	0	0	0

DATE: 12/23/81 TIMES FROM:09.00 TO 10.00

Users whose states are summarized in Table 1 have never been in the ELIGIBLE state, meaning that the scheduler considered that there has always been enough main storage to add them to the multiprogramming set when they were ready to run. (This suggests that the main storage in the CPU was sufficient for the workload.) Notice that in queue users are never shown in an idle condition, because they are removed from queue when they become idle.

One of the problems with aggregating all users is that different types of work may be inappropriately reported together. A way to deal with this situation is to group the data by type of work. At Yorktown this is done on the basis of User Identification (USERID). There is a mapping between USERID and a group name, which makes possible separate reports of state information for interactive users, system functions, service virtual machines, batch virtual machines, etc. This may allow the identification of a service problem for a particular class of work, when there appears to be no overall service problem.

The state sample information can also be used to estimate the actual time spent in each state. In order to do so, multiply the fraction of the samples in that state by the logged-on time. The following formula gives elapsed-time estimates:

$$\text{ELAPSED TIME} = \frac{\text{SAMPLES IN STATE}}{\text{TOTAL SAMPLES}} \times \text{LOGGED ON TIME}$$

Other resource data is also collected with the state data. These data include CPU time (virtual and system) page reads and VIO counts. A special USERID named SYSTEM is used to account for resources not assignable to a particular user. The most important data for characterizing the interactive users is the state sample data. The counts of each state for Q1 users shown in table 1 constitute the Q1 user profile. The profile may be presented as percentages, a bar chart, or a pie chart.

RESOURCE MANAGER DATA.

The resource manager keeps two sets of data, one for Q1 and the other for Q2 (which includes Q3), that contain response and resource data. Resource data include CPU time and VIO count, but no page read count. CPU time attributed to the "SYSTEM" userid is not counted in these data. Response time data includes the time waiting to get into a queue, the time in queue, and a transaction count.

A particular advantage of the data is that Q1, and Q2 resource consumption are separated, providing an inexpensive way of characterizing transactions. The resource manager report shown in Table 2 provides information on the response time, throughput, page reads and CPU time used in Q1. (Similar data can be obtained about Q2 resource use).

Table 2

Queue 1 Resource Manager Data

REPORT SELECTION NAME: SERVICE_BY_TOD

MIN TIME	Q1 DROP RATE	Q1 MEAN Q1 CPU	Q1 PAGE RDS	MEAN Q1 ELIG	Q1 RES PONSE (SEC)
09.01	12.2	0.028	5.531	0.000	0.508
10.00	13.0	0.027	5.927	0.000	0.485
11.00	12.7	0.026	5.377	0.000	0.412
12.00	10.5	0.028	5.071	0.000	0.375
13.00	12.8	0.028	6.587	0.000	0.486
14.00	14.6	0.027	6.566	0.000	0.447
15.00	13.2	0.027	6.629	0.000	0.456
16.00	14.0	0.030	7.014	0.000	0.465
Mean	12.9	0.028	6.137	0.000	0.456

RATES ARE PER SECOND
DATE: 12/23/81 TIMES FROM:09.00.53 TO 16.59.53

SCHEDULER TRACE DATA

Scheduler trace data can be collected in order to obtain CPU, Paging and I/O activity. Records are written each time a user is added to or removed from the multiprogramming set. These data can be divided between Q1 and Q2 and between voluntary and involuntary drops from queue. CPU time attributed to the "SYSTEM" userid is not counted in these data because the SYSTEM is never added or dropped from queue. The primary advantage of this class of data is the separation by queue, separation between involuntary and voluntary drops, and the existence of an I/O count. The disadvantage of these data is that they come in massive quantities, they come at a high cost in terms of both data collection and data reduction. It is necessary to write and process at least three logical records for each transaction run on the system. For this reason it is customary to run a scheduler trace only for short periods of time. In the examples of data that are shown in Table 3 the data was collected for fifteen minutes during the 10:00 AM and 2:00 PM hours. Because the data for both involuntary and voluntary drops are included in the same line, these data become comparable to the Resource Manager data by queue. The comparable data in Table 2 does not match the data in Table 3 because the time periods differ (Table 2 page read is 5.9 vs. 6.6 in Table 3).

Table 3 Scheduler Data by Queue

```
RESOURCE_BY_QUEUE
DATE: 12/23/81
                                    MEAN
            DROPQ    MEAN   MEAN  PAGE
HOUR QUEUE  COUNT     CPU   VIOS  READ

  10     1  10237  0.030    4.6   6.6
  10     2   3173  0.168   40.9   6.6

  14     1  11949  0.029    4.3   6.5
  14     2   3168  0.182   33.6   5.6

INVOLUNTARY AND VOLUNTARY ARE COMBINED
DATE: 12/23/81
TIMES FROM:10.30.00 TO 10.45.00
      FROM:14.30.00 TO 14.45.00
```

The data in table 4 shows the resources used by truly trivial transactions (transactions that complete in Q1). These data form the basis for estimating the time components of trivial transactions.

Table 4 Trivial Transaction Data by Hour

```
RESOURCE_BY_QUEUE
                                    MEAN
            DROPQ    MEAN   MEAN  PAGE
HOUR QUEUE  COUNT     CPU   VIOS  READ

  10     1   8704  0.019    1.9   4.9

  14     1  10442  0.019    1.9   4.9

RESOURCE USE FOR QUEUE 1 VOLUNTARY ONLY
DATE: 12/23/81
TIMES FROM:10.30.00 TO 10.45.00
  AND FROM:14.30.00 TO 14.45.00
```

Q1 RESPONSE ANALYSIS.

The first step in the analysis is to look at Q1 response time. The process of breaking Q1 response time into its components is straight forward. It was already seen, that a state count profile can be produced for all users in Q1 during a specific period of time. This profile represents an *average* Q1 transaction. From resource manager data the average time in queue for all Q1 transactions can be found. The state count percentages are then applied to the average response time to obtain a transaction profile in seconds.

Table 5 shows a complete profile of Q1 transactions for a one hour period of time. The data shown here was taken on a Yorktown system two days before Christmas. This shows a system under relatively low load, where CPU and Paging contention are at a minimum. Under these circumstances the I/O done by the virtual machines is the largest component of Q1 response time. Notice, that in one hour of sampling of user states users were found in Q1 377 times. Care must be taken in order to have enough sample points for the desired accuracy. The number of samples may be increased by either measuring a longer period of time or by shortening the VM/Monitor recording interval (default is one minute). In the example shown the Running and Runnable states, with only 5 samples each, are the least accurate.

Table 5

Profile of an Average Q1 Transaction

USER STATUS NAME	COUNT STATUS QUEUE Q1	Percent	Elapsed Time (sec)
I/O WAIT	171	45.4	.220
PAGEWAIT	72	19.1	.093
DEFER FUNC	8	2.1	.010
INST SIM	38	10.1	.049
RUNNABLE	5	1.3	.006
RUNNING	5	1.3	.006
PSW WAIT	78	20.7	.100
Total	377	100.0	.485

DATE: 12/23/81
TIMES FROM:10.00.00 TO 11.00.00

In interpreting the data in Table 5 we must realize that there are three major types of transactions that are shown as a weighted average. The first type is the Q1 voluntary drop, which is the true trivial transaction. A Yorktown modification assured that only one Q-drop was associated with a trivial transaction in all cases. These transactions use an above average number of page reads and a below average number of I/O operations. The next type are the Q1 involuntary drop transactions, that are then scheduled in Q2.

These transactions have above average I/O and CPU use. Finally long running teleprocessing machines (PVM, VNET) spend much of their time in Q1. A Yorktown modification keeps them in queue during I/O to TP devices if the delay is less than .300 seconds. These machines are shown in a PSW wait condition. The entire 20.7 percent in PSW wait is attributable to these machines.

It is possible to validate the times spent for paging, CPU and I/O by comparing the data to some other system measures.

A scheduler trace during fifteen minutes of the hour (Table 3) showed that Q1 transactions did an average of 4.6 I/O operations per transaction. Table 5 shows that the average transaction waited for I/O to complete 0.220 seconds. Thus, dividing the I/O time by the I/O count indicates that 0.048 seconds were spent waiting for each I/O. A way to calculate the I/O time from other data is to divide the I/O state count for all users by the total state count. This yields the fraction of logged on time spent waiting for I/O, which can be multiplied by the measured logged on time to get the total I/O wait time. This in turn is divided by the measured I/O count to get the time per I/O. On a system wide basis users were found to wait .043 seconds per I/O.

A similar use of page wait state counts and page read counts shows that the average user wait time for a page read was 0.019 seconds. Resource manager data showed 5.9 page reads per Q1 transaction (Table 2) and the profile in Table 5 showed that they took 0.093 seconds. Thus the time per page read would be .016 seconds.

The smallest, and least accurate part of the calculated profile is the CPU time measure of .006 seconds (Table 5), which compares with the .027 measured by the resource manager (Table 2).

With the exception of the CPU time measure the data obtained from different sources do not show great discrepancies, and they can be accepted as reasonably "verified". The state sampling method provides better accuracy for the larger, and more important, components of response time.

TRIVIAL RESPONSE ANALYSIS.

The magnitude of trivial response time can be measured primarily through the use of the scheduler trace. It is also possible to find the average CPU, I/O count and Page read count for these transactions. The components of the response time can then be estimated based on the resource use. It is not possible to use the state sample data directly because at the time the state sample is taken it is not known whether the transaction will ultimately complete voluntarily, or become a Q2 transaction.

The basic technique is to use the state sample measures to find the average wait per CPU second, I/O, page read and to apply that factor to the known resource consumption.

The CPU, VIO and page read consumptions of the trivial transaction (from Table 4) are entered in the resource use column of Table 6. The CPU time is a direct measurement so it may be moved directly to the component column. The state sample measures (Table 5) provide the ratio of runnable states (meaning waiting for the CPU) and running state counts. This ratio is multiplied by measured CPU time to give CPU wait time. The average time per I/O for all users of the system is used as the coefficient for multiplication by the I/O count. Similarly the system wide page read time is multiplied by the page read count to get the page time component.

Table 6 Trivial Transaction Profile

Name	Resource use	Factor	Time Component
CPU Time	.019 Measured CPU	1.0	.019
CPU wait	.019 Measured CPU	1.0 Runnable/Running	.019
I/O Time	1.9 I/O count	.043 I/O time	.082
Page Time	4.9 Pagein count	.019 Pagein time	.093
Subtotal			.213
Def Func	1.9 I/O count	.002 DF Per I/O	.004
Inst Sim	1.9 I/O count	.010 Inst Sim Per I/O	.019
Total			.236
Measured from scheduler trace			.238

DATE: 12/23/81 TIMES FROM:10.30.00 TO 10.45.00

In Table 6 the subtotalled trivial components fall slightly short of the measured total. These components are the largest components, and the ones with the most confidence. The remaining time is accounted for in deferred function and instruction simulation states. These times are related to I/O

operations, so this analysis makes them proportional to the I/O count. The deferred function factor is developed by first dividing the deferred function state count for all Q1 by the I/O wait count for all Q1 (in this case 8 divided by 171). This shows the deferred function time relative to I/O time. This fraction is then multiplied by the mean I/O time (0.043 seconds) to get the deferred function factor (0.002 seconds per I/O). Similarly the instruction simulation factor is developed by dividing the instruction simulation count by the I/O wait count and multiplying by the I/O time.

It is interesting to note that the profile of trivial transactions is significantly different from the profile of all Q1 transactions. Most of the difference is due to the fact that trivial transactions use a very different resources than a Q1 transactions. Some of the difference is accounted for by the twelve percent higher page reads per transaction observed during the trivial transaction analysis. Table 7 lists side side by side the Q1 and trivial transaction profiles. Even on this lightly loaded system paging is the largest component of the trivial response profile. The I/O time is relatively less important to trivial transactions than Q1 transactions, because the number of I/O operations is lower.

The choice of the factor for pagein time shows one of the choices that must be made in this type of analysis. The 0.019 seconds per page read was derived from all state samples of all users on the system. Alternatively, the 0.016 seconds, derived from the analysis of Q1 users, could have been used. VM/SP does not treat Q1 and Q2 page reads differently so the page read time for all users is presumed to be more accurate because it reflects a larger number of state samples.

Table 7 Trivial Transaction Profile

USER STATUS	Q1 TIME (ms.)	TRIVIAL TIME (ms.)	Q1 PCT	TRIVIAL PCT
I/O WAIT	220	82	45	34
PAGEWAIT	93	93	19	39
DEFER FUN	10	4	2	2
INST SIM	49	0	10	2
CPU TIME	6	19	1	8
CPU WAIT	6	19	1	8
PSW WAIT	100	0	21	9
TOTAL	485	236	100	99

Q1 data covers 10:00AM to 11:00AM
Trivial transaction data covers 10:30AM to 10:45AM

Table 8 shows mean response times across 5 minute intervals for trivial transactions. The previous trivial response profile predicted a mean response of 0.236 seconds which is consistent with the 0.238 measured during fifteen minutes of the hour. It also shows the distribution of response times for the interval. The important thing to notice here is that mean response time is not really typical. The median response time is about half of the mean response time.

Table 8 Scheduler Trace Data

REPORT SELECTION NAME: TRIV_RESP_BY_TOD

MIN TIME	MAX TIME	TRAN PER SEC	MEAN TRIV RESP	PCT RESP <=.100	PCT RESP <=.250	PCT RESP <=.500	PCT RESP <=1.0	PCT RESP >1.0
10.30	10.35	7.331	0.268	35.6	35.5	15.2	8.0	5.6
10.35	10.40	9.782	0.230	44.7	30.7	12.6	7.3	4.7
10.40	10.45	10.253	0.215	44.2	33.4	11.9	6.5	3.9
14.15	14.20	10.706	0.195	54.2	23.9	10.5	7.4	3.9
14.20	14.25	10.344	0.207	46.6	29.0	13.3	7.5	3.5
14.25	14.30	11.718	0.245	44.1	28.2	14.4	8.4	4.9

Q1 VOLUNTARY DROPS FROM QUEUE (RESPONSE < 4 SECONDS)

THE USE OF THE TRIVIAL TRANSACTION PROFILE.

A methodology was provided for obtaining the component parts of the trivial transaction. The transaction profile directly leads to remedial measures to be taken if it is desired to improve the transaction response time. It goes almost without saying, that the easiest way of improving the response time is by attacking (if possible) its largest component. In Table 6 the largest component is the paging component. Consequently, if better response time is desired, then the paging response time must be improved, by providing (for example) more main storage, faster paging devices, or limiting the number of users on the system. Similar conclusions can be drawn about I/O or CPU components if they represent the largest component of the response time.

SUMMARY.

A methodology was shown for using state samples of active tasks on a time shared system to provide a profile of the time spent using various resources. These data were then used in conjunction with continuously measured response time data to determine the magnitude of the components in the profile. Resource usage data was used in order to calculate the components of the elapsed time of trivial transactions. Finally the trivial transaction profile so obtained could be used to pinpoint the chief bottlenecks preventing the system from obtaining better response times.

ACKNOWLEDGMENTS

We want to thank Messrs. W. Buco, J. Gindele, J. Greenberg, R. Miles, R. Newson and D. Patterson who have also been contributing their ideas to the project leading to this paper.

REFERENCES.

1. W. Tetzlaff, "State Sampling of Interactive VM/370 Users," *IBM Systems Journal,* 18, No. 1, (1979).

2. W. J. Doherty and R. P. Kelisky, "Managing VM/CMS systems for user effectiveness," *IBM Systems Journal,* 18, No. 1, (1979).

3. A. J. Thadhani, "Interactive User Productivity," *IBM Systems Journal,* 20, No. 4, 407-423 (1981).

4. *VM/370 System Programmer's Guide* Document GC20-1807, IBM Corporation, Data Processing Division, White Plains, New York 10604.

5. Y. Bard, "Performance Analysis of Virtual Memory Time-Sharing Systems," *IBM Systems Journal,* 14,no 1, 366-384 (1975).

A VM/SP Performance Management Information System

John Story

Texas Instruments, Inc
P. O. Box 405 M/S 3407
Lewisville, TX 75067

The analysis and capacity management of any computer system requires data to be collected on a periodic basis. This data can become voluminous and is very often difficult to present in any understandable manner. The purpose of this paper is to present an outline of a Performance Management Information System, and to show how it was implemented at Texas Instruments on an IBM VM/SP system running mostly CMS users. This system gathers the data, analyzes the data, archives the data, produces reports and graphical displays of the data, on a periodic basis.

Key words: Performance measurement; IBM VM/SP; VMAP; performance evaluation; performance prediction; graphical presentation.

1.0 Introduction

The study and evaluation of computer system performance requires a large amount of detailed data for analysis. This data is usually collected by some sort of system monitor (hardware or software, inherent in the operating system or add-on) which yields large amounts of raw data that must then be evaluated by calculating performance variables from the raw data variables. In some cases the software for this process is provided by the monitor vendor. In every case the amount of data can be staggering and requires some systematic method of processing.

This paper describes a system developed at Texas Instruments, Inc. to collect, analyze, report, archive and graphically display the performance data from a medium size installation. Parts of the system are "off the shelf" items from the vendor, and the rest were locally developed.

2.0 Motivation for the System

A Performance Management Information System (PMIS) is necessary for efficient evaluation of a computer system in order to

a. Standardize report format,

b. Standardize reporting methodology,

c. Standardize archive procedures, and

d. Insure continuity of data available.

The data collected by the PMIS is used for

a. Prediction of performance under projected workloads,

b. Tuning the existing system for optimal performance, and

c. Tracking system performance to identify trouble spots.

The evaluation cycle which utilizes the data is depicted in Figure 1. Data from the PMIS is instrumental in making the decisions involved in

a. Tuning,

b. Capacity planning,

c. Workload forecasting,

d. Modeling the system, and

e. Cost/Rate setting.

3.0 Operating Environment

Texas Instruments operates a number of computer systems in support of Engineering and Software Development. VM/SP PMIS operates on a VM (Virtual Machine) system with most users running CMS

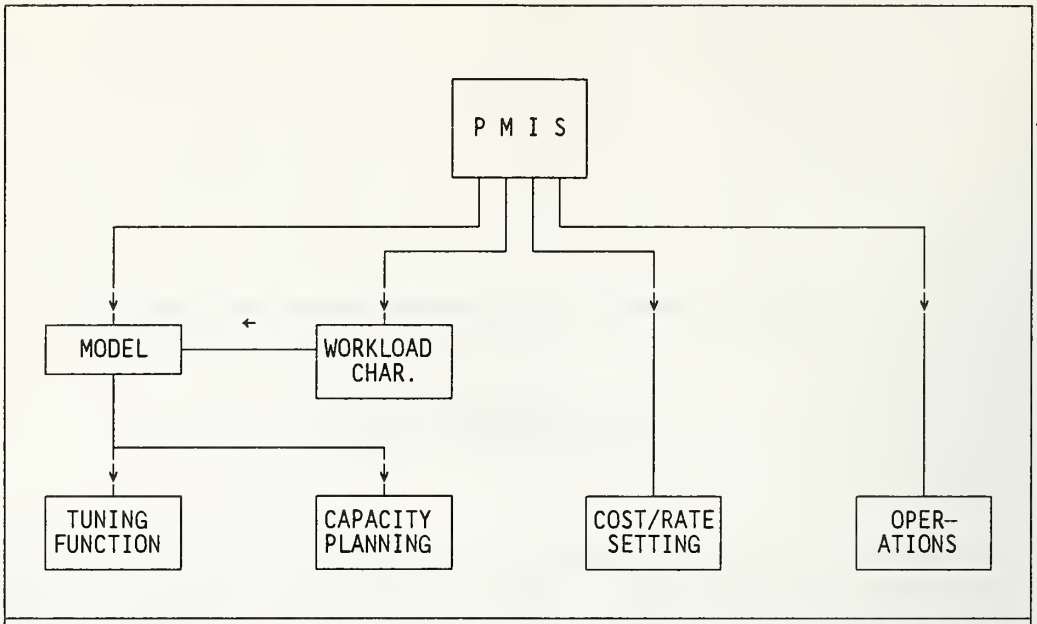


Figure 1. Utilization of PMIS Data in the Evaluation Cycle

(Conversational Monitor System). The workload consists primarily of software development and Engineering development from individual terminals scattered around the corporation. The current hardware is an IBM model 4341 CPU with a mixture of 3350 and 3370 disk drives. The control program of VM (CP) contains a monitor which gathers data concerning system operation, and IBM has a data reduction software package to analyze the data (VMAP).

4.0 Design Criteria

During the design stages of PMIS, the following items were considered

a. Utilize as much current software as possible,

b. Allow full usage of current software incorporated in the system,

c. Provide high quality graphical output of data,

d. Provide for little or no manual intervention in operation, and

e. Archive data for later retrieval.

The end product fulfilled almost all of these design requirements. Both the CP Monitor and VMAP were utilized, high quality graphical output was produced, data was archived for later retrieval, and most of the process is automatic.

5.0 Operation of the System

The daily operation of the system is automatic. Data is collected continuously by the CP Monitor and then, at the end of the day, the data reduction step automatically calculates the performance variables, creates daily reports, and adds to the trend files.

The class of data collected by the monitor and the time of day that data is collected is automatically set and modified at the appropriate times.

On a monthly basis, the daily averages are retrieved and converted to weekly averages for the graphical output and plotted. The plots maintain a 6 month plotting scope, and the new data for this month forces the old data to "roll off" the graph, so that any plot covers the previous 24 weeks. The plots are formatted automatically with the PLOT UTILITY PACKAGE (PUP) developed at TI for output to the plotter. The data files for input to PUP are automatically updated after the weekly averages are calculated.

Also on a monthly basis, the data for the month is transfered to tape for storage as historical data. The basic output of the cycle is the daily reports and the monthly reports. The entire system is diagrammed in Figure 2.

6.0 Description of the Data Base

The IBM product VMAP collects historical data called ACUM TREND files. These files contain values of certain variables that are measured by the CP Monitor and calculated by VMAP. In addition to the variables included in the ACUM TREND files, there are some other variables of interest which are not included in the file. For these other variables, PMIS maintains a separate ACUM file from which to draw historical conclusions. Some of these variables are

a. Prime and non_prime CPU utilization,

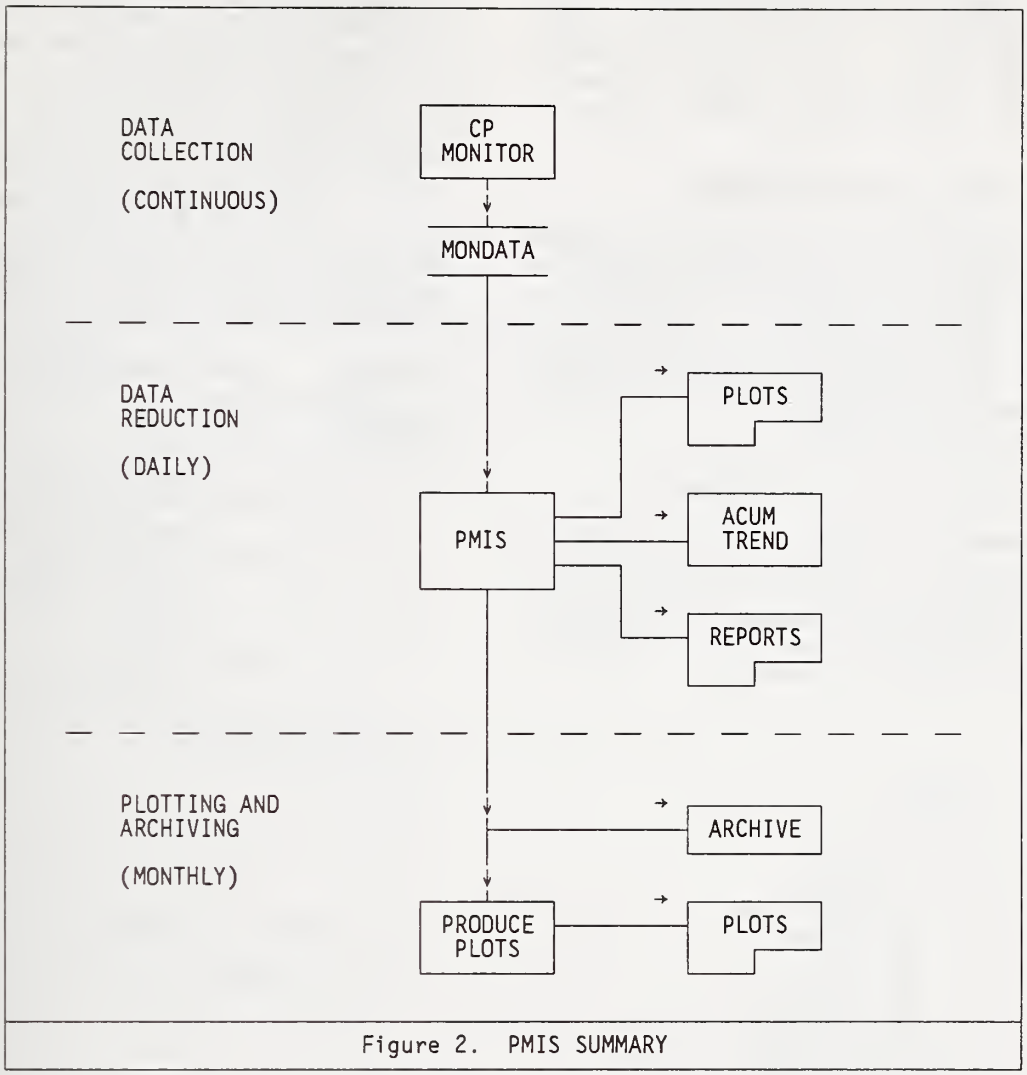


Figure 2. PMIS SUMMARY

b. Response time by time of day,

c. Channel utilization,

d. CPU utilization by user class,

e. 95th percentile of response time, and

f. 95th percentile of expansion factor.

7.0 Processing Techniques

The bulk of the data processing for PMIS is done by the IBM program VMAP. This program creates reports and history files that are used by PMIS as output (reports) and input (reports and history files).

The daily ACUM file created by VMAP is used to create a list of response times from which the 95th percentile of response for each hour of the day is calculated along with the average response for each hour of the day. This information is averaged over the entire month to produce the output shown in FIGURE 2 in section 10.0, "Sample Output of PMIS".

The 95th percentile and average expansion factor for each hour of the day is collected and reported in the same manner as response time, and is shown in FIGURE 4 in section 10.0, "Sample Output of PMIS".

The channel utilization is not collected in the ACUM file, but is reported on the VMAP "OUTDASD" report. The data presented is collected in another historical file and averaged over each week and reported at the end of the month as shown in FIGURE 10 in section 10.0, "Sample Output of PMIS".

The prime shift and non-prime shift CPU utilization is calculated from the statistical reports called "OUTSTAT". Sample output is shown in FIGURE 5 in section 10.0, "Sample Output of PMIS". The day is divided into prime and non-prime time periods, and the VMAP program is run against the data for each period separately to provide prime and non-prime data.

The CPU utilization of user classes is provided by reading the data on the VMAP report "OUTUSER" and gathering the users into pre-deterimined

classes. Currently, the classes represent SYSTEM ids, BATCH processing, DOSVSE, RSCS, VSEVTAM, PVM, one or two large CMS users, and all other CMS users. These classes may be modified at any time. Sample output is shown in FIGURE 7 in section 10.0, "Sample Output of PMIS".

8.0 Descripton of Output

The daily reports for PMIS are identical to VMAP reports. They may include any of the VMAP reports, including the plots produced on the printer. The monthly reports are graphical in nature and show the trend for the last six months or the profile over the 24 hour day. Sample output of both types is found in section 10.0, "Sample Output of PMIS".

9.0 Utilization of Data Reported

The data gathered has been used to classify users into groups in order to characterize the workload. This has been used to simplify the input into some analytic models which then predict the performance of new hardware configurations that are under consideration.

The data has also been used to predict response time under various workloads. A linear statistical model was built which utilized the memory size, number of users, and the average workset size reported by PMIS to predict response time. The results of the study are summarized in Figure 3, which indicates a good match between actual and predicted values of response time.

10.0 Sample Output of PMIS

The following pages contain some sample output reports and plots from PMIS. The sample output is in the following order:

OUTHDR VMAP Header Report

OUTPLOT VMAP Plots

User Activity

User Frequency Distribution

CPU Utilization

Paging

Storage Utilization

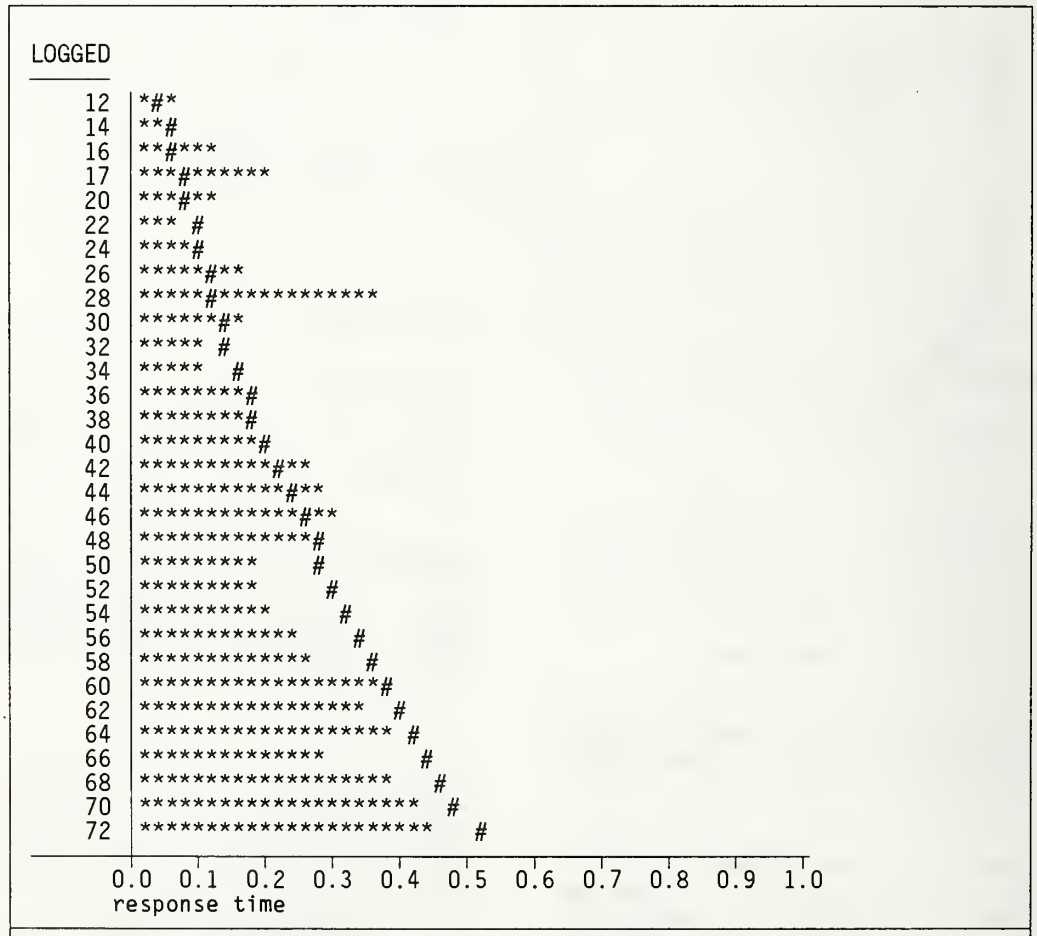


Figure 3. Actual vs Predicted Response Time
Bars of * are the actual times
are the predicted times

	Resource Availability Index
	Expansion Factor Frequencies
	Trivial Response Frequencies
OUTPERF	VMAP System Performance Summary
OUTUTIL	VMAP Resource Utilization Summary
OUTUSER	VMAP User Resource Utilization Summary
OUTCMND	VMAP Partial Command Analysis
FIGURE 1	LICC INTERACTIVE RESOPONSE TIMES
FIGURE 2	LICC RESPONSE TIME PROFILE
FIGURE 3	LICC EXPANSION FACTOR
FIGURE 4	LICC EXPANSION FACTOR PROFILE
FIGURE 5	LICC CPU UTILIZATION
FIGURE 6	LICC CPU UTILIZATION PROFILE
FIGURE 7	LICC CPU UTILIZATION BY CLASS
FIGURE 8	LICC USERID ANALYSIS
FIGURE 9	LICC PAGING RATE
FIGURE 10	LICC CHANNEL UTILIZATION
FIGURE 11	LICC USERS IN RESOURCE WAIT

```
* * * SUMMARY OF VMAP RUN * * * * * * * * * * * * *

  MONITOR INPUT HEADER INFORMATION

CREATION DATE: 05/10/82,    TIME: 08:00:14,     CLASSES: 012 4 6

  SUMMARY DATA

                TI - LICC VM/370 MONITOR ANALYSIS

FIRST TIME SELECTED                    08:00:14
LAST TIME SELECTED                     18:00:00

TOTAL TIME ANALYSED                    09:57:04

NUMBER OF SECONDS ANALYSED               35,824

NUMBER OF RECORDS ANALYSED             1080,149
NUMBER OF RECORDS BYPASSED                    0
NUMBER OF SUSPENSION RECORDS                  0

TOTAL NUMBER OF USERS                       155

CPU MODEL - SERIAL NUMBER     4341 -    10085

USER STARTING MONITOR                  OPERATOR

SOFTWARE VERSION.LEVEL.PTF           01.01.0110

NUCLEUS SIZE                           329,392
V=R AREA                                     0
DYNAMIC PAGING AREA                  7,094,272
FREE STORAGE SIZE                      819,536
INTERNAL TRACE TABLE                   118,784

STORAGE TOTAL                        8,361,984


MONTAPE VERSION/LEVEL                      3.4

MONTAPE OPTIONS SPECIFIED
    PAGTYP= 3370, NPAGDEV=  5, NPAGCYL= 95976
    PERF UTIL PAGE SERV RESMGR MIGRATION RESP USER IO SAVE RESTORE


MONITOR CLASSES ENABLED   # RECORDS

    00  PERFORM    1          1816
    01  RESPONSE   1        169136
    02  SCHEDULE   1        873090
    04  USER       1         34304
    05  INSTSIM    0             0
    06  DASTAP     1          1802
    07  SEEKS      0             0
    08  SYSPROF    0             0
```

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
USER ACTIVITY

BASE VARIABLE - TOD TIME OF DAY

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
L	LOGGED	10.00		33.43	71.00	# OF USERS LOGGED ONTO SYSTEM
A	ACTIVE	10.00		17.12	67.00	# USERS ACTIVE IN A SAMPLE INTERV

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
07:00	1 A L 1	15		15	
07:30	1 A L 1	30		30	
08:00	1 A L 1	30		30	
08:30	1 A L 1	30		30	
09:00	1 A L 1	30		30	
09:30	1 A L 1	30		30	
10:00	1 A L 1	30		30	
10:30	1 A L 1	30		30	
11:00	1 A L 1	30		30	
11:30	1 A L 1	30		30	
12:00	1 A L 1	30		30	
12:30	1 A L 1	30		30	
13:00	1 A L 1	30		30	
13:30	1 A L 1	30		30	
14:00	1 A L 1	30		30	
14:30	1 A L 1	30		30	
15:00	1 A L 1	30		30	
15:30	1 A L 1	30		30	
16:00	1 A L 1	30		30	
16:30	1 A L 1	30		30	
17:00	1 A L 1	30		30	
17:30	1 A L 1	29		29	
	0----1----2----3----4----5----6----7----8----9----0				
TOTAL		644		644	

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
ACTIVE USER FREQUENCY DISTRIBUTION

BASE VARIABLE - ACTIVE # USERS ACTIVE IN A SAMPLE INTERVAL

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	FREQ	15.00		1439.00	105.01	FREQUENCY DISTRIBUTION OF BASE VA

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
6	1.........*	1	30	4		
8	1	1	0	4		
10	1....*	1	15	6		
12	1....*	1	15	9		
14	1....*	1	15	11		
16	1	1	0	11		
18	1.........*	1	30	15		
20	1....*	1	15	18		
22	1...*	1	14	20		
24	1.........*	1	30	24		
26	1....*	1	15	27		
28	1.............*	1	45	33		
30	1....*	1	15	36		
32	1.................*	1	60	45		
34	1................................*	1	105	61		
36	1............................*	1	90	74		
38	1............................*	1	90	88		
40	1.............*	1	45	95		
42	1....*	1	15	97		
44	1....*	1	15	100		
	0----1----2----3----4----5----6----7----8----9----0					
TOTAL			659			

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
CPU UTILIZATION

BASE VARIABLE - TOD TIME OF DAY

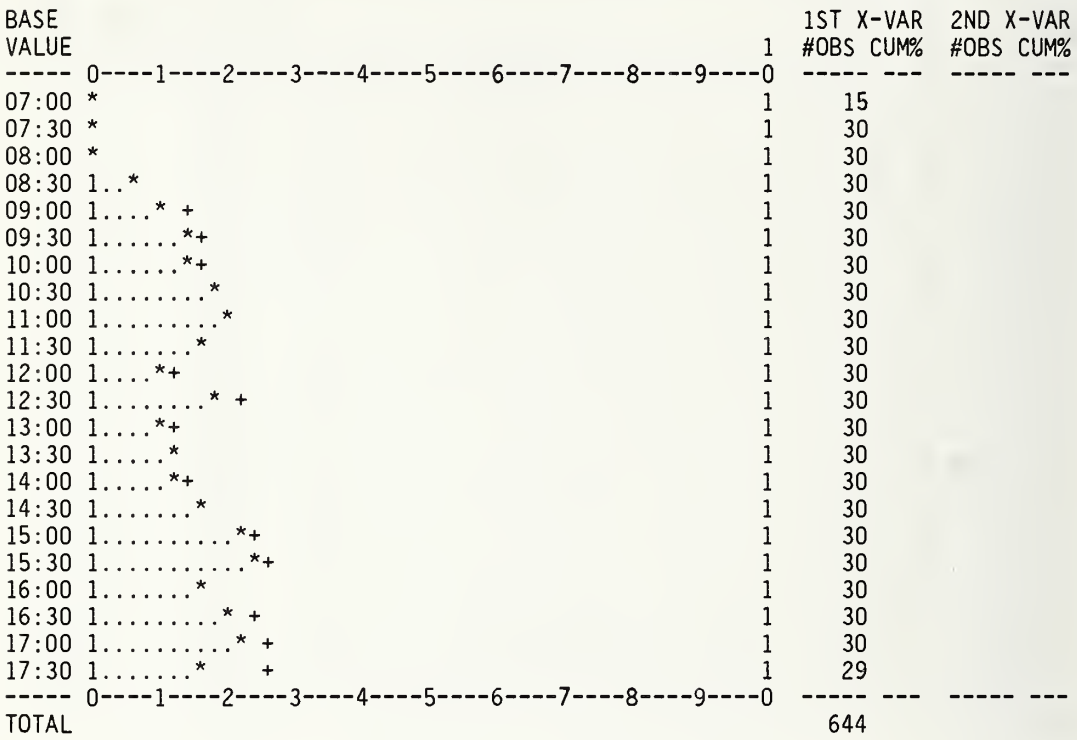
SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	TOTCPU	10.00		53.54	100.00	CPU TOTAL PCT UTILIZATION

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
07:00	1.....* 1	15			
07:30	1.....* 1	30			
08:00	1.....* 1	30			
08:30	1.....* 1	30			
09:00	1.....* 1	30			
09:30	1.....*1	30			
10:00	1.....*1	30			
10:30	1.....* 1	30			
11:00	1.....* 1	30			
11:30	1.....* 1	30			
12:00	1.....* 1	30			
12:30	1.....*1	30			
13:00	1.....*1	30			
13:30	1.....*1	30			
14:00	1.....*	30			
14:30	1.....*1	30			
15:00	1.....*1	30			
15:30	1.....*1	30			
16:00	1.....*	30			
16:30	1.....*1	30			
17:00	1.....*1	30			
17:30	1.....*	29			
TOTAL		644			

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
PAGING

BASE VARIABLE - TOD TIME OF DAY

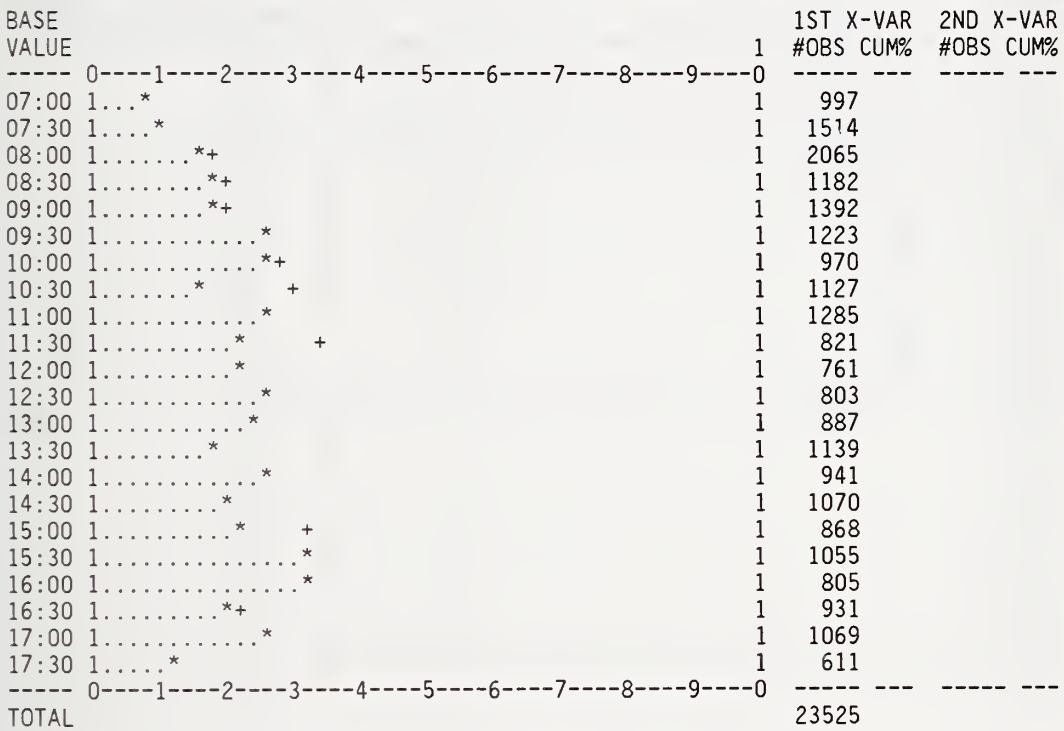
SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	PAGERATE	10.00		7.13	53.80	PAGING RATE PER SECOND
+						MAXIMUM VALUE OF PAGERATE



05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
STORAGE UTILIZATION

BASE VARIABLE - TOD TIME OF DAY

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	STGUTIL	12.50		22.41	61.68	PCT MAIN STORAGE UTILIZATION
+						MAXIMUM VALUE OF STGUTIL



05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
RESOURCE AVAILABILITY INDEX

BASE VARIABLE - TOD TIME OF DAY

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	RAI	0.10		0.80	1.00	RESOURCE AVAILABILITY INDEX

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
07:00	1.....*	15			
07:30	1.....*	30			
08:00	1.....*	30			
08:30	1.....*	30			
09:00	1.....*	30			
09:30	1.....*	30			
10:00	1.....*	30			
10:30	1.....*	30			
11:00	1.....*	30			
11:30	1.....*	30			
12:00	1.....*	30			
12:30	1.....*	30			
13:00	1.....*	30			
13:30	1.....*	30			
14:00	1.....*	30			
14:30	1.....*	30			
15:00	1.....*	30			
15:30	1.....*	30			
16:00	1.....*	30			
16:30	1.....*	30			
17:00	1.....*	30			
17:30	1.....*	29			
TOTAL		644			

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
EXPANSION FACTOR FREQUENCIES

BASE VARIABLE - EXPF EXPANSION FACTOR FOR MAJOR COMMANDS

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	FREQ	5470.00		1439.00	54669.93	FREQUENCY DISTRIBUTION OF BASE VA

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
0	1.....*	1	37028	21		
2	1.....*1	1	54669	53		
4	1.....*	1	52399	83		
6	1......*	1	7704	87		
8	1........*	1	10879	94		
10	1.......*	1	9269	99		
12	1	1	0	99		
14	1	1	0	99		
16	1	1	0	99		
18	*	1	363	99		
20	1	1	. 0	99		
22	*	1	619	100		
TOTAL			172933			

05/10/82 00:00:00 DAILY PLOTS FOR LICC SYSTEM B
TRIVIAL FREQUENCIES

BASE VARIABLE - TRIVRESP TRIVIAL (INTERACTIVE) RESPONSE TIME

SYM	CROSS-VAR	SCALE	ORG	AVG	MAX	DESCRIPTION
*	FREQ	835.00		1439.00	8340.75	FREQUENCY DISTRIBUTION OF BASE VA

BASE VALUE	0----1----2----3----4----5----6----7----8----9----0	1ST X-VAR #OBS	1ST X-VAR CUM%	2ND X-VAR #OBS	2ND X-VAR CUM%
0.0	1.....* 1	1682	6		
0.1	1.....* 1	6175	31		
0.2	1.....* 1	5077	52		
0.3	1.....*1	8340	86		
0.4	1.....* 1	3048	98		
0.5	* 1	75	99		
0.6	* 1	109	99		
0.7	1 1	0	99		
0.8	* 1	75	99		
0.9	* 1	38	100		
TOTAL		24622			

	<----------ELAPSED TIME---------->				<-------FREQUENCY------>			<-------------CPU UTILIZATION------------->			
						% OF		<-VIRTUAL CPU TIME->		<--TOTAL CPU TIME-->	
COMMAND	MINIMUM	MAXIMUM	AVERAGE	STDEV	NO. OBS	TOTAL	CUM %	AVERAGE	ACCUMULATED	AVERAGE	ACCUMULATED
ACCESS	0.5	236.1	49.4	64.3	19	2.8	2.8	0.53	10.12	0.95	18.01
ASMH	2.8	66.9	45.5	37.0	3	0.4	3.2	2.00	6.00	2.68	8.04
ASSEMBLE	16.0	70.8	60.4	16.9	3	0.4	3.7	4.21	12.64	5.48	16.43
BASIC	11.8	244.6	73.1	84.1	7	1.0	4.7	2.40	16.82	4.17	29.16
BM	0.9	4298.6	437.9	1210.2	14	2.1	6.7	0.26	3.66	0.43	5.96
COMPARE	10.3	1019.5	164.3	256.0	14	2.1	8.8	0.80	11.14	1.89	26.50
COPYFILE	1.2	17.3	7.7	6.9	8	1.2	10.0	0.09	0.71	0.23	1.81
COST	6.3	6.3	6.3	0.0	1	0.1	10.1	0.02	0.02	0.10	0.10
CP	0.2	4260.8	939.2	1339.4	22	3.2	13.3	1.50	32.97	2.24	49.36
DEFINE	16.0	1769.9	606.0	752.1	5	0.7	14.1	1.16	5.78	2.19	10.94
DETATCH	1.5	7.1	4.3	0.0	2	0.3	14.4	0.03	0.07	0.06	0.13
DISK	14.6	14.6	14.6	0.0	1	0.1	14.5	0.18	0.18	0.32	0.32
EDIT	16.0	3454.9	2705.5	0.0	2	0.3	14.8	0.70	1.41	1.07	2.15
ERASE	16.0	16.5	16.5	0.0	1	0.1	15.0	0.13	0.13	0.21	0.21
EXEC	0.7	1925.4	642.7	1110.9	3	0.4	15.4	0.07	0.20	0.18	0.55
FLIST	0.7	6462.5	364.1	820.0	296	43.4	58.8	1.69	499.83	2.30	680.54
FORTGI	6.7	5400.2	472.2	1297.9	20	2.9	61.7	11.29	225.89	12.39	247.87
FORTHX	16.0	452.5	156.9	148.0	7	1.0	62.8	19.19	134.35	20.60	144.22
GENMOD	9.5	9.5	9.5	0.0	1	0.1	62.9	0.11	0.11	0.13	0.13
GETMEM	16.0	246.2	142.2	77.0	5	0.7	63.6	5.73	28.63	7.35	36.76
HELP	16.0	131.6	131.6	0.0	1	0.1	63.8	1.81	1.81	2.16	2.16
INDICATE	10.6	1861.5	736.5	987.8	3	0.4	64.2	0.16	0.47	0.36	1.07
IPL	16.0	72.4	72.4	0.0	1	0.1	64.4	0.40	0.40	1.01	1.01
JES*	16.0	140.6	140.6	0.0	1	0.1	64.5	0.76	0.76	1.37	1.37
LINK	1.2	49.8	11.3	16.2	9	1.3	65.8	0.02	0.18	0.05	0.46
LINK990	16.0	149.1	90.5	42.5	4	0.6	66.4	8.37	33.47	10.97	43.87
LISTFILE	4.9	1209.7	227.1	374.8	10	1.5	67.9	0.20	1.97	0.33	3.33
LOAD	2.5	25.0	16.4	9.6	5	0.7	68.6	0.94	4.68	1.51	7.55
MACLIB	2.2	3771.2	531.7	1251.7	9	1.3	69.9	0.12	1.10	0.27	2.41
MAIL	16.0	39.9	39.9	0.0	1	0.1	70.1	0.22	0.22	0.40	0.40
MSG	8.2	2401.4	715.5	1138.6	4	0.6	70.7	0.04	0.15	0.11	0.45
MT	16.0	39.0	39.0	0.0	1	0.1	70.8	0.07	0.07	0.17	0.17
ORDER	11.8	29.6	20.7	0.0	2	0.3	71.1	0.21	0.43	0.30	0.61
PRINT	2.1	1973.3	359.1	739.1	18	2.6	73.8	0.37	6.70	0.80	14.42
PUNCH	0.4	402.7	75.6	134.5	9	1.3	75.1	0.29	2.61	0.48	4.34
PURGE	16.0	45.7	45.7	0.0	1	0.1	75.2	0.01	0.01	0.02	0.02
QN*	14.2	1403.4	157.0	393.7	12	1.8	77.0	0.45	5.43	0.88	10.54
QUERY	2.3	5644.0	604.2	1261.3	67	9.8	86.8	3.78	253.44	7.35	492.64
RORPULL	3.5	3136.3	595.3	1144.6	10	1.5	88.3	1.08	10.77	2.33	23.31
READCARD	1.3	44.0	15.7	17.3	7	1.0	89.3	0.52	3.66	1.07	7.52
REWIND	0.3	0.3	0.3	0.0	1	0.1	89.4	0.01	0.01	0.01	0.01
RUN	16.0	175.1	72.6	88.8	3	0.4	89.9	2.38	7.14	2.85	8.56
SAS	16.0	2050.8	1046.4	0.0	2	0.3	90.2	14.72	29.45	25.73	51.45
SORT*	6.0	6.0	6.0	0.0	1	0.1	90.3	0.01	0.01	0.03	0.03
TAPE	0.0	397.4	118.2	187.9	4	0.6	90.9	2.66	10.63	6.02	24.08
TOISK	16.0	67.7	67.7	0.0	1	0.1	91.1	0.30	0.30	0.66	0.66
TYPE	16.0	4471.1	1118.5	1543.8	8	1.2	92.2	0.71	5.69	1.98	15.86
UPDATE	16.0	35.6	35.6	0.0	1	0.1	92.4	1.55	1.55	2.23	2.23
WHOME	3.6	2117.9	734.9	1198.3	3	0.4	92.8	0.30	0.90	0.71	2.14
XE	16.0	79.2	63.1	0.0	2	0.3	93.1	0.24	0.48	0.38	0.76
XEDIT	3.2	3321.4	273.8	583.5	47	6.9	100.0	1.33	62.52	1.76	82.69

DATE 05/10/82 TIME 08:00:14 SUMMARY OF SYSTEM PERFORMANCE LOG PAGE 1

	<-USER LEVEL->				<--CPU-->		<-MAIN STORAGE->			<--PAGING-->		<IO>	<---RUNNING STATUS---->					<TRNSACTN>		<RESPONSE TIME>		
													<-PCT OF ACTIVE USERS->					RATE PER		VALUES		
	LOG	ACT		RES	TOT	TOT/	PCT	SAT	CONTE	RATE	%PRIME	VIO	<IN RESOURCE WAIT>				VOL	MINUTE				MAJ
TIME	GED	IVE	IN Q	AVAL INDX	PCT	VIRT RATIO	UTIL	FAC TOR	NTION RATIO	/SEC	PAGDEV NEEDED	RATE /SEC	CPU	STG	PAG	I/O	WAIT	TRIV	MINOR	TRIV	MINOR	EXPF
08.00	30	17	2.5	.91	95	1.3	21	0.7	1.0	5	2	41	12	0	0	0	83	5	0	0.0	0.0	2.1
08.15	35	22	3.1	.90	96	1.3	25	1.0	1.0	7	4	23	4	0	0	2	85	37	1	0.2	3.2	2.6
08.30	42	25	3.7	.89	97	1.2	22	0.9	1.0	6	6	24	4	0	0	2	87	3	0	0.1	0.0	2.4
08.45	46	29	4.9	.86	97	1.3	25	1.0	1.0	9	6	50	5	0	1	5	84	42	0	0.2	9.9	3.9
09.00	52	33	6.4	.84	98	1.5	25	1.1	1.0	15	7	61	5	0	1	4	86	6	0	0.3	10.6	10.1
09.15	56	35	8.6	.78	99	1.6	34	1.1	1.0	18	8	91	9	0	2	8	79	52	2	0.4	5.2	8.3
09.30	54	35	9.3	.76	100	1.5	31	1.1	1.0	11	8	78	14	0	1	6	78	5	0	0.3	0.0	7.8
09.45	56	36	9.3	.77	100	1.5	35	1.1	1.0	18	9	78	11	0	1	8	79	59	2	0.4	4.2	11.1
10.00	55	35	7.2	.82	98	1.4	29	1.0	1.0	12	9	66	7	0	2	5	85	3	0	0.4	4.6	7.1
10.00	0	0	0.0	.00	0	0.0	0	0.0	1.0	0	0	0	0	0	0	0	0					
10.00	0	0	0.0	.00	0	0.0	0	0.0	1.0	0	0	0	0	0	0	0	0					
10.15	59	34	5.0	.87	69	1.8	21	1.1	1.0	19	9	63	1	0	2	6	88	89	1	0.4	7.3	4.3
10.30	61	39	7.1	.84	86	1.8	38	1.2	1.0	19	9	67	3	0	2	5	88	6	0	0.3	6.6	20.5
10.45	62	39	6.8	.85	98	1.7	33	1.1	1.0	20	10	54	7	0	2	3	87	65	1	0.3	9.2	5.5
11.00	64	40	6.7	.86	97	1.4	25	1.1	1.0	21	10	56	6	0	2	4	86	3	0	0.7	4.0	5.7
11.15	61	36	6.7	.84	97	1.4	28	1.1	1.0	18	10	62	9	0	1	5	85	62	2	0.4	25.3	5.2
11.30	55	33	8.6	.77	98	1.4	44	1.1	1.0	15	9	56	8	0	1	5	85	4	0	0.6	475.9	17.9
11.45	49	30	6.1	.83	97	1.5	28	1.0	1.0	14	9	74	6	0	2	7	85	41	1	0.3	107.3	4.7
11.59	45	26	5.3	.84	97	1.6	25	1.0	1.0	6	7	70	5	0	1	6	86	0	0	0.4	0.0	1.9
12.00	46	19	5.5	.76	99	1.7	31	0.7	1.0	8	7	74	11	0	0	0	87	11	0	0.2	0.0	15.6
12.15	48	27	5.8	.82	99	1.7	33	1.0	1.0	22	7	66	8	0	0	5	85	29	1	0.2	34.0	5.3
12.30	52	32	6.0	.84	99	1.4	27	1.1	1.0	15	8	40	8	0	1	3	86	2	0	0.2	0.0	9.3
12.45	52	29	6.4	.81	100	1.3	32	1.0	1.0	11	9	32	12	0	0	1	85	51	1	0.2	10.0	5.9
13.00	56	33	7.2	.81	100	1.4	28	1.1	1.0	12	9	44	11	0	1	2	85	4	0	0.4	5.1	8.1
13.15	59	35	6.2	.85	100	1.3	24	1.0	1.0	12	9	44	8	0	1	4	86	81	2	0.2	4.2	5.5
13.30	60	37	5.6	.87	100	1.2	22	1.1	1.0	13	9	39	7	0	1	3	86	3	0	0.2	7.1	4.8
13.45	62	38	7.4	.83	100	1.3	34	1.1	1.0	16	9	50	10	0	1	3	85	82	3	0.3	5.6	7.2
13.59	60	36	8.3	.79	100	1.2	40	1.0	1.0	11	9	45	8	0	1	1	87	1	0	0.1	75.2	5.6
14.00	60	14	5.7	.67	100	1.2	24	0.3	1.0	7	9	27	14	0	0	0	90	11	0	0.3	0.0	1.7
14.15	64	38	6.4	.86	100	1.3	25	1.1	1.0	17	9	49	7	0	2	3	87	74	2	0.3	14.7	5.4
14.30	63	41	7.5	.84	100	1.2	26	1.2	1.0	16	10	37	8	0	1	2	87	6	0	0.3	0.0	8.2
14.45	62	38	7.3	.83	100	1.3	28	1.2	1.0	25	10	45	9	0	2	3	86	89	1	0.4	7.1	5.5
15.00	61	38	11.2	.73	100	1.3	42	1.2	1.0	23	10	40	11	0	2	2	84	5	0	0.5	0.0	24.4
15.15	64	38	10.5	.75	100	1.3	42	1.1	1.0	21	11	38	15	0	1	2	82	71	2	0.4	4.9	11.6
15.30	68	44	10.0	.80	100	1.3	38	1.2	1.0	27	10	64	7	0	2	6	84	2	0	0.9	359.8	38.5
15.45	69	41	10.0	.78	100	1.2	41	1.2	1.0	18	10	50	10	0	2	7	82	59	2	0.4	7.0	9.6
15.59	64	42	11.7	.75	100	1.3	40	1.1	1.0	17	9	54	8	0	1	4	84	1	0	2.1	0.0	8.1
16.00	62	24	7.4	.73	100	1.3	21	0.6	1.0	25	9	68	0	0	0	8	89	28	0	0.2	0.0	6.3
16.15	63	40	6.2	.87	100	1.2	26	1.1	1.0	17	9	49	6	0	2	3	87	69	1	0.3	5.2	4.7
16.30	62	43	8.4	.83	100	1.3	29	1.1	1.0	25	9	53	7	0	4	2	82	3	0	0.9	5.6	9.5
16.45	59	39	8.8	.80	100	1.4	34	1.1	1.0	28	9	60	10	0	2	4	82	74	2	0.4	4.6	7.2
17.00	49	33	5.8	.85	100	1.4	23	1.1	1.0	17	8	48	10	0	2	2	82	3	0	0.1	0.0	8.7
17.15	37	22	3.8	.87	100	1.2	16	0.8	1.0	5	6	21	6	0	0	1	86	18	0	0.2	5.7	3.1
17.30	34	19	3.9	.85	100	1.2	14	0.8	1.0	28	5	47	4	0	0	7	84	30	0	0.1	3.8	2.4
17.45	32	19	3.2	.88	100	1.1	14	0.8	1.0	4	5	28	3	0	0	2	86	39	0	0.1	4.9	1.9
17.59	29	12	2.9	.85	100	1.1	9	0.6	1.0	1	3	10	5	0	0	1	87	26	0	0.1	2.8	2.4
17.59	55	33	6.7	.83	98	1.3	29	1.0	1.0	16	8	51	8	0	1	4	85	33	1	0.3	18.1	6.7

TIME	USERS LOGGED	<--PERCENT CPU UTILIZATION---> <--RUNNING--> TOT	CP	VIRT	IDLE	<WAITING> PAGE	I/O	<--SCHEDULER QUEUES--> DISPATCH LISTS Q1	Q2	ELIGIBLE LISTS E1	E2	ACTIVE AVG K WKSET	VIRT PRIVOP RATE /SEC	<RESOURCE QUEUE LENGTHS> AT END OF SAMPLE INTVL CPU	STG	PAGE	I/O	QWAIT TOTAL	NO. IN DEFER TASK	NO. IN VOLUNTARY WAIT	# SECS IN SAMPLE INTRVL
08.00	30	95	24	72	0	0.0	5	0.7	1.8	0.0	0.0	286	42	2	0	0	0	2	0	25	8
08.15	35	96	20	77	0	0.0	3	1.1	2.0	0.0	0.0	309	32	1	0	0	1	1	0	30	900
08.30	42	97	19	77	0	0.0	3	1.0	2.7	0.0	0.0	251	26	1	0	0	0	1	0	37	900
08.45	46	97	24	72	0	0.0	3	1.7	3.2	0.0	0.0	240	31	2	0	0	1	3	0	38	900
09.00	52	98	32	66	0	0.0	2	2.0	4.4	0.0	0.0	224	51	2	0	0	1	3	0	45	900
09.15	56	99	38	60	0	0.0	1	2.1	6.5	0.0	0.0	220	88	3	0	1	3	7	0	44	900
09.30	54	100	32	68	0	0.0	0	2.0	7.3	0.0	0.0	214	34	5	0	0	2	7	0	43	900
09.45	56	100	32	68	0	0.0	0	1.9	7.4	0.0	0.0	205	36	4	0	1	3	8	0	44	900
10.00	55	98	29	70	0	0.0	2	1.8	5.5	0.0	0.0	199	82	2	0	1	2	5	0	47	900
10.00	0	0	0	0	0	0.0	0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0
10.00	0	0	0	0	0	0.0	0	0.0	0.0	0.0	0.0	0	0	0	0	0	0	0	0	0	0
10.15	59	69	30	39	0	0.0	31	2.2	2.8	0.0	0.0	217	70	0	0	1	2	3	0	52	899
10.30	61	86	37	49	0	0.0	14	2.1	5.0	0.0	0.0	208	398	1	0	1	2	4	0	53	900
10.45	62	98	42	56	0	0.0	2	2.0	4.8	0.0	0.0	202	497	3	0	1	1	4	0	54	900
11.00	64	97	27	70	0	0.0	3	1.9	4.7	0.0	0.0	197	62	2	0	1	1	4	0	55	900
11.15	61	97	26	71	0	0.0	3	1.6	5.1	0.0	0.0	204	109	3	0	0	2	5	0	52	900
11.30	55	98	26	72	0	0.0	2	2.3	6.3	0.0	0.0	233	160	3	0	0	2	5	0	47	900
11.45	49	97	30	66	0	0.0	3	1.4	4.7	0.0	0.0	243	187	2	0	1	2	4	0	41	900
11.59	45	97	34	63	0	0.0	3	1.5	3.8	0.0	0.0	251	254	1	0	0	2	3	0	39	840
12.00	46	99	42	57	0	0.0	1	1.3	4.2	0.0	0.0	270	620	2	0	0	0	2	0	40	16
12.15	48	99	39	60	0	0.0	1	1.2	4.7	0.0	0.0	254	385	2	0	0	1	4	0	41	900
12.30	52	99	28	71	0	0.0	1	1.1	4.9	0.0	0.0	231	143	3	0	0	1	4	0	45	900
12.45	52	100	25	75	0	0.0	0	1.3	5.1	0.0	0.0	247	258	4	0	0	0	4	0	44	900
13.00	56	100	31	69	0	0.0	0	1.8	5.5	0.0	0.0	220	302	4	0	0	1	4	0	47	900
13.15	59	100	22	78	0	0.0	0	1.3	4.9	0.0	0.0	205	61	3	0	0	1	5	0	51	900
13.30	60	100	19	81	0	0.0	0	1.8	3.9	0.0	0.0	210	30	3	0	0	1	4	0	52	900
13.45	62	100	24	76	0	0.0	0	2.0	5.4	0.0	0.0	204	34	4	0	0	1	5	0	53	900
13.59	60	100	19	81	0	0.0	0	1.9	6.4	0.0	0.0	198	38	3	0	0	0	4	0	52	840
14.00	60	100	14	86	0	0.0	0	0.9	4.7	0.0	0.0	159	23	2	0	0	0	2	0	54	18
14.15	64	100	24	76	0	0.0	0	1.7	4.8	0.0	0.0	202	60	3	0	1	1	4	0	55	900
14.30	63	100	18	82	0	0.0	0	1.7	5.7	0.0	0.0	196	55	3	0	0	1	4	0	55	900
14.45	62	100	21	79	0	0.0	0	2.3	5.0	0.0	0.0	214	109	3	0	1	1	5	0	54	900
15.00	61	100	26	74	0	0.0	0	2.2	9.0	0.0	0.0	217	211	4	0	1	1	6	0	51	900
15.15	64	100	25	75	0	0.0	0	2.3	8.2	0.0	0.0	206	193	6	0	0	1	7	0	52	900
15.30	68	100	24	76	0	0.0	0	2.3	7.6	0.0	0.0	193	73	3	0	1	3	6	0	58	900
15.45	69	100	19	81	0	0.0	0	2.3	7.7	0.0	0.0	195	35	4	0	1	3	8	0	56	900
15.59	64	100	20	80	0	0.0	0	4.5	7.3	0.0	0.0	182	44	3	0	0	2	5	0	54	840
16.00	62	100	25	75	0	0.0	0	2.5	4.8	0.0	0.0	171	55	0	0	0	2	2	0	55	17
16.15	63	100	19	81	0	0.0	0	2.0	4.2	0.0	0.0	191	73	2	0	1	1	4	0	55	900
16.30	62	100	22	78	0	0.0	0	2.6	5.8	0.0	0.0	182	69	3	0	2	1	6	0	51	900
16.45	59	100	27	73	0	0.0	0	2.7	6.1	0.0	0.0	199	80	4	0	1	2	6	0	48	900
17.00	49	100	27	73	0	0.0	0	1.2	4.6	0.0	0.0	222	36	3	0	1	1	5	0	40	900
17.15	37	100	17	83	0	0.0	0	1.2	2.7	0.0	0.0	250	24	1	0	0	0	2	0	32	905
17.30	34	100	18	82	0	0.0	0	1.0	2.9	0.0	0.0	277	13	1	0	0	1	2	0	29	900
17.45	32	100	10	90	0	0.0	0	1.1	2.1	0.0	0.0	295	20	1	0	0	0	1	0	28	900
17.59	29	100	6	94	0	0.0	0	1.0	1.9	0.0	0.0	313	22	1	0	0	0	1	0	26	840
17.59	55	98	25	73	0	0.0	2	1.8	4.9	0.0	0.0	216	112	3	0	0	1	4	0	46	35824

DATE 05/10/82 FROM 00:00:02 TO 23:59:59 USER RESOURCE UTILIZATION SUMMARY # SECONDS: 86100 PAGE 1

		<----------CPU---------->					<STORAGE>		<--------PAGING--------->					<---I/O--->		<-SAMPLES->			<------RUNNING STATUS------>						
		RELATIVE				TOT:	WKSET		<PAGE I/O>		<PGDEV SLOTS>			THOUSANDS			PCT	AP	<PCT DELAYS DUE TO>				AP	PCT	
			CUM	<--SECONDS-->		VIRT	K BYTES		THOUSANDS		PRIMARY		OFLO	DISK+	PRINT	NO.	ACT	PCT	WAITING FOR				LOK	VOL	
RANK	USERID	PCT	PCT	TOTAL	VIRT	RATIO	AVG	MAX	READ	WRITE	AVG	MAX	MAX	TAPE	LINES	OBS	IVE	CPU	CPU	STG	PAG	I/O		WAIT	RAI
1	TAPEMON	23	23	9,573	9,487	1.0	93	312	2	1	62	78	0	0	0	871	98		29	0	0	0	0	52	.70
2	DOSVSE	13	36	5,433	2,791	1.9	455	836	4	2	152	217	0	1023	0	1448	99		6	0	0	1	0	85	.92
3	TATCH	13	48	5,277	5,202	1.0	522	972	4	3	208	245	0	29	0	564	96		78	0	1	5	0	0	.16
4	HITNRUN	3	52	1,453	601	2.4	404	1132	10	8	232	510	0	45	0	385	87		15	0	3	11	0	66	.72
5	PREAMP	3	55	1,207	1,137	1.1	362	1264	9	9	227	319	0	11	0	310	94		13	0	2	2	0	77	.83
6	GGOAD	3	58	1,197	1,066	1.1	257	416	7	3	60	90	0	36	5	338	93		18	0	1	9	0	64	.72
7	SCARBORO	3	60	1,081	964	1.1	254	892	7	4	80	118	0	30	8	393	82		21	0	2	8	0	69	.69
8	TATCHM	2	63	982	622	1.6	222	816	1	1	87	191	0	233	0	266	95		37	0	0	53	0	0	.10
9	BBRIAN	2	65	884	795	1.1	217	864	6	4	80	134	0	27	0	312	92		11	0	2	3	0	79	.84
10	TATCHS	2	67	777	712	1.1	424	968	1	1	63	191	0	18	2	103	79		58	0	2	22	0	0	.17
11	GRAPEVIN	2	68	719	622	1.2	354	1832	4	4	104	211	0	30	16	252	91		8	0	2	9	0	75	.81
12	NEWACCT	2	70	686	514	1.3	243	640	7	3	70	106	0	70	3	484	83		2	0	1	9	0	84	.88
13	TANDY	2	71	628	538	1.2	221	564	5	3	63	118	0	27	1	284	94		5	0	2	9	0	70	.84
14	CHRISRA	1	73	546	517	1.1	365	892	1	1	42	117	0	7	0	82	88		4	0	8	7	0	68	.80
15	VSEVTAM	1	74	520	323	1.6	557	2296	4	2	826	849	0	29	0	589	99		2	0	0	0	0	96	.98
16	SSD	1	75	501	447	1.1	427	1248	8	10	183	325	0	9	0	287	77		8	0	5	3	0	85	.85
17	BATCHL	1	76	476	231	2.1	167	496	1	0	64	83	0	151	1	513	30		10	0	0	36	0	83	.54
18	DEFSUP	1	77	466	386	1.2	376	1300	9	10	170	327	0	15	1	329	82		4	0	2	3	0	87	.91
19	STRONG	1	79	457	407	1.1	271	640	5	2	78	114	0	19	0	457	58		5	0	3	3	0	91	.89
20	TST	1	80	451	401	1.1	558	1296	3	3	104	325	0	10	2	121	92		9	0	4	7	0	72	.80
21	PERFB	1	81	436	400	1.1	696	1048	7	0	11	94	0	13	0	18	94		0	0	6	25	0	18	.69
22	OPERATOR	1	82	415	246	1.7	159	264	2	0	9	21	0	92	6	1448	55		1	0	0	5	0	96	.94
23	CHARLIEB	1	83	385	263	1.5	297	656	4	2	70	128	0	34	1	774	48		0	0	1	5	0	95	.94
24	DTSSPEC	1	83	385	323	1.2	183	772	7	4	69	119	0	13	0	490	77		2	0	1	1	0	89	.96
25	SMART	1	84	377	114	3.3	162	284	7	6	71	73	0	0	4	1448	99		0	0	0	0	0	99	1.00
26	NMONITOR	1	85	335	184	1.8	239	680	4	1	31	110	0	29	0	660	27		1	0	3	6	0	96	.90
27	JANE	1	86	328	287	1.1	186	564	3	2	69	88	0	10	0	297	61		7	0	2	6	0	88	.85
28	PVM	1	87	279	106	2.6	161	184	1	1	115	119	0	80	0	1448	53		2	0	0	0	0	98	.98
29	DAILY2	1	87	279	253	1.1	249	864	3	1	102	182	0	6	0	302	56		2	0	2	1	0	95	.96
30	JDS127	1	88	276	243	1.1	208	636	4	2	91	146	0	6	0	412	65		2	0	2	2	0	95	.94
31	GROVESB	1	89	269	165	1.6	234	544	10	5	95	175	0	22	0	534	74		0	0	1	2	0	94	.96
32	PERICLES	1	89	257	188	1.4	274	1232	1	1	36	87	0	34	4	124	77		1	0	0	17	0	80	.82
33	JERRY	1	90	250	132	1.9	310	1148	2	1	67	100	0	22	38	509	19		7	0	5	10	0	95	.78
34	WAZOO	1	90	228	195	1.2	224	620	2	1	65	88	0	8	2	106	96		8	0	3	10	0	74	.79
35	RSCS	1	91	212	73	2.9	141	232	5	1	41	54	0	30	154	1448	99		16	0	0	0	0	83	.84
36	CEL	0	91	193	162	1.2	885	1308	2	2	125	323	0	6	0	118	93		2	0	2	3	0	91	.94
37	TATCHD	0	92	167	163	1.0	709	856	0	0	1	1	0	2	0	10	89		63	0	0	25	0	0	.12
38	ROB	0	92	135	86	1.6	195	424	6	3	73	81	0	14	0	460	67		1	0	2	4	0	96	.94
39	KOHL	0	92	131	106	1.2	299	604	4	2	86	109	0	5	0	241	70		2	0	3	2	0	93	.93
40	VMAP	0	93	127	98	1.3	273	708	0	0	28	71	0	11	4	18	71		8	0	0	58	0	6	.33
41	MARKB	0	93	122	61	2.0	285	628	6	2	82	93	0	13	0	565	44		0	0	3	6	0	96	.91
42	IONDS	0	93	105	56	1.9	165	360	5	3	78	95	0	8	0	397	77		0	0	2	3	0	90	.95
.														7	1	551	51		0	0	2	3	0	97	.95
.														8	1	334	85		0	0	1	1	0	93	.97
.														16	0	599	46		0	0	1	3	0	97	.95
	TOTALS:	100	100	41,848	33,277	1.3	224	2296	304	188	67	849	0	2558	28745461		54		6	0	1	4	0	91	.89

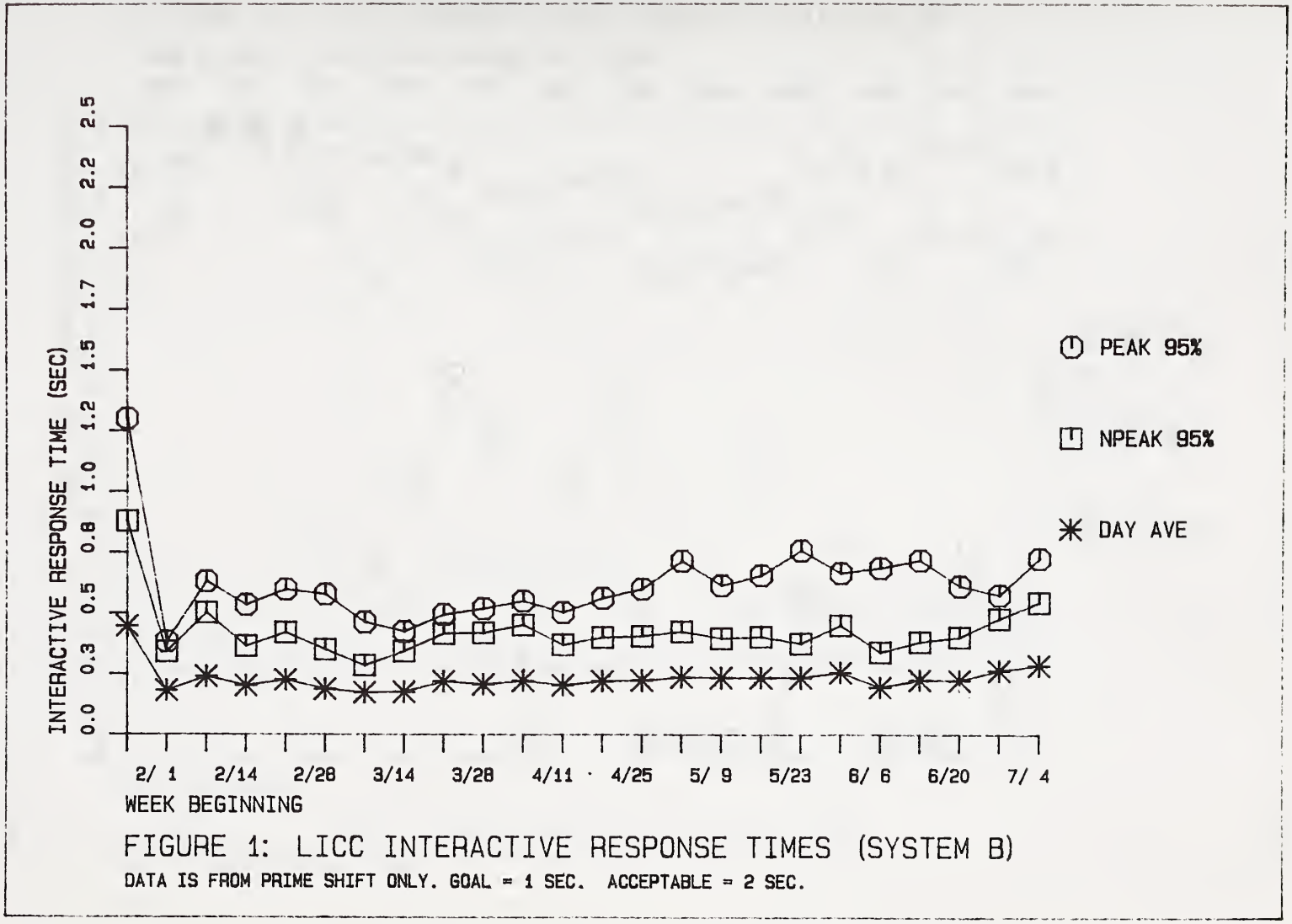


FIGURE 1: LICC INTERACTIVE RESPONSE TIMES (SYSTEM B)

DATA IS FROM PRIME SHIFT ONLY. GOAL = 1 SEC. ACCEPTABLE = 2 SEC.

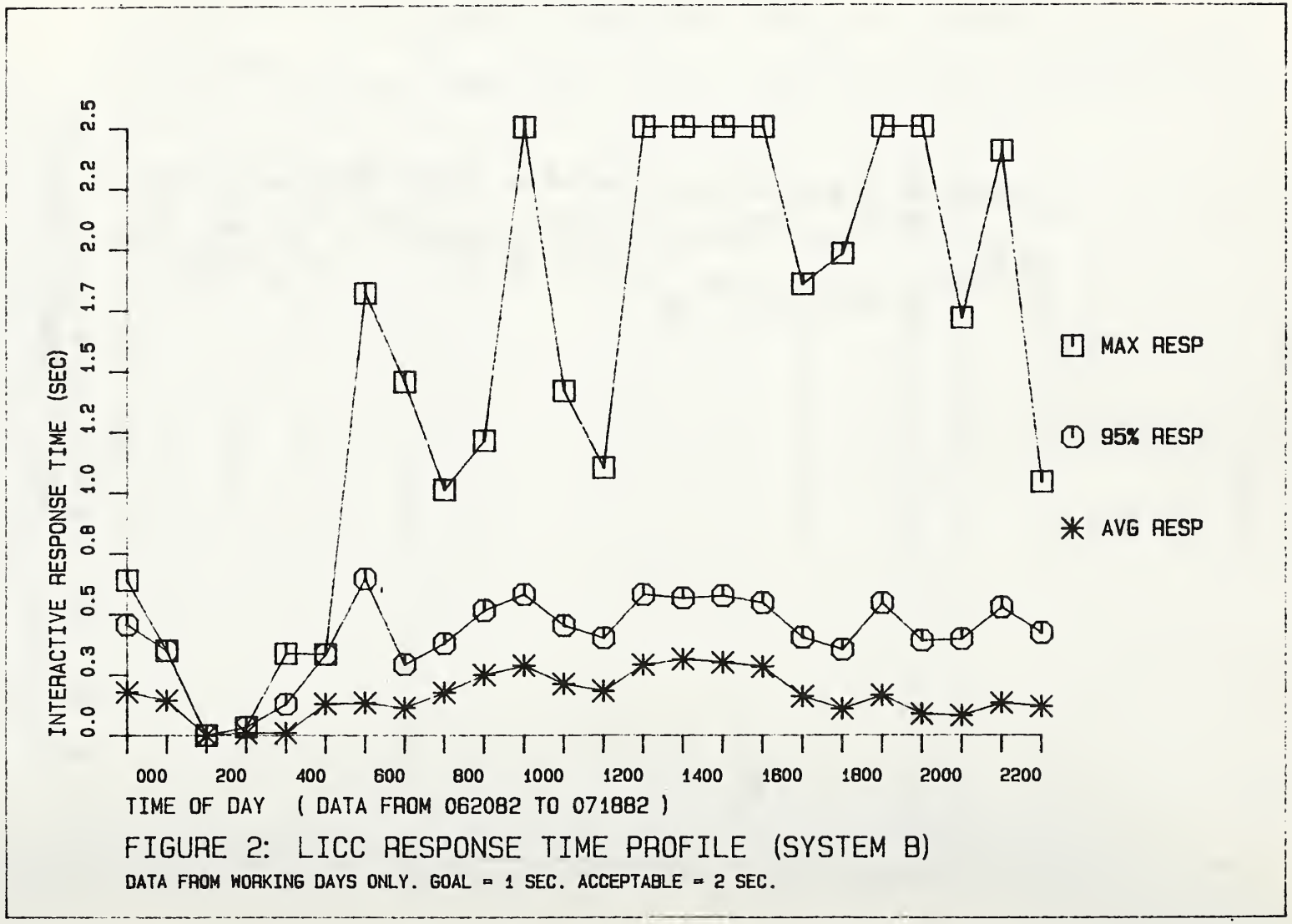


FIGURE 2: LICC RESPONSE TIME PROFILE (SYSTEM B)

DATA FROM WORKING DAYS ONLY. GOAL = 1 SEC. ACCEPTABLE = 2 SEC.

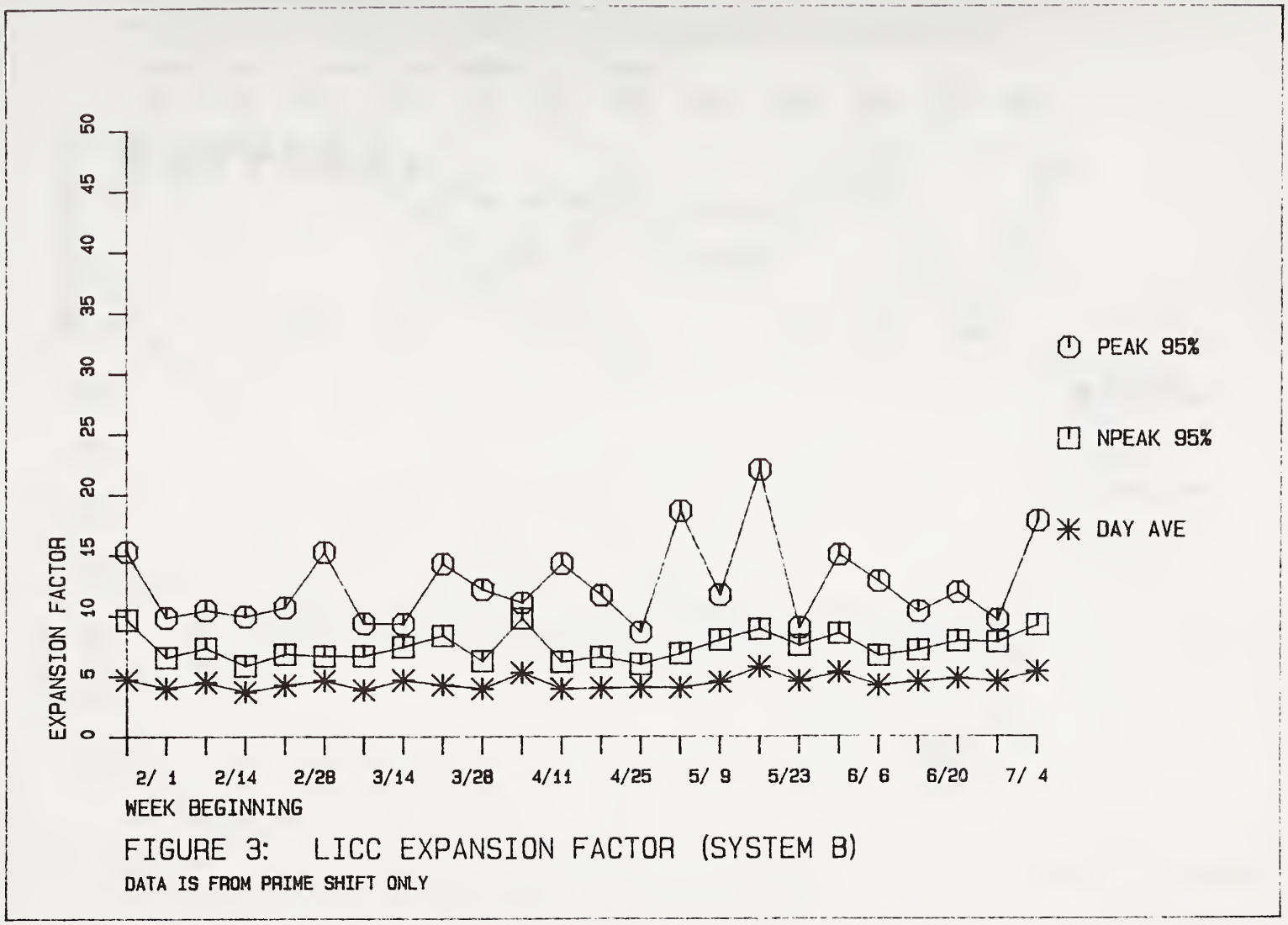


FIGURE 3: LICC EXPANSION FACTOR (SYSTEM B)

DATA IS FROM PRIME SHIFT ONLY

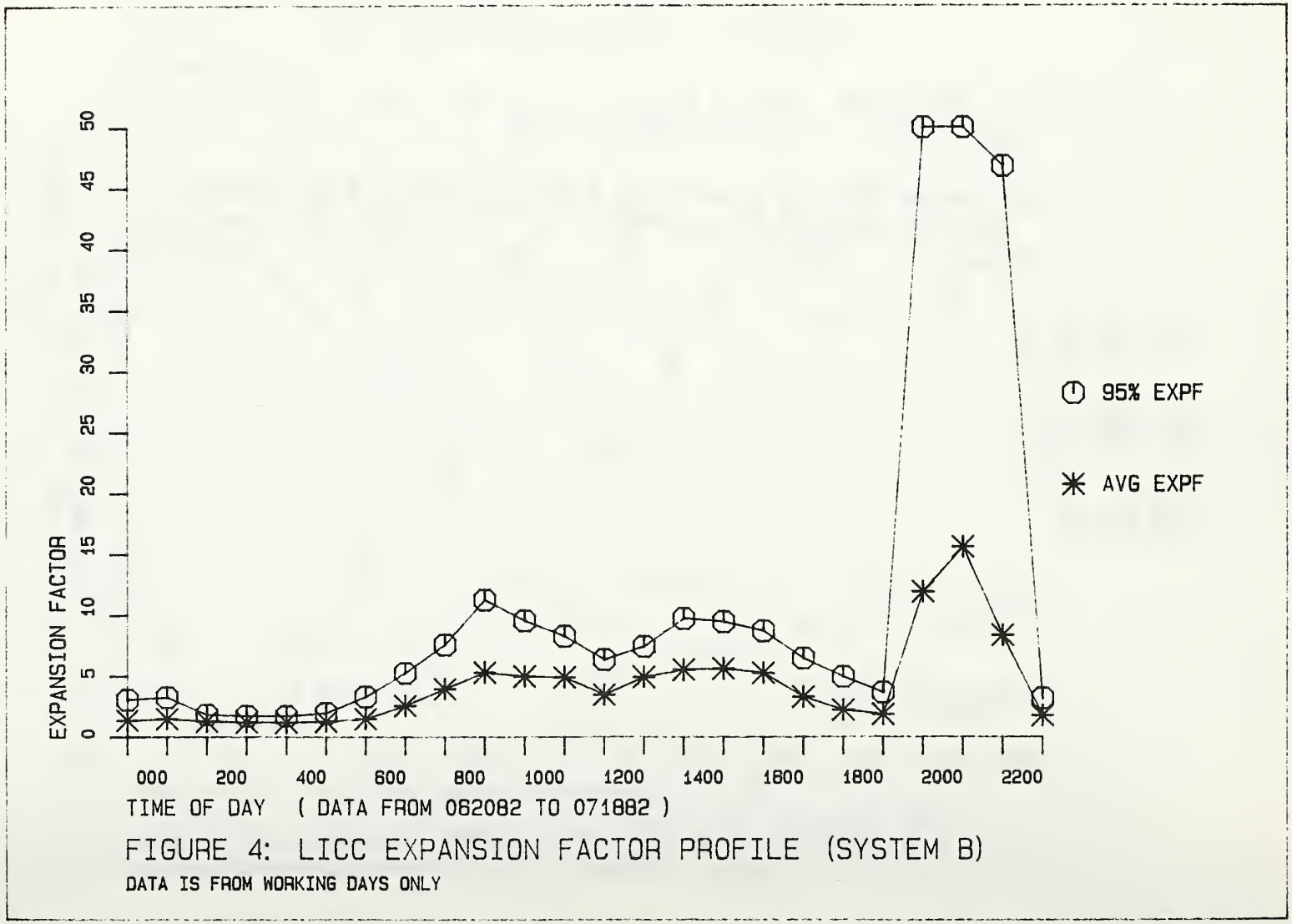


FIGURE 4: LICC EXPANSION FACTOR PROFILE (SYSTEM B)

DATA IS FROM WORKING DAYS ONLY

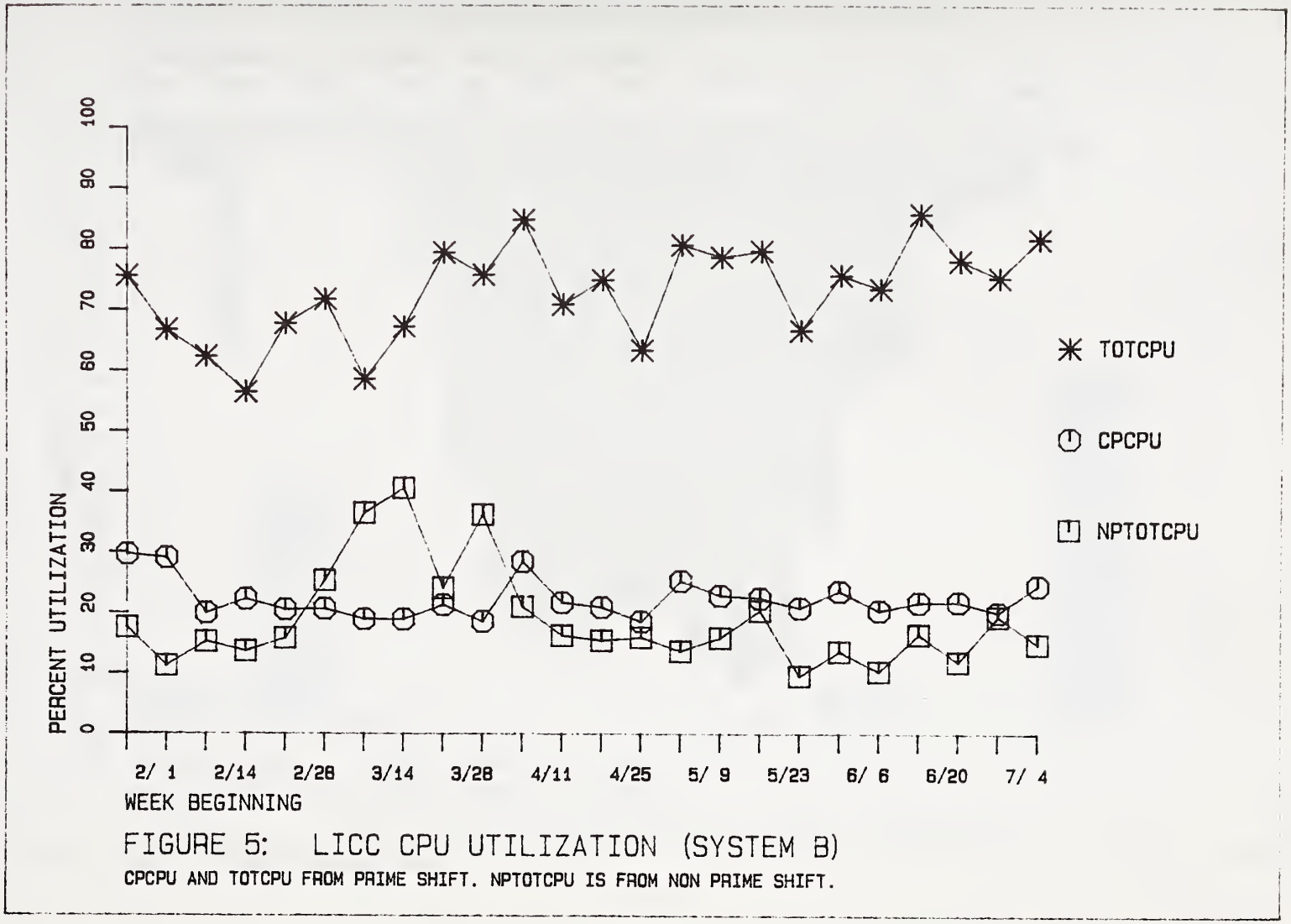


FIGURE 5: LICC CPU UTILIZATION (SYSTEM B)

CPCPU AND TOTCPU FROM PRIME SHIFT. NPTOTCPU IS FROM NON PRIME SHIFT.

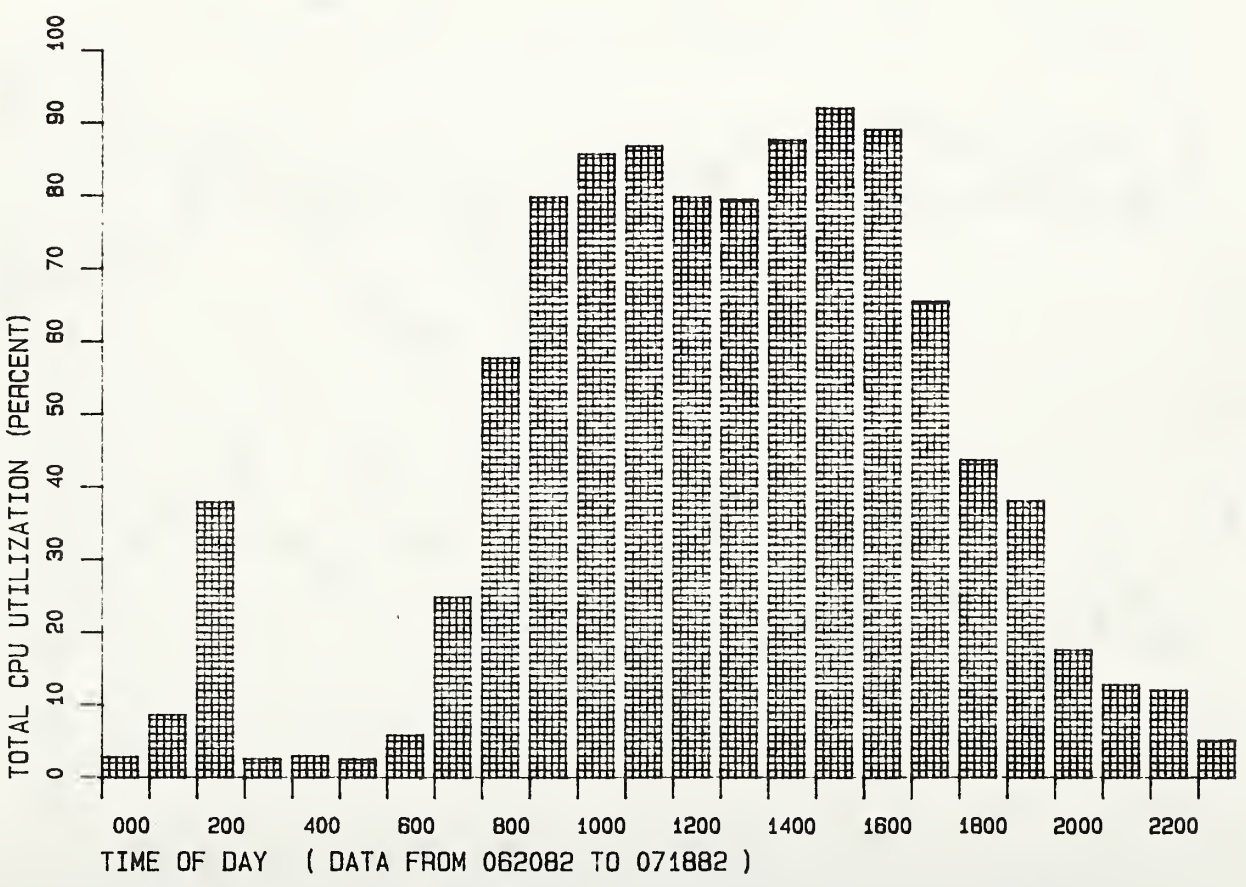


FIGURE 6: LICC CPU UTILIZATION PROFILE (SYSTEM B)

DATA FROM WORKING DAYS ONLY

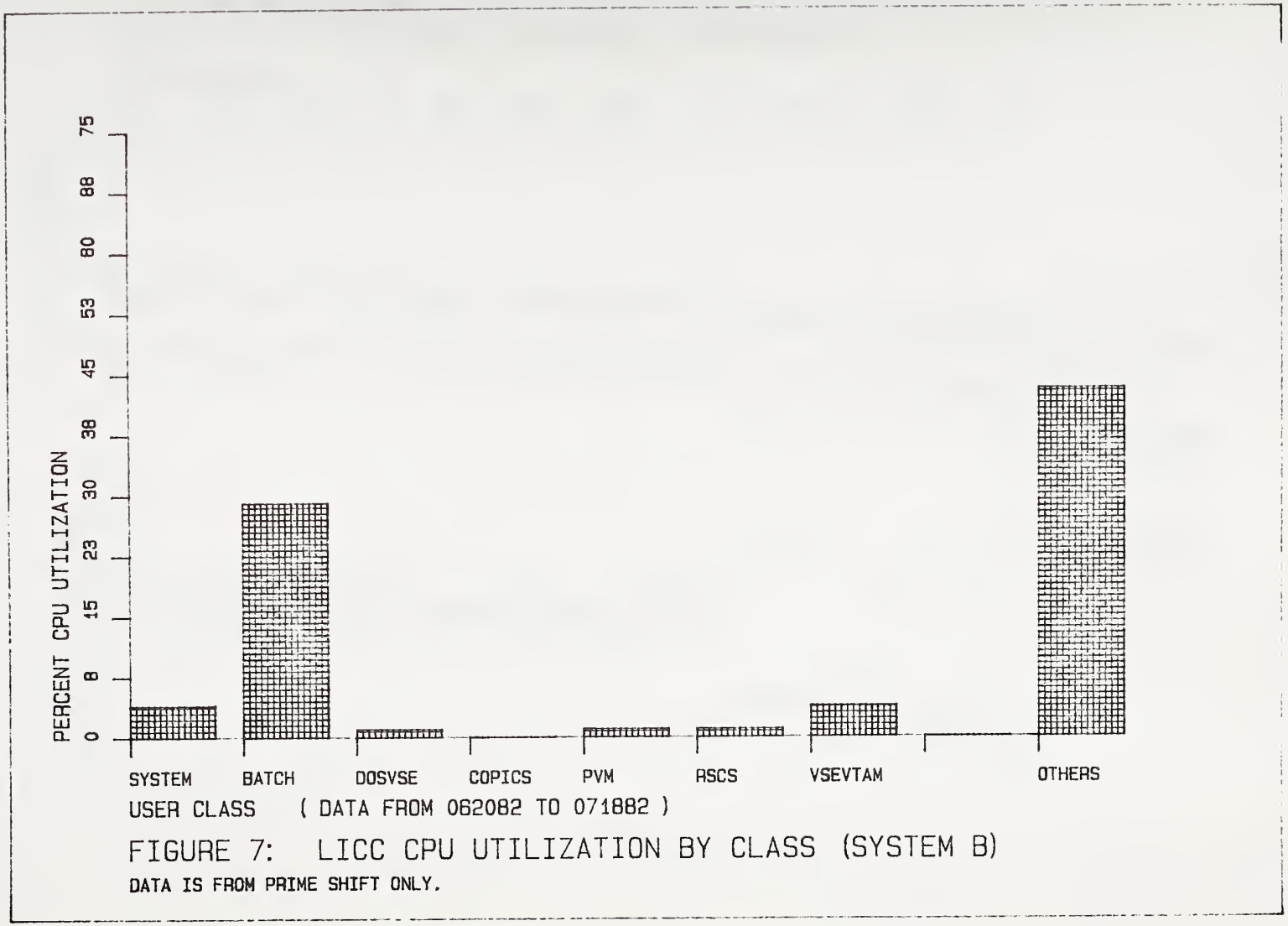


FIGURE 7: LICC CPU UTILIZATION BY CLASS (SYSTEM B)

DATA IS FROM PRIME SHIFT ONLY.

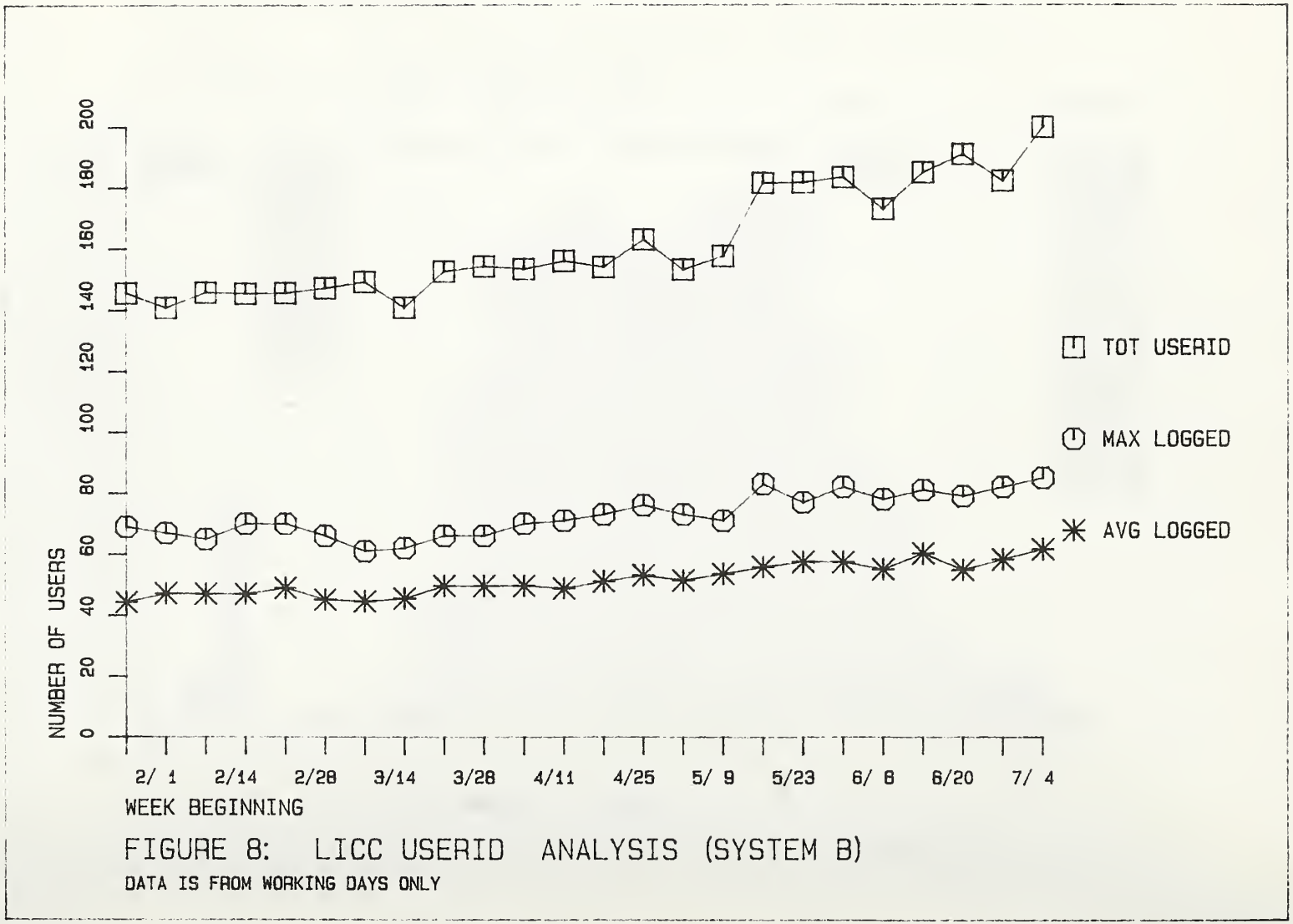


FIGURE 8: LICC USERID ANALYSIS (SYSTEM B)

DATA IS FROM WORKING DAYS ONLY

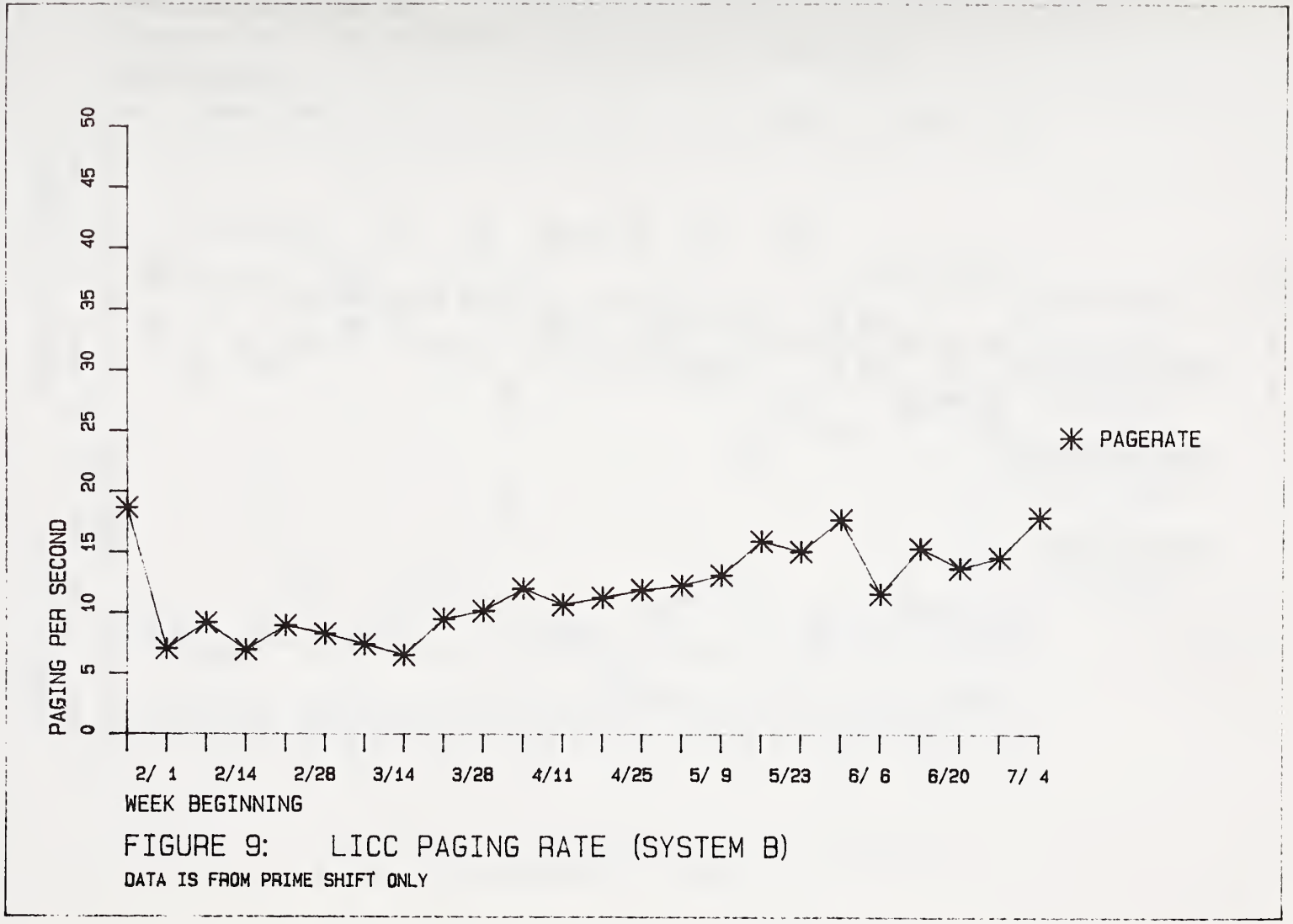


FIGURE 9: LICC PAGING RATE (SYSTEM B)

DATA IS FROM PRIME SHIFT ONLY

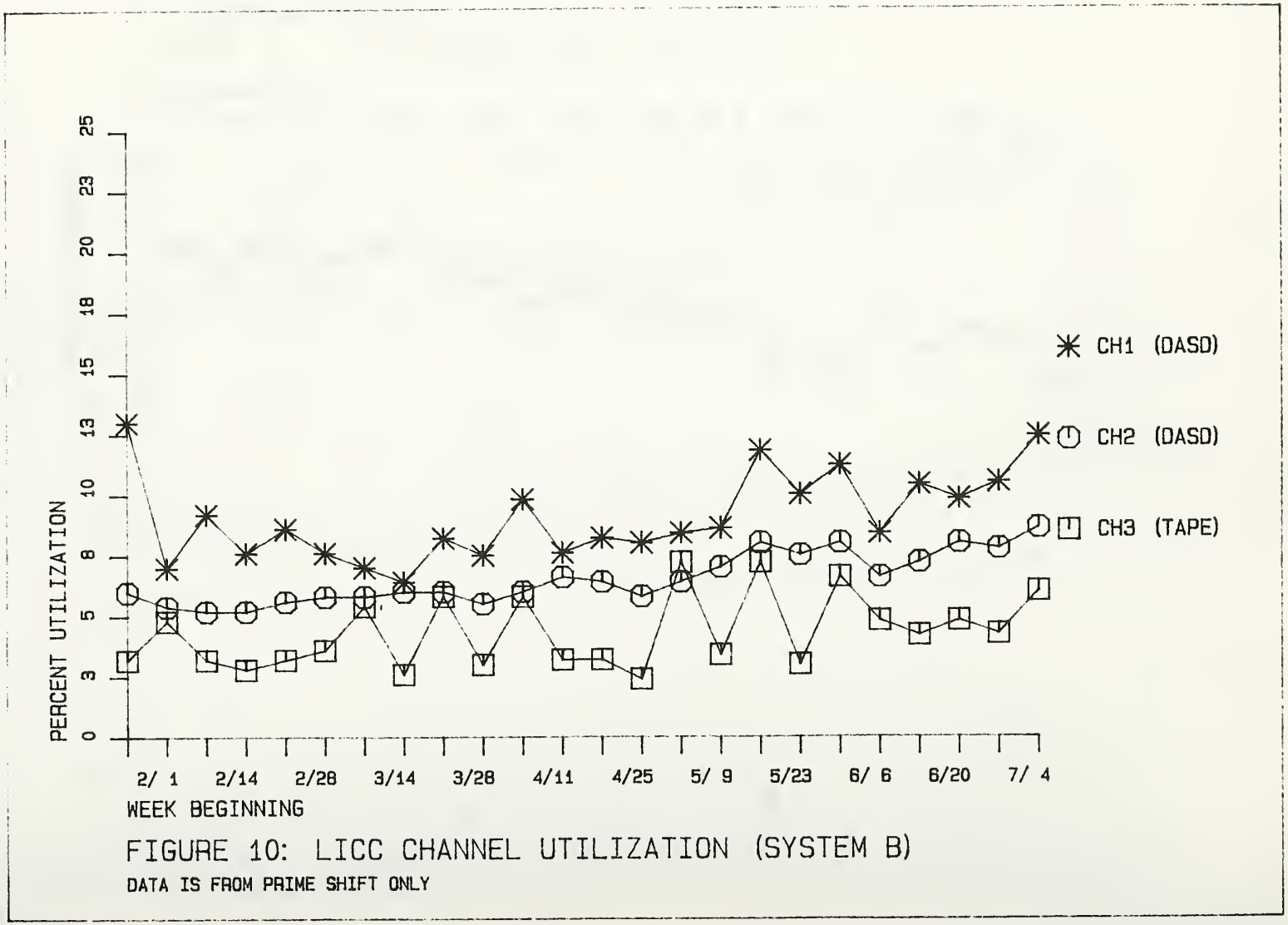


FIGURE 10: LICC CHANNEL UTILIZATION (SYSTEM B)

DATA IS FROM PRIME SHIFT ONLY

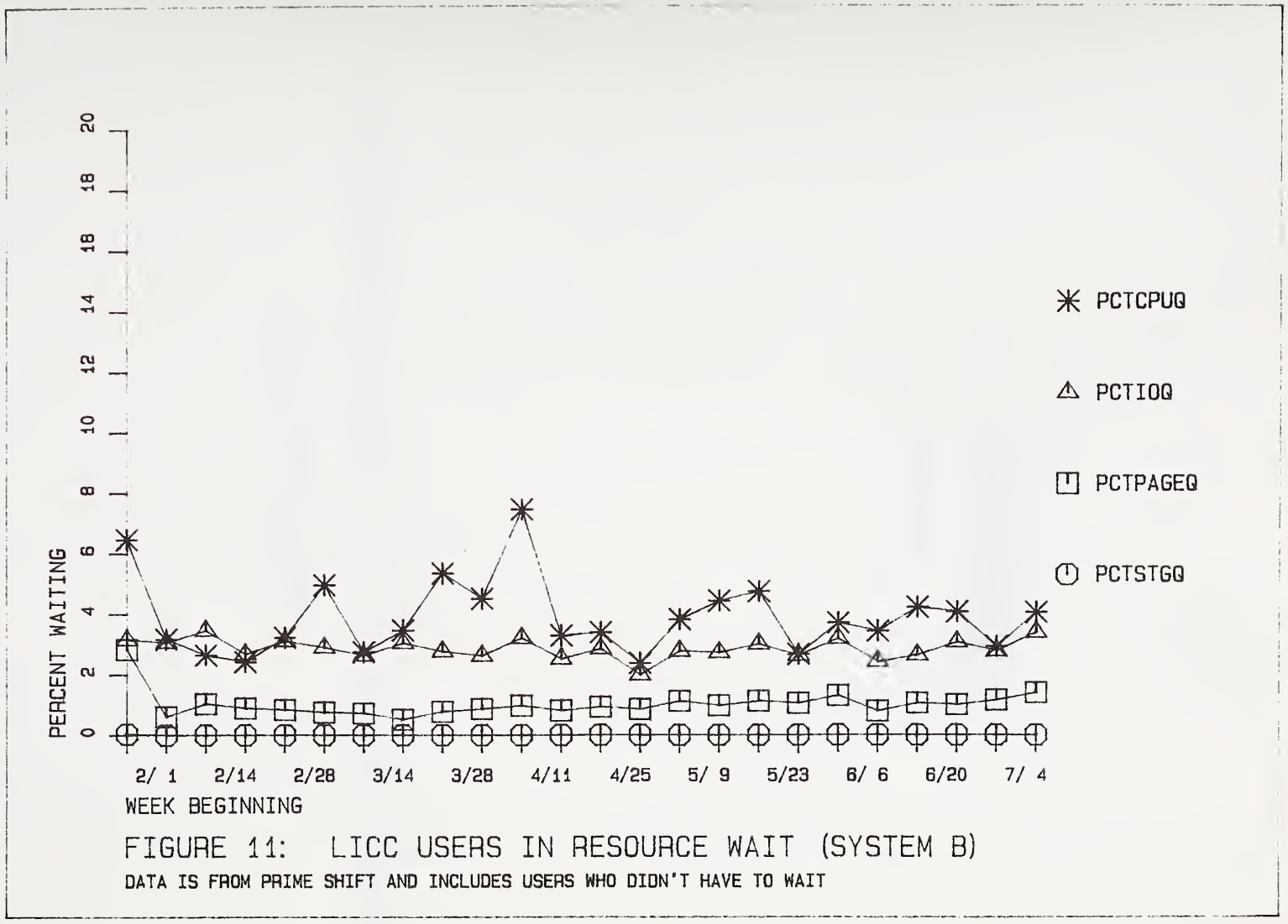


FIGURE 11: LICC USERS IN RESOURCE WAIT (SYSTEM B)

DATA IS FROM PRIME SHIFT AND INCLUDES USERS WHO DIDN'T HAVE TO WAIT

CPEUG82

"Improving Organizational Productivity"

Performance of Local Area Networks

SESSION OVERVIEW

PERFORMANCE OF LOCAL AREA NETWORKS

Bill Hawe

Digital Equipment Corporation
Hudson, MA 01749

This session investigates various aspects of Local Area Network performance analysis. Included are discussions of an analytical framework for studying the performance of shared channel access protocols. This is applied to CSMA, Slotted Aloha and TDMA access methods. Additionally, a detailed simulation study of Ethernet which incorporates a model of the layered architecture above Ethernet is presented. This presents system performance metrics such as waiting time, number of collisions, etc. as a function of the number of users sharing the channel. Finally, results of hardware and software measurements of hosts, interfacing channels, adapters, etc. for a Hyperchannel connecting Digital Equipment Corporation (DEC) and Control Data Corporation (CDC) machines are presented. These compare the contributions of host protocol overhead, network adapter processing and truck transfer rates to the system performance.

A Common Framework for Studying the Performance of Channel Access Protocols

K.K. Ramakrishnan
and
Satish K. Tripathi

System Design and Analysis Group
Department of Computer Science
University of Maryland
College Park, MD 20742.

Abstract

Channel Access Protocols for shared multiple access channels have been widely studied. Performance studies have treated these protocols in isolation using open system models. For comparing various access protocols, we need to arrive at a uniform framework for the access schemes, operating under similar load conditions. We consider a closed system with a fixed number of users utilizing the channel. The channel is aggregated to a single load dependent M/M/1/N server.

Within such a framework, the adaptability of the channel access protocol to changes in the offered load is studied. The Relaxation Time of the channel is proposed as a measure of the sensitivity of the channel to such changes. Example access schemes are studied. In particular, Fixed Time Division Multiple Access (FTDMA), Slotted Aloha, and Carrier Sense Multiple Access (CSMA) schemes are compared.

Key words: Local area networks; channel access; protocols; throughput; relaxation time; sensitivity; slotted aloha; carrier sense multiple access; load dependent ; M/M/1/N queue; transition matrix.

1. Introduction

Local Computer Networks have achieved greater prominence and application in the recent past due to advancing technology, the trend towards smaller systems and the need to share not only computer resources but data as well. The medium of communication is typically a shared channel, with access provided to the nodes in the network in some fixed manner.

As observed by Shoch et al. [Shoc80], message traffic generated in a typical local network is bursty in nature. Several access schemes have been synthesized to account for this burstiness in traffic, and achieve an efficient utilization of the channel [Lam80], [Metc76]. The underlying protocols that appear in use include Fixed Time Division Multiple Access (FTDMA), Slotted Aloha, Reservation Aloha and Carrier Sense Multiple Access (CSMA) schemes.

Considering the channel as an open system, expressions for the throughput and delay in the channel have been obtained for different access schemes, [Klei73], [Lam80], [Buze79], [Fran80], [Toba80], [Alme79]. Performance studies of specific techniques have treated these access

protocols in isolation. Comparison of different access schemes, therefore, has been difficult [Powe81]. To enable a comparison of these schemes, a common framework to study them is needed. We propose a framework in which these access protocols may be studied, under similar load conditions. As an example of using this framework, we study such shared multiaccess channels for their performance, with regard to their capability to adapt to transient changes in their offered load. A measure of the sensitivity of the channel to such changes is proposed.

In this framework, we attempt to reflect the realistic situation of a fixed number of users accessing the channel and determine the response of the channel to the load offered by these users, using models for the access protocols proposed earlier, [Klei73] and [Lam80]. A closed system is considered, with a fixed number of terminals/customers permitted access to the channel. Note that the throughput of the channel and the arrival rate from this set of users are interdependent and also depend on the channel access protocol that is used. We adopt an iterative technique solving for the throughput-delay characteristics of the access scheme and obtain the corresponding arrival and throughput rates. Given these arrival and throughput rates, we approximate the channel behaviour by modeling it as a single load dependent exponential server. In this common framework, a comparison of the FTDMA, Slotted Aloha and CSMA schemes is made.

Because of an increase in contention within a shared multiaccess channel, or because of a fixed access methodology being employed, performance of the channel may degrade with additional load. This degradation in performance and the ability of the channel to adapt itself to such changes in load is an important aspect to evaluate. The channel's First Exit Time (FET) to a 'critical state' has been used as a measure of the stability of the channel in the case of the Slotted Aloha and CSMA access protocols [Klei75b], and [Toba77]. Fayolle et al. [Fayo77] study control policies and the stability of slotted access channels. In our analysis, we consider the "relaxation time" [Klei76] as a measure of the sensitivity of the channel to changes in the load on the channel.

The next section presents a methodology for arriving at a common framework to study the access protocols. In Section 3 we outline a method of obtaining the relaxation time of such an aggregated M/M/1/N server with load dependent arrival and service rates. Section 4 outlines the models used for the individual access protocols and aggregation of the schemes. Section 5 presents experimental results.

2. A Common Framework for Channel Access Schemes

Existing models for channel access protocols study the throughput and delay behavior of the channel, treating it as an open queueing system e.g., [Klei73], [Klei75c]. In actual systems though, there are a fixed number of users permitted to use the channel. The underlying assumption is that each new user can have only one outstanding message at a time. The load offered to the channel, when the channel is treated as an equivalent open system, also depends on the response from the channel to these users. Although open system models of channel access protocols are adequate to study the throughput and delay behavior of the channel in isolation, they may not be useful for comparison between different access protocols. They cannot reflect the interdependency between the load on the channel and the response from the channel to that load. Therefore we model the situation by a closed system, with a fixed number of users M, attempting to utilize the channel, as shown in Figure 1.

We consider the service offered by the channel as the time from the presentation of the packet by a ready user, until its successful transmission. This includes any queueing delays and delays for retransmission that may be required. We further assume that the channel may be modeled by aggregating it to a single M/M/1/N server. As the behavior of the protocol depends substantially on the number of ready users of the channel, we use load dependent arrival and service rates for the aggregated server.

Let the 'think time' for each user be Z. Ready users attempting to transmit queue at the channel. When there are 'i' users queued at the channel, the arrival rate generated by the users to the channel is given by

$$L(i) = (M-i) / Z \qquad (2.1)$$

With this assumed value of the arrival rate (refer to Figure 1), we solve the throughput - delay equations for the channel, given the characteristics of the access scheme that is used [Klei73], [Lam80], treating the channel as an open system. For each access protocol, the

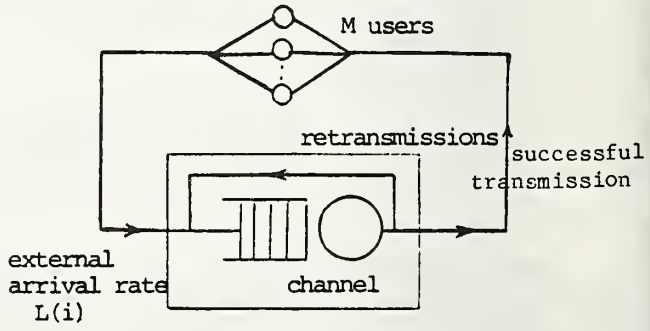


Figure 1.

throughput from the channel is thus given by

$$S(i) = i / D(i) \qquad (2.2)$$

where D(i) is the delay encountered at the channel.

The channel adapts to the load on the system, and thus reaches a state which can be characterized as a steady state. In steady state, the throughput S(i) from the channel must match the average arrival rate L(i) that was assumed. We obtain a new estimate of the arrival rate and iterate by successively solving for the throughput and delay and getting a new L(i), till the error between L(i) and S(i) is within limits. Thus, by treating the channel as an open system, we compute the throughput for a given arrival rate to the channel. Determining the load dependent service rate of the channel in such a system, we can now use it in the closed system shown in Figure 1, where the load dependent arrival rate is given by eq. (2.1). Section 4 elaborates on the models of the access protocols used, when the channel is treated as an open system.

The performance of the access protocol may also depend on the time-out period used or the acknowledgement delay encountered on the channel. The throughput-delay equations incorporate this factor, and a further level of iteration may be necessitated to obtain estimates of the acknowledgement interval. Briefly, the iterative technique is:

```
while | S(i) - L(i) | > E1 do
  Compute L(i) = (M - i) / Z ;
  Assume R ; (ack. interval)
  Compute S1(i) and D1(i);
(*using the open system models*)
  j = 1 ;
    Repeat
      j = j+1 ;
      Compute Sj(i) and Dj(i) ;
      Update R based on Dj(i) ;
    until | Sj(i)-Sj-1(i) |<E1u;
  S(i) = Sj(i) ;
  D(i) = Dj(i) ;
  Update L(i) based on S(i) - L(i);
end do;
```

When a new user is added to the system, this has an impact on the service time of the channel and the load dependent arrival rate is also changed. For each value of N, we need to iteratively solve the throughput - delay equations of the channel. By such an iterative technique, the parameters of the uniform model of the channel (for each of the corresponding access protocols), aggregated as a single load dependent M/M/1/N server are obtained.

For computing the throughput and delay values for the channel, we use models proposed by Kleinrock and Lam [Klei73] (for the Slotted Aloha channel), and Lam [Lam80] (for the CSMA channel). These values determine the parameters for the load dependent M/M/1/N server used to represent the channel. Thus, for the range of offered load on the channel, the performance of the load dependent server matches the proposed models. The load dependency is necessary because the channel performance varies with the number of ready users of the channel. In this framework, the performance of the different access protocols may be studied, and a comparison of the schemes made.

3. Relaxation Time for the Aggregated Server

Within the framework presented in the previous section, we compare the performance of the channel for changes in the offered load for different access protocols. Typically, as a result of the access protocol that is used, the channel adapts to the load offered by a given set of users, and a steady state is achieved. Subsequently, when there is a change in the offered load to the channel, for example with an increase in the number of users attempting to utilize the channel, the performance may degrade before a new steady state is achieved. Within the same environment, each access protocol responds differently to such changes. The relaxation time is used as a measure for comparison of the protocols, using the uniform framework for modeling the channel access protocols. The relaxation time is a 'time constant' for a queueing system to reach a new steady state, after a change occurs in the offered load to the system [Klei75a]. We present here, a technique of obtaining the relaxation time for an M/M/1/N server with load dependent arrival and service rates.

For the M/M/1/N server, the relaxation time of the channel is a measure of the dynamics of the number in the system. In the exact case, expressions for the number in the system would involve an infinite sum of Bessel functions, even for the simple M/M/1 case [Klei75a]. Stern [Ster79] provides an approximation for the time dependent number in the system of an M/M/1 queue by truncating the queue at some finite but arbitrarily large length. A realistic assumption of a finite buffer is thus made. We take a similar approach in our case to limit the states of the Markov chain to a finite length.

Briefly, the approach by Stern is as follows: Assuming Poisson arrivals at a rate of L packets/sec, and exponential service times with a mean of 1/U, the evolution of the probability distribution p(t), is given by

$$\frac{d\underline{P}}{dt} = \underline{Q}\,\underline{P} \qquad (3.1)$$

where $\underline{Q}$ in the model proposed in [Ster79], is a (k+1)*(k+1) matrix. $\underline{Q}$ is then transformed by a change of variable, yielding a tridiagonal symmetric matrix $\underline{A}$.

The eigenvalues E_i, of the matrix $\underline{A}$, are provided by the explicit closed form solution of

$$p(t) = p_0 + \sum_{i=1}^{K} c_i p_i e^{-UE_i t} \qquad (3.2)$$

where $p_i = E^{1/2} u_i$ and $c_i = u_i^T E^{-1/2} p(0)$

where u_i is the i^{th} eigenvector of $\underline{A}$. From the above, it is shown that the "time constant" or relaxation time r of the queue is given by

$$r = 1 / (UE_1) \qquad (3.3)$$

where E_1 is the smallest non-zero eigenvalue of the matrix $\underline{A}$.

We use this method of obtaining the relaxation time for our load dependent M/M/1/K queue. Both the service and arrival rates in our case, however, are load dependent, and the matrix $\underline{A}$ can no longer be symmetric. The Markovian state diagram is shown in Figure 2.

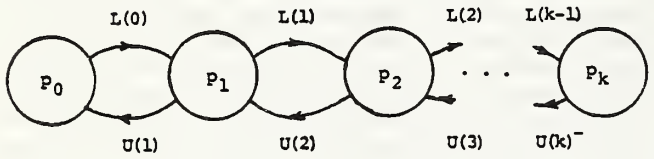


Figure 2

From the above, the rates of departure and arrival are as follows, from each state:

$$\frac{dp_0(t)}{dt} = -L(0)p_0(t) + U(1)p_1(t)$$

$$\frac{dp_n(t)}{dt} = L(n-1)p_{n-1}(t) - (L(n)+U(n))p_n(t) + U(n+1)p_{n+1}(t) \quad \text{for } 0 < n < k \qquad (3.4)$$

For n = k,

$$\frac{dp_k(t)}{dt} = L(k-1)p_{k-1}(t) - U(k)p_k(t)$$

Writing these down as a vector, we obtain

$$\frac{d\underline{P}(t)}{dt} = \underline{P}(t)\,\underline{A} \qquad (3.5)$$

where $\underline{A}$ is the following matrix:

$$\begin{bmatrix} -L(0) & L(0) & 0 & \cdots & 0 \\ U(1) & -(L(1)+U(1)) & L(1) & & \\ 0 & U(2) & -(L(2)+U(2)) & & \\ \vdots & 0 & U(3) & & \\ & & & L(n-1) & \\ & & & -(L(n)+U(n)) & \\ & & & U(n+1) & \\ & & & & L(k-1) \\ 0 & 0 & 0 & 0 & -U(k) \end{bmatrix}$$

We numerically solve for the eigenvalues of $\underline{A}$ and obtain the smallest non-zero eigenvalue. If E_1 is this smallest eigenvalue, then the relaxation time is given by

$$r = 1/(UE_1) \qquad (3.6)$$

given a mean service time of 1/U seconds.

We present in the next section specific models used for the access schemes compared, and arrive at the parameters for the aggregated M/M/1/N server to enable computation of the relaxation time.

4. Access Scheme Aggregation of Access Protocols

With the development of Section 2, we consider the following access protocols for comparison:

(1) FTDMA
(2) Slotted Aloha
(3) CSMA

For each of these schemes, given a particular arrival rate to the channel, we can obtain the throughput and delay relationships by considering the channel to be a single open system. The considerations involved in each of the access schemes are given below. The underlying model of the channel for the Slotted Aloha and CSMA channels used have been shown to be good, in comparison to the actual performance of the protocols in [Klei73], and [Lam80].

4.1 Fixed Time Division Multiple Access

With Fixed Time Division Multiple Access (FTDMA), the channel access is divided into slots of equal but fixed lengths. The slots are allocated to users in a fixed, round robin fashion. The channel delay involved is the sum of the synchronization time of the user with the corresponding slot and the transmission time itself, which are both fixed.

Given a closed system of M users, and a normalized packet transmission time T = 1, the synchronization delay DS is, on the average equal to M * T/2. The synchronization delay does not depend on the total number of users ready to transmit, and thus is load independent. The channel delay is

$$D = M * T/2 + T \qquad (4.1)$$

The arrival rate from the set of M users, with a finite think time Z, between successive generations of packets is given by eq. (2.1). The throughput of the channel is given by eq. (2.2), i.e. S(i) = i/D. The aggregation of the communication channel as a single M/M/1/N server

accounts for the queueing delays as part of the service of the channel. The load dependent service rate for the aggregated server is given by

$$U(i) = S(i) \qquad (4.2)$$

We thus have the requisite parameters for the model proposed in Figure 1.

4.2 Slotted Aloha

Users utilize the slotted channel, with the restriction that transmission can only begin at the start of a slot, although it may be any arbitrary slot. Collisions occur when more than one user attempts to transmit at the beginning of a slot, thus requiring retransmission of collided packets.

The aggregation is performed using the analysis of Kleinrock and Lam [Klei73]. The effective throughput rate, S(i) is given by

$$S(i) = Gq_t / (q_t + 1-q) \qquad (4.3)$$

where
G = the offered channel traffic, as a result of external traffic as well as packets retransmitted after collisions,

$$q = [\, e^{-G/K} + G/K\, e^{-G}]^K \qquad (4.4)$$

where q = 1 - P(newly generated packet collided)

$$q_t = \frac{K-1}{K}\, e^{-G} \qquad (4.5)$$

and q_t = 1 - P(recollision of blocked packet)
K = number of slots over which the user attempts to retransmit a collided packet, with a uniform probability.

A further relationship, for stable operation of the Slotted Aloha channel, is given by, [Robea], [Robeb]

$$L(i) = G\, e^{-G} \qquad (4.6)$$

By solving the non-linear eqs. (4.3)-(4.5), we obtain the throughput S(i) for a given offered load G, which in turn is obtained for each value of L(i), by solving the condition for stability (4.6). The final load dependent arrival rate L(i) is given by eq. (2.1).

The delay at the channel, D(i) is

$$D(i)=R + (1-q)/(q_t)*R + 1 + (K-1)/2 \qquad (4.7)$$

where R = delay to ensure packet was successfully received - i.e., the time for receiving an acknowledgement. Here q and q_t depend on the value of G and thus are load dependent.

The iterative solution of the throughput-delay equations is performed, till the channel throughput matches the arrival rate generated by the fixed number of users. The value of R was assumed to be 2 * D(i). As a result,a further level of iteration was required as outlined in Section 2.

Treating the service offered by the channel as that from presentation of a packet by a ready user till the successful transmission of that packet, the service rate is given by

U(i) = i/D(i) , where
i = number of ready users,
D(i) = delay in the channel with i ready users.

4.3 Carrier Sense Multiple Access

The Carrier Sense Multiple Access protocol has become the access scheme of choice in local networks. Several existing local networks have adopted this scheme or variations of it, such as in the Ethernet [Metc76], the NBSNet [Carp80] and FordNet [Biba79]. Further, the CSMA scheme and variations, with carrier detection and abort protocols have been widely studied [Fran80], [Bern80], [Lam80], [Span79] and Kleinrock and Tobagi [Klei75c].

We adopt the model proposed by Lam [Lam80] for the CSMA protocol with carrier detection (CSMA-CD). The channel is considered to be slotted in time, the slot size being 'a', the propagation delay in the channel. The user senses the channel for the presence of a transmission and contends for its use only when the channel is sensed to be idle. Collisions in the channel cause users involved in the collision to abort transmission immediately. The transmission probability in the next time slot is assumed to be 1/e, which yields good results [Lam80]. A ready user is considered to transmit in the next time slot, following any successful transmission, with probability 1.

Consistent with the other models, we assume that the service offered by the channel is from presentation of a packet by a ready user till the successful transmission over the medium. Thus, delays resulting from retransmissions due to collisions are also incorporated into the service time.
Let s be the probability of successful transmission in the next time slot, given a collision has occurred, (assumed = 1/e). We outline below the results for the delay encountered in the channel from [Lam80].

The mean message delay is

$$D(i) = \bar{x} + \frac{T}{s} + \frac{T}{2} - \frac{(1 - p_o)(2/L(i) + sT - 3T)}{2(B^*(L(i)s - (1 - p_o))} + L(i)\ \frac{(\bar{x}^2 + 2\bar{x}(T/s) + T^2(1+2(1-s)/s^2))}{2(1 - L(i)\ (\bar{x} + T/s))}$$

where $T = 2a$, and $\bar{x} = b_1 + a$,

where b_1 = mean value of the packet length, and

$$\bar{x}^2 = b_2 + 2b_1a + a^2$$

where b_2 = second moment of the packet length distribution.

$B^*(s) = P^*(s)e^{-sa}$, where

$P^*(s)$ is the Laplace transform of the probability distribution function $P(x)$ for the message length. p_j, the probability of 'j' new arrivals (ready users) in a time slot is given by

$$p_j = (LT)^j\ e^{-LT} / (\ j!\ ),$$

$j = 0,1,2....$

where L = the arrival rate.

As outlined in Section 2, we iterate on the packet delay D(i) and the arrival rate generated by the fixed number M of users/terminals, which have a mean think time of Z.

The service rate (load dependent) for the CSMA channel is given by

$U(i) = i / D(i)$

5. Experimental Results

The common parameters of interest for the three access protocols are the normalized packet transmission time, T=1, and the "think time", Z = 200*T. The load dependent arrival and service rates L(i) and U(i) are obtained by the technique outlined in Section 2 and 4, for the three schemes, for different values of the number of users attempting to utilize the channel. The parameters of the closed system with the M/M/1/N server for the channel match the values of L(i) and U(i) obtained from the iterative technique. For the FTDMA technique, a solution using the iterative technique was not required as the scheme is a fixed one, where the service offered by the channel is not dependent on the number of ready users, but only on the total number of users of the channel. The parameters L(i) and U(i) are thus directly available.

For the cases considered here, with arrivals generated from a set of M users, FTDMA was seen to have the lowest service rate while Slotted Aloha had the highest. For example, with M = 15 users generating traffic, and with 5 ready users with packets for transmission, FTDMA has a service rate of 0.588235, CSMA has a U(5) of 0.791228, while Slotted Aloha has a rate of 2.654352. The corresponding arrival rates for the set of 15 users, with i = 5 is 0.050. Tables 1,2 and 3 in Appendix 1 show the load dependent arrival and service rates for the three access protocols for M = 15. The iteration performed was to obtain the correct throughputs, when M users are present, and hence, the load dependent service rates. On the other hand, the arrival rates, are directly obtained according to eq. (2.1), for the closed system models.

Given the models for each of the access protocols, we study their behavior with respect to their adaptability to changes in load. The measure used is the relaxation time. Using the set of load dependent arrival and service rates L(i) and U(i) respectively, we obtain the relaxation time for each of the channel access protocols, as outlined in Section 3. Table 1 shows the relaxation time for each of the channel access protocols, for varying number of users, M. For example, for M equal to 15, the relaxation times for the different access schemes are as follows:

Slotted Aloha = 0.462612
CSMA = 3.341915
FTDMA = 9.240624

Table 1 also provides the corresponding average service rates for the different access protocols. The number of ready users, results in the load dependency factors for the arrival and service rates L(i) and U(i), from which the relaxation times were computed.

The relaxation time obtained represents the adaptibility of the channel to changes in load. For the given load conditions, FTDMA is seen to be the poorest in its adaptability, having the largest relaxation time. This reflects the fact that the FTDMA protocol is a fixed methodology, and as a result does not adapt to changes in the load offered to the channel. Slotted Aloha on the other hand is seen to adapt fastest to load changes, having the smallest relaxation time. It must be noted that the approximation used, to obtain the correspondence between L and G, for the Slotted Aloha channel limits the maximum arrival rate to 1/e. A peak in the throughput-offered traffic curve occurs at this value and hence, higher arrival rates lead to instability in the solution of the non-linear equation.

Figure 3 shows the graph of the relaxation time for the three access schemes studied, as the number of users that utilize the channel is varied. In the region of operation, for the load factors considered here for the Slotted Aloha access protocol, an increase in the throughput results as the number of users is increased. The relaxation time decreases with an increase in the number of users of the channel. It is seen that when an increase in the arrival rate (by an increase in the number of users utilizing the channel) results in an increase in the throughput of the channel, the adaptability of the access protocol to variation in the load is good. Thus, in this case, we notice that the relaxation time is lower. CSMA indicates a gradual increase in the relaxation time, with increase in the number of users, and appears to be in between the two other protocols, for the range of load values considered.

6. Conclusions

We have proposed a technique of obtaining in a uniform framework, models for shared communication channels. By considering a closed system with a fixed number of users utilizing the channel, and iteratively solving for the throughput - delay equations of the channel with its associated access protocol, we obtain the parameters for the model of the channel. We have used this common framework for modeling the FTDMA, Slotted Aloha and CSMA access schemes.

Given such a model of the channel, as a M/M/1/N server, with the load dependent arrival and service rates, a comparison of the different protocols has been made. The performance of the channel under transient conditions, and its adaptability to changing load was of interest. The relaxation time has been used as a performance measure to compare the transient behavior of these schemes.

For the given load conditions, FTDMA is seen to have the maximum relaxation time. Thus, the time the channel takes to reach a new steady state is longer than for the random access techniques studied. Given the load conditions, Slotted Aloha has the smallest relaxation time, indicating that it adapts to the change in load fastest. In conclusion, we note that although the load conditions considered here are not exhaustive, the relaxation time can be used as a parameter to characterize the transient behavior of such multiaccess communication protocols, providing additional insight into the applicability of particular access protocols.

Acknowledgements

Computing support for this research was provided by the Computer Science Center of the University of Maryland. We also gratefully acknowledge the suggestions of the members of the System Design and Analysis Group of the Department of Computer Science, of the University of Maryland.

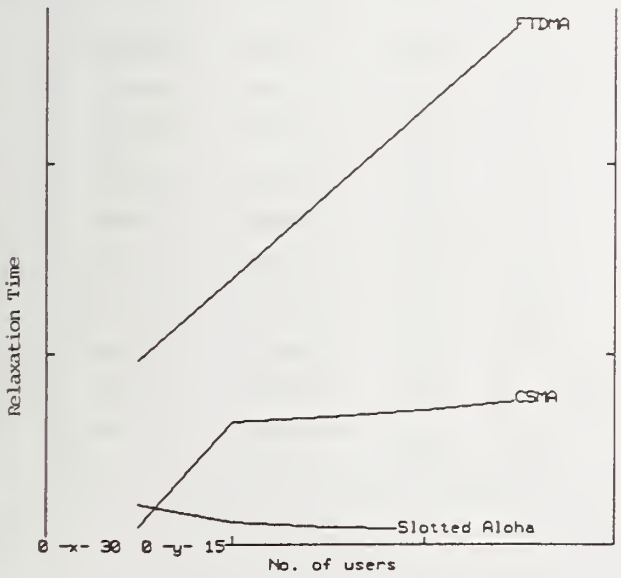


Figure 3: Relaxation Times for Access Protocols

REFERENCES

[Alme79]
Almes, G. T. and Lazowska, E. D., The Behavior of Ethernet-Like Computer Communications Networks, Proceedings of 7th SOSP, Asilomar, pp. 66-81, December 1979.

[Bern80]
Bernard, G., Non-Persistent CSMA - Abort Protocol for the Access to the Channel of Local Computer Networks, ERA N452 du CNRS, University of Paris-SUD, November 1980.

[Biba79]
Biba, K. J. and Yeh, J. W., FordNet : A Front-End Approach to Local Networks, Proceedings of Local Area Communication Network Symposium, Boston, May 1979.

[Buze79]
Buzen, J. P., Denning, P. J., Rubin, D. B., and Wright, L. S., Operational Analysis of Markov Chains, Technical Report 79-1, BGS Systems, January 1979.

[Carp80]
Carpenter, R. J. and Sokol, J., Serving Users with a Local Area Network, Computer Networks 4, 5, October/November 1980.

[Fayo77]
Fayolle, G., Gelenbe, E., and Labetoulle, J., Stability and Optimal Control of the Packet Switching Broadcast Channel, Journal of the Association for Computing Machinery 24, 3, pp. 375-386, July 1977.

[Fran80]
Franta, W. R. and Bilodeau, M. B., Analysis of a Prioritized CSMA Protocol Based on Staggered Delays, Acta Informatica, 13, 1980.

[Klei73]
Kleinrock, L. and Lam, S. S., Packet Switching in a Slotted Satellite Channel: Performance Evaluation, Proceedings of the NCC, Montvale, N.J., AFIPS Press, 1973.

[Klei75a]
Kleinrock, L., Queueing Systems, Vol I: Theory, John Wiley, New York, 1975.

[Klei75b]
Kleinrock, L. and Lam, S. S., Packet Switching in a Multiaccess Broadcast Channel: Performance Evaluation, IEEE Transactions on Communications 23, 4, pp. 410-423, April 1975.

[Klei75c]
Kleinrock, L. and Tobagi, F. A., Packet Switching in Radio Channels : Part I - Carrier Sense Multiple-Access Modes and Their Throughput-Delay Characteristics, IEEE Transactions on Communications COM-23, 12, December 1975.

[Klei76]
Kleinrock, L., Queueing Systems, Vol II: Computer Applications, John Wiley, New York, 1976.

[Lam80]
Lam, S. S., A Carrier Sense Multiple Access Protocol for Local Networks, Computer Networks, 4, April 1980.

[Metc76]
Metcalfe, R. M. and Boggs, D. R., Ethernet: Distributed Packet Switching for Local Computer Networks, Communications of the ACM 19, 7, pp. 395-403, July 1976.

[Powe81]
Powell, D. R., Performance Evaluation and Comparison of Dependable Channel Access Technologies for Locally-Distributed Computing Systems, Proceedings of The 2nd International Conference on Distributed Computing Systems, Paris, France, pp. 256-270, April 1981.

[Robea]
Roberts, L., Arpanet Satellite System Notes 8, NIC Document 11290,

[Robeb]
Roberts, L., Arpanet Satellite System Notes 9, NIC Document 11291,

[Shoc80]
Shoch, J. F. and Hupp, J. A., Measured Performance of an Ethernet Local Network, Communications of the ACM 23, Dec. 1980.

[Span79]
Spaniol, O., Modeling of Local Computer Networks, Computer Networks 3, 5, November 1979.

[Ster79]
Stern, Thomas E., Approximations of Queue Dynamics and their Application to Adaptive Routing in Computer Communication Networks, IEEE Transactions on Communications COM - 27, 9, September 1979.

[Toba77]
Tobagi, F. and Kleinrock, L., Packet switching in radio channels: Part IV - Stability considerations and dynamic control in carrier sense multiple access, IEEE Transactions on Communications COM - 25, 10, pp. 1103-1120, October 1977.

[Toba80]
Tobagi, F. A. and Hunt, V. B., Performance Analysis of Carrier Sense Multiple Access with Collision Detection, Computer Networks 4, 5, October/November 1980.

Appendix 1

Table 1: Fixed Time Division Multiple Access - Load Dependent Rates.

i	Arrival Rate	Service Rate	Delay
0	0.075	0.00	0.00
1	0.070	0.117647	8.50
2	0.065	0.235294	8.50
3	0.060	0.352941	8.50
4	0.055	0.470588	8.50
5	0.050	0.588235	8.50
6	0.045	0.705882	8.50
7	0.040	0.823529	8.50
8	0.035	0.941176	8.50
9	0.030	1.058824	8.50
10	0.025	1.176471	8.50
11	0.02	1.294118	8.50
12	0.015	1.411765	8.50
13	0.01	1.529412	8.50
14	0.005	1.647059	8.50

Table 2: Slotted Aloha - Load Dependent Rates.

i	Arrival Rate	Service Rate	Delay
0	0.075	0.00	0.00
1	0.070	0.482086	2.074320
2	0.065	0.988693	2.022873
3	0.060	1.514607	1.980712
4	0.055	2.071170	1.931275
5	0.050	2.654352	1.883699
6	0.045	3.251973	1.845034
7	0.040	3.887956	1.800432
8	0.035	4.534892	1.764099
9	0.030	5.205057	1.729088
10	0.025	5.922789	1.688394
11	0.020	6.644872	1.655412
12	0.015	7.391759	1.623430
13	0.010	8.163238	1.592505
14	0.005	8.997494	1.555989

Table 3: CSMA - Load Dependent Rates.

i	Arrival Rate	Service Rate	Delay
0	0.075	0.00	0.00
1	0.070	0.492173	2.031804
2	0.065	0.642290	3.113856
3	0.060	0.716507	4.186981
4	0.055	0.761330	5.253965
5	0.050	0.791228	6.319294
6	0.045	0.812550	7.384164
7	0.040	0.828506	8.448943
8	0.035	0.840897	9.513656
9	0.030	0.850797	10.578315
10	0.025	0.858891	11.642927
11	0.020	0.865630	12.707507
12	0.015	0.871330	13.772053
13	0.010	0.876213	14.836571
14	0.005	0.880443	15.901079

PREDICTING ETHERNET CAPACITY - A CASE STUDY

Madhav Marathe
Bill Hawe

Digital Equipment Corporation
HL 2-3/C09
77 Reed Road
Hudson, MA 01749

"How many users can I support if I install an Ethernet in my installation?" This is a question asked by many installation managers these days. Their worry is understandable because Ethernet bandwidth requirements of a typical user are not widely known. In this paper we estimate these requirements for a specific environment and a specific communications protocol. The environment chosen for this study was the program development or the time-sharing environment in a large University. The communications protocols assumed were similar to the existing Decnet protocols. The methodology presented here can be applied to other environments and protocols as well.

In order to calculate the Ethernet bandwidth requirements of a typical user we used a two step approach. First, we measured how users are using an existing timesharing system, and then we extrapolated this usage to a hypothetical Ethernet based timesharing system. We assumed that users on an Ethernet based timesharing system will issue the same commands as they do on present systems. This will be true at least initially. We therefore used workload measurements we had performed at several large Universities which do not currently have Ethernet based timesharing systems. On an Ethernet based system, commands issued at a terminal will cause data and control packets to be transmitted over the Ethernet. Depending on the configuration and the communication protocol used, some fraction of the characters to and from the terminals, to and from the disks and to the printers will be transmitted over the Ethernet. In order to make this distributed system work, there will also be some protocol control packets. We combined the user data packets and the protocol control packets together and calculated the total load offered by each user. We then used a packet level simulation model of the Ethernet to estimate the number of users at which the Ethernet will be saturated.

Our results indicate that several thousand users are required to saturate the Ethernet. We therefore expect that in this environment, other components such as the processors or disk servers will become bottlenecks before the Ethernet bandwidth is exhausted.

Key words: Ethernet, Ethernet performance, Ethernet simulation, higher level protocols, layered architecture, user level workloads, time-sharing, interactive program development.

1. Introduction

Both marketplace "pull" and technology "push" are the driving forces presently causing local area networks to become cost effective mechanisms for interconnecting a broad variety of devices within a moderate geographical area [DIGI82 and appendix A]. Ethernet is becoming a major commercial local network product and it is expected that a number of installation managers are considering installing an ethernet in their installations. One of the

questions many of these managers have is whether the ethernet has enough bandwidth to support the network traffic in their environment. In this paper we present a case study that we conducted for a university environment. Similar analysis can be carried out by interested installation managers for their specific installations.

Most installation managers are interested in determining the capacity of the ethernet in terms of the number of active (i.e. logged in) users it can support rather than in terms of the raw bandwidth of so many bits per second. In the traditional time-sharing environment the term "support" implies an acceptable level of response time at the terminal. We would have liked to use the same notion of "support" except that predicting the terminal response time in any environment is a very complex problem due to the large number of factors involved. We therefore decided to estimate the practical upper bound on the number of users when the limiting resource is the ethernet. This was done using a simulation model to determine the number of users that will saturate the ethernet i.e. that will cause the idle time on the wire to become zero.

2. Methodology

The most important characteristics of ethernet traffic are the packet size distribution and the packet interarrival time distribution. These are the input parameters to the packet level simulation model used in this study. Figure 1 displays the methodology used to arrive at these parameters. To simplify the analysis, all users were considered identical to each other. It was therefore necessary to determine the packets generated by or on behalf of a typical user every second. This can then be multiplied by the number of users to get the total packet load in the installation. The packets required by a user can be broken down into the data packets the user needs transfered and the protocol packets needed to acomplish this. These two can be further subdivided in order to get down to the parameters we can estimate. For example, the data packets the user needs transfered consist of characters to and from the user's terminal, the disk blocks to and from a file server, the characters going to the printer and mail messages going to other users on this ethernet or other remote locations. The protocol packets consist of acknowledgements and other flow control messages generated by various layers of the network software.

Determining the packets generated per second per user then boils down to estimating these parameters for the workload and protocols expected to be present on the ethernet. Not knowing exactly how such local networks will be used in the future, we made an assumption that, at least initially, users will be executing the same commands as before. This let us measure an existing installation which did not use an ethernet and from these measurements estimate the amount of data a user will need transfered when an ethernet is installed. We also made an assumption that the protocols used on the ethernet will be similar to the current Decnet protocols which were designed for long-haul networks. Obviously, Decent protocols for use over ethernet are expected to more efficient than the current protocols, so using current protocols provided us with a conservative estimate of the overhead needed to acomplish the user's data transfer needs.

3. Performance Metrics

As mentioned above, here we concentrate on the performance at the Ethernet level rather than at the terminal response time level. The delay through the Ethernet as a function of the offered load is the most important performance metric. The delay is often small compared to the delays at moving head disks and processors. The delay through the Ethernet consists of the waiting time to acquire the ether and the actual transmission time of the packet. The waiting time consists of the time spent defering to ongoing transmissions (see appendix A) and time spent in collisions and backoffs. The main parameter controlling the waiting time is the ratio of the one way propagation delay (ie: half the slot time) to the average packet transmission time. This is called "alpha". The performance improves as this ratio is made smaller [MARA80], [SHOC80]. This is because packets are exposed to collisions only during the first slot time of their transmission. Once a packet has been on the wire for that length of time it should not experience a collision (see Appendix A). Under heavy load the throughtput will be better if alpha is smaller [SHOC80].

The number of collisions a packet experiences in attempts to transmit is another interesting metric. Each collision causes the backoff range to be doubled. One would hope that, on the average, a packet does not experience many collisions. Measurements [SHOC80] and simulations [MARA80] have shown that

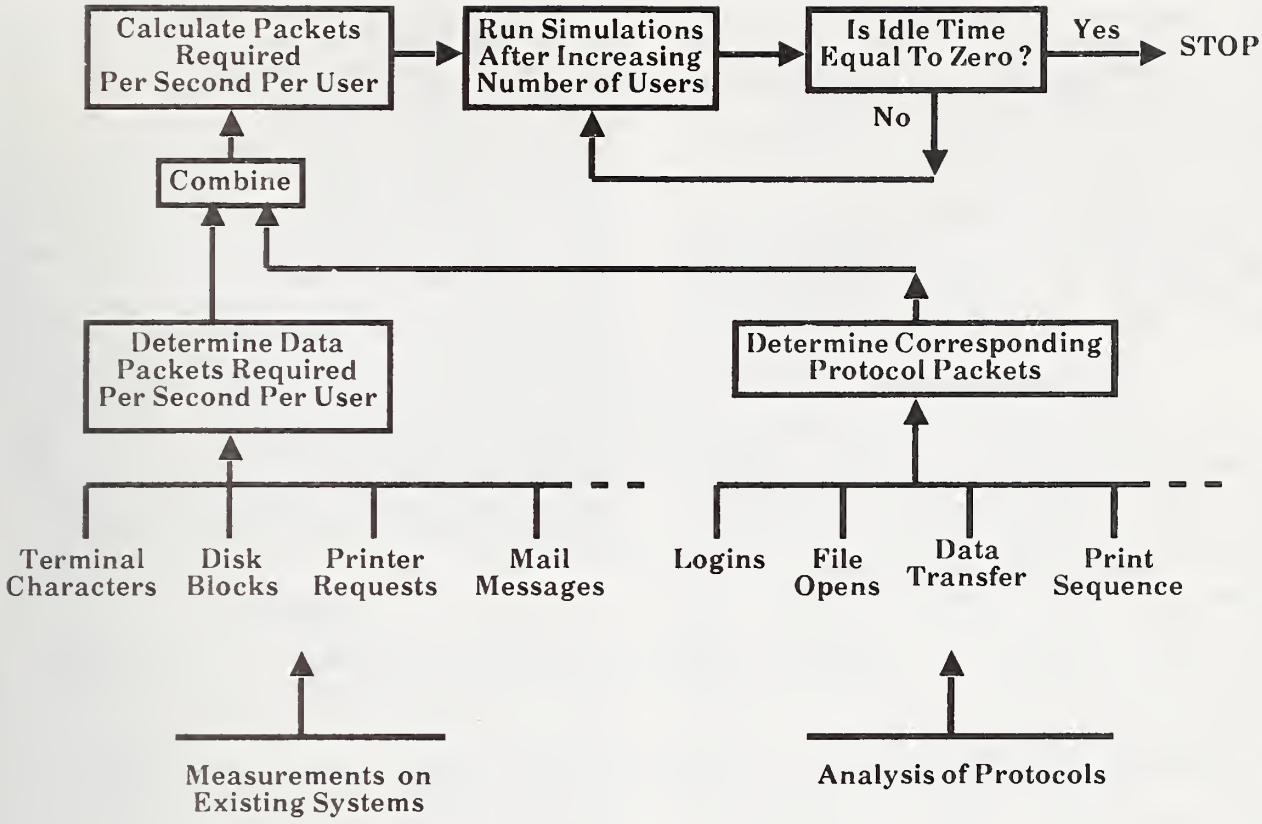


Figure 1. Analysis Methodology.

there are few collisions in typical systems.

One could devise other metrics relating to the higher level protocols such as number of packets transmitted for each user message, etc. However, here we examine worst case scenarios and do not pursue that topic. It should be noted that the higher layers often dictate the performance of the network and therefore they should be carefully studied [MQUI80]. They will produce extra packets for each user packet transmitted. These control packets contend with the data packets for the resources of the shared channel (Ethernet). They also contend with other applications for resources (CPU cycles and memory) at the transmitter and receiver. Here we only address the issues relating to the shared channel.

4. Program Development Workload

4.1 Configuration Assumptions

This study deals with the behavior of Ethernet in the interactive time-sharing and program development environments. There are many installations which fall in this category. Our analysis is based on the measurements at one such installation - a large University with a number of large hosts presently connected to each other by conventional direct connections. We asked the question: "What will the traffic on the Ethernet at this University look like if an Ethernet was installed today?". We hypothesized that for the near future, the university will still have the dumb terminals (asynchronous, character mode) that are being used today and that these will be connected through terminal concentrators to the Ethernet. Some terminals will still have direct connections to hosts since it is not likely that existing hardware will be thrown away. However, the users of those terminals still will generate Ethernet traffic in transfering files, sending mail, etc. The hosts will continue to have local secondary storage which will be used for swapping, paging and temporary workfiles. We assumed some level of file transfers and mail messages between hosts. Since we could not extrapolate the current traffic of this type into the superior sharing environment of the Ethernet, we assumed

three somewhat arbitrary levels for traffic of this type. Figure 2 displays the configuration we expect the university to have after an ethernet is installed. The figure indicates the types of devices present on the network. Since in this study we are estimating the number of users limited only by the Ethernet bandwidth, it is assumed that enough hosts, terminals, disks, etc. are added to the system to support the number of users used in the simulations.

4.2 User Profile

The workload can be specified by descriptions of the activities of the users. Users perform operations such as file edits, links, compiles, executes, etc. They also perform typical "house keeping" operations such as directory listings, file copies and deletes, etc. They send and receive mail and communicate with other users using interactive message facilities. The characteristics of the users were measured during heavy usage periods for several days at the University. I/O as well as program image related data was collected [JAIN82]. Table 1 summarizes some of the major points of interest in the user I/O characteristics. The table contains the mean value of several interesting statisitics. It is important to note that many of these statistics had bimodal and trimodal distributions. This means that more than the mean is required to fully understand the data.

In deriving the total network traffic generated by each user, the data and control packets generated at each protocol layer as a result of a user transaction were totaled and used to drive the Ethernet simulation. The amount of disk traffic present on the Ethernet will change with time as more intelligent servers and workstations are added to the system and as usage patterns change due to those new capabilities. We therefore have varied the load due to disk traffic in the simulation. Various amounts of the user disk traffic were sent over the network. Access rates of 0.00567, 0.0085 and 0.017 accesses/second/user were used. These correspond to 3.3%, 5%, and 10% of the accesses generated by each user to the local disk in this environment.

5. Results

Figure 3 contains a histogram of the Ethernet packet sizes generated by the user interactions coupled with the protocol model. The packet size includes user data (if any), the preamble, CRC and all other protocol fields from all protocol levels. The main contributor to the relatively large number of small packets (64 to 100 bytes) is the higher level protocol control packets. As mentioned previously, we have assumed the worst case for all protocol exchanges. This means that there are no piggybacked acknowledgements, etc. This imposes the

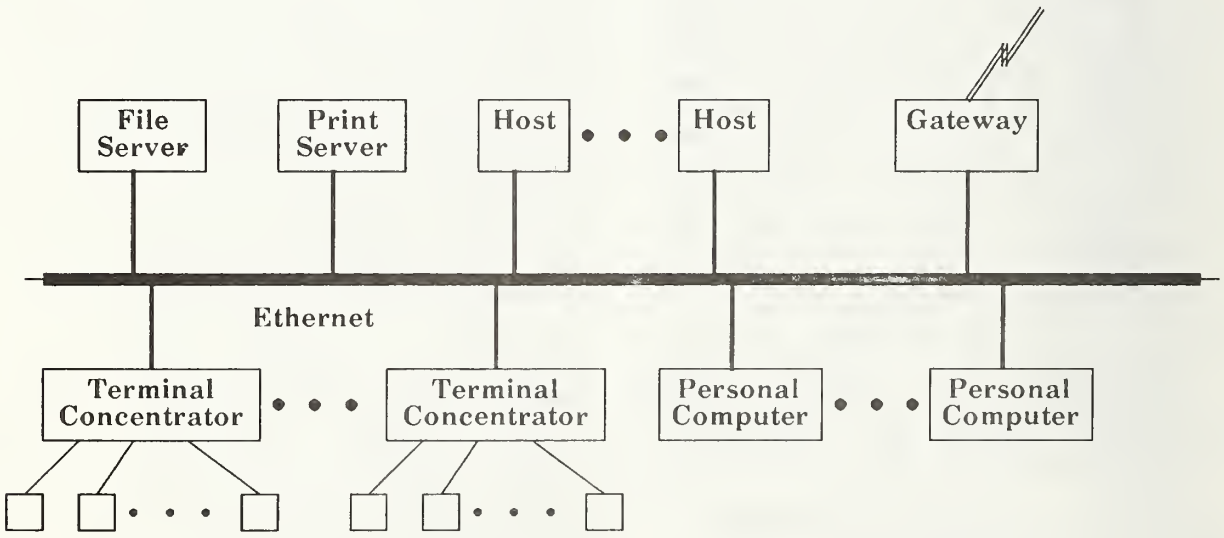
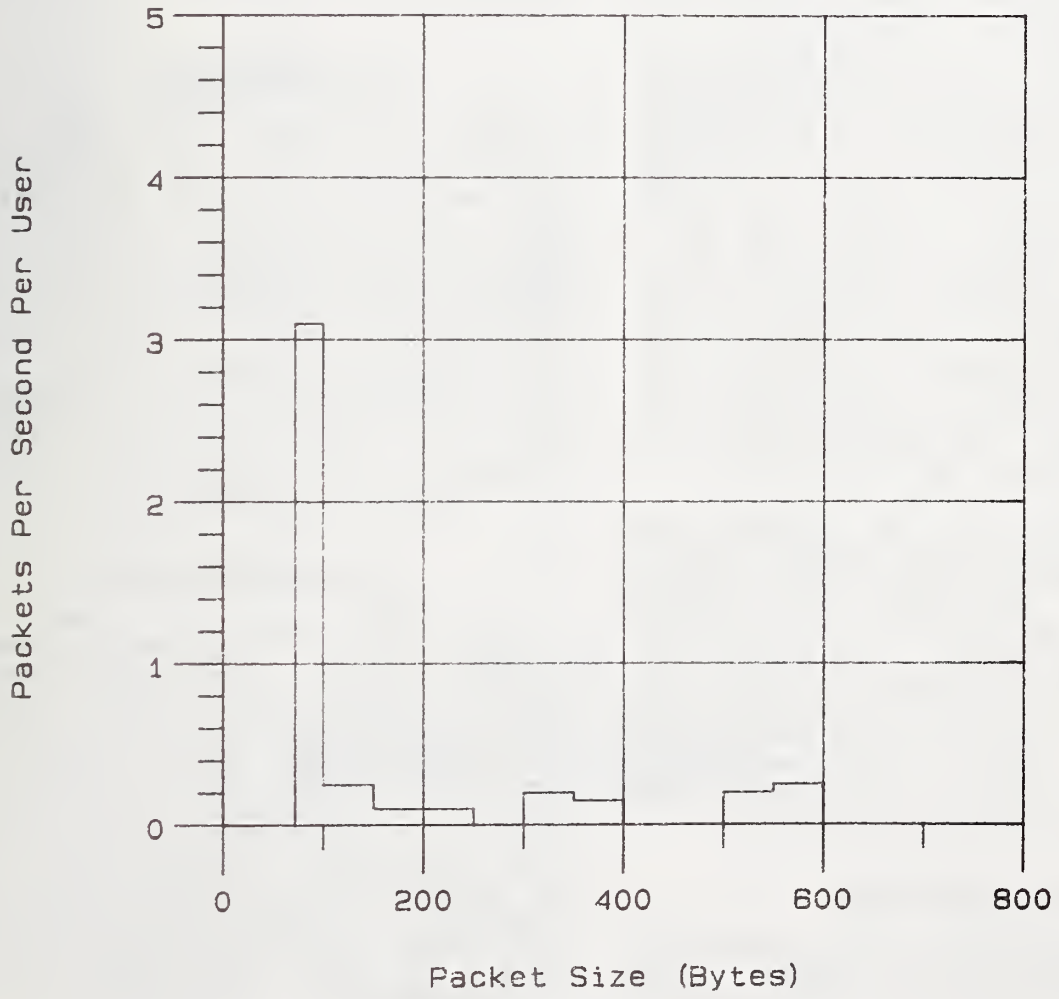


Figure 2. Local Area Network Components

PARAMETER	VALUE
1) Avg. Session Duration	1307 seconds
2) Avg. Input Size (Term -> Host)	10.7 bytes
3) Avg. Input Rate (Term -> Host)	0.16 inputs/sec
4) Avg. Output Size (Host -> Term)	26.5 bytes
5) Avg. Output Rate (Host -> Term)	0.34 outputs/sec
6) Avg. Printed Character Rate	2.91 chars/sec
7) Avg. Remote File Access Rate (Assumed Light Usage, See Text)	0.00567 accesses/sec
8) Avg. File Access Size (Directed Locally or Remotely)	3584 bytes/access

Table 1. "Per-User" Workload Summary

Figure 3. Packet Size Frequencies (Low Remote Disk Traffic)



heaviest load due to protocol control traffic. Since these are generally small packets, this distribution poses a demanding load on the Ethernet and should produce conservative results for this user workload.

Figure 4 shows the Ethernet offered load versus the number of users for this workload. The Ethernet specifications indicate that a maximum of 1024 taps may be connected to an Ethernet. The simulation conforms to that rule. Note that several users can share a tap. This is the case with terminal concentrators and hosts that have local terminals generating Ethernet traffic. In the figures presented here, the "number of users" corresponds to actual users - not to physical transceiver taps (of which there is a maximum of 1024). Figure 5 shows the idle time on the Ethernet going to zero at the overload points. Note that this occurs for an unusually large number of users.

Figure 6 shows the mean waiting time versus the number of users. Figure 7 shows the 90th percentile of the waiting time. The waiting time is defined as the time from when the packet becomes ready for transmission until it begins successful transmission. It includes all time spent defering, colliding and backing-off. As mentioned previously, three levels of remote file traffic were simulated. The "low level" corresponds to an access rate of 0.00567 accesses/user/second. The other two are for one and a half and three times the load due to that component. The waiting time in figure 4 should be compared to other waiting times contributing to the response time at the terminal. For example, waiting time at a disk is about a seek time (40 milliseconds) and the waiting time for a processor can be several tens of milliseconds depending on the load.

Figure 8 shows the number of attempts required to successfully aquire the channel as a function of the number of users. The number of attempts includes all collisions as well as the one successful attempt which aquires the channel. Note that even at an overload point with 2000 users, a given packet experiences an average of only one collision before a successful transmission. Figure 9 shows the 90th percentile of the number of attempts.

6. Conclusions

The results of the simulation indicate that the Ethernet has sufficient bandwidth to serve large numbers of users of the type characterized by the time-sharing workload. In practice, one generally does not operate the system with the steady state load near the system limits. The finite rate at which the hosts, disks, users, etc. can generate and process information will prevent the steady state loading from achieving this level.

The waiting time experienced in attempting to gain access to the channel was shown to be small compared to other sources of delay such as disks and processors. The number of collisions experienced by a packet attempting to acquire the channel was also shown to be quite low - even in the heavily loaded regions.

In summary, we can say that the Ethernet seems to be well qualified to carry the type of traffic experienced in the time-sharing environment. It has the capacity to support large numbers of users in this environment. Installation managers are encouraged to carry out similar analysis of their computing environment.

6.1 Discussion

Here we have shown that the Ethernet is capable of handling the traffic generated in this time-sharing environment. To build an effective network, the operation of the higher level protocols must be examined. The delays encountered due to processing and queueing can result in poor user perceived performance if care is not taken in their implementation. One should also examine other environments to see how similar or different they might be and how this affects performance. For example, the office environment is very important.

7. Acknowledgements

We would like to thank our colleagues in the Systems Performance Analysis Group, especially Rollins Turner, for obtaining the workload measurements as well as their help in analyzing the large amounts of data. We also wish to thank them, and others in Distributed Systems, for insights regarding the modelling of Ethernets in this environment. Finally, the people in Systems Performance Analysis and Distributed Systems Product Management who reviewed this paper deserve special thanks for their many useful comments and suggestions.

Figure 4. Offered Load (%)

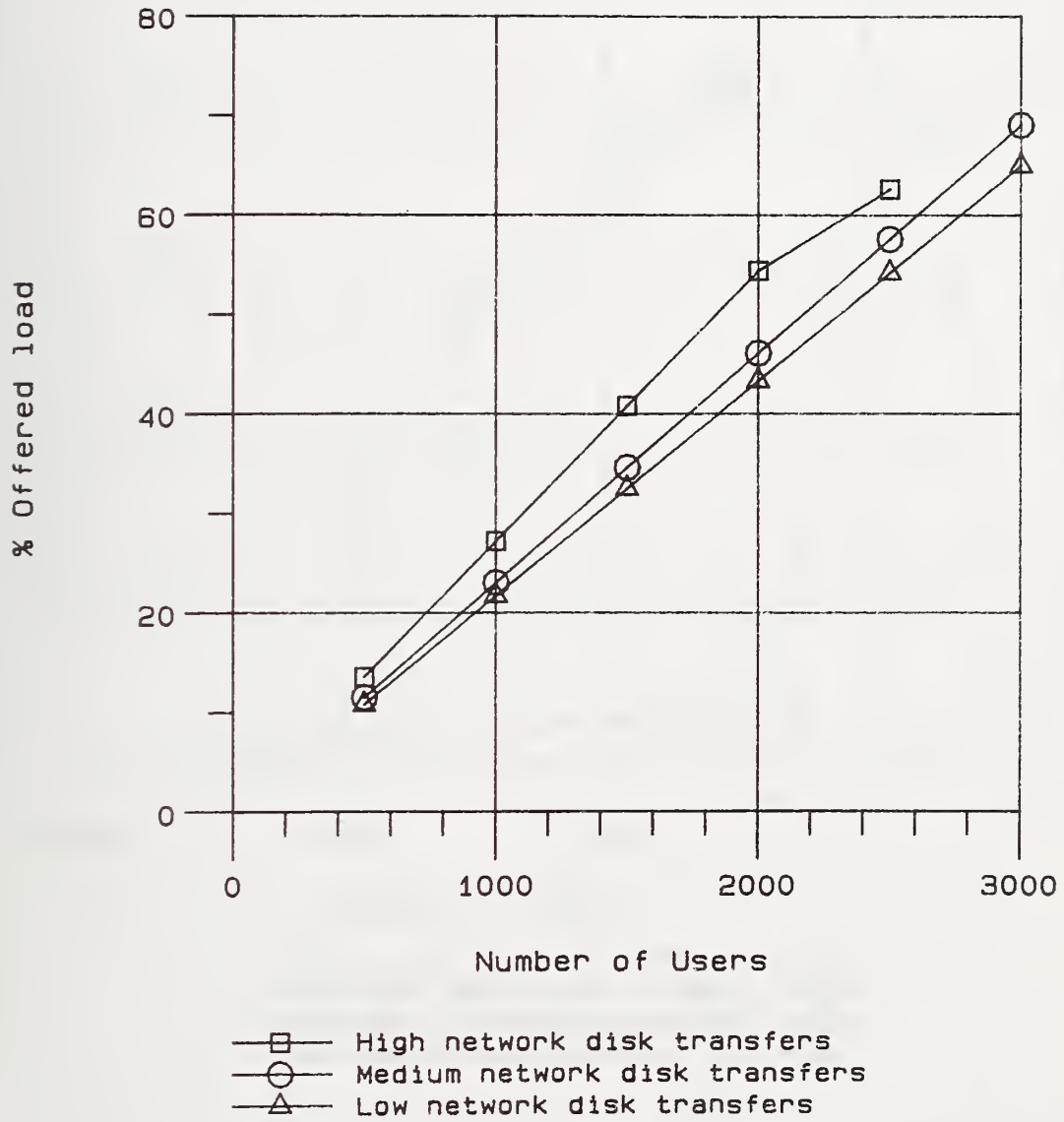


Figure 5. Percent idle time on cable

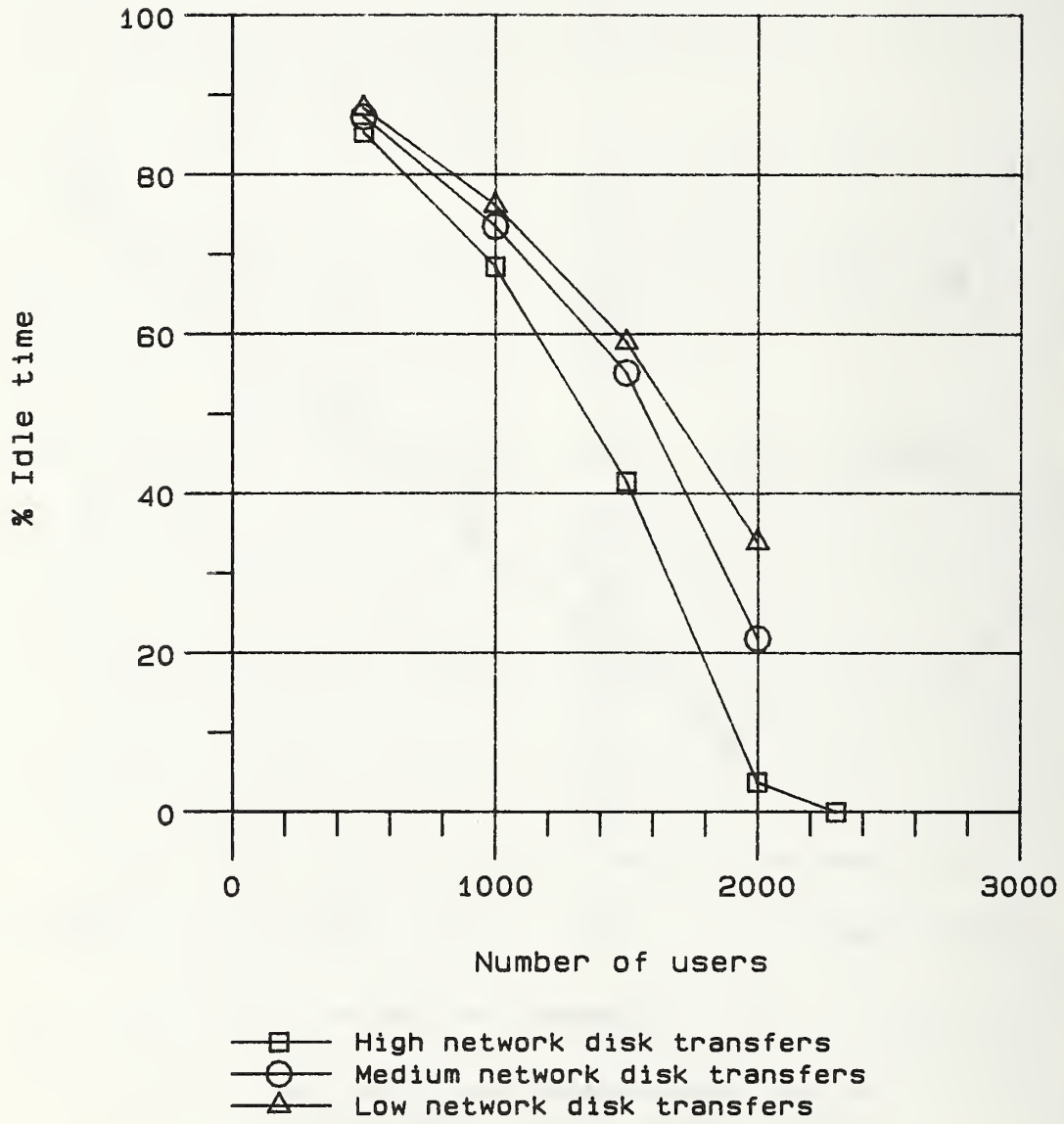


Figure 6. Mean waiting time

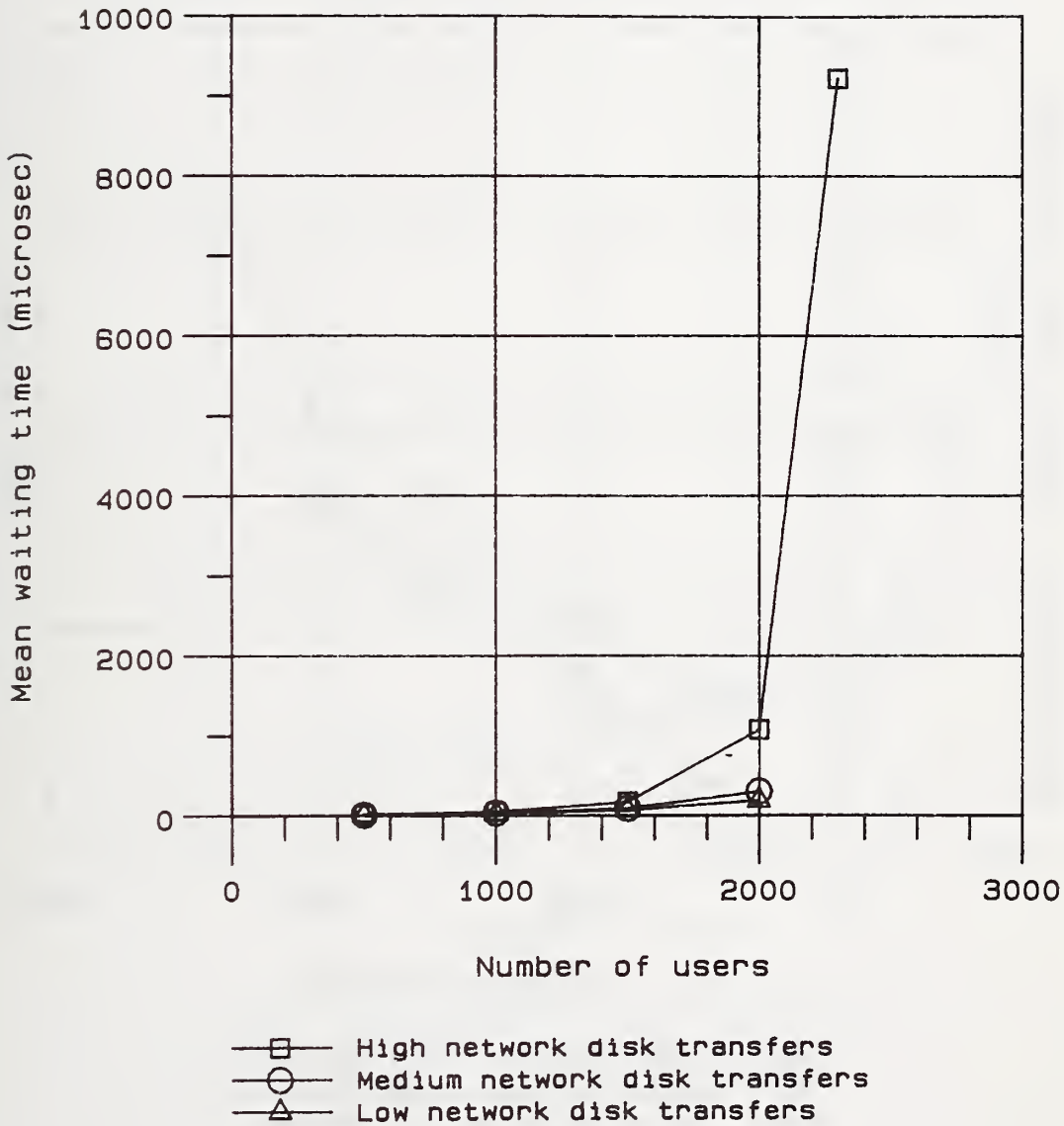


Figure 7. 90 Percentile of waiting time

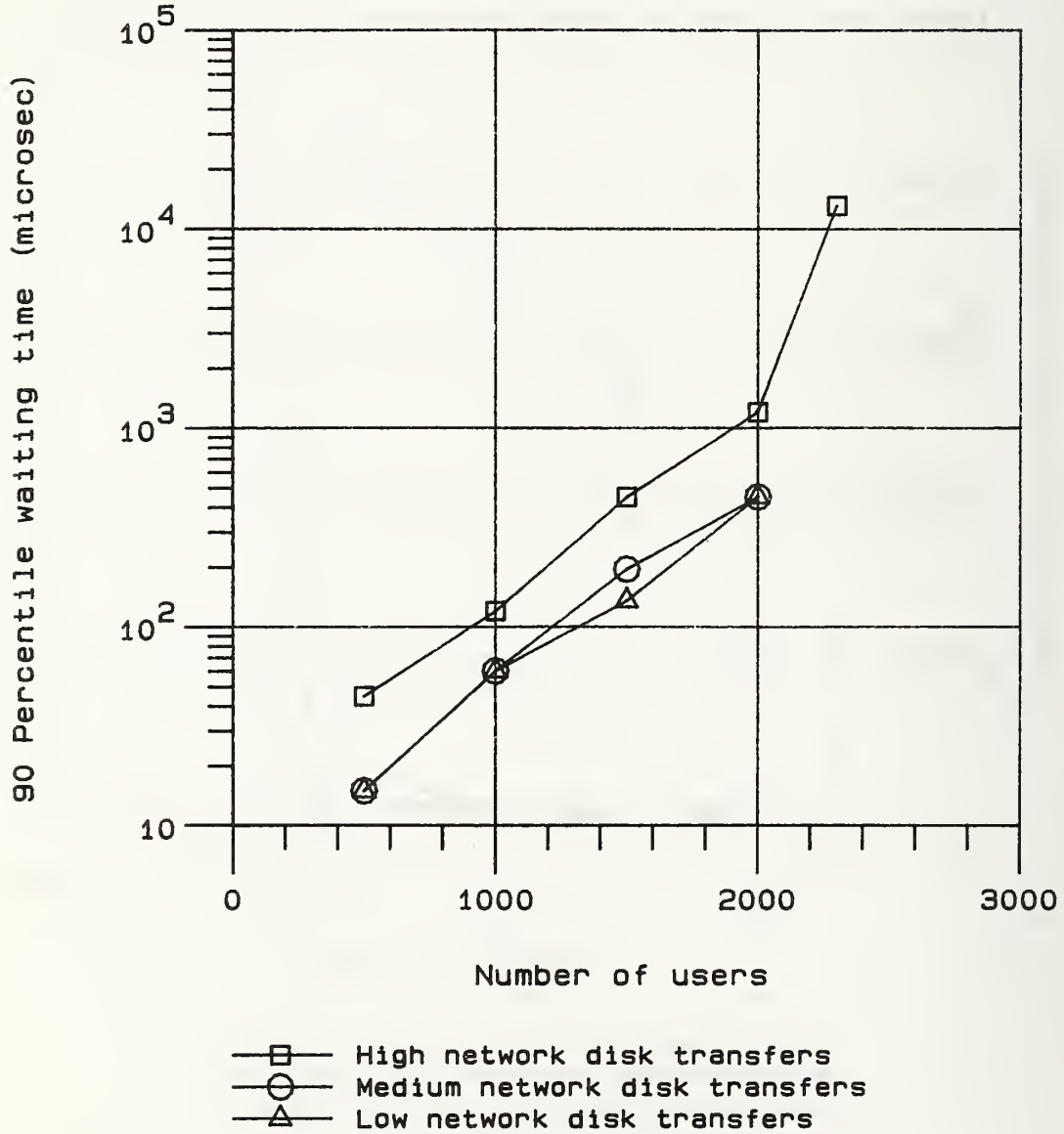


Figure 8. Mean number of attempts

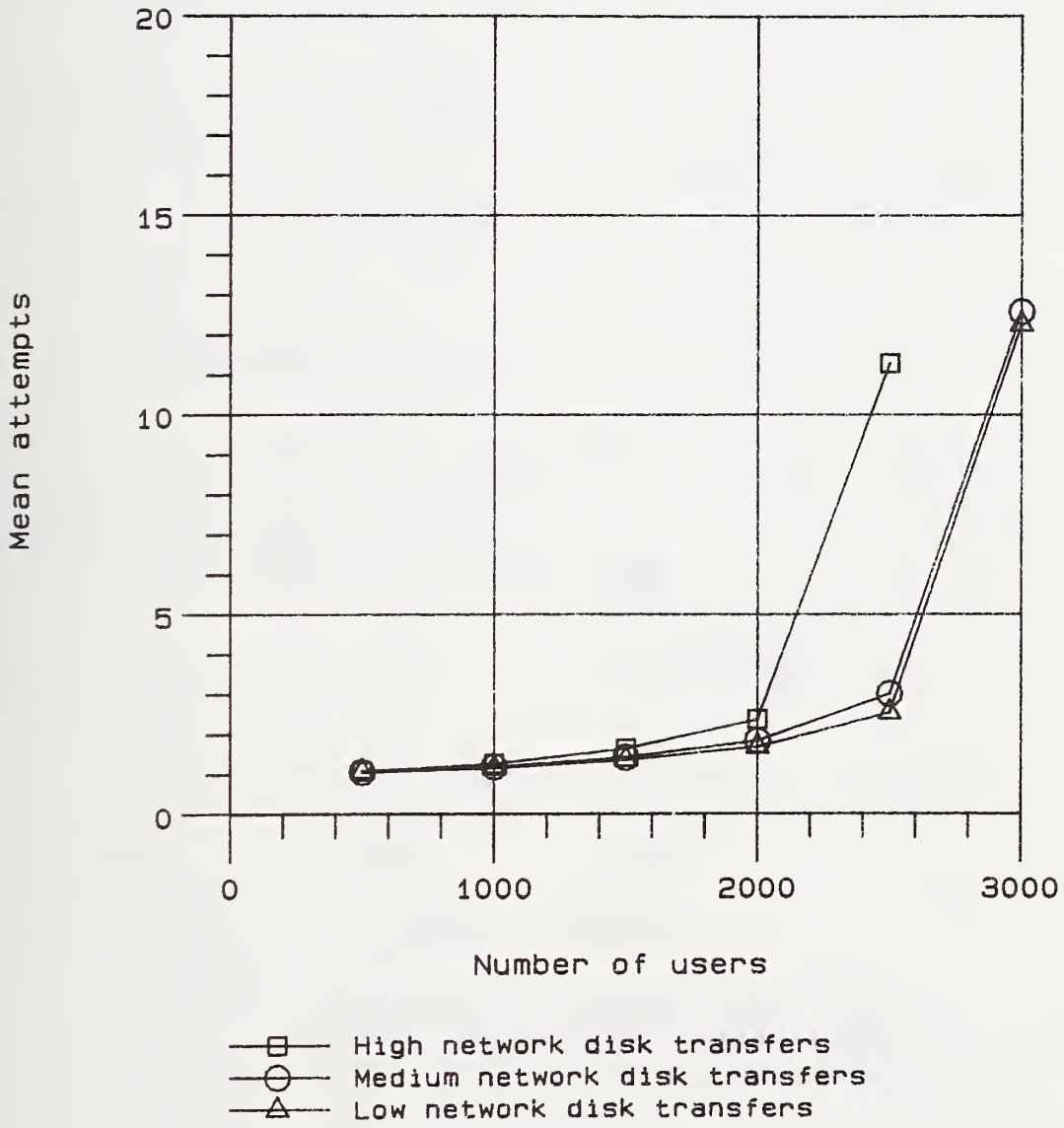
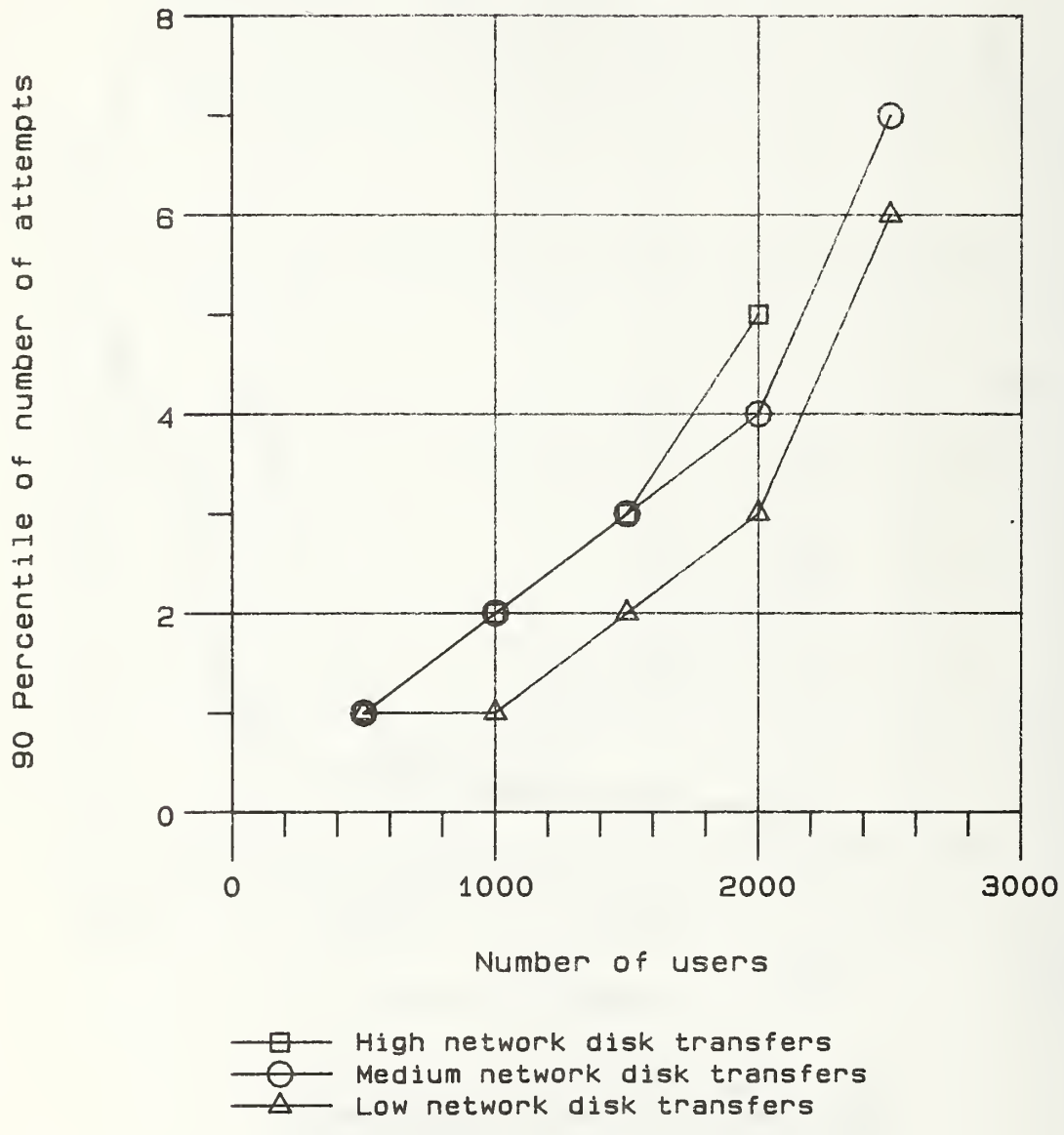


Figure 9. 90 Percentile of number of attempts



8. References

[COTT80] **Technologies For Local Area Computer Networks**, I. Cotton, Computer Networks, Vol.4, No.5, Oct/Nov 1980, pgs. 197-208.

[DAP80] **DECnet Data Access Protocol Functional Specification**, Version 5.6.0, Digital Equipment Corporation, October 1980.

[DECN80] **DECnet Transport Functional Specification**, Version 1.3, Digital Equipment Corporation, March 1980.

[DIGI80] **The Ethernet - A Local Network**, Version 1.0, Digital, Intel, Xerox, September 1980.

[DIGI82] **Introduction To Local Area Networks** Digital Equipment Corp. 1982. Order number EB-22714-18

[FREE80] **Updated Bibliography On Local Computer Networks**, H. Freeman, K. Thurber, ACM Comp. Comm. Review, Vol.10, No.3, July 1980, pgs. 10-18.

[JAIN82] **Workload Characterization Using Image Accounting**, R. Jain, R. Turner CPEUG 1982, Wash. D.C., October 1982.

[MARA80] **Design Analysis Of A Local Area Network**, M. Marathe, Comp. Network Symp., Wash. D.C., Dec. 1980, pgs. 67-81.

[METC76] **Ethernet: Distributed Packet Switching For Local Computer Networks**, R. Metcalfe, D. Boggs, Comm. ACM, Vol.19, No.7, July 1976, pgs. 395-404.

[MQUI80] **Local Network Technology And The Lessons Of History**, J. McQuillan, Computer Networks, Vol.4, No.5, Oct/Nov, 1980, pgs. 235-238.

[NSP80] **DECnet Network Services Protocol Functional Specification**, Version 3.2.0, Digital Equipment Corporation, October 1980.

[SESS80] **DECnet Session Control Functional Specification**, Version 1.0.0, Digital Equipment Corporation, November 1980.

[SHOC80] **Measured Performance Of An Ethernet Local Network**, J. Shoch and J. Hupp, Comm. ACM, Vol.23, No.12, Dec. 1980, pgs. 711-721.

[WECK80] **DNA: The Digital Network Architecture**, S. Wecker, IEEE Trans. Comm., COM-28, No. 4, April, 1980, pgs. 510-526.

APPENDIX A

A.1 Local Area Networks

Local Area Network interconnection schemes such as the Ethernet provide the the framework in which one can construct systems which provide sharing of resources in an effective manner. Two aspects of local networks which help achieve this goal are their speed and the fully-connected nature of their configurations.

To date, no one has come up with a standard definition of local area networks. However, most Local Area Networks do exhibit some general characteristics. Generally, they span areas of up to a few square kilometers. They are often contained completely in one or a small number of buildings. They usually have data rates in the range of 1 to 10 megabits/second. One group or organization almost always has complete control over the operation of the network. Since users are generally from one organization, there is a strong desire to access shared devices such as print servers, file servers, gateways, hosts, databases, etc. As a result, full physical connectivity is desirable. Because of the technology employed and the restricted size of the network, one observes lower bit error rates compared to conventional long-haul networks.

Because of the Local Area Network's speed it usually gets used for not only the traditional network communication but also for handling I/O traffic for shared disks, printers, etc. The personal computer workstations of the future will introduce a new class of traffic on the network. However, in the near future, the traffic on the local area network will consist of host/terminal traffic, host to host file transfers, mail, etc., specialized device traffic (print servers, etc.) and gateway traffic. We have made use of this fact in modelling the workload on these networks. More information on Local Area Network technology and architectures can be found in [COTT80] or [FREE80].

A.2 Ethernet

In this paper we are concerned with a Local Area Network built using an Ethernet [DIGI80], [METC76]. Ethernet uses a broadcast mechanism (coaxial cable) and a distributed access procedure to allow for sharing of the channel. The procedure is called Carrier Sense, Multiple Access with Collision Detection (CSMA/CD). Nodes on the Ethernet can sense on-going transmissions and defer theirs until the channel is idle. They also have the ability to monitor the channel while transmitting to determine if any other stations are also attempting to transmit. Once an idle channel is sensed a station may transmit. Because of the propagation delay on the wire, two or more stations may sense an idle channel and attempt to transmit simultaneously. This results in a collision. In order that all stations (including the one transmitting the packet) can "hear" the collision it is required that all packets be greater than a certain minimum size. That size is determined by a parameter called the "slot time". The slot time is slightly greater than the round trip propagation delay. Any station involved in a collision must stop sending the packet and reschedule the transmission. The algorithm used to determine when the next attempt should be made is called the truncated binary exponential backoff algorithm. Basically, every time a station is involved in a collision it backs off (ie: waits) a random amount of time whose mean is doubled every time it experiences a collision. The backoff time is reset after a successful transmission. This algorithm has the advantage of being fair to all nodes on the Ethernet since the same algorithm is executed by all. Ethernet performance is fairly robust. It degrades slowly and recovers well from momentary overloads [MARA80], [SHOC80].

The day to day operational performance of a 3 Mbps Ethernet is reported in [SHOC80]. It is interesting to note that the utilization of the channel was quite low. Less than 0.03% of the packets transmitted were involved in collisions while 99% acquired the channel with no latency.

One of the main reasons for Ethernet's popularity is because it uses a passive broadcast medium. This results in very reliable operation. Ethernet interfaces can be built using VLSI technology and thus made fairly inexpensive. Multi-vendor environments can be implemented by adhering to interface specifications at any of several levels. For instance. one may choose to provide compatibility at the wire tap, the transceiver cable, the port, higher level protocols, etc. Because of the heterogeneous enviornments in which Ethernets are used one can expect to see a great variety of traffic distributions. In this paper we have studied the traffic generated in a University environment and have predicted the performance of the Ethernet when used to satisfy the needs of that enviornment.

EVALUATING LOCAL NETWORK PERFORMANCE

JONAS HERSKOVITZ

Hughes Aircraft Company
El Segundo, California

ABSTRACT

This paper examines the factors involved in evaluating the performance of a local computer network. The configuration of the network consists of multiframe Digital Equipment Corporation (DEC) and Control Data Corporation (CDC) host computers connected using Network System Corporation (NSC) HYPERchannel Adapters. Software and hardware measurement experiments were implemented to definitize the performance characteristics of major subsystem components in the network system such as host computer, network adapter and interfacing channels. Measurements compare the contributions of host protocol overhead with network adapter processing and trunk transfer rates in defining network performance.

Key words: Local networking; computer network; performance evaluation; measurement; network performance; mathematical modeling.

1. INTRODUCTION

To solve a particular distributed processing requirement, Hughes Aircraft Company's Space and Communications Group embarked upon the development of an integrated system of hardware and software products. JANET (Joint Applications Network) comprises a major portion of this system. The JANET software provides intercomputer communications among a group of nonhomogeneous mainframes in a local point-to-point data network. Other elements of the system provide remote timesharing access, file transfer and general graphics display processing capabilities.

Utilizing the network concept in configuring the system increased system flexibility and processing power. At the same time, greater configuration complexity compounded the difficulty of evaluating overall system performance. Although multimainframe system performance in a network environment is of great concern, the immediate problem centered about evaluating a JANET local high speed network, with a goal of applying these techniques during network development and in the operational phase. To assist in monitoring and maintaining the network in an operational environment, JANET development included a network monitoring system.[1]

Although a number of studies have been undertaken to study the performance characteristics of networks of under 10 megabits per second,[2,3] comparatively little has been done to evaluate high speed local networks, i.e., a network with data transfer capabilities exceeding 50 megabits per second.[4] This lack of high speed network performance evaluations may be due to a number of factors, including cost considerations, time and effort involved, relative difficulty, inadequate measurement tools and a lack of performance criteria. All too often, performance measurements lag behind system design and development. This lag in performance evaluation methodology is particularly evident in high speed local area networks.

In the JANET development however, critical time dependent applications, such as file transfer and display support software, required the full exploitation of the network adapter

capabilities. Therefore, in support of network development, a parallel investigation was initiated to evaluate parameters affecting network performance. As a result of this effort, methods and measurement tools were devised which evaluated high speed network protocol timing at various interface levels.[5] Also, as a consequence of the investigation, viable methods and techniques evolved for measuring and evaluating network performance. Though the measurements cited in this paper are oriented specifically toward JANET, the approach is applicable in evaluating almost any local network.

2. JANET NETWORK ENVIRONMENT

The JANET network architecture grew out of a requirement analysis and an industry survey conducted in 1977. Major operational changes mandated the network approach. The primary change involved the conversion of large computer applications to a standard Control Data NOS operating system and the distribution of new functions to minicomputers. Distributed processing necessitated a reorientation of data sharing between programs. Another major factor influencing the networking approach centered about the elimination of specialized and obsolete graphics hardware with its inherent programmed dependencies. As an added benefit, off-loading display processing to minicomputers relieved the large mainframe for more efficient application oriented processing.

JANET's configuration, illustrated in Figure 1, employs Network System Corporation (NSC) HYPERchannel™ processor adapters in a local environment for communication support to transfer large volumes of data at high rates of speed between various host computers.

Coaxial cables interconnect data trunks between adapter devices, driving the trunk at a specified data rate of up to 50 megabits per second. Each NSC adapter contains a host dependent I/O device interface, a common nucleus microprocessor with control and buffer memories,

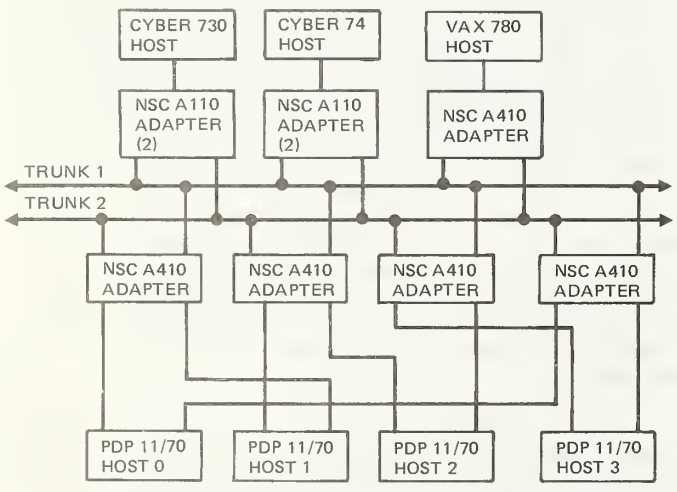


Figure 1. Janet's Development and Test Configuration

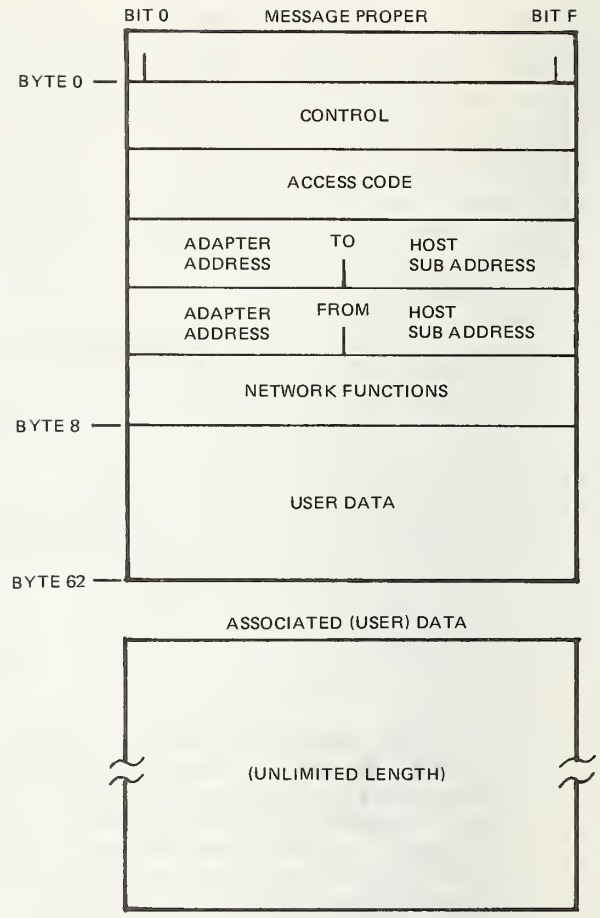


Figure 2. Network Packet Structure

and trunk drivers which accommodate one to four data trunk interfaces.

Among communicating hosts, NSC adapters provide a reservation protocol, where the receiving adapters are "reserved" for the entire duration of the communication. Between themselves, adapters employ a prioritized "Carrier Sense Multiple Access" (CSMA) trunk protocol.[6]

Defined and formatted network messages directed by the sending and receiving hosts provide the means for communication between interfacing processors. Network messages consist of the "message proper" and optionally associated (user) data as shown in Figure 2.

The message proper contains 64 bytes (8 bits/byte) of information part of which indicates the source host, some fields control the message destination and function, and there is limited space for user defined data. Optional associated (user) data of unlimited length follows the message proper, though in a practical sense memory buffer sizes in communicating hosts constrain the length of data transmitted in a single transaction.

Host specific processor adapters connect a processor channel to data trunks which function with other adapters to send and receive network messages. NSC's model A410 adapters provide the

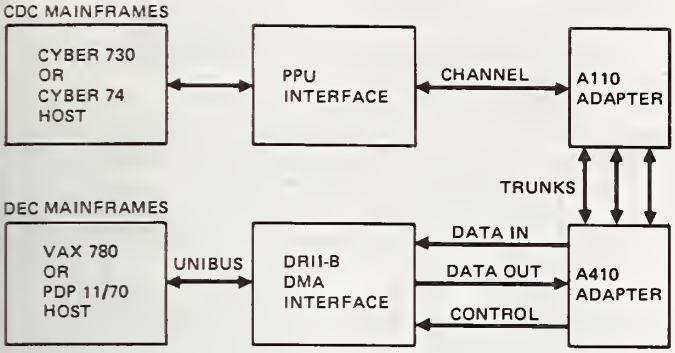


Figure 3. Local Network Configuration of CDC and DEC Mainframe Processors

communication interfaces with DEC's UNIBUS on the PDP 11/70 and VAX 780 computers; the NSC model A110 adapter, counterpart to the A410, interfaces with the CYBER PPU channel. Figure 3 indicates the NSC adapter interfaces with the CDC and DEC mainframes.

The DR11B I/O attachment to the UNIBUS serves as a general purpose direct memory access (DMA) device, and operates directly to or from mainframe memory, moving data between the UNIBUS and the user A410 adapter device. In an analogous manner, a dedicated peripheral processing unit (PPU) in the CYBER system senses and controls the movement of data on the channel to the A110 adapter and to CYBER memory.

Each host mainframe contains network software to support intermachine and intramachine processing. JANET software is viewed as three levels of network protocol handling routines coexisting under each host's operating system, as illustrated in Figure 4.

At the application level, the Network Interface Package (NIP), resides in the application field length or task partition. The core resident JANET Control Program (JCP) schedules all incoming and outgoing messages, multiplexes connections, and routes messages to their destination host. NIP handles the communication between the user application and JCP residing in its local host. The Adapter Interface Program

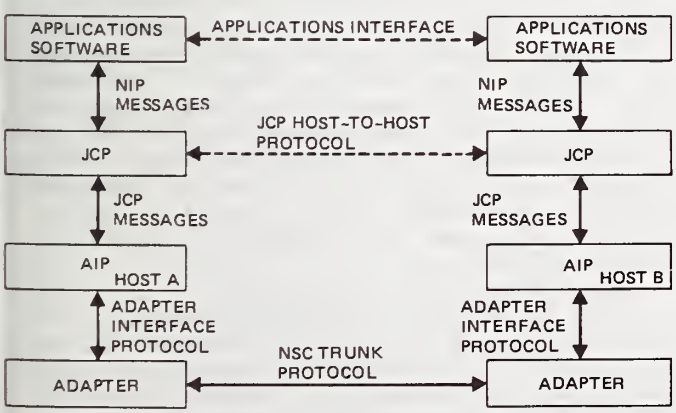


Figure 4. Layered Janet Protocols

(AIP) represents the I/O hardware driver for a particular adapter type, which issues or receives appropriate function code sequences, controlling the movement of network messages and user data.

In addition to the network protocol handling routines unique to each host's operating system, the HYPERchannel adapters employ between themselves, a trunk access protocol based on a Carrier Sense Multiple Access scheme with prioritized staggered delays.

In examining JANET's network configuration, we observe three primary hardware components involved in network communication:

1) The network data trunks

2) The communicating interface units, NSC Adapters

3) The processor I/O channel devices

Each of these components acts asynchronously within the network and displays its own performance characteristic. The communicating interface units, usually called adapters, occupy a unique status. Positioned at the nodal points of the network, adapters intercept and forward communications and data transfers between processors. Its strategic system location and its inherent role as a communication device, put the network adapter at the focal point for evaluating network performance.

3. MEASUREMENT APPROACH

Even before integrating of JANET protocols, network performance measurements were obtained by using customized driver programs for communicating between tasks. Basically, the tasks in each host echoed the transfer of specified data back and forth a given number of times. The test software timed the transfer sequence, using the system clock as the reference. As network software development progressed, application level test programs written in Fortran incorporated Network Interface Protocol (NIP) requests such as network reads and writes (NETRD and NETWT) to evaluate task-to-task communication.

Software instrumentation of test driver code to obtain application-to-application level protocol timing, though elementary, proved quite valuable. By varying the length of data transferred in a controlled network environment and measuring the elapsed duration, we deduced both the software protocol overhead and the network transfer rate. Inferences were made from an analysis of the acquired test data that a linear relationship exists for determining network transfer time. The model assumed that network delays were defined by a ratio of the data length transferred and the bus transfer rate, plus invariant protocol overhead effects. Because test data showed a high degree of linearity, a simple mathematical model fit well and proved useful for evaluating network performance.

Single stream point-to-point timing tests fit the linear mathematical model equation:

$$T_n = (L_d/R_b) + P_o$$

where

T_n = is the network transfer time

R_b = is the effective bus transfer rate

L_d = is the data length transferred

P_o = is the protocol overhead time

As network complexity increased, this mathematical model proved simplistic and did not explain anomalous network behavior, particularly prevalent in a multitasked environment. In addition, expansion of network functions mandated more precise and definitive measurements. Maintaining the network and upgrading the system to increase its efficiency required a greater understanding of network component behavior and more detailed performance measurements.

To accomplish a more detailed comprehension of network implementation we decided to employ sophisticated hardware monitoring measurement techniques in conjunction with software instrumentation.

Hardware monitoring techniques were selected since they provide an exceptional passive method of instrumenting and measuring network activity. This measurement approach does not introduce an artifact which would affect network performance, but requires the installation of a monitoring device capable of resolving high speed signals. The Dynaprobe D7916 Hardware monitor chosen has a pulse count resolution of between 20 and 40 nanoseconds and measures elapsed time parameters to within 100 nanoseconds, all within tolerances needed for measuring network performance.

The D7916 hardware monitor illustrated in Figure 5 contains 32 probes which detect and decode input signals passing through externally programmed logic circuitry. Occurrences, decoding and logical combinations of these input pulses, simultaneously feed 16 buffered output counters or timers for recording on magnetic tape. Data reduction software processes the captured information and provides convenient statistical and graphical reports for analysis and evaluation.

Regardless of the measurement techniques used in performance measurements, most serious considerations center on the definition of information required, determining the feasibility of

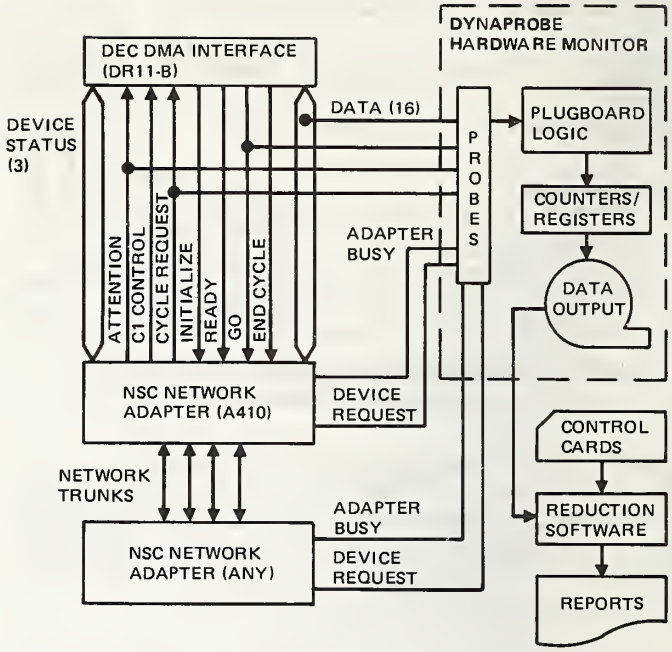


Figure 5. Dynaprobe Hardware Monitoring and Measurement Configuration

obtaining that information and finally understanding and applying the results in a positive manner to effect ultimate system improvement.

4. NETWORK PERFORMANCE PARAMETERS

Earlier in this paper, a primary critierien used for evaluating network performance was cited, namely message timing, which is defined as the end-to-end timing delay or the elapsed time from initiating message transfer until receipt and acknowledgement. By assuming a linear network model and measuring the transfer time for messages of various lengths an estimate was made of the network protocol overhead and the effective transfer rate. This performance measurement approach, implemented by instrumenting software, proved adequate in a dedicated environment where conflicting activity does not introduce communication delays in the network and processing delays at the nodes. However, during normal operations contention effects contribute heavily to timing delays. In addition, in any measurement process we want to avoid the introduction of a software artifice which slows down network functions. It is most desirable to have a measurement technique which is transparent to the network software yet measures system activity and performance and retains its applicability during the developmental and operational phase. As a consequence of our efforts the use of hardware monitoring techniques gave us an insight on how to accomplish this most difficult performance measurement task.

In examining alternative options, for measuring network performance characteristics in satisfaction of development and operational requirements, we decided to combine elements of software instrumentation with hardware monitoring techniques.

Selecting hardware monitoring for network performance measurement was not an easy choice since the use of this technique was never demonstrated in evaluating performance of a high speed, state-of-the-art network. Documentation of signals that could be probed in the adapter device was practically nonexistent. On the positive side, we had Dynaprobe monitoring equipment and some prior measurement experience in its use and there was a definite need for its application.

Before implementing the hardware monitor measurements, we researched a multitude of network adapter and interfacing signals to ascertain their suitability for measuring network activity. Probing signals of interest, while driving the network with predetermined instrumented software, permitted validation and calibration of hardware measurements.

The objective of these experiments was to select a list of measurable parameters indicative of network performance. Some network performance characteristics chosen for measurement are:

1) Program-to-program network transfer time

2) Data transfer rates

3) Network software overhead

4) Effective data trunk transfer rates

5) Effective UNIBUS transfer rate

6) Contention delays

7) Network protocol timing measurements

8) Network bottlenecks

The resolution of these objectives makes for an interesting discussion and the remainder of this paper will deal with some network measurements and their implication in evaluating network performance.

5. NETWORK ADAPTER PERFORMANCE CHARACTERISTICS

Lying at the nodal points of the network, NSC's adapters provide a convenient test bed for interrogating communication between mainframe processors. For that reason we initiated our research by examining network documentation and deriving a test point library. Table 1 shows a partial list of defined signals and their physical system location.

Concurrent software and hardware measurements of network components were performed even before implementation of the full JANET protocol. Its primary purpose was to evaluate component hardware and to establish a baseline for evaluating firmware changes.

Table 1. NSC Adapter Test Point Library

SIGNAL	TEST POINT	LOGIC BOARD
BUSY	A04	AD3DS
RECEIVE DATA	B01	AD3DS
TRANSMIT DATA	A02	AD3DS
TRUNK FRAME*	D05	AD3DS
TRUNK TRANSMITTING*	E03	AD3DS
TRUNK LAST BYTE	A05	AD2C
BYTE COUNTER CONTROL	E04	AD2C
TRUNK FUNCTION DECODE ENABLE*	F05	AD2C
REQUEST TO SEND*	C06	AD6
CARRIER ON	D01	AD6
DEVICE REQUEST	A06	AD5CB
DEVICE WRITE	C03	AD5CB

*SAME COUNT AS BUSY

In the measurement process, we evaluated actual component performance as opposed to vendor specification. An example of a series of parametric measurements performed to evaluate NSC adapter performance characteristics is shown in Table 2. The table lists measured results obtained from software and hardware monitoring experiments using test drivers which echo data back and forth between two PDP 11/70 host computers.

By integrating some understanding of the network adapter's role in network communication we judiciously chose and derived the network and adapter performance characteristics shown in Table 3.

Knowing the message length (64 bytes or 512 bits per message) and the amount of user data transferred across the network for each test, we combined trunk busy measurements to calculate the message and user data transfer rates for the entire spectrum of tests. We derived the trunk transfer rate for user data by assuming that in a dedicated environment message processing (i.e., trunk busy time) remains invariant and indepen-

Table 2. Parametric Measurements of Network Transfer Signals.

SIGNAL MEASUREMENT	TYPE	MESSAGES ONLY	20 BYTES MESSAGE	200 BYTES MESSAGE	4000 BYTES MESSAGE
BUSY	COUNT	10,000	22,1B2	22,000	2B,065
BUSY	TIME*	0.0842	0.1672	0.2233	1.4B08
RECEIVE DATA	COUNT	2,000	4,1B2	4,000	6,000
RECEIVE DATA	TIME*	0.0147	0.0225	0.0513	0.6637
TRUNK FRAME	TIME*	0.0285	0.0532	0.0818	0.7021
TRUNK TRANSMIT	TIME*	0.0418	0.0829	0.1111	0.7396
TRANSFER FUNCTION ENABLE	TIME*	0.0B20	0.1623	0.21B4	1.4745
DEVICE REQUEST	COUNT	64,357	B4,387	266,924	4,081,123
DEVICE REQUEST	TIME*	0.0220	0.0288	0.0907	1.3957
DEVICE WRITE	COUNT	32,000	42,387	134,925	2,034,951
DEVICE WRITE	TIME*	0.0110	0.0146	0.0459	0.7045
TEST DURATION, SEC		11	22	22	63
BYTES TRANSFERRED EACH WAY		64,000	84,000	264,000	4,064,000

*IN SECONDS

Table 3. Network Performance Characteristics.

TEST PARAMETER	MESSAGES ONLY	20 DATA BYTES/ PASS	200 DATA BYTES/ PASS	4000 DATA BYTES/ PASS
TEST DURATION, SEC	11	22	22	63
KILOBYTES TRANSFERRED, TOTAL	128	168	528	8128
KILOBYTES/TEST SEC	11.64	7.64	24.00	129.02
ADAPTER TRUNK BUSY TIME, SEC	0.0842	0.1672	0.2233	1.4808
KILOBYTES/SEC TRUNK BUSY	1,520	1,004	2,365	5,488
MESSAGE AND DATA TRANSFER, KBITS	1,024	1,344	4,224	65,024
MESSAGE AND DATA TRANSFER RATE*	12.2	8.0	18.9	43.5

* MBPS

dent of the associated data block size. Test results indicate that the message transfer rate without associated data measures 12.2 megabits per second. The "message proper" accompanied by associated data incurs additional overhead, which for small data transfers reduces the effective trunk transfer rate even further. As the packet size transferred increases, the effective transfer rate also increases. For example, when transferring 4000 byte data blocks the test measurements show a trunk transfer rate of 43.5 megabits per second, close to the specified 50 megabit transfer rate.

Since NSC touts their HYPERchannel transfer rate at 50 megabits per second, the difference between the effective and theoretical transfer rate measures the adapter processing delay or overhead in driving the network trunk.

This network performance experiment not only evaluated the effective network transfer rate using NSC's HYPERchannel adapters, but also identified and validated vital parameters that measure network activity such as "trunk busy time" and the extent of data received or sent through the network adapter. Network monitoring was also applied in evaluating the effects of microcode changes during a mandated adapter trunk board retrofit program.

Table 4 compares network performance parameters obtained before and after adapter retrofit using the identical network driver software and measurement tests. Notice that the adapter retrofit program reduced trunk throughput by 3 to 6 megabits per second, and that percentagewise the change was most significant when transferring short data packets.

The comparison of trunk transfer rates, before and after adapter retrofit, is shown graphically in Figure 6. In a semilog plot of trunk transfer rate versus packet size we observed lower trunk performance after retrofit and a similarity in transfer behavior between the curves. In both instances there is a marked drop in trunk transfer rate when the message is accompanied by a small amount of associated data. An enhancement which allows embedded user data within the message proper (see Figure 2) was justified on the basis of these measurements.

Table 4. Comparison of Network Performance Before and After Adapter Trunk Retrofit.

TEST PARAMETER	MESSAGES ONLY	MESSAGES & 20 DATA BYTES	MESSAGES & 200 DATA BYTES	MESSAGES & 4000 DATA BYTES
TEST DURATION 1000 PASSES	11 SEC	21 SEC	22 SEC	63 SEC
TOTAL BYTES TRANSFERRED	128,000	168,000	528,000	8,128,000
ADAPTER PRIOR TO RETROFIT				
BUSY COUNT	10,000	22,182	22,000	28,065
BUSY TIME	0.0842 SEC	0.1672 SEC	0.2233 SEC	1.4808 SEC
TRANSFER RATE*	12.2	8.0	18.9	43.5
ADAPTER AFTER RETROFIT				
BUSY COUNT	10,000	27,325	26,616	32,167
BUSY TIME	0.1081 SEC	0.2758 SEC	0.3294 SEC	1.6249 SEC
TRANSFER RATE*	9.5	4.9	12.8	39.9
DELTA CHANGE*	2.7	3.1	6.1	3.6
PERCENT CHANGE	-22	-39	-32	-8

* MEGABITS PER SEC

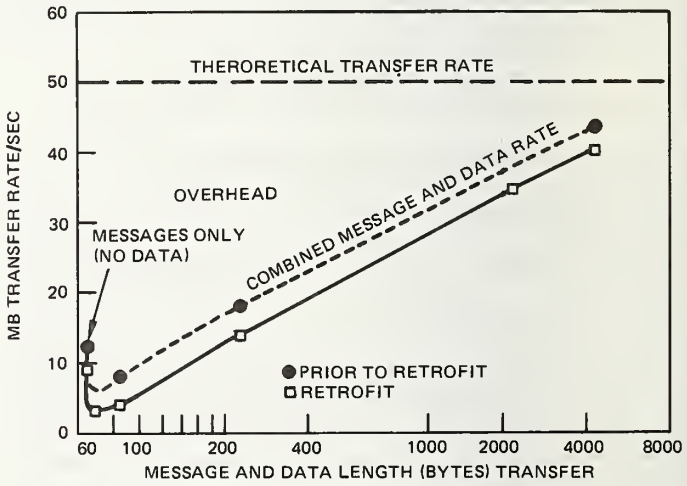


Figure 6. Effect of Adapter Retrofit on Trunk Transfer Rate

Though important in establishing a performance baseline and valuable in evaluating NSC Network Adapter and trunk throughput capabilities, these preliminary measurements were only preparatory to measuring network performance with JANET's full protocol installed.

6. JANET PERFORMANCE EVALUATION

Evaluating JANET's unique software system, with its layered protocol proceeded in parallel with integration and test, using many of the techniques evolved in evaluating network adapter

and trunk performance. As the understanding of network behavior increased, the performance investigation broadened from rudimentary end-to-end transfer timing and network adapter trunk transfer capability to include the effects of peripheral channel transfer rates.

Augmented measurements included network interface devices such as the DR11B. The DR11B interfaces with DEC's UNIBUS, acts as the DMA (direct memory access) device, which controls and shuttles data to and from the network adapter. Some measurement experiments were designed specifically for diagnostic purposes, such as monitoring attention interrupts which signal adapter interface processing. Other measurement experiments combined NSC's network adapter load parameters with signals obtained on the UNIBUS backplane to evaluate potential system bottlenecks and determine the relative activity contributions to network loading emanating from each host. Table 5 shows parametric measurements resulting from a variation in data block size transferred, using full network protocols.

Standard linear regression techniques were applied using end-to-end timing measurements shown in Table 5, to calculate the nominal protocol overhead,

P_o = .02 sec

and effective bus transfer rate,

R_b = 437 megabytes per second.

Inserting these values into the mathematical model gives the network end-to-end network timing delay,

$$T_n = \frac{L_d}{437} + .02$$

in seconds as a function of length of data transferred in kilobytes. A 0.996 correlation coefficient indicates a good linear relationship between the measurements and the mathematical model. The least square calculated network transfer rate of 437 megabytes per second also compares favorably with the maximum data transfer rate measurement of 440 megabytes per second contained in Table 5.

Hardware measurements also resolved performance concerns related to the following:

1) UNIBUS Transfer Rate vs Network Transfer Rate

2) Relative component loading of the network

3) Network trunk saturation levels

Table 5. Full Network Protocol Performance Measurements Transferring Data Between Two PDP 11/70s.

BLOCK SIZE TRANSFERRED, BYTES	2	198	1998	3996
MEASURED PARAMETERS				
1) TEST TIME, SEC	39.30	42.08	50.07	57.75
2) END-TO-END TIME, SEC	0.020	0.021	0.025	0.029
3) ADAPTER TRUNK BUSY, SEC	0.484	0.544	1.124	1.846
4) UNIBUS BUSY, SEC	0.810	1.171	4.510	8.208
5) ADAPTER TRUNK BUSY, %	1.23	1.32	2.24	3.19
6) BUS BUSY, %	2.06	2.85	9.00	14.21
7) BUS/ADAPTER RATIO	1.67	2.15	4.01	4.45
8) KILO BYTES TRANSFERRED	132	520	4,124	8,120
9) DATA TRANSFER TIME, SEC	–	1.78	10.77	18.45
10) DATA TRANSFER RATE, KILO BYTES PER SEC	–	292	383	440

The UNIBUS transfer rate was obtained from direct software and hardware measurements. For example, in transferring 8.12 x 10^6 bytes of information between two PDP 11/70 processors the UNIBUS was busy 8.2 seconds (see Table 5) for an average transfer rate of 0.99 x 10^6 bytes per second. However, the UNIBUS transfer rate is sensitive to many contending mainframe processing factors and varies considerably.

Other interesting aspects to the data presented in Table 5 and graphically as a log plot in Figure 7, give the relationship between UNIBUS busy and adapter trunk busy to measure the relative component loading of the network. The loading factors applied in a multitasked environment suggest the components most likely to cause constriction and the load level at which to expect contention or saturation. Note that UNIBUS busy (loading) exceeds adapter trunk busy for the entire transfer spectrum tested and the UNIBUS/adapter ratio increases as the block size increases indicating that under synchronous circumstances UNIBUS saturation occurs before any significant trunk contention.

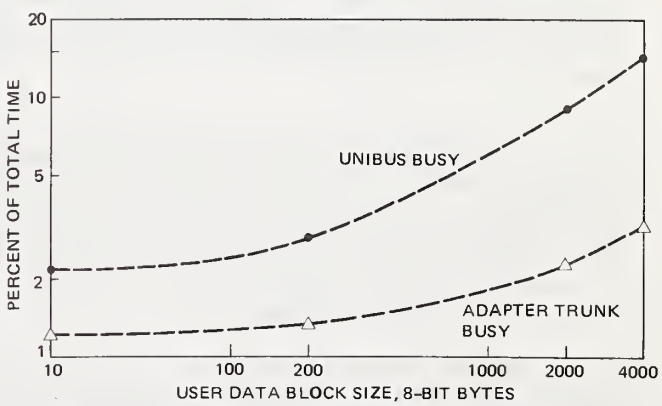


Figure 7. Comparison of Unibus and Adapter Loading as a Function of Block Size Transferred

After examining many component factors contributing to network performance, one would still want to know what measurements indicate the vitality and health of an operational network. Certainly, there is no single answer to this question, but based on the research and evaluations done during network development, I conclude that performance parameters selected should have these attributes:

1) Sensitive to network activity

2) Common to all hosts

3) Simple to monitor

Based on these considerations, operational performance of a network system is best viewed from the adapter host interface. Choosing network performance parameters within the adapters is not readily apparent but, after careful consideration, two stand out among the rest. The first parameter, "adapter busy", gives an immediate indication of the level of network activity attributable to a host node. The second parameter, "device request", counts data transferred, producing the value network traffic per unit time and, combined with the first parameter, "busy", yields the network transfer rate.

To date, we have not performed any network wide performance measurements combining more than a single network adapter. However, I anticipate that in the near future we will apply a combination of software and hardware measurement techniques for monitoring multiple network adapters to evaluate the operational performance of the JANET system.

SUMMARY AND CONCLUSIONS

This paper has described the development of a highly sophisticated JANET network system and how methods and procedures were devised to measure and evaluate system and component performance, in parallel with development. The discussion traced the evolution of testing from strictly software instrumentation to multifaceted hardware measurement. Performance measurements progressed from specially designed network driver software to evaluating the fully configured JANET system.

Installation of software and hardware monitoring techniques were delineated for measuring total performance and evaluating primary network components, such as NSC's network adapters, and DEC's UNIBUS and DR11B device interface, to identify their relative performance capabilities and to highlight conditions most likely to cause contention and system bottlenecks.

The evaluation indicated correspondence between test measurements and a linear mathematical model which defined network timing delay in terms of protocol overhead, data length transferred, and bus transfer rates. Direct, hardware monitoring experiments measured actual network adapter trunk transfer rates which, when compared with vendor specification, determined the scale of adapter firmware overhead as a function of user data size transferred.

Measurements established a network performance baseline to evaluate the effects of the NSC adapter retrofit program and provided justification for software enhancements to improve system performance.

Recommendations were offered for monitoring selected and significant system performance parameters common to all operational nodes which conveniently evaluate and control high speed network performance.

ACKNOWLEDGEMENTS

The author wishes to acknowledge the contributions of Paul Valrie and Dave Schreima who patiently devised, instrumented, and executed test software to obtain a multitude of timing measurements. Thanks also to Karen Stier and Francine Schwartz for their assistance in collecting and reducing the measurements into a distinguishable form for presentation.

REFERENCES

1) Murphy, J. L., "Centralized Control and Monitoring of a Distributed Local Network", Proc. Sixth Conference on Local Networks, Oct. 1981, pp. 93-99.

2) Shoch, J. R. and J. A. Hupp, "Measured Performance of an Ethernet Local Area Network", CACM, Vol. 23, No. 12, Dec. 1980.

3) Almes, Guy T. and Edward D. Lazowska, "The Behavior of an Ethernet-like Protocol", Proc. Seventh Symposium on Operating System Principles, December 1979, pp. 66-81.

4) Franta, W. R. and J. R. Heath, "Performance of HYPERchannel Networks: Parameters, Measurements, Models and Analysis", Technical Report 82-3, Jan. 1982.

5) Herskovitz, J., "Network Protocol Timing", Proc. Sixth Conference on Local Networks, Oct. 1981, pp. 73-81.

6) Christensen, G. E., and W. R. Franta, "Design and Analysis of the Access Protocols for HYPERchannel Networks", 3rd U.S.A.-Japan Computer Conference 1981, pp. 86-93.

CPEUG82

"Improving Organizational Productivity"

Benchmarking and Remote Terminal Emulation

SESSION OVERVIEW

BENCHMARKING & REMOTE TERMINAL EMULATION

Dr. Bernard Domanski

The College of Staten Island
Staten Island, N.Y. 10301

Historically, remote terminal emulation has had broad application; benchmarking, capacity management, and in procurement activities. Four new RTE and RTE-related tools are presented, with particular attention paid to their respective designs, uses, and advantages. Three of these are true minicomputer based RTE's, while one is a load driver designed to exercise a large, teleprocessing database management system. The key issue addressed is "human engineering"; that is, by making tools easy to use, the accuracy of the results and the productivity of the tools' users increase significantly.

327X EMULATOR PACKAGE FOR SYSTEM RESPONSE TIME EVALUATION

Mary Christ

International Business Machines
Kingston, N. Y. 12401

The purpose of this paper is to introduce the 327X emulation package as a unique, state-of-the-art performance evaluation tool. The main objective of this package is to measure the system availability that a typical end user experiences. This monitoring tool was developed to address the many problems encountered when attempting to accurately measure, and in turn provide data that adequately reflect actual user response time. The emulator package focuses on this problem by using a Series/1 to simulate a remote 327X user. In simulation, the emulator issues commands and calculates the transaction turnaround time. With it's ability to measure response in a controlled environment, utilizing sidestream processing, this IBM internal use only response monitor provides an optimal solution to the problems plaguing 'pre-emulator' monitors.

Keywords: Accurate data; end user; host independent; monitor; network; performance; remote; response time; Series/1; sidestreaming; simulated commands; 327X emulator.

1. Introduction

The 327X emulator package (327X refers to any one of the 3270 series IBM terminals) was originally developed to address the problems of accurately measuring online response times. As its development progressed, it became quite apparent that the tool not only met the challenges of response time monitoring, but it also provided additional measurement capabilities. This paper describes the overall objectives that the developers attempted to achieve while creating the package. An actual case study will be used in order to provide a clearer and more conceptual understanding of the necessary software and hardware requirements for this performance analyzer. Also included are post-processing techniques and several future considerations.

2. Overall Objectives of the Monitor

The main objective of the 327X emulator is to measure the system availability that a typical user experiences. In general, the performance data logged on most systems is not indicative of actual user response time. This monitoring package uses a Series/1 to simulate a remote 327X user. The Series/1 appears to the host as a regular user issuing various commands. The Series/1 calculates actual user response time by measuring the length of time it takes the host to respond to a particular transaction (e.g. LISTC command.) The monitor also addresses other problems that previous reporting tools have been unable to resolve. One main difficulty is that different users groups on the same online system often experience totally opposite views of system response time. Generally, the reason for this is because each user group requires different resources. Another point that

should be considered is that response time is affected by the quality rather than the quantity of the functions simultaneously occurring in the system. In both instances the 327X emulator would be a very helpful tool, since it functions in a controlled measuring environment. It is controlled in the sense that specific commands can be executed to reflect the usage of the system by a particular group. This ability to simulate actual online sessions allows for the reenactment of system resource usage, thereby producing meaningful performance data.

The monitoring package also uniquely focuses on the trends towards systems management. Typically, systems management is approached with a mainstream philosophy. That is, management is performed on the host with tools such as: Network Problem Determination Aid (NPDA), Network Performance Analyzer (NPA) and Network Communication Control Facility (NCCF). Now, however, the emulator package allows for sidestreaming management. Sidestreaming, which often compliments host dependent tools, is an improvement over mainstreaming for the following reasons:

- It increases process control.
- The 327X emulator (Series/1) is not part of the failing host.
- It is host release independent.
- It is simplistic and inexpensive.

Note that a Series/1 seems to be the most adaptable distributed processor to meet the preceding requirements.

3. Functional Description

The 327X performance analyzer can be used to monitor the response time for any number of applications. (e.g. A certain site has one monitor each for TSO, IMS, and CICS.) Each monitor sends transactions that are relative to its specific application. The Series/1 accomplishes this by using a script data set that contains multiple pairs of lines. The first line in each pair is the actual transaction to be sent. The second line is the expected response image. The monitor logs the actual response time, the transaction sent, and a rating of either GOOD, MARGINAL, INADEQUATE, or NO RESPONSE (in accordance with the preset level for a particular transaction), in a log data set at the Series/1. It also prints the current log information on an IBM 4974 printer for immediate access hardcopy output. The log file provides a means for detecting trends or changes in the measurement data. The package also provides the capability to format different IBM 497X terminals as operator information consoles. These consoles can display the current transaction being sent, the response received, the rating of the response time and a comparative analysis of previously defined response thresholds. Also, the terminals can act as early warning message centers for the operators, by providing information about slowdowns, bottlenecks and shutdowns in the system. The emulator software can easily be modified to allow the monitor to be almost entirely automatic. The monitor can be automated to perform the following tasks:

- Logging on/off the host at any specified time of day.
- Logging back on the host after an IPL.
- Loading in all of the necessary programs for the monitor into the Series/1 memory.
- Starting up the necessary lines and programs for Series/1 host communication.
- Switching between data sets to send different transactions for distinct user groups.

Perhaps the most significant aspect o the performance monitor is the fact that the environment can be controlled. The core of the package is the data set containing the transactions to be sent and the expected responses to be received. A system center can arrange service level agreements with its users, and then proceed to use the monitor to analyze the agreements for performance. A site can be even more specific in its agreements, by including EXACT transactions and their projected (GOOD, MARGINAL, INADEQUATE) response times.

Example #1

A group of TSO users agree on these levels of response time for the ALLOCATE command:

GOOD < 2 sec.
MARGINAL >= 2 sec. & < 5 sec.
INADEQUATE >= 5 sec.

Example #2 (Actual Case)

At the case study site, the monitoring environment is controlled in a rather unique manner. In addition to the regular TSO commands that are issued and recorded, CLISTS using a specific amount of service units are executed to provide

greater control. The amount of service units (su) to be used by a TRIVIAL, MODERATE and/or COMPLEX command were determined to be as follows:

TRIVAL < = 200 su
MODERATE > 200 su & < = 1400 su
COMPLEX >1400 su

(Three clists are executed - one for each level of complexity.)

(See figure 2 for a sample report of clist transactions.)

The data collected can be processed at the Series/1 or it can be sent back up to the host via a Remote Job Entry line. The Series/1 includes RJE capabilities in the basic software package. The actual methods for post-processing the data depend upon the available resources and the needs of a particular site. For the purposes of this report, the post-processing methods used at the case study site will be examined.

The case study site configuration serves as an excellent example of the typical hardware requirements for a monitor of this type. The site monitors prime shift (0800-1700) TSO user response time on a 3033 MVS system. The Series/1 automatically logs itself off of the host at 1700. At this time, an operator switches the Series/1 from a bisynchronous line to an RJE line. The data is then sent to a 3032 host by means of batch job submitted on the RJE line. See Figure 1 for the sample hardware configuration.

4. Database Usage for Problem Reporting

The collection of end user response time data is of course the purpose of this monitor. Specifications for storage and retrieval of the data is site dependent. The case study monitoring system has a very effective post-processing and database(update/storage/retrieval) schema. An overview of the case study's performance evaluation database follows.

The sample site submits a batch job on a daily basis from the Series/1 to the host via an RJE line. This job adds the daily data into a weekly generation data set. At the end of the week, a job is run that incorporates the weekly data into an online (Statistical Analysis System) SAS database. [2] This database can be queried both online and in batch, with the use of easily written SAS programs. One of the outstanding points of this database is that it is user friendly. Any analyst, manager or user can request statistics about specific transactions for any period of time. SAS allows one to plot, graph and list the information. Another feature that the case study queriers utilize is color graphic capabilities. The site has several 'canned' SAS programs that perform superb color graphic plotting and charting. (The querier only has to specify transactions and periods for the statistics.) (See figure 2 and figure 3 for generated report samples.)

5. Summary

The 327X emulation package is an incomparable measurement tool insofar as representing actual user sessions. The ramifications of this tool are quite significant. It is possible to simulate the exact transactions that a remote user might execute in a typical session. Until recently, the emulator software was designed to handle only remote terminals because bisync protocol was used. Now however, a package for SDLC local terminal performance analysis is currently being developed. Basically, the same theories will be applied in the creation of the software and hardware requirements. However, due to the nature of the hardware involved with local terminals, this performance analyzer will be more difficult to implement. Problem determination can be simplified merely by examining the data logged by the monitor. For instance, one could determine transactions that cause bottlenecks and optimal response periods. Perhaps the most important information that can be gleaned from the data, is whether service level agreements are being met. The 327X emulator data can also be used in conjunction with other performance data such as SMF and RMF. In fact, a performance database could be created that incorporates all the data collected on a system(s). This database could then provide a common, analytical basis for retrieval and reporting of any performance data element. The case study site uses the MVS Integrated Control System by Morino and Associates for this purpose.[3] Since every site has different situations that would require this type of monitor, it is difficult to project usage. However, it seems reasonable to assume that many large system sites could benefit from this performance monitor. Actually, any site that has multiple applications would find the package beneficial. The future of this type of measurement is promising - the user wants statistics that reflect the response time that he experiences!

References

This paper was based on software development work done by Michael Lentz. The performance analyzer package has been upgraded many times in order to resolve inadequacies discovered by various users. Also, this IBM internal use only package has incorporated several valuable suggestions as specified by other performance interested parties. For these reasons, the package can be considered to be a 'melting pot' of input and information from numerous groups.

[1] Lentz, Michael. IBM Burlington GTD.

[2] SAS Institute Inc. SAS User's Guide, 1979.

[3] Morino Associates, Inc. MVS Integrate Control System. MICS User's Guide, 1979.

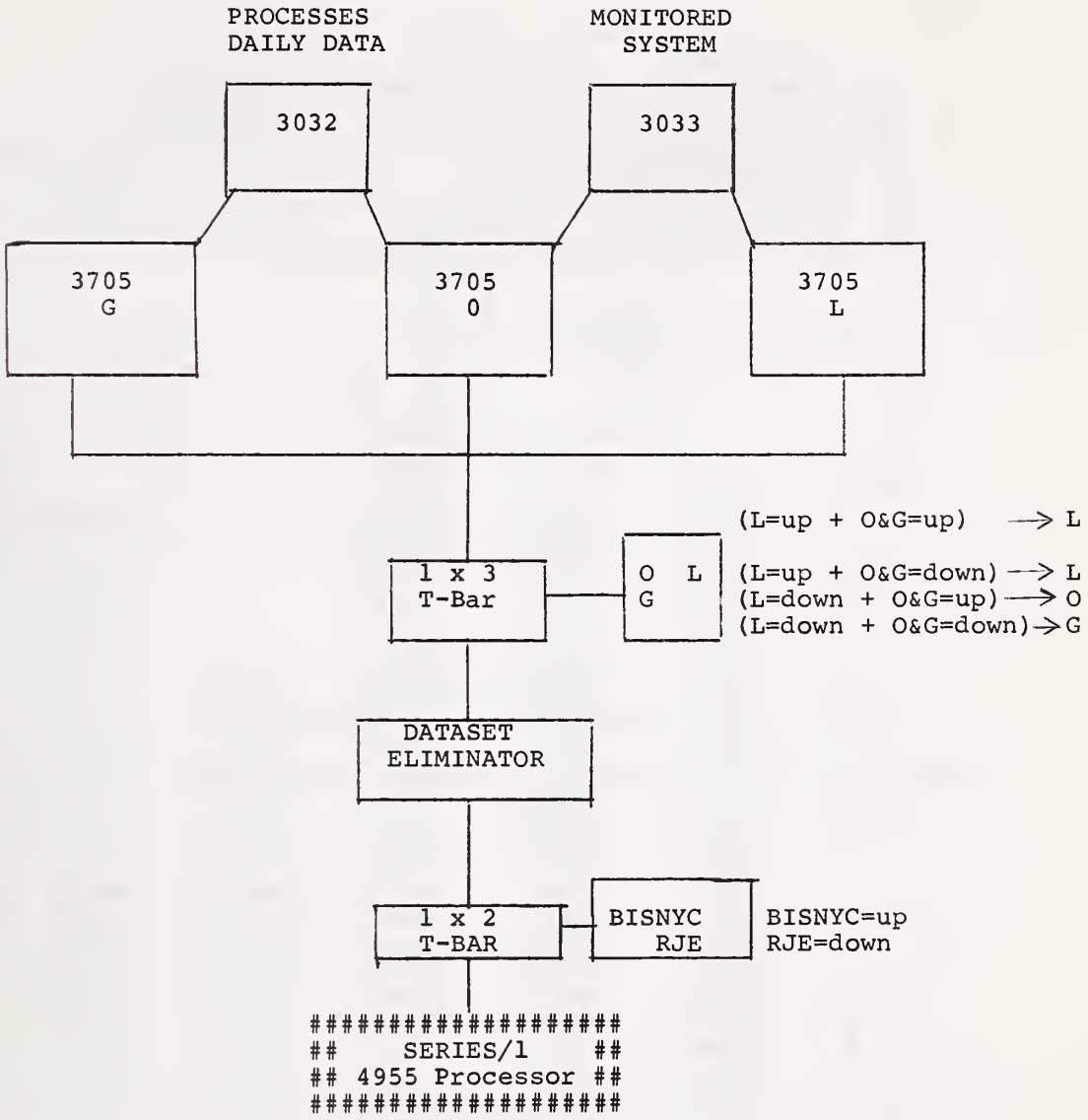


Figure 1. Emulator Hardware Configuration Using a Series/1

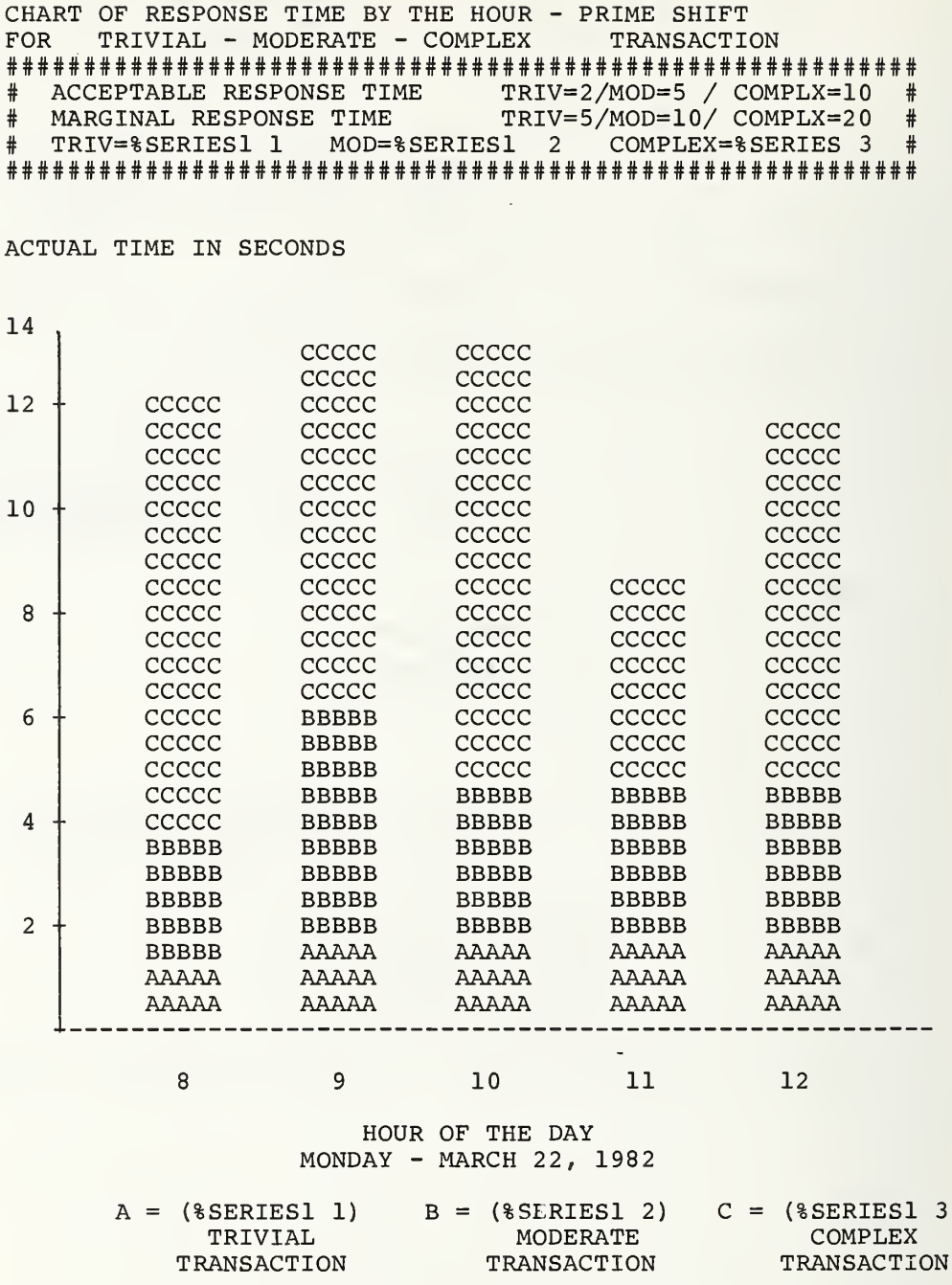


Figure 2. Sample Chart of Hourly Response Time Evaluation

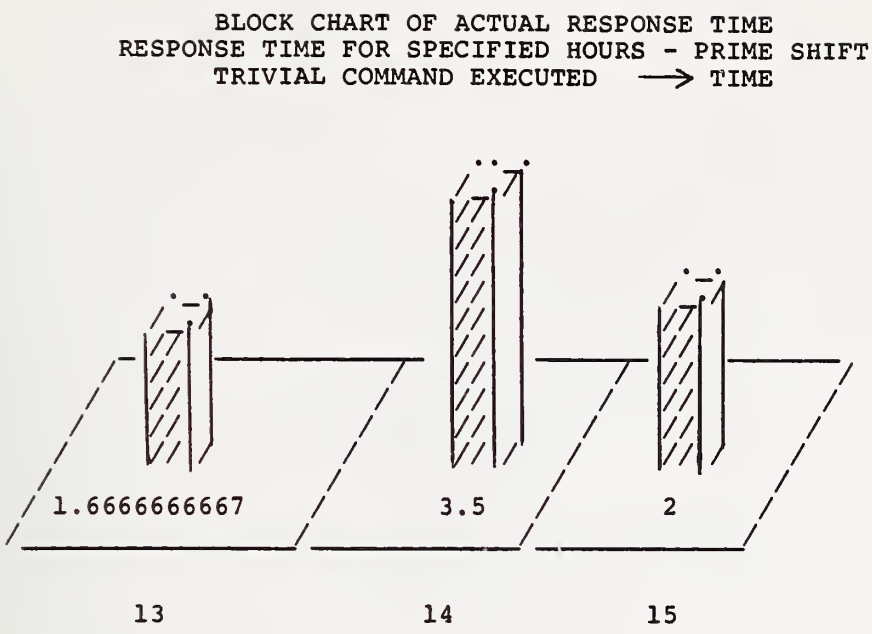


Figure 3. Sample Block Chart of Hourly Response Time Evaluation

THE DESIGN AND APPLICATION OF A REMOTE TERMINAL EMULATOR

Michael Proppe

Computer Sciences Corporation
Systems Division
Falls Church, Virginia 22046

Barry Wallack

Command and Control Technical Center
The Pentagon
Washington, DC 20301

This paper discusses the design and application of a Remote Terminal Emulator (RTE) developed by Computer Sciences Corporation (CSC) for the Command and Control Technical Center (CCTC). This RTE will be used in the Worldwide Military Command and Control (WWMCCS) community for testing network component performance and new versions of software prior to release. The topics covered include the functional capabilities of the RTE, the significance of these capabilities in achieving design goals, and the use of the RTE as a performance evaluation tool.

Key Words: Interactive System; Performance Evaluation; Remote Terminal Emulation; Remote Terminal Emulator; System Under Test.

1. Introduction

The RTE discussed in this paper is an interactive performance evaluation tool designed for a variety of test environments. It has been implemented in FORTRAN on a Data General Eclipse S/140 minicomputer with 128KB of memory and 10 communication lines. The protocols supported include full- and half-duplex asynchronous teletype, the VIP synchronous protocol used with Honeywell's visual information projection stations, and the synchronous Remote Line Printer (RLP) protocol that is used in the WWMCCS environment. The VIP synchronous lines operate in a multidrop mode, each line supporting a maximum of 32 devices. The same RTE design is also being implemented by CSC on a Honeywell Level VI system. Previous RTEs have been implemented for the Navy on a Data General MP200 microcomputer with 64KB of memory. These RTEs support 15 asynchronous communication lines or 2 BISYNC lines, and are packed in a suitcase for field testing.

This paper discusses the types of environments in which remote terminal emulation is applicable and functional capabilities included in the RTE design to provide a flexible evaluation tool.

2. Background

In evaluating an interactive system, the bulk of the work is defining exactly what is to be tested and designing a scenario to reflect the desired workload. The objectives of an emulation test must be defined clearly to interpret properly the resulting data. Whether in the experimental or procurement environments, tests usually fall into one of the following categories: stress test and breakpoint analysis, configuration performance tests, benchmark performance tests, and functional and operational performance tests.

Stress tests and breakpoint analyses measure system response to heavy load conditions. Configuration performance tests compare the performance of one system configuration versus alternate configurations. Benchmark performance tests compare the performance of different systems, usually with similar hardware configurations and scenarios, to determine whether one is more suitable for a given application. The functional and operational performance tests determine whether the System Under Test (SUT) can satisfactorily perform a set of functions and operations, under expected load conditions,

in an evolving hardware or software environment. Tuning systems, optimizing SUT software, and testing new releases of software also fall into this category.

RTEs have been in existence for some time and have been designed to emulate these types of tests. Although their usefulness to the Automatic Data Processing (ADP) community has been limited by their specialized nature and their high development and production costs, remote terminal emulation continues to evolve as the most preferred method for tests that require large volumes of terminal traffic. An RTE is a driver that is external to the SUT. It connects to the SUT through its communication device interfaces, either locally or through a network. The RTE then interacts with the SUT as if it were a set of terminal devices or operators. As an external driver, an RTE can alter the workload placed on the SUT to measure performance and to approximate a given operating environment. Remote terminal emulation provides the advantage of a controlled, repeatable test environment and the means to assess the performance of various systems and configurations. This offers a powerful tool for evaluating telecommunication applications because the user can determine performance before investing in additional equipment or systems programming, and it eliminates the cost and inaccuracies of terminal operators during live tests. Some past emulators have been developed for a specific SUT, limiting their use to tests with a particular vendor's system. Others require modification of SUT software to send prompt characters to the RTE each time a user request has been processed or does not provide verification of SUT responses (1). The RTE discussed here has been developed to incorporate as many features as possible into a small, cost-effective RTE system.

3. The RTE Design

Our design goal was to produce a flexible system that is easy to use and applicable to a broad range of test types. Studies of past RTEs and General Services Administration (GSA) specifications (2) provided a baseline definition of the types of functions expected in an emulation. Our approach was to perform as much processing as possible either before or after the emulation session to minimize the amount of processing performed by the RTE task while connected to the SUT. This effectively separated the emulation session into three phases: the pre-emulation phase, the emulation phase, and the post-emulation phase. Figure 1 depicts the relationship between these three phases. During the pre-emulation phase, a SUT-independent scenario is defined, the scenario is translated into a SUT-specific dialog or script, the script is prepared for RTE execution, and the RTE is configured for the emulation session. In the emulation phase, the RTE interacts with the SUT in an online mode, using the information provided

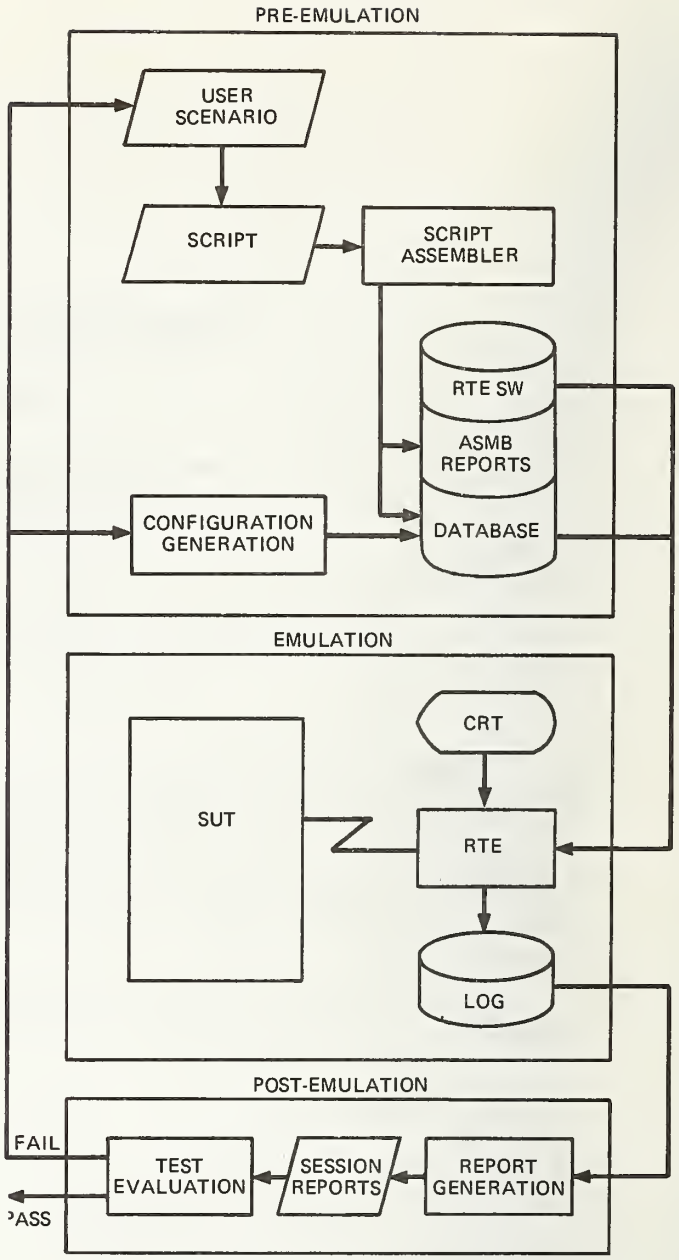


Figure 1. Emulation Phases

in the user scripts, and records session data and statistics. During the post-emulation phase, a series of reports may be generated using the information gathered by the RTE during the emulation session. These reports provide statistics on the emulation session and a chronological log of events. This phased approach uses offline utility programs on the RTE system to perform the bulk of the processing associated with preparing and reporting data involved in an emulation. The following paragraphs discuss how each phase is addressed in the RTE design.

3.1 The Pre-emulation Phase

The cornerstone of an emulation session is the user scenario. The scenario is defined in a

high-level description, usually an English-language format, and describes the types of actions and functions to be performed by the RTE. It includes data on user think times, transaction delivery distributions, maximum response times, line protocols, and line speeds. The system-independent scenario transactions must be translated into the SUT-specific stimuli and responses necessary to achieve the desired result. (A stimulus is the text to be sent to the SUT by the RTE, and a response is the message originated by the SUT.) Communication protocol messages are not included. These stimulus and response transactions are then coded into a script that the RTE will use during the emulation session. Since the majority of the user's work involves designing and coding scripts to be used by the RTE, it was important that the method selected to code scripts be flexible and easy to use. This resulted in the specification of the Script Development Language (SDL).

The SDL provides a high-level method for defining what the RTE is to do during the emulation session. Each transaction is described in a set of script command lines. There is the "S" command line to define the stimulus text to be sent to the SUT, the "L" command line to define how many characters are expected in the response (protocol characters are not included), and the "C" command line to define contingency actions the RTE is to take in the event of protocol errors or expected response mismatches. The contingency actions allow the RTE to ignore the error, resubmit the transaction up to "N" times, where "N" is specified by the user, abort testing over the line in error, or jump to another portion of the script. Command lines that are optional in defining a transaction include a "T" command line to define the maximum amount of time the RTE will wait for a SUT response, a "W" command line to define the user think time, which is the amount of time the RTE will wait between the receipt of the response for the current transaction and the transmission of the next stimulus, and an "R" command line to specify response text that will be compared with the actual SUT response. SUT responses are verified selectively by including the "R" command line; without it, no response verification takes place. To verify a SUT response, the user specifies a maximum of 71 characters of the response in the "R" command line, and the position in the response where the character string begins. During the emulation session, the RTE looks for the specified text in the SUT response and returns a "match" or "mismatch" status on the verification. The user may select a portion of the expected response--a key phrase particular to that response--to be verified. If this key phrase is found in the SUT response at the location specified by the user, and if the received character count matches the expected count, the RTE considers the message validated. There are also command lines to declare default think and wait times to be used when none are defined explicitly within a transaction. These default values may be changed later in the script by redefining them in subsequent default declarations. Other features include comment lines that may be inserted anywhere in the script, octal and hexadecimal number representations (for special characters not found on a standard keyboard), and the optional generation of file name extensions. The file name extensions allow the RTE to use the same script over multiple lines, yet avoid contention when multiple users try to access the same file on the SUT. The RTE will automatically generate the extension, in the range of "AA" to "ZZ", which is associated with each emulated user and append it to the file name prior to transmission. As a result, each emulated user has a unique SUT file and avoids access collisions. An example of two transactions that logon and logoff a system using basic script command lines would be coded as follows:

```
S 0001 /LOGON:  SYSTEM-82/
L 0032
R 0002 /SYSTEM READY/
C      /ABORT/

S 0002 /LOGOFF/
L 0029
R 0004 /TIME IS:/
C 0003 /RESUBMIT/
```

In this example, text strings are enclosed within the "/" delimiters. The numbers on the "S" command lines represent stimulus sequence numbers used to order the script. The expected response length for the first transaction is 32 characters, and that of the second transaction is 29 characters. Portions of the SUT response are verified in each transaction, as noted in the "R" command lines. In the first transaction, the RTE will look for the text string "SYSTEM READY" beginning at position 2 of the SUT response. The second transaction verifies that the text string "TIME IS:" is found in the SUT response beginning at the fourth character position. If an error occurs during the emulation, the contingency actions coded in the script are executed by the RTE. In this example, if a protocol error or verification error occurs on the first transaction, the RTE aborts testing over the line in error. The number in the "C" command line for the resubmit specifies how many times the transaction will be retried before shutting down the line in error.

Scripts are entered using the system editor. The SDL-coded scripts are then assembled using the Script Assembler utility. The Script Assembler makes two passes through the user's script. The first pass parses the script, reports any syntax errors, and optionally generates an assembly report detailing each transaction. The second pass is specifically designed to minimize the calculations required by the RTE during real-time interaction with the SUT, and the storage space for the script, by compressing the script information and calculating the pointers and offsets used by the RTE in

accessing data within the database. The result of the script assembly is a disk-resident database file that is loaded by the RTE at runtime.

Once the scripts have been assembled, a utility is used to configure the communication lines. The configuration utility uses editor-like commands to add or delete lines, change or examine line characteristics, and list the configuration. Abort and exit commands terminate the configuration utility, and a help command summarizes valid commands and entry rules. This allows the user to specify which lines will be active during the emulation, their protocol type and line speeds, poll and select addresses if necessary, and the script that will be executed on each line. This information is then stored in the database with the assembled script.

3.2 The Emulation Phase

The emulation phase constitutes the real-time execution of the script by the RTE. Upon startup, the RTE prompts the user for the name of his database file and establishes connections to the SUT over the designated lines. Once connected, the emulation of users begins without the need for operator intervention. The RTE automatically reports error conditions and progress information to the user and maintains a condensed chronological log detailing the events on all active lines. The event information includes stimulus transmissions, response reception and verification status, response time data, script iteration completions, errors, and executed contingency actions. In addition, the user can query the RTE for statistics and status information while the RTE is executing, turn the logging feature on or off, or halt the session.

The RTE software executes in a multitasked environment with the main RTE task managing the database, scheduling transactions, verifying responses, and logging event data. This task is not concerned with the protocol type being used on a given line; auxiliary tasks are used to support each type of protocol that is active during the emulation. These communication tasks manage the interface with the SUT, send the stimulus text (characters sent by an emulated device are transmitted with no intercharacter delay time), and read in the SUT response, which is then passed to the RTE task through a shared memory area. Each communication task is allocated a segment of this area to maintain its Transaction State Table (TST) and communication buffers. The TST contains the current state of the executing transaction on each line. The states include idle, sending data, waiting for a response, data received, error condition, or done executing the script. The RTE controls the communication tasks by manipulating their TSTs and providing data to be sent to the SUT.

A separate task provides the user interface to the RTE. The interface task executes at a low priority and accesses data structures used by the RTE and communication tasks to report execution statistics to the user. This allows the user to examine the number of characters or transactions processed, halt the session, turn logging on or off, and display error and iteration counts, line states, or the position of each line within its script.

3.3 The Post-Emulation Phase

In the post-emulation phase, reports are generated using the information gathered by the RTE. The condensed log is expanded into a formatted report using a log expansion utility. The user may expand the entire log or select a time window within the log. The log report includes a line-by-line description of all actions taken by the RTE during the emulation session and their associated execution time. The statistical data is processed and formatted using a report generation utility. The statistical reports provide data such as the average, minimum, maximum, and standard deviation of response times for each transaction. Additionally, the reports include actual-versus-expected response accuracy, character throughput rates, error counts, and response time frequency distributions.

3.4 Design Considerations

The RTE has been designed to be applicable to a broad range of tests. Some of the features implemented respond to general requirements of an RTE, such as logging, response verification, masking of fields within SUT responses, consistent transaction delivery, test repeatability, and a user interface to the RTE to verify correct operation. Other features enhanced the operation of the RTE: the file name extension for scripts that can be executed over multiple lines, iteration specifiers for scripts that may be executed iteratively to achieve a steady-state load on the SUT, the ability to specify null responses or stimuli, automatic error limit specifiers to shut down a line if a specified error threshold is exceeded, and contingency actions to allow the RTE to recover from error conditions. Certain RTE features were included for stress tests and breakpoint analysis applications. In such types of tests, throughput is the overriding concern. To deliver the performance required during stress tests, the RTE code and run-time database are entirely memory-resident during execution. This eliminates any unnecessary disk I/O due to program swapping, overlay usage, and data retrieval. Also, it limits the disk accesses to those for the blocked writes to the log. The time-consuming task of verifying every character in a SUT response was reduced by supplying selective response text verification and comparing the received character count with the character count specified by the user. The processing required for the RTE to generate transaction delivery distributions is offloaded by allowing the user to specify the distribution using transaction think time command lines within the

script. This also removes limitations on the number and types of distributions that may be designed into the script.

A larger design consideration in the RTE was expandability. Although the current RTE supports 10 communication lines, it was felt that the capability must be present for easily expanding both the number of lines and the number of protocols that are supported. The modular approach used in the RTE design facilitates such enhancements. Additional protocol types may be added to the RTE system by including the appropriate task to manage the communication interface and expanding the shared memory area to include table and buffer space for the added task. The number of lines supported may be increased in much the same manner.

4. Conclusions

During its development, the RTE was tested with a variety of SUTs to verify its operation and ease of use with different systems. These systems included Tandem 16, PDP 11/70, Datanet 355, VAX 11/780, Data General MP200, and Data General Eclipse S/140. The test objectives were to load the RTE, as opposed to loading the SUTs, to verify that the implemented functions operated correctly during testing. In this manner, RTE tuning was performed based on the service statistics gathered during the emulation sessions. As a result of these tests, communication line service algorithms were adjusted to offer quicker service to the active lines, and portions of the RTE code were optimized to provide more efficient execution. Inclusion of the features necessary in a stress test environment has also enhanced RTE performance in the other types of test environments.

The RTE developed by CSC has demonstrated its ability to fullfill the objective of producing an emulator that is a cost-effective alternative to perform evaluations of interactive systems under a variety of test conditions and SUT types. It will be used within the WWMCCS community under varying load conditions, for performance testing of network components that are involved in the WWMCCS Intercomputer Network (WIN). RTEs will also be used to load the WIN network and provide performance data.

Enhancements are planned to provide additional communication support, system memory, and user functions, and to increase the allowable size of user scripts. One enhancement will allow the user to start or stop selected lines through the user interface. Another will provide an automatic script generator that can connect the operator on the RTE system to the SUT and generate RTE scripts based on the data exchanged during the terminal session.

Although the features implemented in the RTE are not all-inclusive of those available on other RTEs, incorporation of additional features is one of the continuing development goals.

ACKNOWLEDGMENTS

The authors would like to acknowledge the support of Dr. David Karlgaard, Vice President of CSC's Systems Architecture Activity and the contributions of his colleagues. In particular, Mr. Paul Rice and Mr. Charles Owlett of CSC; and Mr. George Gero of CCTC had key contributions to the development of the RTE.

REFERENCES

[1] Watkins, Shirley W., and Abrams, Marshall D., Computer Science and Technology: Survey of Remote Terminal Emulators, National Bureau of Standards Special Publication 500-4, April 1977.

[2] General Services Administration, Use and Specification of Remote Terminal Emulation in ADP System Acquisitions, Report FPR 1-4.11, GSA/ADTS, Washington, DC, August 1979.

DESIGN OF AN EXTERNAL TEST DRIVER FOR PERFORMANCE EVALUATION

Agu R. Ets
John H. McCabe

Analytics
7680 Old Springhouse Road
McLean, VA 22102

The procedure for benchmarking a teleprocessing-oriented ADP system is expensive and complex. Current approaches, whether manual or computer-based, are labor intensive and produce inconsistent data. An external test driver which can emulate a number of remote devices promises to reduce costs and improve the accuracy of teleprocessing benchmarking. Current test drivers are very specialized and therefore limited in general application. Generalized external test drivers can be cost-effective if their base of application is broad enough and if their test input and output processes can reduce personnel requirements. This paper presents a design for such a generalized external test driver. The design is independent of target systems and incorporates many features which support the test director. These features include scripting the test, defining test data, running the test, and analyzing the resultant data. Technically, the design is based on remote terminal emulation, with emphasis on simplified station characteristic definition, transparency to the system under test, and driver efficiency. When implemented, the design will provide externally-driven testing for a broad range of applications such as benchmarking, configuration management, software upgrade verification, stress testing, and system enhancement studies.

Keywords: External test driver; performance evaluation; remote terminal emulation; system design; teleprocessing systems; testing.

1. Introduction

Computer performance evaluation is critical to the orderly development and growth of ADP systems. Performance evaluation supports both the management and technical staff during pre-acquisition benchmarking, post-acquisition quality assurance, and ongoing configuration verification. Batch systems can be adequately tested by duplicating production runs while measuring various performance parameters. With the advent of teleprocessing, the creation of a creditable test load on the host system became a problem of distribution, coordination, and control of many resources.

1.1 Automated Techniques

The cumbersome nature of manual performance evaluation methods led to the evolution of automated techniques, which improved consistency and accuracy in the measurement of performance parameters. Furthermore, the results were automatically recorded and could be analyzed and evaluated as extensively as the user required. Unfortunately, each automated approach was oriented toward a unique set of testing objectives — a specific host, a specific configuration, specific peripherals, or specific software. The high development cost of automated techniques restricted their application to large systems, to large-scale procurements, or to the manufacturers themselves.

1.2 Remote Terminal Emulation

In August 1979, GSA published FPR 1-4.11 [1][1], a handbook on the use of remote terminal emulation in ADP system acquisitions. The handbook defined general capabilities for remote terminal emulators (RTEs) that may be used by procuring agencies to substantiate system

[1]Figures in brackets indicate the literature references at the end of this paper.

performance prior to acquisition. GSA recognized the cost-effectiveness of a generalized RTE which could reduce acquisition costs while providing greater assurances that the acquired system will perform as expected.

1.3 Generalized External Test Driver

The basic premise of generalization necessitates an external test driver which can interact with the system under test (SUT) without requiring modification of the SUT. The external driver also can provide greater latitude in the off-line support of script preparation and post-test analysis. The relative merits of internal and external test drivers were thoroughly discussed by Shirey [2].

The resources of the external driver's independent processor can be used to isolate the SUT from the processes of stimulus preparation, test computations, journaling, and analysis. To test teleprocessing systems, the external test driver would emulate remote terminals with their corresponding user activity.

The RTE concept has provided the basis for the design of a generalized external driver. This particular driver is called System for Evaluation of Computers based on Universal Remote Emulation (SECURE). SECURE is a comprehensive test driver and can be used in a range of applications, but is especially effective in testing teleprocessing systems. In order to capitalize on its cost-effectiveness, a generalized external test driver must have:

- Convenient setup procedures
- Minimal manpower and resource requirements
- Minimal unique tailoring requirements
- Consistency in data measurement

These features are the basis for the SECURE design considerations and design approach.

2. Design Approach

The SECURE design approach began by listing specific objectives which supported the features stated in Section 1.3. The design and implementation implications of each objective were evaluated. The statement of the objectives was refined to ensure that collectively these objectives satisfied the growing interest in teleprocessing testing. Based on the objectives, a specific list of requirements was compiled. These requirements then defined the design domain and served both to guide and constrain the design effort.

2.1 Design Objectives

The design of SECURE had to satisfy the following objectives:

- Ease of use
- Emulation of remote terminals
- Flexibility and expandability
- Real-time testing operation
- Automated data collection
- Comprehensive user support facilities

Ease of use is a primary objective whose importance was stressed by Shirey [2]. The SECURE must have user-oriented script preparation, user-friendly test execution and monitoring, and a general application of non-technical dialogue. A corollary objective is to minimize requirements for technical personnel (i.e., programmers) in the setup, execution, and post-test analysis.

Emulation of remote terminals is a key objective. The SECURE design must allow the characteristics of any type of terminal to be easily incorporated into the driver, and must allow the emulation to be accomplished with full transparency to the SUT. The emulation must accommodate user characteristics as well as the terminal's technical characteristics. The evaluation of this objective led to closer study of the emulation process. It was found that to stimulate a SUT and impose a realistic load, the SECURE should not exceed the rate of interaction for a normal user/terminal combination. If real-time concerns could be efficiently addressed, variances in the emulation process would not need to be weighted for processing delays. Analysis showed that the emulation variances consisted of both user and terminal characteristics. Thus, the emulation design objective became one of applying these characteristics in a digital form that could be interpreted as a time-delay element.

The SECURE must be flexible enough to drive any of the commonly used teleprocessing systems. It must be expandable to allow any number of terminals and a sufficient number of terminal types to be emulated. The number of emulated terminals is what loads and stresses the teleprocessing system. In order to produce valid performance data, the loading must be real. The design of SECURE allows modular expansion to a maximum of 256 terminals, the maximum stated in [1].

Only a real-time emulation can fully stress a teleprocessing system. At real time, SECURE must be transparent to the SUT. As real time is approached, the use of compensatory processing in emulation would be minimized. A corollary objective is the ability to monitor the test in real time and to throttle its execution. The real-time processing objectives can be achieved if I/Os can be kept ahead of emulation and SUT demands.

With real-time testing, automated data collection becomes a corollary objective. The data recorded during real-time testing is journaled for post-test analysis and evaluation. With automated collection, sufficient data can be recorded to support in-depth analysis of system performance.

SECURE is a comprehensive system in which the final objective is to provide a complete set of user support capabilities including script generation, test data preparation, and post-test analyses. Since [1] was the basis for SECURE, it will also serve as a standard for post-test analysis. This standard establishes performance measurement in terms of throughput, turnaround, and response times, which are to be measured with time stamps, computed, and entered at the appropriate processing points. The reporting defined in [1] was based on scenarios, functions, and input/output pairs. In the SECURE design, it was necessary to distill the description of these items, mesh them with the requirements of the generalized RTE, and define them in terms most appropriate to script development.

2.2 Design Requirement

The design for SECURE must:

- Allow interactive development of test scripts
- Provide a scripting language that is easily understood and used
- Permit a non-programmer to establish characteristics of each user/terminal combination
- Emulate up to 256 terminals
- Be transparent to SUT
- Allow automatic data journaling
- Permit interactive setup of analysis and extraction functions
- Support FPR 1-4.11 report formats

3. System Design

A number of terms introduced in [1] were defined and described in a generic sense and were intuitively biased toward acquisition benchmarking. The generalized nature of SECURE necessitated a different biasing and the introduction of selected new terms. For the sake of design consistency, it was necessary to develop the following definitions:

- TEST DATA — A collective term for the set of request items to be processed during emulation runs.
- ITEM — An emulation test element consisting of an output to the SUT and an input from the SUT (I/O pair). May also be referred to as request item and response item.
- INSTRUCTION — A single script directive defining a rudimentary action to be taken by SECURE during the emulation process.
- SCENARIO — At a selected terminal, a single user session from log-on through log-off.
- FUNCTION — A set of related instructions that compromises a complete operation (e.g., log on, print file, move paragraph).
- PORT — At a single workstation, a combination of terminal type and typical users having definable characteristics.

Test data are the collection of stimuli that will invoke processes in the SUT. The collection resides on a test data file (TDF) along with other data germain to test execution (e.g., special journal logs, console displays). The TDF serves as nothing more than a reference source for the script instructions which will drive the test. Each TDF entry is termed an item and can be referenced via its item identification code.

The script is developed and managed in a functionally-oriented, user-friendly manner. It is not actually classified as a script until it is transitioned to an executable form. In this form, the script consists of a series of instructions which tell test subsystem modules what actions are to be taken. Most instructions will reference TDF items on which such actions are to be performed. Scenario and function essentially refer to various levels in the hierarchical script development process. A third (highest) level is classified as test description, principally due to its role in describing the overall scope of any given test. All three are discussed in context in the paragraphs that follow.

Port is a new term that is significant to the design of SECURE. Its importance is derived jointly from a definition of the emulation process and a need to collectively address emulation factors (e.g., users, terminals, devices). The ability to define the characteristics of a port, to employ those characteristics in the execution of emulation algorithms, and to selectively evaluate performance on a port basis is the keystone in a generalized structure.

The application of SECURE to any given test consists of three distinct steps: preparation, execution, and analysis. Each step has a mode of operation, a unique interactive dialogue, and functional independence from the others. Based on these attributes, SECURE is divided into three subsystems:

- Script Development
- Test Execution
- Post-Test Analysis

The script development and post-test analysis subsystems are off-line operations. Together they interactively support the user with tools and capabilities to design, describe, and analyze a test. Test execution occurs in real time and only requires user interaction when it is necessary to augment the self-driven test. Figure 1 depicts major elements of the system organization. The design features and functions of each subsystem are described below.

3.1 Script Development Subsystem

The script development subsystem supports three functional areas required to set up tests:

1. Test data development and maintenance
2. Script preparation and maintenance
3. Port characteristics maintenance

These operations may be conducted in either an interactive or batch mode, though the design

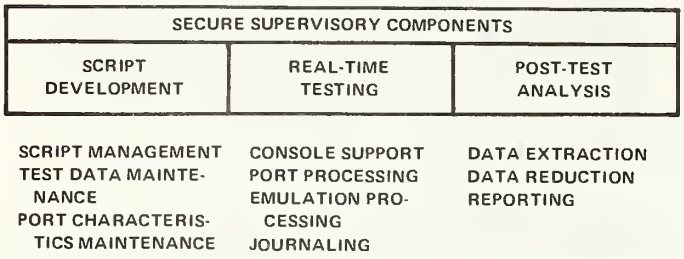


Figure 1. SECURE System Organization

details address only the interactive mode for its user-friendly attributes.

SECURE defines scripting as the combination of interactively conducting operation 1 and then invoking the semi-interactive processes of operation 2. Operation 1 supports the development of information that will be converted to an executable form by operation 2. The SECURE scripting concept is based on test items which are formed when a fundamental set of items is built into a test data file (TDF). Each item consists of a character string recognizable by the SUT and capable of invoking SUT processing. Items are defined to maximize their applicability to various functions. The degree to which planning and anticipation are involved has a direct bearing on the volume of test data that must be generated.

The next step in the scripting process is the definition of functions. These are interactively entered into a single-function file, each entry being individually maintainable. Each function is identified and defined by the user. This allows the user to employ a structure and reference mechanism that is both familiar to him and relevant to his particular test.

Tables 1a and 1b depict hypothetical TDF and function file segments, respectively. The TDF consists of a list of test data items (TEST DATA column) and their corresponding identifiers (ITEM ID column). The user structures identifiers in a form that is manageable and easily referenced. The function file entries are referenced by function ID, which is also user-structured. Each function identifies the TDF to which it refers (TDF ID column) and bears a user-defined title which aids the user in subsequent references. The remaining function composition is simply a list of operators and operands.

The samples have three functions and five items. The "01xx" series of item IDs has been reserved for log-on items, while the 09xx series is reserved for various log-offs. One must assume that each item has been wrapped in characters that are required by the SUT for recognition and processing. Function LA will cause two separate items to be sent to the SUT. Once they have been processed by the SUT, Mary

Table 1a. Sample Test Data File

TEST 01

ITEM ID	TEST DATA
0100	Smith, Mary
0110	Elephant
0120	Doe, John
0130	PASSWD1
0990	Bye

Table 1b. Sample Function File

FUNCTION ID	TDF ID	FUNCTION TITLE
LA	TEST01 SEND 0100 SEND 0110	Log on M. Smith
LB	TEST01 SEND 0120 SEND 0130	Log on J. Doe
LZ	TEST01 SEND 0990	Log off general

Smith will be logged on at a port designated by an associated scenario. If the Smith port were to include user processes, additional functions would be developed. Eventually, Smith will be logged off by the general log-off function, LZ. The entire sequence is ordered by a scenario (discussed below) addressing the Smith terminal session.

NOTE: Every operator is the instruction SEND, which means that the item (operand) is going to the SUT for processing. Other instructions (e.g., TYPE, LOG, REPEAT, WAIT) use the same basic format. The REPEAT operand is the number of repetitions, and the WAIT operand is the number of minutes the port remains in a suspended state.

After defining functions, the user is ready to assemble scenarios. The pattern is much the same as with functions. Scenarios are identified and titled in a style and vernacular that is suited to the user's line of thought. The scenario itself consists of a list of the functions comprising the session and of the identification of the port at which the scenario is to be conducted. In the hypothetical examples, the Smith scenario might appear as:

Scenario	MS
Title	Mary Smith
Port	22
Functions	LA,LZ

Test description is the last scripting step that requires the entry of data by the user. It consists of a test title and a list of scenarios in order of execution.

At some subsequent point, the user (test director) verifies each file. He indicates to the system his intent to have the actual test execution script prepared. The system allocates a token file for script retention. Preparation proceeds from the top down, as shown in figure 2. In accordance with its scenario list, the test description expands into a table of ports and functions. The table then is processed sequentially; each function is expanded into instructions; and a token file entry is made for each instruction. During test execution, the token file and TDF are merged into an executable script.

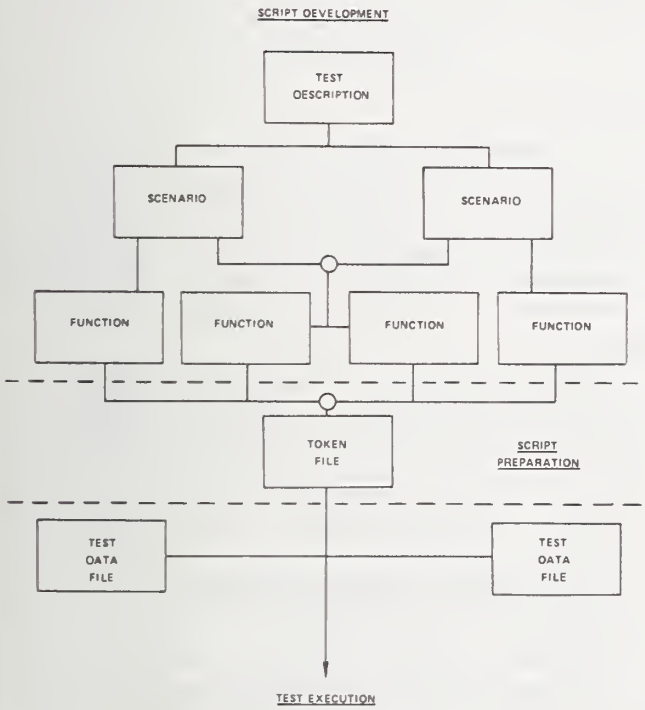


Figure 2. Script Preparation Flow

Important features of the SECURE scripting concept are that the only requirements are minimal volumes of test data and the general ability of the user to describe the entire test in a friendly, relatable manner. Test data may be referenced and employed repeatedly; functions may play a variety of roles by simply altering the TDF reference; scenarios may be applied to numerous workstations by altering port IDs; and scenarios may be executed repetitively through multiple entries in the test description.

The third operation supported by this subsystem is port characteristics maintenance. This operation is totally divorced from scripting and may be performed at any point preceding execution of the test. The operation consists of interactively entering parameters to a port characteristics file. At the outset of test execution, the file is loaded into memory, where the characteristics are accessed for emulation processing.

3.2 Real-Time Testing Subsystem

The major processes of the test execution subsystem are:

- Port Processing
- Emulation Processing
- Journaling
- Console Support

Figure 3 shows the functional relationships among these processes.

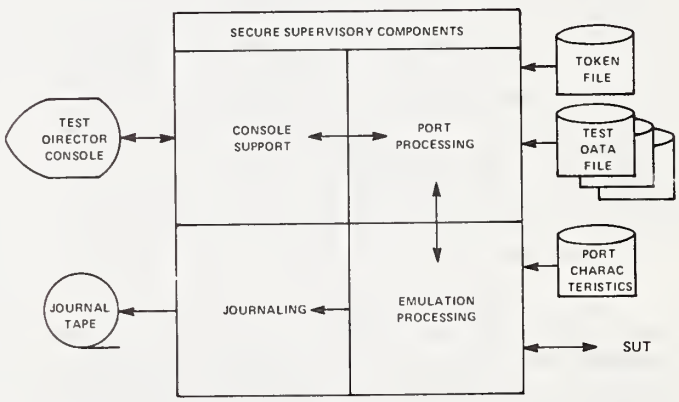


Figure 3. Testing Subsystem

Console support allows the test director to initiate the test, monitor its progress, throttle it, and suspend or terminate it. Port processing marries the token file and TDF in preparing processing packets for the SUT. Emulation processing executes emulation algorithms and controls interaction with the SUT. Journaling records all transactions initiated by the script, responses returned from the SUT, and supporting data necessary for evaluation. When a test is begun, a port status table (the heart of the stimulation process) is built, and port processing commences with port-by-port initialization. The token file (sequentially addressable within the port) is read. A referenced item is pulled from the TDF, and a script item is assembled (as described above) in a corresponding port buffer. The item remains in the port buffer until required by the emulation processor to feed the SUT. Emulation is stimulated by an empty emulation buffer and the port status table. When stimulated, port buffer contents are transferred to the emulation port and corresponding port characteristics are "pulled in" to feed the emulation process.

Characteristics include:

1. User key-in rate
2. User think time
3. Line speed of terminal in emulation (TIE)
4. Line speed of SUT terminal
5. Directional characteristics of TIE line
6. Type of characters in TIE communications
7. Port type

The emulation process is essentially a time delay, and the emulation algorithm considers only characteristics 1 through 4 and 7. All emulation computations are related to item size, and the result is a stamp for the time at which the item is due out from the emulation buffer to the SUT. Each emulation buffer contains a full set of evaluation and identification data comparable to journal record format. Only the data portion is sent to the SUT, but the entire buffer remains intact. When the SUT responds, its data overwrite the request in the same emulation buffer. As each emulation buffer is filled, its contents are transferred directly to the journal with all analysis data attached. As soon as journaling is completed, the emulation buffer is completely overlayed with the next item in the port buffer. Lengthy requests and responses are processed as a series of packets, with each packet identified relative to its position within the series. Time stamps are placed on all packets as logged, but only the first and last in a series have significance.

The journaling process handles both direct and indirect entries. The indirect entries are response emulation buffers, in standard journal format, which evolve from emulation processing. Request buffers are essentially ignored because they would contribute nothing to reporting within the analysis subsystem. Direct entries may be generated at the console (by the test director), or they may result from a log instruction imbedded in the script (token file). Both forms of direct entries are supported by a LOG module. Direct entries are supported for the purpose of annotating the journal and recording specific test milestones which the test director regards as pertinent to post-test analysis. Direct entries are made in a variable format and include time stamps and identifying data.

3.3 Post-Test Analysis Subsystem

The analysis subsystem supports review, extraction, and reduction of data collected during test execution. Subsystem processing is oriented to the production of the three reports specified in [1]:

- Scenario summary report
- Function summary report
- Interactive reponse-time summary report

SECURE has modified these reports to shift their emphasis from procurement to generalization and script organization. These modifications are most noticeable in the report headings and selection parameters. Furthermore, the application of scenarios is such that they are not logically related. Thus, scenario group reporting is treated as a report-time function in which the user identifies the scenarios in each group. Function and response-time (item) reports summarize all usage within the test or specified test period. All three reports default to the hierarchy of the script organization so that an unspecified report for an entire test/period would list items within functions within scenarios.

The analysis subsystem also provides a unique report which reflects the design approach of SECURE. The Log Summary Report shown in figure 4 summarizes all test/period activity for a port on a single page.

TEST: _____ TEST START: _____ TEST END: ___ ___ DATE: _____ PAGE: __ ___

LOG SUMMARY REPORT

SUMMARY PERIOD START TIME: _____ STOP TIME: _____ DURATION: _____

PORT: ____ ____ _____

	SCENARIOS	FUNCTIONS	ITEMS
THROUGHPUT			
NUMBER IN PERIOD	_____	_____	_____
NUMBER COMPLETED	_____	_____	_____
PERCENT COMPLETED	_____	_____	_____
COMPLETION RATE	_____	_____	_____
TURNAROUND			
AVERAGE	_____	_____	_____
MINIMUM	_____	_____	_____
MEDIAN	_____	_____	_____
MAXIMUM	_____	_____	_____
RESPONSE TIMES	_____	_____	_____

Figure 4. Log Summary Report

All test characteristics are retained in a directory, which is available for display at the user's terminal upon request. The display lists data that will aid the user in his report selection and in his definition of extraction parameters.

4. Implementation Considerations

During the design of SECURE, several ways of implementing the system were considered. One implementation approach which showed promise is based on a mini- and microcomputer hardware configuration as depicted in figure 5. The central component is a 256K-byte minicomputer which completely supports the scripting and analysis functions. During test execution, the minicomputer processes all software except the direct SUT interface, which is under control of very high-speed microprocessors.

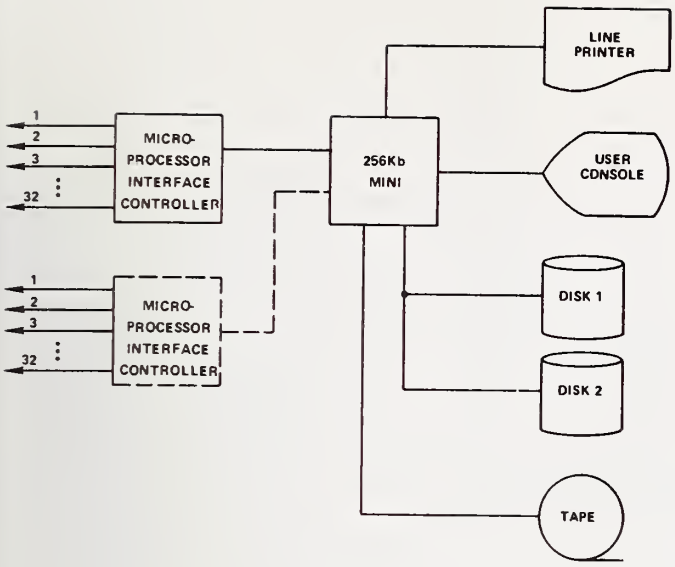


Figure 5. Hardware Components of SECURE Implementation

The entire design is modularly expandable in 16-port increments. One microprocessor is required to support every two increments of port expansion. Buffering, scheduling, and status maintenance within the minicomputer are all dynamically expandable in 16-port increments, as dictated by the port characteristics file.

Buffering is a critical function in the mini/micro architecture. Test execution at near-real-time speed requires the efficient use of buffers to pass data to and from the SUT. In budgeting minicomputer memory, an initial 32-port limitation is imposed (on the assumption that a definition of memory expansion requirements will be derived from the experience of an initial implementation). Disk inputs from the token and test data files are simplex buffered. Journal outputs are blocked, and two screen buffers are allocated for the console. The design requires two buffers (port and emulation) for each port, at 512 bytes each.

Most design software may be considered static and independent of the system targeted for implementation. Static software includes the operating system, I/O handlers, SECURE supervisory software, and most of the non-I/O-related SECURE software. Test-specific software, which is subject to SUT characteristics, includes line handlers (to the SUT) and the terminal emulators. Memory and processing impacts, resulting from expansion beyond 32 ports, should be easily accommodated through corresponding modular expansion.

The SECURE design is both hardware and language independent. The intent has been to achieve a single installation cost that provides a cost-effective improvement over alternative testing methods. Accordingly, the hardware requirements have been scaled in a trade-off with software. The scripting and analysis subsystems impose support demands well within the capabilities of most high-order languages.

References

[1] GSA Handbook FPR 1-4.11, Use and Specifications of Remote Terminal Emulation in ADP System Acquisitions, August 1979.

[2] Shirey, Robert W., "Quality Assurance Tools," in Computer World, May 19, 1980, In Depth/31-49.

CPEUG82

"Improving Organizational Productivity"

Abstracts: Tutorials and Case Studies

TUTORIAL ON CHARGING SYSTEMS IN THE FEDERAL GOVERNMENT

Dean Halstead

FEDERAL COMPUTER PERFORMANCE EVALUATION
AND SIMULATION CENTER (FEDSIM)
Washington, D.C. 20330

On September 16, 1980 the Office of Management and Budget(OMB) issued its Circular No. A-121 entitled "Cost Accounting, Cost Recovery, and Interagency Sharing of Data Processing Facilities. This Circular requires Federal Agencies to implement policies and procedures to (1) account for the full cost of operating data processing(DP) facilities, (2) allocate and report all DP costs to users according to the services received, (3) recover DP costs from external DP users, (4) recover DP costs from internal users when deemed appropriate by the agency, (5) share excess DP capacity with other agencies, and (6) evaluate interagency DP sharing as a means of supporting major new DP applications. In order to satisfy these requirements, Federal agencies will have to develop and implement a DP charging system. Circular A-121 clearly reflects the desire of the Federal Government to begin to manage its DP facilities in a more business-like manner.

By requiring most Government DP facilities to implement a DP charging system, OMB has followed the lead established by many private industry organizations over the last ten years. During this time, private industry took a closer look at the role DP charging systems have in enabling senior management to better manage DP facilities. Many senior officials determined that (1) managing the DP facility should be no different than managing any other department in the organization and (2) installing a DP charging system helps to manage the DP facility like other departments. This point-of-view resulted in a dramatic increase in the number of private industry organizations installing DP charging systems.

Clearly the time has come when all Federal DP managers need to take a closer look at the managerial and technical issues involved in the development and implementation of a DP charging system. The intent of this tutorial is to help CPEUG attendees gain a better understanding of DP charging systems by providing (1) an introduction to the major managerial and technical issues involved with DP charging systems; (2) a detailed example of the most difficult part of developing a DP charging system, the rate-setting process; and (3) some background information on an upcoming National Bureau of Standards' Federal Information Processing Standards guideline entitled "Guidelines for Developing and Implementing a Charging System for DP Services."

Key words: Charging systems; chargeback systems; costing; pricing; DP accounting; billing systems; cost accounting.

REALLY IMPROVING SOFTWARE MANAGEMENT

Thomas B. Cross

Director - Cross Information Company
Boulder, Colorado

ABSTRACT

In the future new tele/conferencing tools will be used to improve productivity in management information systems. One of these tools called computer tele/conferencing may prove to be an important system in the development and management of software.

There is considerable research to support the evidence that software development, user, and management personnel do not generally work well together, even when a team management is used. This can be attributed partly to the fact they have approached their relationship almost entirely ignoring the needs of the ultimate consumer - the user.

Critical new tools are needed to create both a productive and innovative environment for software development efforts. Integrating computer conferencing as a primary vehicle in enhancing these efforts will be pursued in this article. With new electronic information systems, the process of systems development may improve dramatically. Computer conferencing is an exciting new tool for management communications. It is a computer software system which generally allows for:

- Electronic mail - person-to-person communications
- Bulletin Board - access to listed announcements
- Conferencing - many-to-many communications with special areas for electronic meetings. These areas are accessed by people with approval of the conference manager much like a normal meeting. Discussions are held on various topics, specific interests, and work activities. Each user has private files or scratchpads which are kept online for ease of use. Notes or comments can be sent either to other conferees or to the discussion file.

According to Clifford Barney noted expert in this field, "computer teleconferencing is the ability to conduct an ongoing meeting with personnel in different geographic locations. An electronic message system is used to record communications among meeting participants. Each person involved in the meeting can access, read, and respond to these communications, regardless of whether other participants are communicating simultaneously or not. The system thus provides a verbatim log of the meeting, and the asynchronous method of participation offers extraordinary flexibility, especially if meeting members travel frequently (always there) or are in different time zones. The technique has proven to be highly effective for managing ongoing project activities."

SYSTEMS DEVELOPMENT

Using computer conferencing can be used to organize the systems or software development process. However, bringing a group of software programmers together and designating them a systems development team does not necessarily make them a cohesive, productive team. Even when a great deal of care is taken in selecting team members, the result is not an effective team unless the members view themselves, and are viewed by others, as integral participants, each with a significant function.

Some of the major development projects demand a team approach and quite often, all software development requires some team effort. It may only be the user, the programmer, and the software manager. In most situations, however, many users and programmers are at work on a project which may eventually impact many thousands of employees throughout the company.

It is important to recognize that all members of the systems development team (programmers, users, and managers) should each play a distinctive and significant role. In addition, each should be able to acquire some additional status and influence by working with the team. Thus, participation as a team member can be overwhelmingly positive and satisfying.

Experience indicates that in practice it is difficult to achieve this very ideal outcome. Understanding the social structure of the team in terms of individual roles, staus, and power can help isolate and address some of the problems that intefere with attaining the goal.

The members of a systems development team may have many external requirements as well. They may have other projects as well to track, normal continuing duties and responsibilites within that office. This splitting of responsibilities can cause a member to view him/herself as being only peripherally committed to the team.

With computer conferencing, the team can participate in a way far easier than with any other system. CC allows each person to participate "at once." Everyone can 'have the floor' at the same time, everyone from next door to the other side of the world. That is because, with CC systems, you are 'always there'. CC systems have the capacity to allow many people to be working on the same conference or discussion simultaneously which means a potentially greater work output. In addition, where it is difficult to measure work output of 'information workers,' CC provides a record and filing system as well as a personal notepad area for private, secure work away from the conference. For the first time, we can begin to really track the progress of a project from inception to completion allowing software management, new staff, or observers to participate at any point along the way. Also, CC allows conferees to always go back to the beginning of the work and review discussions all along the way.

In developing software it is important to solicit input from others in the department or company -- CC allows many people who may not have direct responsibility or job-related activities to participate. Many people have wide and diverse interests. Engineers, for example, are sometimes fascinated by and may have fresh ideas about marketing. CC facilitates and encourages input by others because the system is designed to allow input and comments from people as 'observers.' These people can review the discussion and may add their own comments.

The name of each person is included when they develop a position or idea and add it into the discussion in a CC system. Each person, additionally, has full opportunity to present positions, raise issues, and even fillibuster without being supressed by the group. Correspondingly, shy or inhibited persons can participate without being worried by the meeting 'bullies' or by the 'Hollywood syndrome' (theatre and acting skills).

Notes can be sent from one person to another, increasing involvement and participation without dominating or detracting from the foreward movement of the conference. Personal friendships can be developed between colleagues over long distances without the limitations of 'telephone-tag.'

Neither rain, nor sleet, nor sickness, nor distance, nor time, will restrict your participation in the meeting. With CC, each person is 'always there' and everyone is 'always on time' for the meeting. Travel and

other meetings are the typical causes of delays in the scheduling of meetings. Such conflicts can disrupt for weeks the scheduling and the timely impact of a meeting's purpose.

Most meetings sometimes must be scheduled so far in advance that interest in them, as well as their ultimate impact, are lost. With CC, the meeting is not dependent on any one person's time schedule. A meeting can go on for long or short periods of time. This is controlled and facilitated by the conference manager who may, in fact, be located in another city. Even when the manager is down the hall, computer conferencing keeps people in touch. Assignments can be given, people can respond, argue, or discuss work without having to physically go to a meeting.

Most groups exist to perform tasks. In meetings, the task involves information gathering, discussion, problem solving, and implementing decisions, and evaluating the outcome of the group's work. Many of these functions require many other people to help support the group, such as secretaries, researchers, and writers. In addition, much of the meeting overhead is concentrated on the organization rather than the task at hand. CC systems provide an organized structure which facilitates the major activities such as, initiating, information seeking, information giving, opinion giving, elaborating, coordinating, evaluating, and energizing. In this way, the group can 'get down to business' rather than spending expensive time on the agenda.

CC lets people concentrate on the substance of issues, rather than the disorganized form they inevitably take. How many times have you confronted the same problem over and over? Multipled by those of close working colleagues, and those of other people in other departments, in other divisions, and in other corporations, ad infinitum, the result is that the wheel is continually reinvented but the problem isn't solved. The saying that goes, "my problems are your problems," actually means that we spend most of our time attacking the same problem, but without making enough headway so that, when it occurs again, we use the same approach over again to solve it.

It is an important aspect of all software project management to document the work. How many times have you gotten minutes from a meeting or really know the status of a project? And, if/when you do get the minutes, how often do they actually convey what really took place at the meeting? CC provides a verbatim transcript. Each person's comments are in their own words, not just the meeting secretary's notes. There is no "body language" to interpret, no bullies to shy from, and no confusion about what somebody really meant. How many times have you sat in a meeting and lost the drift of the meeting, wondering what the person is trying to say, as he or she jumps from point to point? CC structures a meeting in such a way that its purpose and content are not lost.

In a system like MTX - MATRIX system there is a meeting 'status' indicator letting each person know where the meeting is and what has occurred. And, at any point in the meeting, everything that has already transpired can be reviewed. CC also allows a participant to read the speaker's remarks without having to respond immediately. Consequently, it is not so easy for someone to undermine and 'sabotage' the speaker. A friendly rather than confrontational atmosphere is thus created. Notes can be made, questions raised, and comments added without distractions. This allows the speaker to respond to the question when he or she is prepared to and chooses to respond.

It is often said of committees and meetings that 'the camel was a horse designed by a committee.' And, surely, everyone at some point has been frustrated by 'Moscow delegates' -- conference participants who cannot vote without first going back and checking with the boss. CC can provide for assisting members in a way that is rarely, if ever, found in a meeting, i.e., private conferences.

Private notes and discussions can occur during the computer conference allowing participants to determine interests and positions of other people. They can 'test the waters' with fellow workers before bringing the issue to the group.

Consensus building is usually the most difficult and the most challenging

goal of meetings. A true 'meeting of the minds' and a real commitment to putting decisions into action are far easier to bring about through CC. This is because CC allows members to work privately, without formality and scheduling requirements. Members can pair-off to work on the various sub-issues of the problem. In many cases, the monetary savings aren't as important as improvements in productivity.

CC facilitates candid discussion which, in turn, means that people pay more attention. A higher quality of discussion and thinking are the results. Communication and morale are improved because people talk with their supervisors on an equal basis. And, when workers come up with an idea, it is attributed to them. The commitment to 'making it work' is therefore greater.

PROGRAMMER TRAINING

One dynamic use of computer conferencing is for training and career development. Some of the specific learning needs for computer software personnel are:

- Programming - languages, operating systems, applications
- Analytical techniques - probablity, queuing theory
- Systems Analysis Methods - structure and hierarchical data flow
- Systems Design Methods - standards, logical structure, and integrity
- Computing Technology - hardware configurations
- Storage Technology - file management, data base management
- Communications Technology - error detection, data communications
- Project Planning and Estimating - project teams, work scheduling

With CC systems, each of these concepts as well as others can be placed online, updated as necessary, and accessible by all members of the organization at their convenience - at home or at the office. Since CC systems allow a person to access the system at will at their own convenience, a person can go to class in their pajamas! There are no actors or performing skills required of the teach or student.

Using a CC approach, expensive teaching time is reduced, employee time away from work, travel time, and access by other employees for career development.

Another area applicable for using a CC system is in the documentation area. Before we discuss the project CC concept in detail, a review of traditional documentation would be useful for contrast.

DOCUMENTATION

Previous approaches documented the work typically long after the software system has been developed and installed. With computer conferencing, as the project analysts progress through the development cycle, they collect information, record it on notes, and then produce documentation at key project milestones to signify completeness. With CC, all of that work is now kept online. As the cost for development and the number of systems failures grows, management demands greater assurance of success. Thus, with increasing cost accountability, there is real concern over projected risks and whether the intended objectives are attainable within budget. This concern is branding the traditional approach unsatisfactory; a more systematic method for development is required.

In the course of creating an information system, analysts and specialists accumulate a very large quantity of data. When a team of creative people is working toward common objectives, the preparation and maintenance of easily available, well-organized, and up-to-date project documentation is essential to success. With a CC system, all work documentation, meetings, and conferences occur online, just as they would in normal live meetings even though they do not replace all live reviews. However, with all the information online, when live meetings take place many of the problems have been identified and the group can concentrate its efforts rather than reinventing the wheel.

For example, some of the discussions would be:

System Operator Manual
User Manual
Installation Manual
Operating System Interfaces

Within each of the discussions, similar outlines could be developed to present online documentation development by many sources, including daily review of work effort by management.

CONCLUSION

Computer terminal conferencing is a dynamic, easy to use concept for managing the systems development effort, programmer training, and documentation. These are but a few examples of using the 'computer manage the computer effort' in ways that simplify efforts, reduce costs, and save time in an increasingly competitive environment.

KEY ADVANTAGES OF COMPUTER TELECONFERENCING

They are:

ENHANCE COMMUNICATIONS

- Improved access to personnel
- Improved coordination of group activities with rapid and timely messaging to individuals and groups as well as rapid paperless forwarding of messages
- Better coordination of project activities
- More effective use of resources - "You control the meeting time"
- Ability to attend several meetings at diverse locations in a single day - "Being there without going there"
- Quicker solution of problems
- Increased reaction and anticipation to problems - "Always there"
- Availability of diverse and distant participants

SIMPLIFY ACTIVITIES

- Automatic records of messages
- Efficient automated file searching and retrieval
- Delivery impervious to weather, holidays, and labor disputes
- Improvement in time management with elimination of no-contact telephone calls - "Telephone-tag"
- Elimination of unnecessary meetings
- Written records of all meetings
- Participation at user's convenience "Always ontime"
- Portability and remote access to files

LOWERING COSTS

- Reduced photocopying volume and expense
- Reduced paperwork volume and mailing expense
- Reduced labor and travel costs - "No more waiting at the airport"
- Reduce real costs of telephone calls

SAVING AND MANAGING TIME

- Faster delivery of information
- Geographic and time independence - "Around the clock messages"
- Shortened messages and reading times
- Reduction in paperwork, paper handling, and postage costs
- Reduction in office space used for files
- Allows time for 'thought' in responding to important issues

APPLICATION OF SOFTWARE PERFORMANCE ENGINEERING TECHNIQUES

Dr. Connie U. Smith

Duke University
Computer Science Department
Durham, N.C. 27706

The objective of this case study is to present the concept of performance engineering, a software engineering discipline used in the development of large-scale software systems to ensure that they meet performance goals. The concept of performance engineering is now well understood and analysis tools and techniques have been developed which facilitate the prediction of performance characteristics of software designs before coding begins. Unfortunately, problems are typically encountered when the techniques are applied to large-scale software systems during their development.

The application of software performance engineering techniques to one such system is described. Emphasis is on the nature of the problems encountered and proposed solutions to them. Suggestions are made for future work in this area.

TUTORIAL ON WORKLOAD FORECASTING

Helen Letmanyi

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

This tutorial will provide participants with a detailed overview of the organizational approach to the workload forecasting process. The tutorial is recommended for those who have an interest in forecasting workload requirements via quantitative forecasting techniques.

A brief review of the ADP life-cycle will first be discussed to identify the workload forecasting process as an integral part in the life-cycle management of an ADP system. Next, the tutorial will discuss forecasting in general, and some basic forecasting terminology necessary for the understanding of the remainder of the tutorial. Then, a step-by-step approach to the workload forecasting process will be identified. The importance of having definite objectives and goals prior to performing a forecasting process will be discussed. Emphasis will be placed on the importance of an organizational approach to forecasting by translating mission requirements into processing requirements (through workload levels) either in terms of ADP operation or resource usage requirements. Also, the analysis of the forecast results will be discussed.

Analyzing Queueing Network Models Of Computer Systems: A Tutorial On The State Of The Art

Edward D. Lazowska
Department of Computer Science
University of Washington
Seattle, Washington

and

Kenneth C. Sevcik
Computer Systems Research Group
University of Toronto
Toronto, Canada

This tutorial describes recent advances in evaluating queueing network models -- obtaining performance measures such as utilization, residence time, queue length, and throughput from parameters such as workload intensities and loadings. These advances make it possible to quickly evaluate models of large systems (hundreds of devices and dozens of workload components) and to incorporate system characteristics such as memory constraints, priority scheduling, and complex contemporary I/O subsystems.

A model is an abstraction of a system: an attempt to distill, from the mass of details that is the system itself, exactly those aspects that are essential to the system's behavior. Once a model has been *defined* through this abstraction process, it can be *parameterized* to reflect any of the alternatives under study and then *evaluated* to determine its behavior under this alternative. Using a model to investigate system behavior is less laborious and more flexible than experimentation, because the model is an abstraction that avoids unnecessary detail. It is more reliable than intuition, because each particular approach to modelling provides a framework for the definition, parameterization, and evaluation of models. Of equal importance, using a model serves to enhance both intuition and experimentation. Intuition is enhanced because a model makes it possible to "pursue hunches" -- to efficiently investigate the behavior of a system under a wide range of alternatives. Experimentation is enhanced because the framework provided by the particular approach to modelling being used gives guidance as to what experiments are required in order to define and parameterize the model.

Queueing network modelling is a specific approach to computer system modelling in which the computer system is represented as a *network of queues* which is evaluated *analytically*. A network of queues is a collection of *service centers*, which represent system resources, and *customers*, which represent users or transactions. Analytic evaluation involves using packaged software to efficiently solve a set of equations induced by the network of queues and its parameters.

Queueing network models have become important tools in the design and analysis of computer systems because, for many applications, they achieve a favorable balance between accuracy and efficiency. In terms of accuracy, a large body of experience indicates that queueing network models can be expected to be accurate to within 5 to 10 percent for utilizations and throughputs and to within 25 to 50 percent for residence times. This level of accuracy is consistent with the requirements of a wide variety of design and analysis applications. Of equal importance, it is consistent with the accuracy achievable in other aspects of the computer system analysis process, such as workload characterization.

In terms of efficiency, queueing network models can be defined, parameterized, and evaluated at relatively low cost. Definition is eased by the direct correspondence between the attributes of queueing network models and the attributes of computer systems. Parameterization is eased by the relatively small number of high level parameters that must be specified. Evaluation is eased by the recent development of algorithms whose running time is only a few seconds for models of realistic systems.

These efficient evaluation algorithms are the first subject of this tutorial. We survey their evolution, and describe the way in which they work. A spectrum of algorithms have been developed, providing the possibility of trading off between cost and accuracy. Some analysis techniques produce bounds on answers, indicating ranges in which the values must lie. Others produce point estimates of performance values. Some algorithms provide exact solutions to the underlying equations derived from the computer system; others produce approximate solutions by obtaining better and better estimates through iteration.

The efficient evaluation algorithms described above are applicable only to systems satisfying certain assumptions. Thus, in order to retain this necessary efficiency, we must sacrifice some generality in those system characteristics that can be directly represented by our models. Characteristics that cannot be directly represented typically arise in areas such as memory management (memory constraints, swapping, paging), I/O subsystems (disks with rotational position sensing, loosely coupled multiprocessors), and CPUs (overhead, priority scheduling). We incorporate these characteristics in our models indirectly, by transformation of the basic parameters.

Significant progress has been made in recent years in the range of system characteristics that can be efficiently represented in this way. These techniques are the second subject of this tutorial. We present a number of them, which are interesting in their own right and which are illustrative of the general approaches that are used.

CATEGORIES OF BACKUP STRATEGIES

Susan K. Reed

Institute for Computer Sciences and Technology
National Bureau of Standards
U.S. Department of Commerce
Washington, DC 20234

The basic strategies for backup operations during recovery from disaster are described. A number of possible variations are included. Some pointers are given in how to select strategies.

Key words: Backup operations; contingency planning; disaster recovery; empty shell; reciprocal aid; recovery center; redundant facilities; shared contingency facility.

1. Introduction

There are a number of possible strategies which can provide partial relief in the event of a computer center contingency. A contingency is defined in this case as any natural or man-made event which prevents the processing of information by destroying the facility or equipment, rendering the facility unreachable, or depriving the facility of utilities and supplies. However, there are several variations within each strategy, and because there are so many possibilities, each one must be evaluated. In addition, selecting one strategy for each facility may not always be sufficient. Different scenarios will require different remedies; for instance, loss of air conditioning for a few days might point to the need for service bureau backup while loss of a whole facility for a month would more likely point to a recovery center.

Especially among commercial facilities, there is considerable disparity in services offered under similar but different names. Even in the friendly atmosphere of two redundant facilities owned by the same organization, it is easy to assume that the other party is doing something that they are unaware is a requirement. In the hectic activity following a disaster, it is impossible to foresee all misunderstandings. For this reason, it is imperative to take nothing for granted and to get all agreements in writing. This is not to say that commercial vendors of contingency services are not reliable but only to suggest that there are so many possible variations, different meanings, and possibilities for misunderstanding that every care should be taken to assure that there are no surprises. It is a good idea to have a prepared list of questions on hand for interviews.

It should be noted that none of these solutions is ready and waiting for a disaster to occur. All must be carefully researched, evaluated, contracted for, planned, and tested in advance of a disaster.

2. Commercial Solutions

2.1 Service Bureaus

The purpose of the service bureaus is well known and their utility under contingency circumstances is obvious. It is well to find a bureau which will store source and object code and data on their premises. All of these must be kept updated and should be tested as often as monthly. Testing can be combined with regular runs. Service bureaus are ideal backup solutions for jobs which need to be run once in any given period, such as payroll. The warning about making arrangements in advance holds true here. Service bureau time cannot be expected to be available upon demand.

There is even a company which, for a fee, will find available processing time that is not being used by its owners. Arrangements are made entirely through this third party, rather than with hardware owners.

2.2 Empty Shells

These are large, unfurnished spaces which can be leased to house computers and related processing equipment. Usually they

are equipped with raised flooring, air conditioning, telecommunication lines and electric wiring are in place but not connected. Shells are usually available from the same vendors who supply recovery centers. In fact, there is usually a planned progression from recovery center to empty shell if the disaster is serious enough to require offsite backup for more than 60 to 90 days. At a minimum, it takes about three weeks to equip an empty shell.

2.3 Recovery Centers

These are fully equipped, furnished spaces which include a computer and all hardware normally necessary. As a rule they also usually have raised flooring, air conditioning, fire protection and warning devices, telecommunications lines and physical security. The hardware with which these centers are equipped has been carefully chosen to meet the requirements of a large number of organizations. Often if a customer will not be able to operate without a specific piece of equipment, it will be installed for exclusive use at the customer's expense.

Problems to be considered are hardware compatibility, cost of moving personnel and maintaining them away from home, site security, number of others wanting to use facility following a disaster, willingness of personnel to leave home during a disaster, availability for testing.

A number of vendors offer both empty shell and recovery center services in one location. In such cases there is usually a limit on the length of time a recovery center may be occupied, e.g., 60 days, before the client is required to equip the empty shell and continue operations there. Some vendors also lease time in their recovery centers for overload operations but customers for this service must vacate the premises immediately in case of a disaster. These variations in available services are mentioned to show that a wide range of services is available, but that each vendor has a different approach to the disaster and contingency planning problems.

Most contingency centers have resident staffs of experts who will help customers with risk analysis, planning, documentation, installation, etc., or they will do the entire task for them.

3. Non-commercial Solutions

3.1 No Hardware Backup (Reversion to pre-computer processing)

It may be that an organization determines that no hardware backup is necessary. If so, this decision should be based on a risk analysis and should be carefully documented. It should not be considered a reason for not needing a contingency plan. It may be that the data processing functions not sufficient critical and that manual processing will be temporarily satisfactory.

If it is the case that manual processing, or whatever was used prior to obtaining a computer, will suffice for a limited period of time, then it will be necessary to prepare a contingency plan for such processing, assigning personnel to tasks, providing all necessary equipment and supplies, describing the system in detail, including flow diagrams.

3.2 Reciprocal Agreements (Mutual Aid)

These are formal, written, signed agreements between one or more parties, each of which has a computer facility with excess time available. The facilities must be completely compatible and must maintain this compatibility throughout the period of the agreement. This probably works best between similar organizations with common business interests, e.g., two banks. It is extremely difficult to maintain compatibility between systems because the smallest change in either facility can effect it. The time requirements of a company can also change very rapidly, reducing excess time.

3.3 Membership in Shared Contingency Facility

These are essentially the same as recovery centers, but they are typically formed by a group of similar businesses which use identical hardware, e.g., inssurance agencies, for their own exclusive use. Management is selected by and is at the behest of the member organizations. Even though these facilities are not commercially available, the fees are nearly comparable to those of commercial recovery sites. Ample time for testing is allowed. Usually the facilities may be used for overload except in the event of a disaster.

As more and more recovery centers become available, there are fewer and fewer shared facilities. It appears that the task of managing such a facility and arriving at agreements with all the members is an impossible task better left to an impartial management.

3.4 Geographically separate locations (Redundant Facilities)

This is probably the best solution to disaster planning if the installation of two complete computer facilities is financially feasible. If a government agency has the rare opportunity to design its facilities from scratch, it can probably financially justify the need for sufficient redundant hardware to support backup operations. If each geographically separated part of the facility is large enough to carry the critical load (determined by risk analysis), there will be little disruption of operations due to disaster. While it is not necessary for both locations to have identical hardware, each must have enough to support the critical workload.

The separate parts of the facility must be located geographically far enough apart that they will not simultaneously be affected by the same natural hazards or electrical outages.

BENCHMARK CONSTRUCTION AND VALIDATION USING SYNTHETIC SOFTWARE (A TUTORIAL OUTLINE)

Bruce D. Grant

Systems Architects, Inc.
Denver, Colorado 80228

This tutorial examines the use of synthetic software in the development of acquisition benchmarks. As such, characteristics of synthetics and the methodology employed to use them effectively are explored. When properly utilized synthetics offer significant advantages over application software in many cases. They are relatively simple to select, modify, and tune to particular benchmark categories and requirements yielding significant savings in time and resources. Acceptance of synthetics has been relatively slow due to problems related to successful mapping of the synthetic to the characteristics of existing and planned workloads and a susceptibility to the effects of optimization. While still worthy of note, these problems may be significantly reduced through proper technique during the developmental process.

1. Introduction

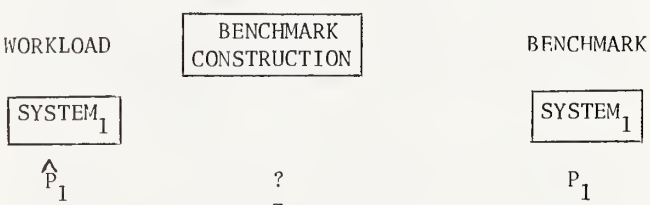
The process of competitively acquiring a computer involves many complex steps requiring a significant amount of preparation. Benchmark development is undoubtedly one of the most time and resource consumptive portions of any procurement.

The benchmark is clearly a highly critical step in the acquisition process. Properly constructed, a benchmark provides a fair test of the features and performance capabilities of vendors' equipment against the procuring organization's future workload. Improperly constructed benchmarks have often provided information of negative value in that inappropriate equipment appeared most suitable and was procured based on benchmark results.

The two major potential problem areas in the benchmark development process are the forecasting of future workloads and the representation of those workloads by a set of benchmark problems. This tutorial examines the second of these through the use of synthetic programs and support routines.

2. Problems in Achieving Representativeness Using Application Software

The following figure 'Test of Benchmark Representativeness', borrowed from Dr. Dennis Conti [1], supports the concept of an ideal linear relationship between 'actual' workload (including forecasts) and the benchmark workload.



PERFORMANCE VARIABLES ($\hat{P}_1$, P_1):

- Response Time
- Turnaround Time
- Transaction Rates
- Processing Charges

Figure 1. Test of Benchmark Representativeness

Other valuable performance criteria include elapsed time and various resource oriented performance measures. These may include CPU time, total I/O per test, channel or device I/O, swapping and paging activity, memory utilization measures (maps, etc.), and system overhead statistics. When synthetic software is under consideration, other pertinent performance data

may include instruction mix information for the particular benchmark category to aid in tailoring the synthetic.

Application software often poses many obstacles to representative and manageable benchmark development. Many benchmark categories simply cannot be effectively represented by application software. The functions to be represented may not exist on the present computer or the associated application software may be biased in favor of (or entirely dependent upon) the incumbent vendor's machine. Another problem arises when the software and/or data contains information subject to security and privacy restrictions. While not an insurmountable obstacle, its' correction may be very time consuming.

In yet other cases, the category could be represented by application software, but it may be too cumbersome or resource consumptive to do so. The application software may require significant familiarization or modification time by either developer or vendor. In cases of long system lives, application software may prove too inflexible for the changing requirements of the particular category (e.g., an increasing amount of query versus update in a DBMS scenario). The benchmark problem may require incredibly large data or transaction files which may change for each potential augmentation point, suggesting synthetic data generators. Within an RTE benchmark, all of these problems may be magnified by the number of terminals and iterations involved.

The key problem of the representativeness and suitability of application software must ultimately be examined from two perspectives: 1) Does the selected software adequately test the specific category's functions and workload over the system life?; and 2) Will the selected software do (1) in an unbiased and economical (i.e., time, labor, materials) manner?

Many benchmarks are developed with one or both of these conditions unsatisfied. Figure 2 illustrates the development process from step 7 of FIPS 75 to benchmark completion.

3. Synthetic Benchmark Programs and Support Routines

Synthetic software, including synthetic benchmark programs, data generators, and validation routines, avoid the previously described limitations when properly designed or selected and tested for the particular benchmark category. As per Fleming and Rucks [2], a good synthetic possesses the following qualities: "(1) it consists solely of the constructs of a generally used standard high level language (e.g., ANSI FORTRAN or ANSI COBOL); (2) it exercises a typical set of programming langauge functions; (3) it provides a set of user controllable parameters for synthesis of an application workload into a synthetic workload; and (4) it produces repeatable, predictable, and verifiable results."

The advantages of such software are obvious: high transportability, ease of mapping to processing features (assuming adequate variety of functions with the synthetics), ease of tuning to resource requirements, and an associated reduction in costs for the entire benchmark development effort and, in many cases, the LTD itself.

Synthetics have been criticized primarily for not effectively representing the instruction mix of the specific benchmark category. Nearly any program could be 'tuned' to consume identical amounts of CPU, I/O, or memory per unit time as the category under examination. The type, number, frequency, and order of instruction's is another matter, however. 'Resource oriented' synthetics are to be distinguished from 'functionally oriented' synthetics. The former merely mapped workloads at detailed system resource levels without adequate consideration for the order and complexity of functions (including instruction mixes) involved.

Work performed by Fleming and Rucks [2], the U.S. Department of Agriculture, the National Bureau of Standards, and Systems Architects, Inc. has attempted to address this problem. FORTRAN, COBOL, and DBMS synthetics were examined by the author with respect to the four qualities described above, with particular emphasis placed on instruction mix flexibility. Subsequent 'tests' of the representativeness of these programs have been performed by using program analyzers, which record instruction mix information, and the more traditional accounting logs and software monitors. When a synthetic passed a 'suitability test' and closely approximated the expected or 'real' workload by means of traditional performance measures and instruction mix analysis, it was judged to be successful in representing the designated category. Figure 3 illustrates the testing and tuning process of a typical benchmark development effort employing synthetics.

Synthetic support routines, primarily data and transaction generators, also can yield significant savings during development of a benchmark. Parameterized software which generates a large number of unique records in a rapid, controlled manner offers great flexibility when creating and tuning a particular benchmark problem. Additionally, by means of modification of 'seed' values, greater control is maintained over 'live' data when the benchmark is run. (Unique 'seeds' may be provided when the LTD is run.) Storage requirements are minimized for transmittal purposes and the vendor may use the routines for sizing purposes prior to the LTD.

While not synthetics per se, the following software can be of great assistance in testing, validation, and tuning synthetic software.

It is often extremely helpful to create

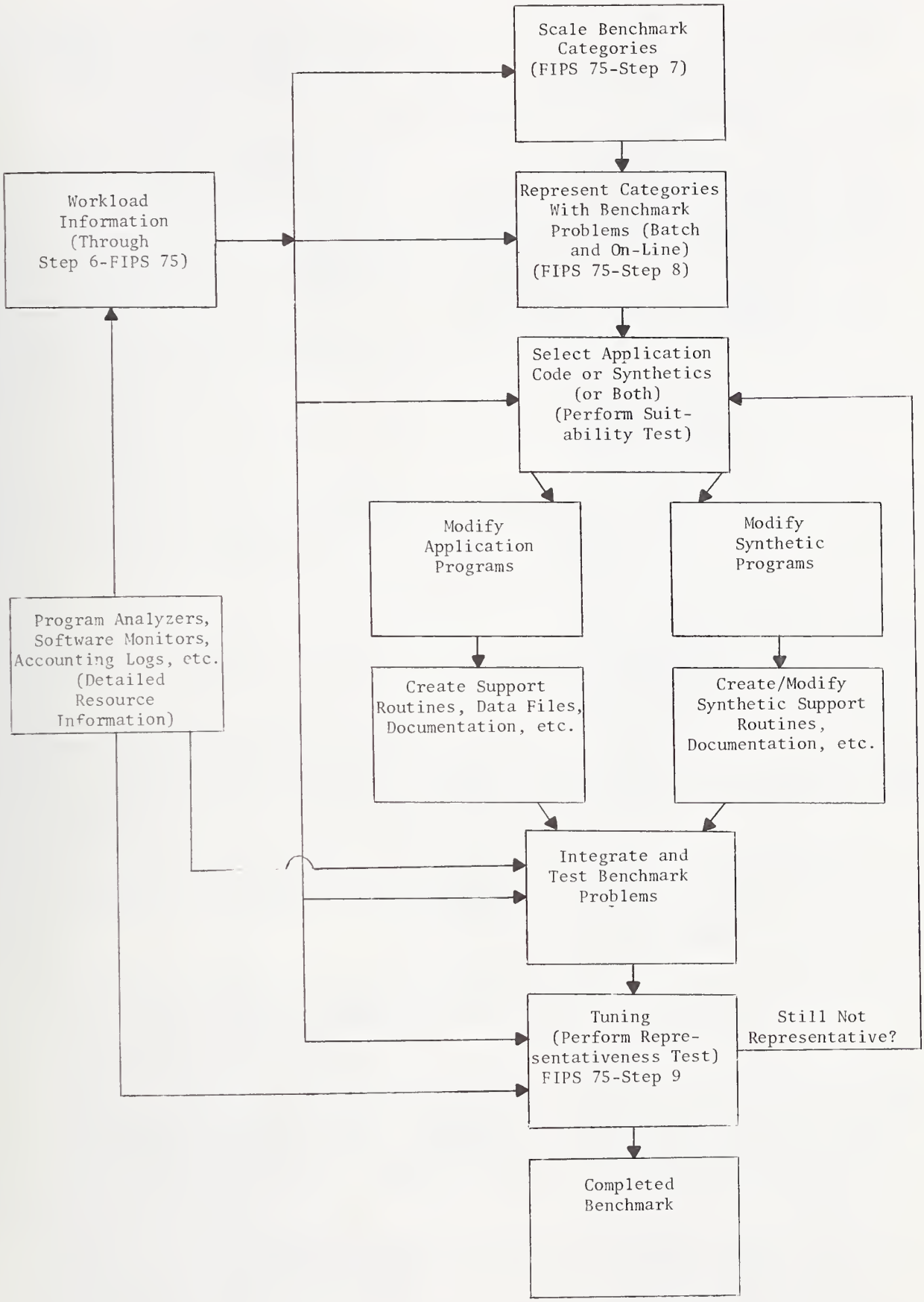


Figure 2. Benchmark Development

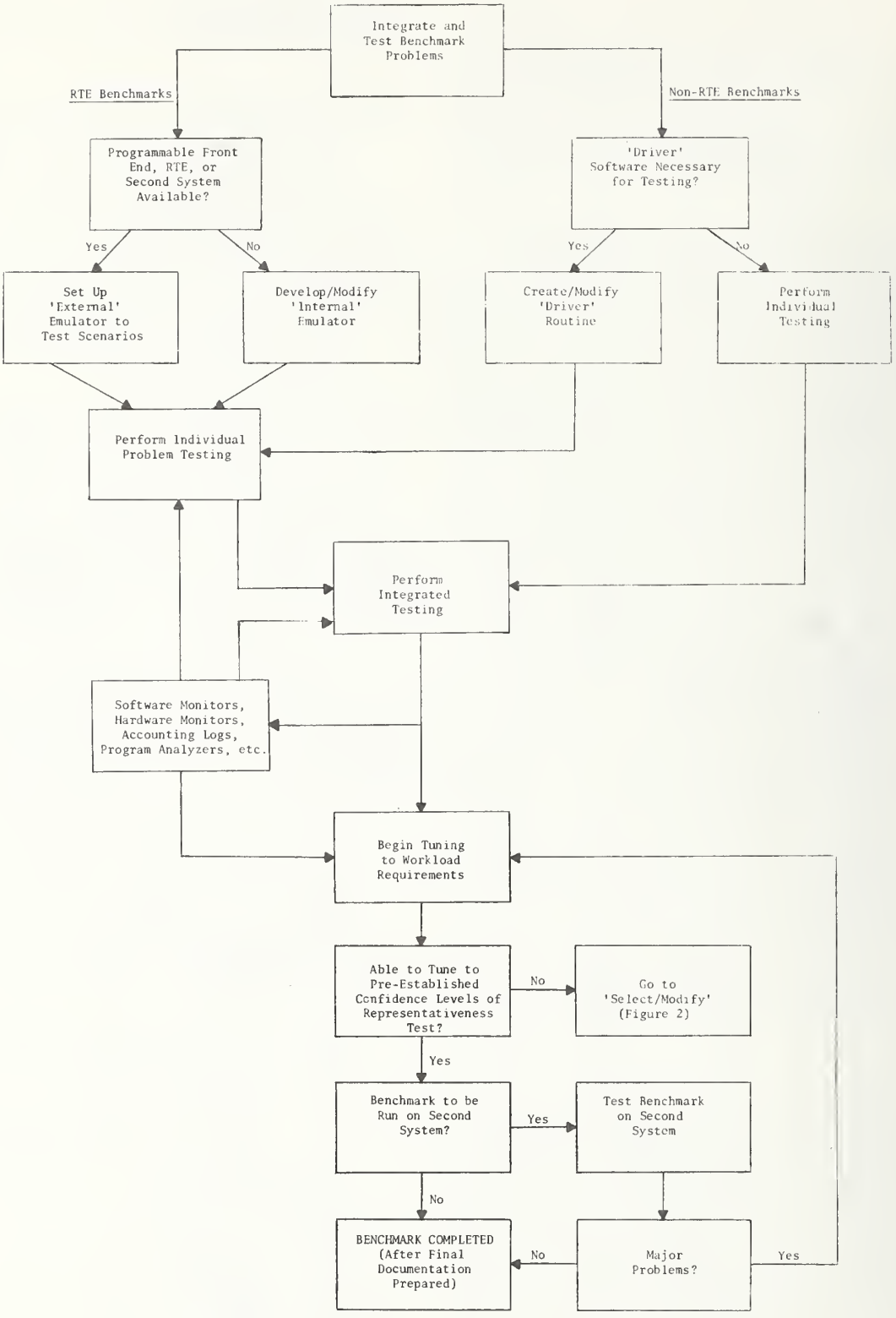


Figure 3. Testing and Tuning the Benchmark

parameterized report generation routines for validation purposes. By developing report software which retrieves every 'nth' record of an output file, control is maintained and output requirements can be significantly reduced.

Perhaps more importantly, internal emulators fulfill an important need for an RTE benchmark where an external RTE-type device is not available. A driver program, preset for type and think time delays, with an internal monitoring or checkpoint capability, is of great assistance. Such a program can provide response, turnaround, and transaction rate information. System software and programming are often required to effectively balance the load, however. The primary limitation of an internal emulator is in the inability to factor in the corresponding communications overhead and factor out the effects of the emulator itself.

4. Benchmark Development Steps Using Synthetic Software

The following steps describe techniques and questions to be resolved in the construction of benchmarks using synthetic software. These steps commence with FIPS PUB 75 Step 8.

STEP A: PERFORM SUITABILITY TEST

For Each Benchmark Category:

- Of Existing (or Suggested) Application Codes:
  a) Does it map to that category's functions over the entire system life?
  b) Is it biased towards the incumbent?
  c) If privacy or security considerations, is the effort required to 'sanitize' prohibitive?
  d) Would interface or familiarization requirements on the vendor's part be excessive?
- Of Synthetics:
  a) Are functionally oriented synthetics available which map to the category? If not previously developed and available, can they be expeditiously and economically developed? Does the synthetic functionally resemble the category's requirements in terms of:
     --Language (COBOL, FORTRAN, etc.)
     --Processing mode and type (e.g., interactive, ISAM update)
     --Range of capabilities (see 'c' below)
  b) Is a detailed breakdown of each category's functions (including instruction mix analysis) available or readily obtainable? If not, what is the degree of difficulty involved in developing (b)?
  c) What percent of the category's functions (including language instructions) can be expressed with the synthetic? (Should be > 90%)
  d) Is the modification required to achieve (c) prohibitive?

STEP B: SELECT OR CREATE REPRESENTATIVE PROGRAMS

Based upon each benchmark's requirements, the weight or importance of each item in Step A will vary. The procuring organization should evaluate the 'Suitability Test' results in conjunction with guidance provided by FIPS 42-1 and FIPS 75.

STEP C: MODIFY PROGRAMS AND CREATE SUPPORT ROUTINES (IF REQUIRED)

- For Each Program:
  a) Modify to achieve maximum functional representation.
     --Processing requirements
     --Instruction mix
- For Each Benchmark Category:
  a) Assess trade-offs of using 'live' data versus synthetically generated data or transactions.
     --Are data storage and transmittal requirements excessive?
     --Would synthetic generators simplify the benchmark effort for both parties (as opposed to the use of existing 'live' data)?
     --Does 'live' data pose security or privacy problems? Is 'sanitization' difficult?
     --Is data or transaction generation software readily available? If not, what level of effort is required to develop it?
  b) Would parameterized validation routines simplify the evaluation process?
     --Primarily for benchmarks with large data requirements.
     --Is the development effort justified?

STEP D: INTEGRATE AND TEST BENCHMARK PROBLEMS

- For Each RTE Benchmark:
  a) Is an RTE or external driver system available?
     --If YES, does benchmark (and procurement) justify its use (perhaps for extended period) as testing tool?
     --If YES, are resources available (primarily manpower and communications capability) to make it feasible?
     --If YES, and it is the development systems' front end, can it be programmed to effectively function as an external emulator?
     --If NO, is an internal emulator available for development system?
     --If NO and an internal emulator is not available, does benchmark (and

procurement) justify its development as a testing tool?

- For Each Non-RTE Benchmark:

  a) Does benchmark complexity justify 'driver' software for effective testing?

- For Each Benchmark Category:

  a) Perform individual problem testing.
  --For synthetics, is sufficient level of detail in analytical data available (functional and resource performance information)?

  b) Perform integrated testing.

STEP E: FINE-TUNE BENCHMARK AND PERFORM REPRESENTATIVENESS TEST

- For Each Benchmark Problem:

  a) For application software based problems:
  --Does benchmark workload equal or closely approximate actual or forecasted workloads? Suggested performance measures include:
    - Response Times
    - Turnaround Times
    - Transactions Rates
    - Processing Charges or System Resource Measures

  b) For synthetic software based problems:
  --Do the same conditions as in (a) hold true?
  --Does synthetic instruction mix closely approximate actual mix?
  --Do system resource measures (memory, CPU, I/O, utilization, etc.) closely approximate expected results?

5. Summary and Conclusions

Synthetics are applicable to a variety of workloads and benchmark requirements. These must be carefully analyzed with respect to many of the factors described previously. In order to limit the effects of software and compiler optimization, it is also important that the developer map as many characteristics of the actual workload as possible and does not rely exclusively on synthetics.

Various sources of functional synthetic benchmark routines are available; the best are undoubtedly the National Bureau of Standards 'Synthetic Library' of COBOL and FORTRAN routines.

References

1 Conti, D., Acquisition Benchmarking, Supplemental Proceedsings, CPEUG, 17th Meeting, November 1981.

2 Fleming, P. M., Rucks, A. C. A FORTRAN Synthetic Program for Benchmarking. Weatherbee, J. E., ed. Computer Performance Evaluation Users Group: CPEUG 15th Meeting; 1979 October 15-18, San Diego, CA. Washington, DC: U.S. Government Printing Office: 193-199.

3 Conti, D. M. Use of Synthetic Benchmarks for Estimating Service Bureau Processing Charges. Washington, DC: National Bureau of Standards (NBS), Institute for Computer Sciences and Technology (ICST). 1976 July. 47p.

4 Conti, D. M. Findings of the Standard Benchmark Library Study Group. Washington, DC: National Bureau of Standards (NBS), Institute for Computer Sciences and Technology (ICST). 1979 January. 49p.

5 National Bureau of Standards. Federal Information Processing Standards (FIPS) Publication. Guidelines for Benchmarking ADP Systems in the Competitive Procurement Environment. Washington, DC: NBS, 1977, May 15, FIPS PUB 42-1.

6 National Bureau of Standards. Federal Information Processing Standards (FIPS) Publication. Guidelines for the Measurement of Interactive Computer Service Response Time and Turnaround Time. Washington, DC: NBS, 1978, August 1, FIPS PUB 57.

7 National Bureau of Standards. Federal Information Processing Standards (FIPS) Publication. Guideline on Constructing Benchmarks for ADP System Acquisitions. Washington, DC: NBS, 1980, September 18, FIPS PUB 75.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS SP 500-95	2. Performing Organ. Report No.	3. Publication Date October 1982

4. TITLE AND SUBTITLE

Proceedings of the Computer Performance Evaluation Users Group 18th Meeting - CPEUG 82 - "Increasing Organizational Productivity"

5.

Proceedings Editor - Carol B. Wilson

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

Institute for Computer Sciences and Technology
NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Institute for Computer Sciences and Technology
National Bureau of Standards
Department of Commerce
Washington, DC 20234

10. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 82-600622

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

These Proceedings record the papers that were presented at the Eighteenth Meeting of the Computer Performance Evaluation Users Group (CPEUG 82) held October 25-28, 1982, in Washington, DC. With the theme, "Improving Organizational Productivity," CPEUG 82 reflects the critical role of information services in the productivity and survival of today's organization. To meet this challenge, the scope of CPE must be expanded to address performance issues in all aspects of information systems (hardware, software, facilities, communications, personnel, policies, and procedures) throughout the system life cycle. The program was divided into three parallel sessions and included technical papers on previously unpublished works, case studies, tutorials, and panels. Technical papers are presented in the Proceedings in their entirety.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

benchmarking; capacity planning; chargeback systems; computer performance management systems; queuing models; resource measurement facilities; simulation; supercomputers; workload characterization.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

414

15. Price

$11.00

USCOMM-DC 6043-P80

U.S. GOVERNMENT PRINTING OFFICE: 1982-360-997/2258

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $18; foreign $22.50. Single copy, $4.25 domestic; $5.35 foreign.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW., Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Services, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services, Springfield, VA 22161, in paper copy or microfiche form.

U.S. Department of Commerce
National Bureau of Standards

Washington, D.C. 20234
Official Business
Penalty for Private Use $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

SPECIAL FOURTH-CLASS RATE
BOOK