NBS SPECIAL PUBLICATION 405

U.S. DEPARTMENT OF COMMERCE / National Bureau of Standards

Benchmarking and Workload Definition: A Selected Bibliography with Abstracts

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards [1] was established by an act of Congress 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 consists of the Institute for Basic Standards, the Institute for Materials Research, the Institute for Applied Technology, the Institute for Computer Sciences and Technology, and the Office for Information Programs.

THE INSTITUTE FOR BASIC STANDARDS provides the central basis within the United States of a complete and consistent system of physical measurement; coordinates that system with measurement systems of other nations; and furnishes essential services leading to accurate and uniform physical measurements throughout the Nation's scientific community, industry, and commerce. The Institute consists of a Center for Radiation Research, an Office of Measurement Services and the following divisions:

Applied Mathematics — Electricity — Mechanics — Heat — Optical Physics — Nuclear Sciences [2] — Applied Radiation [2] — Quantum Electronics [3] — Electromagnetics [3] — Time and Frequency [3] — Laboratory Astrophysics [3] — Cryogenics [3].

THE INSTITUTE FOR MATERIALS RESEARCH conducts materials research leading to improved methods of measurement, standards, and data on the properties of well-characterized materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; and develops, produces, and distributes standard reference materials. The Institute consists of the Office of Standard Reference Materials and the following divisions:

Analytical Chemistry — Polymers — Metallurgy — Inorganic Materials — Reactor Radiation — Physical Chemistry.

THE INSTITUTE FOR APPLIED TECHNOLOGY provides technical services to promote the use of available technology and to facilitate technological innovation in industry and Government; cooperates with public and private organizations leading to the development of technological standards (including mandatory safety standards), codes and methods of test; and provides technical advice and services to Government agencies upon request. The Institute consists of a Center for Building Technology and the following divisions and offices:

Engineering and Product Standards — Weights and Measures — Invention and Innovation — Product Evaluation Technology — Electronic Technology — Technical Analysis — Measurement Engineering — Structures, Materials, and Life Safety [4] — Building Environment [4] — Technical Evaluation and Application [4] — Fire Technology.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides technical services designed to aid Government agencies in improving cost effectiveness in the conduct of their programs through the selection, acquisition, and effective utilization of automatic data processing equipment; and serves as the principal focus within the executive branch for the development of Federal standards for automatic data processing equipment, techniques, and computer languages. The Institute consists of the following divisions:

Computer Services — Systems and Software — Computer Systems Engineering — Information Technology.

THE OFFICE FOR INFORMATION PROGRAMS promotes optimum dissemination and accessibility of scientific information generated within NBS and other agencies of the Federal Government; promotes the development of the National Standard Reference Data System and a system of information analysis centers dealing with the broader aspects of the National Measurement System; provides appropriate services to ensure that the NBS staff has optimum accessibility to the scientific information of the world. The Office consists of the following organizational units:

Office of Standard Reference Data — Office of Information Activities — Office of Technical Publications — Library — Office of International Relations.

---

[1] Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington, D.C. 20234.
[2] Part of the Center for Radiation Research.
[3] Located at Boulder, Colorado 80302.
[4] Part of the Center for Building Technology.

Benchmarking and Workload Definition: A Selected Bibliography with Abstracts

Josephine L. Walkowicz

Systems and Software Division
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Frederick B. Dent, *Secretary*
NATIONAL BUREAU OF STANDARDS, Richard W. Roberts, *Director*

Issued November 1974

Library of Congress Cataloging in Publication Data

Walkowicz, Josephine L.
Benchmarking and workload definition.

(National Bureau of Standards Special Publication 405)
Supt. of Docs. No.: C13.10:405
1. Electronic digital computers—Evaluation—Bibliography.
I. Title. II. Title. Workload definition. III. Series: United States. National Bureau of Standards. Special Publication 405.
QC100.U57 No. 405 [Z5642.2] 389'.08s [016.0016'4] 74-17210

National Bureau of Standards Special Publication 405

Nat. Bur. Stand. (U.S.), Spec. Publ. 405, 45 pages (Nov. 1974)
CODEN: XNBSAV

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1974

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
(Order by SD Catalog No. C13.10:405). Price $1.05

Benchmarking and Workload Definition: A Selected Bibliography with Abstracts

Josephine L. Walkowicz

These 85 citations to the literature of benchmarking and workload definition were selected from a longer list of documents, encompassing a somewhat broader scope, that was submitted to Federal Information Processing Standards (FIPS) Task Group 13 in response to a request made to attendees of the Task Group's Planning Session held on July 12, 1973, at the National Bureau of Standards. One of the topics discussed at the Planning Session was the collection of a selected bibliography on workload definition and benchmarking. The bibliographic effort was to be directed not so much toward exhaustiveness as toward the development of a bibliography that the attendees had found useful and would, therefore, recommend to other workers in the field. Of the approximately 250 citations submitted to the Task Group, these 85 were selected on the basis of two criteria: (1) the item dealt primarily with benchmarking or workload definition; and (2) hard copy was available at the Institute for Computer Sciences and Technology. The citations are arranged alphabetically by last names of the first authors. Each citation has an abstract, a classification category assignment, and a list of key words. The category assignments are made from a classification scheme that was developed for the collection and that is used here as a Category Index to the Bibliography. A Key Word Index is also provided.

Key words: Benchmarking; bibliography; computer performance measurement; computer procurement; workload definition.

Introduction

A Planning Session to review the objectives, scope, and proposed program of work for FIPS Task Group 13, Workload Definition and Benchmarking, was held at the National Bureau of Standards on July 12, 1973. One of the topics discussed at the Planning Session was the collection, as a possible ongoing activity of the Task Group, of a selected bibliography on workload definition and benchmarking. The effort was to be directed not so much toward the development of an exhaustive bibliography of the field as toward a bibliography of literature items that the attendees had found useful and would, therefore, recommend to other workers in the field.

To initiate this effort, all attendees were invited to submit lists of such documents. The response to this invitation yielded a collection of approximately 250 citations to the literature of benchmarking and of computer performance measurement.

In addition to citations, attendees submitted descriptions of the benchmarking process in competitive computer procurements, and some suggestions for key words to describe that process. In organizing the material for further use by the Task Group, it became apparent that the work might be of immediate value to a wider audience, and a decision was made to publish the bibliography as a National Bureau of Standards document.

The *Bibliography* consists of 85 citations selected from the original collection on the basis of two criteria: (1) the item dealt primarily with benchmarking or workload definition; and (2) hard copy was available at the Institute for Computer Sciences and Technology. The number in parentheses which terminates each bibliographic entry has significance only as a local file reference. Each citation has an abstract, one or two numbers indicating classification category assignment, and a list of key words. The citations are arranged alphabetically by last names of first authors.

The classification scheme was developed on the basis of a consolidated description of the benchmarking process, as envisaged by Planning Session attendees, and augmented to include a general category to accommodate necessary discussions of the measurement process and related background topics. Items are assigned to one or more categories, as appropriate to reflect their content. The classification scheme is presented on page 40, where it also serves as a Category Index to the publication.

The Key Word Index also begins on page 40. The key words assigned to each citation were taken from the literature itself; no attempt was made to standardize terminology or to decide whether or not one author's job mix was another's workload. Needless to say, neither the classification scheme nor the list of key words can be considered exhaustive or definitive of the field; both require further development and testing against a larger collection of documents.

In addition to regular distribution of NBS Special Publications, this publication will be made available to FIPS Task Group 13 for further use in whatever continuation effort the Task Group deems necessary or desirable. Reader comments are invited regarding the general value and interest in expanding the scope of the bibliography, and will be gratefully received by the author.

Bibliography

1. Abrams, Marshall D., George E. Lindamood, and Thomas N. Pyke, Jr., Measuring and Modelling Man-Computer Interaction, in Association for Computing Machinery, *Proceedings of the SIGME Symposium*, February 1973, pp. 136–142, 11 refs. (6430166)

   The paper describes the Dialogue Monitor developed at the Institute for Computer Sciences and Technology as a tool for the measurement of computer services, particularly those provided by interactive systems. The model of service used here is concerned with performance measurement external both to the user and to the computer, and focuses on the dialogue which takes place between the two.

   External performance measurement completes the analytic and stimulus approaches used by Karush; service may be measured as delivered to actual users or to an artificial stimulus.

   The character, with two associated descriptors, is the unit of measurement. The first descriptor is the identity of the character's source; the second descriptor specifies the time of occurrence of the character. The character itself is explicit; its source is implicit in the communications discipline; and an external clock provides the time of occurrence.

   With these simple data several models of the man-computer dialogue have been developed. The models differ primarily in the degree of the interactiveness of the dialogue. The paper presents two models and the terms used to describe each. The data stream model is described in terms of idle time, think time, computer burst, user burst, computer interburst time, user interburst time, computer burst segment, user burst segment. The stimulus-acknowledgement-response model is described in terms of acknowledgement delay, acknowledgement time, acknowledgement character count, system response time, system transmit time, system character count, user think time, user transmit time, and user character count.

   The operation of the monitor is described briefly and some analysis of the data is provided. (JLW)

Category: 1.2

Key words: Dialogue monitor; idle time; man-computer interaction; measurement tools; models; think time.

2. Arbuckle, R. A., Computer Analysis and Thruput Evaluation, *Computers and Automation*, 15:1 (January 1966) pp. 12–15, 19. (6430197)

   "The real criterion for measuring system performance is thruput. Yet many evaluations use only internal comparisons to rate a system's overall performance." Add-time, instruction time, instruction-mix, and kernel problem comparisons may provide relative internal performance figures for specific cases, but these are generally applicable to comparable computer families. In thruput evaluation, the power of a system must be measured in terms of how fast it can perform the complete job.

   In this connection, two type of benchmark problems are suggested: one which estimates time, another which reports actual running times. How well either can evaluate thruput depends essentially on two major considerations: how well the benchmarks reflect actual jobs; and how well they characterize the total workload. Production job runs are identified as the "best way" to measure a system's performance. The widespread use of generalized compilers provides the capability to run actual production jobs or systems with entirely different organizations. The choice of selecting jobs to reflect total system load still remains, even with this approach. Hardware monitors can be used to evaluate and tune system components. The article concludes with an example of the use of a hardware monitor to improve performance of an IBM–7094 system. (JLW)

Categories: 3.0; 1.2

Key words: Application benchmarks; computer systems: hadware monitors; IBM–7094; throughput.

3. Bell, T. E., Computer Performance Analysis: Measurement Objectives and Tools, The RAND Corp., Santa Monica, Calif., Rept. No. R–584–NASA/PR, February 1971, 32 pp., 27 refs. (6430172)

   This report suggests a number of objectives for computer system measurement and analysis beyond the commonly accepted one of tuning. Objectives in computer operations—identifying operational problems and improving operational control—mean that personnel in this area should become familiar with new tools and techniques. Computer system simulators should be concerned with model validation as well as model development. Installation managers need results from this field in order to select equipment, trade man-time for machine-time, and tune installed equipment.

Data collection tools for use in measurement and analysis are necessary to fulfill these objectives. These tools range from simple, inexpensive ones—audio and visual indicators, operator opinions, and logs—to the more sophisticated hardware and software monitors. Each of the simple tools can provide initial indications of performance, but hardware and software monitors are usually necessary for a thorough analysis. Five binary characteristics can describe a monitor: (1) implementation medium, (2) separability, (3) sample portion, (4) analysis concurrency, and (5) data presentation. An analyst should determine the characteristics his analysis requires before choosing a product.

Recognizing objectives and choosing measurement tools are two important steps in a performance analysis study. This report deals with these two topics so that analysts can proceed to four more difficult and critical topics. Modeling, choosing a data collection mode, experimental design, and data analysis deserve at least as much attention as examining data collection tools. (Author)

Category: 1.2

Key words: Computer performance analysis; computer performance measurement; measurement tools; simulators; validation.

4. Bell, T. E., B. W. Boehm and R. A. Watson, Computer System Performance Improvement: Framework and Initial Phases, The RAND Corp., Santa Monica, Calif., Rept. No. R–549–PR, August 1971, 55 pp., 4 refs. (6430187)

   This report distills selected RAND experience and research in the measurement and evaluation of computer system performance into a set of practical guidelines for organizing the initial phases of an effort to improve the performance of a general-purpose computer system. The report is designed primarily as an aid for "getting started" and provides a procedural framework which consists of seven phases. Only the initial three phases are discussed in detail in this report.

   Phase 1 is "understanding the system," and a Preliminary Questionnaire is suggested for this purpose. The Questionnaire asks general, descriptive questions about organization, workload, hardware and software, and the accounting system. For Phase 2 which is "analyzing operations," a Detailed Questionnaire is suggested as a guide to the kind of data gathering which must be undertaken in order to analyze computer system performance. The details required identify characteristics of operations, of jobs, and of the system. A Questionnaire on current measurement and evaluation activities is also suggested. Phase 3 is an aid to installations in developing performance improvement hypotheses; methods of analysis are suggested that provide a transition from analyzing operations to formulating hypotheses, and a number of general hypotheses appropriate to particular problem situations are presented. (Modified author)

Category: 1.2

Key words: Computer performance analysis; guidelines; questionnaires.

5. Boehm, B. W., Computer Systems Analysis Methodology: Studies in Measuring, Evaluating, and Simulating Computer Systems, The RAND Corp., Santa Monica, Calif., Rept. No. R–520–NASA, September 1970, 42 pp., 17 refs. (6430186)

   The report is a summary of the results of four studies on computer systems analysis and simulation performed under contract to NASA.

   One of these studies was on measurement and evaluation of computer systems. Among the critical areas cited in this context is that of "a strong instability in gross measures of multiprogrammed system performance (central processing unit utilization, throughput, etc.) with respect to changes in load characteristics, disk data set allocation, and scheduling algorithms. Small changes in load characteristics, etc., can easily produce large changes in multiprogrammed system performance. This phenomenon has the following significant operational implications: (1) significant improvements in CPU utilization or throughput (usually at least 30 percent; sometimes over 300 percent) can be realized from investments in tuning multiprogrammed computer systems; (2) computer systems selected and procured because of their performance on a series of benchmark jobs can lead to disastrous mismatches if great care is not taken to assure that the benchmarks are fully representative; and (3) as workload characteristics change with time, the maintenance of a well-tuned computer requires a continuous rather than a one-shot effort."

   Thus considerable study of the pertinent interactions is necessary before the key contributing factors are isolated. In situations arising from several dominant factors, use of the simplest explanation as a basis for decision can lead to "highly dysfunctional" results.

   A good example of this phenomenon is provided in one of the studies. Performance measures of an IBM–360/65 (in terms of the percentage of CPU cycles productively utilized) indicated an increase after the addition of 50 percent more of

core memory and several additional disk drives. However, a more detailed analysis of the data indicated that the increase actually correlated with a decrease in the average number of jobs resident in the increased memory, and was primarily due to an "otherwise undetected" increase in average CPU usage by individual user jobs. Analysis also indicated that the increased performance was due as much to decreases in the I/O characteristics of the workload as it was to configuration changes. (JLW)

Category: 1.2

Key words: Computer performance evaluation; computer performance measurement; multiprogrammed computer systems; system tuning; workload representation.

6. Boksenbaum, Melvin, Results of Benchmark Comparison of the Performance of IBM–360/85 and 370/165, Memorandum Rept., 2 Apr 71, 11 pp. (6430146)

   The author states that the benchmark is a valid and useful tool in a comparison of the 360/85 with the 370/165. The reasons given for this are several. The performance of the 360/85 is known so that the relative performance of the 370/165 will be meaningful. Both machines are available with similar peripheral equipment and operating systems software. The programs selected for the benchmark are representative of the installation workload.

   The benchmark consists of three parts: (1) a compute-bound linear programming problem run standalone; (2) an I/O bound monthly financial closing job also run standalone; and (3) a job stream of 29 programs taken from the normal installation workload.

   The first two programs cited above represent the two extreme types of processing by a computer system and their performance by the 370/165 is a good measure of that computer's capabilities.

   The 29 programs comprising the job stream were carefully selected in order to represent every type of job run on the 360/85. Programs from each user department were selected on the basis of departmental monthly computer usage. The package includes a daily financial run which spans the job stream, six linear programming jobs of various sizes, and 22 other programs whose elapsed running times vary from one to fifteen minutes and which are coded in PL/I, COBOL, assembly language, and object code. The benchmark has a run time of approximately one hour on the 360/85. Summary data are presented in tabular form.

   The results indicate that performance of the 370/165 should be approximately 90 percent of the 360/85. Turnaround time, peripheral utilization and overtime usage will not be significantly affected. However, the replacement is expected to effect a cost reduction of more than 20 percent, so that net increase in performance is significant. (JLW)

Category: 3.3

Key words: Application benchmarks; benchmark run analysis; computer performance measurement; computer systems; IBM–370/165; IBM–360/85; workload representation.

7. Bookman, Philip G., Barry A. Brotman and Kurt L. Schmitt, Use Measurement Engineering for Better System Performance, *Computer Decisions*, 4:14 (April 1972) pp. 28–32. (6430167)

   The article suggests, in the absence of a discipline of "measurement engineering," an engineering-like approach which will provide a methodology for the application of measurement tasks. This approach is illustrated by a case study of a data center at Allied Chemicals which had an IBM–360/50 and a 360/40 processing a workload consisting of local batch, remote job entry, and online systems. Through the use of the Configuration Utilization Evaluator (CUE) and of the Data Set Optimizer (DSO) the system was tuned and optimized so that the net results were the dropping of the 360/40 system and an annual saving of about a quarter of a million dollars. The author considers hardware and software monitors necessary tools to a measurement methodology, without which it would be impossible to obtain information on what parts of the system are in need of improvement or which ones are not operating at optimal capability. (JLW)

Category: 1.2

Key words: Computer performance measurement; configuration evaluators; data optimizers; measurement engineering; software monitors; system optimizing; system tuning.

8. Brocato, Louis J., Getting the Best Computer System for Your Money, *Computer Decisions*, 3:9 (September 1971) pp. 12–16. (6430198)

   The article describes a method for evaluating vendor proposals, based on weighting all of the required system elements and dividing the score by dollar costs. The "best" system then is benchmarked. This is a departure from current practice in which all vendors are required to perform the benchmark. If the benchmark run of the "best" system is successful, then that system is selected for procurement. If the benchmark fails, then the "next best" system is benchmarked, and so on, if necessary, until a contract is awarded. (JLW)

Category: 1.3

Key words: Proposal evaluation.

9. The Benchmark, or What It Takes to Measure Up *Sperry Univac Review*, 3:1 (1973) pp. 8–9. (6430239)

   A brief overview of Univac's benchmarking facilities and practices. Of note are the following observations. This [benchmark test] "may last as long as six hours for a UNIVAC–9700 or even five eight-hour days for a UNIVAC–1110. It normally includes a time demonstration in a stopwatch environment and a functional test of basic software capabilities." One benchmark facility includes an IBM–360/40 [which] "allows us to run our system directly against the competition's . . . It also helps us find specific problem areas in a program of a 360 user who is a potential customer." The benchmarking facility is considered to be a skill center continually building conversion knowledge. From 500 to 1,000 operational programs may be represented in a customer conversion which is usually a longer process than the pre-sale benchmark. At the Marketing Test Center in Eagan Township, Minnesota, there are usually 2 to 5 benchmarks in process, with customers in attendance, with another 10 to 15 benchmarks in various stages of completion. "Today's customers want to actually witness performance, so that benchmarks precede virtually all procurement. An average benchmark here takes from 6 to 12 weeks, including preparation. This includes approximately 100 hours of actual computer time and from 60 to 75 total man weeks. The time can vary greatly. We've had benchmarks requiring as little as four hours from start to finish. Others have taken up a year." The first successful 1110 benchmark was completed in November 1972 at this facility, and some fifty more have been completed since then. In addition, two or three 1106 benchmarks are also run each week. This facility also does final checkout on specific benchmark configurations of benchmark programs prepared in totally executable form by the Eastern Test Center located in Washington, D.C. (JLW)

Category: 3.0

Key words: Benchmark costs; benchmark facilities; benchmark practices; benchmark times; UNIVAC.

10. Buchholz, W., A Synthetic Job for Measuring System Performance, *IBM Systems Journal*, 8:4 (1969) pp. 309–318, 6 refs. (6430101)

    Performance in this article is defined quantitatively in terms of the running time of a given job. As a yardstick serves to measure length, so a synthetic job can serve to measure those characteristics of a computer to which the job is sensitive. There are several requirements for a synthetic job: (1) it can be stated as a machine-independent procedure; (2) it must be meaningful over a wide range of computer systems; and (3) it should be short enough to permit accurate measurement, and yet not so long that measurement becomes a burden—this requires a cyclic procedure with running time directly proportional to the number of repetitions.

    The synthetic program which is used as an illustration is written in PI/I and is modeled after a file maintenance procedure which makes heavy use of I/O devices. The program has a compute kernel of variable length and its storage requirements may be varied also, so that it is possible to simulate compute- and I/O-bound situations.

    In multiprogramming situations a well-controlled job stream, consisting of synthetic jobs with known properties, can be used as a measure of system throughput and of slowdowns in jobs resulting from concurrent processing of other jobs.

    The author makes no claim as to the representativeness of the synthetic program of a real file maintenance application. However, the adjustable parameters of the program can serve to approximate the steps of a real program and thus provide a tool for simplifying the testing of benchmarks. Though a synthetic program cannot represent all of the complexities of a real program. it can be much better controlled and can provide the user with details on what is being measured and what the limitations are.

    Similarly, a synthetic file maintenance program alone may not be able to model all of the steps of a real application. Also there is the inherent danger that a single measure of performance may lead to designing a system tuned to a particular job. "But if several dissimilar programs are used, a system that does well in all of the items is likely to do well on real jobs. A small collection of parameterized procedures, imitating such operations as sorting and matrix computations, may well prove to be adequate standards of comparison from which a user can select those most appropriate for his application." (JLW)

Category: 3.2

Key words: Machine independence; synthetic program (PL/I); synthetic program requirements.

11. Buckley, Fletcher J., Estimating the Timing of Workload on ADP Systems: An Evaluation of Methods Used, *Computers and Automation*, 18:2 (February 1969) pp. 40–42, 9 refs. (6430194)

    The article discusses and evaluates methods used for timing of specified workloads by various computer systems. Three methods are discussed: (1) mathematical calculations of CPU and I/O

time required for execution of a workload which is described usually by a detailed layout of files and by a generalized description of programs; (2) simulation; and (3) benchmarks.

The validity of the first method is questionable for several reasons. These are the grossness of the program descriptions, the subjective elements in estimating numbers and "average" instructions, and the disregard of the effects of the operating system as well as of the efficiency of the compiler.

Simulation includes the same basic methodology coupled with techniques which would provide better workload representations. These techniques include use of more detailed and complete object program descriptions, use of probability tables to permit study of random events, and incorporating compiler characteristics into the simulation process. To make simulation a valuable tool in the evaluation process, the author suggests that objective means of measuring and testing basic parameters of computers and of operating systems and better methods of workload descriptions are required. He adds, however, that "additional work along these lines seems impractical and appears to be on the extreme fringe of the state-of-the-art."

Benchmark techniques seek to represent the total workload by the use of representative programs. Advantages of the benchmarking process are that the actual operating system, the actual compiler, and the actual machines proposed by a vendor are tested. Abstractions of these components are, therefore, unnecessary. Several objections are raised to the benchmarking process: (1) uncertainty about representativeness of benchmark programs and of test data; (2) the cost in time and dollar resources of providing benchmark programs and data; (3) cost of conversion of benchmark programs to fit vendor computers; (4) cost of the benchmark runs; (5) difficulties in obtaining a precise configuration of a vendor's system; and (6) "the apparent impracticability of attempting to completely benchmark all aspects of a workload, for example, 30 remote users in a real time situation."

Partial solutions to the problems inherent in benchmarking are suggested. The programs constituting a given workload can be categorized into classes, and the significance of each class can be determined by the major portion (in terms of running time) each class requires. Two sets of data are also suggested—one to be used by the vendor to test a benchmark program which had been modified to run on a particular computer, the other set for a customer-supervised run to obtain actual running time data.

Actual costs of running a benchmark are not considered to be substantial. A truly representative benchmark can be run in 2 hours, which at $300 an hour (IBM–360/50) is not a large fraction of the total benchmark cost. Analysis of results is, of course, a must in the case of the benchmark which, in the author's opinion, can produce good results, even if the benchmark is not a panacea. (JLW)

Category: 2.2

Key words: Benchmark costs; benchmark time; computer systems; IBM–360/50; workload representation; workload timing.

12. Budd, A. E., A Method for the Evaluation of Software. Volume 1. General Description Including Benchmark Considerations, Mitre Corp. Bedford, Mass., Tech. Rept. No. MTR–197, August 1966, 36 pp. (ESD–TR–66–113, Vol. I; AD–639586) (6430200)

    The report describes a three-step procedure for evaluating software: (1) select the software component categories needed by the user; (2) select individual features of each category by weighting the importance of each measure of software capability; and (3) assign appropriate weights to the features selected.

    Nine software categories are identified: (1) procedural language compilers; (2) operating, executive, or monitor systems; (3) sort and/or merge routines; (4) output report generators; (5) languages for simulation; (6) symbol manipulation languages; (7) data management systems; (8) general utility library routines; and (9) problem oriented language systems.

    Fourteen measures are then identified as those useful to any installation considering the addition of one of the software components to its facility. The choice of measures to be so used was dictated by considerable care in selection of characteristics that: (1) are significant; (2) cover the complete range of software components; (3) are separable and sufficiently distinct to be of general utility; (4) are measurable in some realistic way; and (5) allow relative comparison of some attribute.

    The suggested measures are listed below. The report presents with each measure a quantity that is measurable, a related quantity for purposes of comparison, and a method of assigning a value to the measure in cases where this is not obvious. A short title follows each measure.

    (1) Decreased Programming Time (Programming)
    (2) Decreased Checkout Time (Checkout)
    (3) Conserve Compilation Time (Compilation)
    (4) Conserve Execution Time (Execution)
    (5) Framework for Efficient Utilization of I/O Equipment (I/O Utilization)

(6) Economies in Object Program (Product Economy)
(7) Framework for Efficient Utilization of Secondary Storage (Secondary Storage)
(8) Flexibility for Hardware Growth or Change (Growth Flexibility)
(9) Framework for Optimizing Simultaneous Use of I/O and Central Processing Units (Simultaneity)
(10) Decreased Facility Maintenance Cost (Maintenance)
(11) Minimized Operator Intervention (Intervention)
(12) Increased Machine Independence (Independence)
(13) More Standardized Documentation (Documentation)
(14) Less Initial Programmer Training (Training)

Once the user/evaluator has selected features significant to his requirements, he can determine their relative advantages and which features vary from one vendor or machine to another. A basic format for this process is suggested.

The report concludes with a section on general benchmark considerations. A table presents timing requirement standards broken down by recognizable points in time and timing quantities suitable for each. For observers and reporters of the results of a benchmark demonstration, a guideline-questionnaire is presented in 8 parts: (1) General and Administrative Questions; (2) Benchmark or Source Program Development; (3) Translation of Source Program; (4) Test and Debug; (5) Object Program Operations; (6) Potential Software Deficiencies; (7) Potential Software Extras; (8) General Comments. (JLW)

Categories: 1.2; 3.6; 10

Key words: Benchmark timing standards; guidelines; questionnaires; software characteristics; software classification.

13. Calingaert, P., System Performance Evaluation: Survey and Appraisal, *Comm. ACM*, 10:1 (January 1967) pp. 12–18.

The state of the art of system perfomance evaluation is reviewed and evaluation goals and problems are examined. Throughput, turn-around, and availability are defined as fundamental measures of performance; overhead and CPU speed are placed in perspective. The appropriateness of instruction mixes, kernels, simulators, and other tools is discussed, as well as pitfalls which may be encountered when using them. Analysis, simulation, and synthesis are presented as three levels of approach to evaluation, requiring successively greater amounts of information. The central role of measurement in performance evaluation and in the development of evaluation methods is explained. (Author)

Category: 1.2

Key words: Computer performance measurement; measurement parameters; system availability; throughput; turnaround.

14. Campbell, D. J. and W. J. Heffner, Measurement and Analysis of Large Operating Systems During Systems Development, in AFIPS, *Proceedings of the Fall Joint Computer Conference*, 1968, Part 1, pp. 903–914, 3 refs. (6430224)

The paper reports on experience in the development of GECOS III which led to the development of a series of techniques for the measurement and analysis of the behavior of operating systems. Both hardware and software measurement techniques used in developing GE's operating systems are discussed; software techniques include simulation and system recording.

System recording encompasses four techniques: (1) system design that allows for adequate measurement; (2) built-in system auditing techniques; (3) event tracing; (4) performance analysis and recording.

The authors' experience with each of these is reported in detail, with summary output data presented in graphic form. The single lesson learned is that continuous measurement is an absolute necessity for top efficiency in operating systems. "It is truly amazing how seemingly minor changes in a system can have profound effects on overall performance." (JLW)

Category: 3.6

Key words: Event counters; event tracing; GECOS III; operating system measurement; operating systems; simulation; software performance analysis; system auditing; system design; system monitoring; system recording.

15. Cantrell, H. N. and A. L. Ellison, Multiprogramming System Performance Measurement and Analysis, in AFIPS, *Proceedings of the Spring Joint Computer Conference*, 1968, pp. 213–221. 5 refs. (6430173)

"Performance analysis consists simply of trying to answer the question, 'Where does the time go?' and given an answer, applying a subjective judgment as to whether the amount of time spent on a given function is reasonable." These concepts (as embodied in the classical theory measurement/revised theory/revised measurement/cycle) are here applied to an analysis of performance and software of the GE–625/635 GECOS II operating system.

GECOS II keeps a running total by program (system as well as application) of all processor.

channel, and device time used. Analysis of accounting data produced by GECOS II revealed that the question, "Where does the time go?" could not be answered from this data. A tentative theory, "The extra time goes into overhead," was advanced. Analysis of overhead processor time revealed the fact that overhead processor time was significant but not excessive, and that there was significant idle time in a supposedly heavily loaded system. The theory, "The extra time goes into overhead," was dropped and replaced by the question, "Why is processor idle time so high?" Analysis of octal memory dumps was made and led to the conclusion that overhead was not the problem, and that "whatever the problem was, it could not be exposed by analysis of overhead time summaries."

This analysis also yielded a "succession of well defined, understood, and evaluated system performance bugs . . . which were corrected in the prototype versions of the next software system distribution." The resultant throughput improvement on customer sites averaged 30 percent and ranged from 10 to 50 percent, depending on the load mix.

Application of the "Where does the time go?" principle to analysis of individual program performance yielded some interesting results. For example, one COBOL program was found to be spending most of its total time in the machine in opening, closing, and checking the labels of a small transient tape file. Another interesting result was really an unexpected discovery of a solution to the problem of determining where compute time goes within a program. By using a high density sampling method and interrupting an executing program according to some statistically independent pattern, it was noted that the frequency with which the interrupt location falls within a particular instruction sequence is proportional to total program time spent in executing that instruction sequence. This method was applied to a variety of programs and proved to be a valuable tool for tuning long running and/or frequently used programs, as well as in locating compute time performance bugs. The first application of this tool to the FORTRAN compiler increased compiler speed by 27 percent.

The current implementation of both system and individual program performance measurements is done by MAPPER which is loaded as a user program. MAPPER runs are made over a period of several hours of normal system operation and require one online printer, 4K of core, and penalty of 10 percent in reduced machine performance. A sample of output of the monitor is provided in the article. Data output from these runs show that the following performance factors can be identified and evaluated: (1) hardware configuration bottlenecks—I/O channels, tapes, drums, disc, core, etc.; (2) indications of inefficient I/O strategy—blocking, buffering, etc., in individual user programs; (3) the effectiveness or ineffectiveness of the actual multiprogramming mix; (4) the effects of machine room procedures upon system performance; (5) the relative efficiencies of various optional strategies of machine operation with enough information to define why this way is better than that one; (6) possible operating system performance bugs.

Another output of MAPPER is the major event report which provides a tool for measuring and understanding the detailed operation of the operating system.

In addition to the above, some intangible results are reported. Of particular interest and significance to benchmarking is the capability here to accumulate data on total time used by each program, and sorting the results to arrive at the top ten (or twenty or fifty) programs. These top ten, typically, appear to account for 50 to 80 percent of the total load. This implies that a very few programs account for a large portion of installation efficiency and throughput. These programs are ideal candidates for fine tuning, especially in view of the fact that the tools discussed here have been successfully applied to considerable reductions in program running times. Mixing the top ten with appropriate short jobs can do much to optimize throughput and increase capacity of the installation. (JLW)

Category: 1.2

Key words: GECOS II; operating system measurement; operating systems; program timing; program tuning; software monitors; software performance analysis; throughput; workload representation.

16. Commission on Government Procurement, *Report of the Commission on Government Procurement,* Volume 3, Chapter 5, Special Products and Services, December 1972, pp. 45–54. (6430201)

    Because of its recent dramatic growth as an industry, and its importance to all government operations, automatic data processing equipmen was one of the products singled out for special treatment by the Commission on Government Procurement.

    The Commission's findings regarding benchmarks are reported here *verbatim.*

    Benchmarks. These are a series of computer programs designed by the using office as representative of the workload that will be processed. Benchmark requirements are incorporated in Government solicitations for

computers. The Business Equipment Manufacturers Association (BEMA), in a poll of equipment manufacturers, obtained a general estimate that as much as 50 to 80 percent of the cost of bidding is tied to benchmarks, mainly because they are individually designed and developed separately for each procurement.

Acquisition procedures should be tailored to the equipment that is being acquired. Often more than half of an agency's budget for an information system may be expended before the specifications are released to industry. The use of standard benchmarks would substantially reduce these costs.

Recommendation 14. Develop and issue a set of standard programs to be used as benchmarks for evaluating vendor ADPE proposals.

Top-management officials give a great amount of attention to equipment oriented details. The emphasis is on computer performance and evaluation rather than ADP personnel costs. For example, each new system now involves a completely new set of benchmark programs with the attendant personnel costs of their development. One solution would be to develop a set of standard programs for use as benchmarks. At the time of acquisition, an appropriate sample of the standard programs would be selected. Vendors and procuring personnel would become familiar with the standard programs, and only the program mix would change from one procurement action to another. The Center for Computer Science and Technology, an organization in the NBS Institute for Applied Technology, is a logical choice to develop standard benchmark programs.

Category: 1.3

Key words: Benchmark; computer procurement policy; procurement methods; specifications development.

17. Crowding, Edward F., A Controllable Synthetic Job Stream for Benchmarking, IBM Systems Development Div., Poughkeepsie, N.Y. Rept. No. TR 00.2173–1, 6 June 72, 13 pp., 2 refs. (6430149)

    Benchmark job streams have been widely used in comparing performance of hardware/software systems. This paper recounts experiences in defining, collecting, and executing representative job streams.

    A new benchmarking approach is presented—the synthetic job step. A control card specifying desired characteristics is input to the synthetic step and controls its execution. The technique has resulted in faster, less expensive and more accurate benchmark tests for systems performance evaluation. (Author)

Category: 3.2

Key words: Accounting data; benchmark run analysis; job stream representation; synthetic job step; synthetic jobs; synthetic job stream; workload analysis; workload characteristics.

18. Denning, Peter J., A Statistical Model for Console Behavior in Multi-User Computers. *Comm. ACM*, 11:9 (September 1968) pp. 605–612, 3 refs. (6430168)

    The ability of a computer system to communicate with the outside world efficiently is as important as its ability to perform computations efficiently. It is quite difficult to characterize a particular user, but rather easy to characterize the entire user community. Based on the properties of this community we have postulated a hypothetical "virtual console." No claim is made that a virtual console behaves like any actual console, but the entire collection of virtual consoles models the collection of actual consoles. Using the model we answer questions like: How many processes are suspended waiting for console input? What is the maximum rate at which a process can execute? What bounds can be set on overall buffer requirements? Answers to these and similar questions are needed in certain aspects of operating system design. (Author)

Category: 1.3

Key words: Input/output design; models; multi-user terminals; operating system design; statistical models; virtual console.

19. Denning, Peter J., The Working Set Model for Programming Behavior, *Comm. ACM*, 11:5 (May 1968) pp. 323–333, 17 refs. (6430169)

    Probably the most basic reason behind the absence of general treatment of resource allocation in modern computer systems is an adequate model for program behavior. In this paper a new model, the "working set model." is developed. The working set of pages associated with a process, defined to be the collection of its most recently used pages, provides knowledge vital to the dynamic management of paged memories. "Process" and "working set" are shown to be manifestations of the same ongoing computational activity; then "processor demand" and "memory demand" are defined; and resource allocation is formulated as the problem of balancing demands against available equipment. (Author)

Category: 1.3

Key words: Models; multiprogrammed computer systems; paging; program analysis; program behavior; program models; resource allocation; scheduling; statistical models; storage allocator; working set model.

20. Department of the Army, Development of Standard Benchmarks, in *Management Information Systems. Information Processing Systems Exchange,* Pamphlet No. 18–10–2, May 1973, pp. 1–8.

The article describes the joint effort undertaken in September 1972 by the Departments of the Army, Navy, Air Force, and the Defense Supply Agency to develop standard benchmarks. The standard benchmarks appear to be a less costly alternative to benchmarks composed of user programs. The description of workloads by a number of standard, variable parameter programs (or standard benchmarks) appears to be feasible with the increasing use of task oriented benchmark programs (update, sort, etc.) instead of application benchmark programs, like payroll, inventory, etc.

Overall control of this project has been assigned to the Department of the Army. Estimates of the development effort are presented in a PERT chart, and expected to require roughly 70 man-months over a 10–12 month period. Maintenance of the program library is estimated to require two people full time. This may more than double with the expansion of the library for future applications, such as scientific. Development of initial programs and distribution to suppliers is expected in the Fall of 1973.

Though not claimed as a panacea solution to the current benchmark dilemma, certain advantages to standard benchmarks are cited: (1) reduced preparation time and cost for users; (2) reduced cost and response time for suppliers; (3) flexibility in use; and (4) a wider base of users. The major disadvantages cited include: (1) high initial cost; (2) non-universality (in that standard benchmarks are not feasible in procurement of time-sharing, real time, process control, data management applications, or in those situations where standard programming languages will not be used); and (3) lack of user confidence.

The initial concepts were refined at the Contributors' Symposium on Standard Benchmarks which was held at the U.S. Army Computer Systems Support and Evaluation Command in September 1972. Comments from Symposium participants indicated manufacturer willingness to support this endeavor which would be beneficial to them as well as to the Government. The main concern of manufacturer participants was the development of programs that would permit all suppliers to represent capabilities of their equipment accurately and fairly, at low cost.

A flowchart for a typical, modular synthetic program for updating is given. Also, there is a chart indicating an interface between the user approach to the preparation of the benchmark and that of the supplier. (JLW)

Category: 3.2

Key words: Contributors' Symposium on Standard Benchmarks; standard benchmark program library; standard benchmarks; synthetic benchmark program library; synthetic benchmarks.

21. Department of the Army, Test Data Generators, in *Management Information Systems. Information Processing Systems Exchange,* Pamphlet No. 18–10–3, September 1973, pp. 1–8. (6430191)

The article suggests a definition for test data generation: "Any manual or automated process or method by which test data (records or files) may be generated on the basis of controlling parameters."

A review of current test data generation methods and parameters indicates that software test management practices have not evolved beyond initial practice of using existing data sets or creation of test data sets from hand-coded test data. This is considered to be a significant factor in the ever increasing costs of software development, failures to meet production deadlines, and malfunctions or failures of new programs. Current use of test data appears to be generally limited to testing of new programs and changes or modifications thereto.

Available software for automated test data generation is described under three categories:

(1) "*Repeaters*"—packages capable only of repetitive data generation with little user control.
(2) *Media-to-media copying and reformatting software*—generally requires much human effort to create adequate test data.
(3) *Test data generators*—packages which generate test data sets according to user supplied dimensions and specifications.

In order to be classed as automated test data generation software, a package should require manpower outlays only to set specifications, to prepare control operation statements, and to run control cards. The general characteristics of such software are discussed, together with implementation philosophies and their impact on both quality and costs.

Use of test data generation software can accrue to considerable advantages to an installation, such as reductions in development costs as well as in production losses due to programmer errors. However, few of these advantages will be realized unless the staff is properly trained in the use of the packages and unless software development management policies and procedures are directed toward maximizing the potential benefits of test data generators. (JLW)

Category: 5.1

Key words: Test data generators.

22. Department of Defense Computer Institute, Air Force Computer Evaluation and Selection Procedures, no date, 187 pp. (6430185)

The material has been extracted from an operating manual in use by the Air Force ADPE Selection Office for computer evaluation and selection. The segments included provide a complete overview of the constraints and considerations that are employed by the Air Force. Material not included is concerned with internal office procedures and other matters in greater detail than necessary for course requirements.

The Selection Office uses a combined cost effectiveness/technical evaluation technique. Embodied here is the concept of a probabilistic workload representation against which the vendors' equipment is tested in the live test demonstration which is mandatory in accordance with a SECDEF memorandum dated July 20, 1966, and entitled, "Management and Use of the Electronic Computer." Benchmark problems comprise the live test demonstration, and must be processed by each vendor participant. The criteria specified for the benchmark are first "that the problem must be well defined so that a specific, representative test can be given. The problem must be unambiguous to ensure that each manufacturer is demonstrating against the same criterion and should be written in a higher level language. . . The primary validation tools are the representative benchmark problems and functional demonstrations of certain proposed features which the vendor must satisfactorily run using the equipment and software proposed." (JLW)

Category: 1.3

Key words: Application benchmarks; benchmark specifications; computer performance evaluation; computer selection procedures; Department of the Air Force; workload representation.

23. Dopping, O., Test Problems Used for Evaluation of Computers, *BIT*, 2:4 (1962) pp. 197–202. (6430199)

The article describes the choices and kinds of benchmarks ("test problems") designed by a Swedish governmental committee (in cooperation with the Swedish Board for Computing Machinery) for use in a procurement of computers for Sweden's census and tax collection operations.

Five test problems, "all very carefully defined" were given to manufacturers for benchmark runs: (1) punched card-magnetic tape conversion, including a number of plausibility checks; (2) sort of various numbers of items of varying lengths and varying-length keys; (3) printout of long lists from magnetic tape, with some editing; (4) input/output; (5) internal operations.

Results of benchmark runs are presented for all five programs on the Bull G–30, IBM–1401, ICT–1301, RCA–301, and SAAB D–21 computers. The figure of merit used in rating each system was the division of speed by price to measure the "price per unit of work," assuming that the computers are always fully utilized, with any overcapacity sold, so that all computer hours represent the same value. (JLW)

Categories: 11; 3.3

Key words: Application benchmarks; Bull G–30; computer systems; IBM–1401; ICT–1301; RCA–301; SAAB D–21; Sweden.

24. Drummond, Mansford E., Jr., *Evaluation and Measurement Techniques for Digital Computer Systems*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1973, 338 pp., bibliography.

Several chapters (or parts of chapters) of this monograph are of interest to benchmarking. Chapter 4, "Systems Analysis Techniques," discusses some of the simpler methods useful in determining throughput and/or response time attributes. The concept of "relative systems throughput" is introduced and defined as follows: "Relative systems throughput is a relative estimate of the performance of a computing system when compared to some base computing system. It is the ratio of the time of computation *of a given load* on the base system to the time of computation of *that same given load* on the comparing system."

The base system is defined as the hardware configuration, the system programs, the application programs, and the data processed by these programs. *Formula timing* is a technique used to provide timing estimates for performance of certain applications. *Profile conversion* provides a way of estimating boundaries of expected job or system performance for a range of processing or configuration options. The third analytic technique presented in this chapter is the synthetic model or an estimate of system time which is a combination of subsystem times with assumed overlap factors. Subsystem times, in turn, include device times which are structured from data set activity. The synthetic models can be structured to accommodate sensitivities such as interjob set up time, media switching time, operator messages and response times. An enormous amount of detail is required for such analysis; the author presents several algorithms for developing synthetic models.

The author discusses benchmarks in a chapter devoted to data acquisition. He notes the gross data requirements in using application programs in the evaluation process, and suggests a way of overcoming these requirements by the "use of an application program known as a 'benchmark.' A benchmark is some particular programmed procedure with some amount of associated data which has been chosen in such a way as to impart meaning to the originator of the benchmark. . . A critical point to keep in mind is that there is no requirement for a benchmark to represent any application or expected loading on a system; its only requirement is that it have meaning to the originator."

The application or set of applications that use a major portion of the computer system's time should be used to structure the benchmark which "has the requirement that it meets in its principal attributes the applications which it is purporting to either represent or span. Notice that the artificial program does not necessarily have the requirement of representing the calculating procedure of any other particular program."

The artificial program can also be used to determine sensitivity of system parameters. A figure displays a synthetic job structure, and there is a brief discussion on how to put together a synthetic job. This particular section concludes with a brief paragraph on "an alternative to obtaining actual applications, sample applications, benchmarks, or what have you, is to measure attributes of actual applications as they are being processed in their normal environment." Data on the incidence of use of particular applications may then be used to predict performance under varying parameters or configurations.

Chapter 9 discusses and illustrates methods of presenting the results of an evaluation process to a nontechnical audience. (JLW)

Category: 1.2

Key words: Application benchmarks; artificial benchmarks; benchmarks; formula timing; job profile; measurement parameters; synthetic job structure; synthetic system model; system profile.

25. Drummond, M. E., Jr., A Perspective on System Performance Evaluation, *IBM Systems Journal*, 8:4 (1969) pp. 252–263, 6 refs. (6430096)

The author presents a historical summary of the growing complexities of performance evaluation. Three primary types of evaluation are noted: classification; comparative evaluation; and absolute evaluation. "An absolute evaluation is one that produces time estimates for the performance of a required function or operation." Absolute evaluation may be a step in a comparative evaluation or it may be an end in itself. In structuring an absolute evaluation, not only the appropriate techniques must be used, but the choice of data to be used in the evaluation of a specific application requires that the data in some way represent that application.

In some situations, data requirements may be enormous. To overcome this, use of an application known as a benchmark is suggested. The author defines a benchmark as "a particular programmed procedure with some associated data chosen in such a way as to impart meaning to the originator of the benchmark." For the commercial data processing community the classical benchmark problem is payroll gross-to-net calculation, while for the scientific it is matrix inversion.

The synthetic program is noted as an alternative to the actual application or benchmark. While the synthetic program must match the principal attributes of the applications it purports to represent or span, it need not necessarily produce the results of any particular program. Since the parameters of synthetic programs may be varied, sensitive functions may be studied, and results of the evaluation can provide tables or graphs for extrapolation of performance for particular applications. (JLW)

Category: 1.2

Key words: Application benchmarks; computer performance evaluation; historical summary; synthetic benchmarks.

26. Ferrari, Domenico, Workload Characterization and Selection in Computer Performance Measurement, *Computer*, 5:4 (July/August 1972) pp. 18–24, 22 refs. (6430202)

The article discusses workload characterization as one of the most general, central, and difficult problems in computer system performance evaluation. The main objective in selecting a workload is that it be fully representative of the real workload. Analysis of current selection techniques indicates that they are generally primitive in respect to workload representation.

Yet the evaluation of system performance requires that the drive workload be representative of the real workload. Three basic types of techniques for obtaining the drive workload are analyzed, and the relative merits of each are discussed. These three techniques are: (1) *natural*—use of the real workload directly without manipulation; (2) *artificial*—design and implementation of a workload independent of the real workload, and (3) *hybrid*—assembly of a workload from parts of the real workload.

The author concludes that the major differences among these techniques are those between the

natural on the one hand, and the artificial and hybrid on the other. Natural workloads require longer measurement sessions, usually in their own environment and user community. A natural workload does not require any preparation, since it is already in existence, and measurements performed on a system driven by a natural workload cause very little degradation in performance due to interference of measurement tasks. Shorter measurement sessions are one of the basic reasons for use of non-natural techniques. Another advantage is the capability to run an artificial or hybrid workload on a compatible system at any location. On the other hand, non-natural workloads take time to develop, and are therefore more costly than natural workloads. Also a system driven by a non-natural workload cannot serve its real users at the same time.

A comparison of hybrid with artificial workloads indicates that development of the former is less expensive. "However, hybrid techniques require that the real workload be not only measurable (which is always necessary for the validation of a non-natural workload), but also available for recording, and this may be difficult in some installations. Moreover, . . . artificial jobs (especially of the synthetic type) are more flexible and, if carefully adjusted and validated, can provide better representativeness with the same session duration or equal representativeness with a shorter duration." (JLW)

Categories: 3.0; 2.2

Key words: Artificial benchmarks; hybrid benchmarks; natural benchmarks; workload description; workload representation.

27. Forest Service, U.S. Department of Agriculture, Computer Job Load Analysis. Volume I. General Description Job Load Analysis System; Volume II. System and Program Documentation, Washington, D.C., 1970, Prepared by Computer Learning & Systems Corp. (PB 201–120, PB 201–121; 6430272; 6430273)

    The two-volume report presents the results of a study prepared by the Computer Learning and Systems Corporation for the purpose of defining Forest Service computing requirements and of developing and implementing a plan for meeting these requirements.

    A mechanized process was used to define and analyze the computer workload. This process included the following major features: (1) a complete inventory of all computer programs in use and under development; (2) an inventory of all existing and proposed applications; (3) a computerized data base describing (1) and (2) above; (4) provision of capability to extract and summarize information from the data base; and (5) development of an automated method for selection of simple programs representative of the total workload.

    The representative programs were extracted from the data base by a program called CIDE (Case Input Data Extracts) whose output provided the required workload description for input into the CASE (Computer-Aided System Evaluation) simulation system. CASE was used to simulate runs of defined workloads on alternative equipment configurations and system software.

    All of the computer programs used in the workload analysis are described by detailed flowcharts and narrative descriptions of major steps in the programs. In addition to this, five appendices to the reports present the forms used for description of programs and applications together with instructions for use of the forms, instructions for the validation process and preparation of related forms, instructions for coding data for keypunch forms, and a description of the use made of CASE in the study. The latter appendix also presents sample pages from several reports produced by the CASE simulation system. (JLW)

Categories: 11; 2.2

Key words: Automated workload definition; workload analysis; workload definition; workload representation.

28. Good, John and Bruce A. M. Moon, Evaluating Computers for the New Zealand Universities, *Datamation*, 18:11 (November 1972) pp. 96–99. (6430164)

    The article reports performance data on a batch of 34 FORTRAN programs which were used to measure CPU power needs and performances in the procurement of new equipment for New Zealand's universities. The Burroughs 6700 was chosen for the five largest institutions with adjacent communications links to serve the smaller institutions.

    The benchmark consisted of programs designed to investigate FORTRAN language characteristics and particular features of the hardware being tested. Collectively the benchmark was a model of one university's job stream. Most programs were "live" ones written by university users as part of their work, and a realistic proprotion of the programs contained known bugs. By running the benchmark as a datum on an IBM–360/44 and timing the performance with an IBM–2989 Basic Counter Unit, the characteristics of the benchmark were known: it was compiler-intensive in comparison with the job stream; it was I/O intensive; the benchmark was 24.6 percent CPU active while the job stream it modelled was 35 percent CPU active.

The major purpose of the exercise was to provide a base level of performance against which to measure the throughput of each candidate system. A model of a FORTRAN job stream was admittedly narrow, but the only practical benchmark for the New Zealand universities. The heavy investment in FORTRAN and lack of expertise in other languages made the FORTRAN requirement fundamental to university purposes.

In rating machine performance in the execution of the benchmarks, two parameters were of interest: (1) disc-to-disc (sometimes tape-to-tape) elapsed time; and (2) CPU time required to execute the benchmarks, minus overhead. The overhead costs for the 360/44 were not deducted because this overhead is minimal. The CPU time thus calculated was considered to be a measure of the raw power of the compiler/CPU/core speed combination, while the disc-to-disc elapsed time recorded the amount of this power which was made available by the operating system/backing store/core size combination. These figures are tabulated for the nineteen hardware configurations tested. Another table shows the powers of these same configurations in terms of throughput of the 360/44 CPU rounded to a scale of 19. According to this rating, the UNIVAC–1108 and the CDC–6400 were found to have a throughput capacity 19 times that of the original 360/44 configuration. (JLW)

Categories: 3.3; 11

Key words: Benchmark characteristics; benchmark run analysis; computer system; FORTRAN job stream; IBM 360/44; job stream representation; New Zealand; university computing center.

29. Gosden, J. A. and R. L. Sisson, Standardized Comparisons of Computer Performance, in Popplewell, C. M. (Ed.), *Information Processing 1962,* (North-Holland Publishing Co., Amsterdam), 1963, pp. 57–61. (6430189)

    The paper describes in detail the method used by Auerbach to measure the performance of a computer system on a file updating task. This is one of several benchmarks used in Auerbach's *Standard EDP Reports* to present comparative performance characteristics for several configurations of most computer systems. The other benchmark tasks used for this purpose are matrix inversion and sorting.

    Performance of each computer system is measured at three levels: basic operations; operation of individual devices; and operation of particular configurations. "A detailed and comprehensive timing procedure" is deemed to be the basic requirement for accurate and consistent results. From these basic figures performance measures can then be generated for a large number of different cases. Factors and parameters which need to be considered in developing a "typical" task for a typical configuration are spelled out in considerable detail. (JLW)

Category: 3.3

Key words: Standard benchmarks.

30. Greenbaum, Howard J., A Simulator of Multiple Interactive Users To Drive a Time-Shared Computer System, Massachusetts Inst. of Tech., Cambridge, Mass., Project MAC Report, MAC–TR–58 (Thesis), January 1969, 181 pp., 5 refs. (6430205)

    The thesis describes the use of a PDP–8 to simulate up to 12 users typing from prepared scripts to provide a terminal load generator for performance measurement purposes. The Simulator can accommodate users at either the IBM–7094 Compatible Time-Shared System (CTSS) or the GE 645 Multiplexed Information and Computing Service (MULTICS) at M.I.T.; it is flexible enough to simulate users on most time-sharing systems which use Bell 103H-compatible data sets. Initial implementation of the Simulator System was for support of a maximum of four users.

    The Simulator recognizes and verifies responses transmitted to it by the system. In addition, the Simulator emulates the user's think time between lines of input. Actions of the simulated users are controlled by "Scripts," prepared in advance, which contain information necessary to perform the checking and verification of responses as well as the emulation of think time. The scripts are encoded in a special language and converted to magnetic tape files which may then contain a library of scripts. A SCRIPT LOADER program moves designed script files to the PDP–8 disk unit; the program also does some editing for proper script formatting. The simulation process is controlled by the scripts on the disk unit and progresses until scripts of all users have been exhausted or until a "fatal" error occurs. All messages transmitted during the simulation are recorded on magnetic tape which may be listed and analyzed by one of several other programs which comprise the Simulator System. The largest problem facing a prospective user of the system is identified as the design of the scripts so that these do, indeed, represent a normal load to the system.

    Seven appendices are provided with the report; the first of these is entitled, "Users Handbook to Simulator System Operations." (JLW)

Category: 5.2

Key words: Computer systems; PDP–8; terminal load generator; user simulation; user scripts; workload generator.

31. Hahn, S. G. and E. V. Hankam, Kernel Analysis of Elliptic Partial Differential Equations, *IBM Systems Journal*, 5:4 (1966) pp. 248–270, 6 refs. (6430177)

The procedure used here in evaluating computer performance for a large class of problems leads to the development of a set of formulas which have problem-dependent and machine-dependent parameters which yield a measure of speed when computer time specifications are substituted. These formulas comprise that particular set of instructions in the program which is repeated over and over again. This set is defined as the "kernel" and includes also the I/O operations necessary for transferring data into main storage.

The program is for solving partial differential equations by the extrapolated Liebmann method. Kernel programs are given for the IBM–360 and 7094 computers for comparative performance evaluation of the two systems. Timing formulas desired include time needed for calculation as well as for read/write operations. The general effect of several I/O devices on the timing formulas is also discussed.

In summarizing their findings the authors noted several interesting observations. Considerable insight was gained into the economic use of computers when large blocks of data can be transmitted at the same time. In this particular method of solution of the problem involved, use of a computer with very large main storage with relatively slow access appears to be preferable to other I/O devices. Seek time of disks would considerably slow down data transmission.

An appendix provides the two kernel programs used—for the IBM–360 in both short and long arithmetic, and in single as well as double precision for the IBM–7094. (JLW)

Categories: 3.1; 1.2

Key words: Comparative computer performance measurement; computer systems; IBM–360; IBM–7094; kernel programs; partial differential equations.

32. Hatfield, D. J. and J. Gerald, Program Restructuring for Virtual Memory, *IBM Systems Journal*, 10:3 (1971) pp. 168–192, 14 refs. (6430180)

Program reference patterns can have a more profound effect on paging performance in a virtual memory system than page replacement algorithms. This paper describes experimental techniques that can significantly reduce paging exceptions in existing, frequently executed programs. Automated procedures reorder relocatable program sectors, and computer displays of memory usage facilitate further optimization of program structure.

Experience to date has shown that the display of a virtual memory use pattern is a good diagnostic tool. The automatic sector reordering technique brings noticeable improvements in paging performance where there is room for improvement, reducing the necessary working space (for a given number of exceptions) by as much as one-third to one-half. Optimal page sizes for programs appear to depend on (besides physical I/O timings) complicated patterns in the use of virtual memory, about which little is known.

In consideration of possible extensions of these techniques, it was found that program sector duplication may be desirable. A comparison of the number of communications among sectors to the number of communications within sectors can be used to measure the goodness of the modularity of the program. Data areas in a program may be defined as sectors and their communications examined. This intelligence could be built into optimizing compilers for improved program performance in a relocation environment. (Modified author).

Category: 1.3

Key words: Optimizing compilers; paging exceptions; program optimization; program reference patterns; program relocation; virtual memory systems.

33. Hicks, Harry T., Jr., The Air Force COBOL Compiler Validation System, *Datamation*, 15:8 (August 1969) pp. 73–74, 76–77, 81. (6430208) (6430208)

The article describes the design criteria of the Air Force COBOL validation system, the internal structure of the system, and its use in measuring a given compiler against Standard COBOL.

Design criteria included many very important to good benchmarks: language independence in methods for specifying the contents of programs and for generating the programs themselves; adaptability to many computers and to changes and variations in the COBOL language; modularity, with easy transfer of statements between modules; and a capability for thoroughly testing a given compiler. Efficiency was not a design criterion, because it was felt that any increase in efficiency would be at a sacrifice of the system's machine independence.

Several uses other than compiler validation, were noted for the system. These include: aid in debugging and adapting existing programs to compiler modifications; identifying differences between a present compiler and the compiler to which conversion is being planned; and a rapid way of developing familiarity with an operating system as well as with the compiler. (JLW)

Category: 3.6

Key words: Benchmarks; COBOL validation; Department of the Air Force; design criteria; test program generators.

34. Hillegass, John R., Standardized Benchmark Problems Measure Computer Performance, *Computers and Automation*, 15:1 (January 1966) pp. 16–19. (6430203)

    The article describes techniques used in developing standardized benchmark problems which have been used to produce relative measures of computer system performance. These were published in a rigidly standardized format for more than 60 different computer systems in the *Auerbach Standard EDP Reports* for several problems: matrix inversion, sorting, and file updating.

    The techniques are illustrated in a file updating benchmark designed to handle 10,000 master file records. Performance of a number of systems (then) available commercially in the standardized file updating problem is compared graphically. The graph is designed to show trends in internal processing power of computer systems. It would appear from this information that, dollar for dollar, third-generation computers (those whose first delivery came after January 1, 1964) are twice as fast as their predecessors.

    In discussing the validity of this technique, some interesting observations are made. Among these is the observation that in the case of benchmark problems which measure sorting and matrix inversion speeds, timing data derived from the techniques used by Auerbach usually agree to within 10 percent of timing data provided by computer manufacturers for their standard routines to do the same jobs. A further observation made is that in the few cases where it was practical to actually run the Auerbach benchmark problems on different computers, there has been quite close agreement between estimated and actual processing time data. (JLW)

Category: 3.0

Key words: Application benchmarks; standard benchmarks.

35. Holdsworth, D., G. W. Robinson and M. Wells, A Multi-Terminal Benchmark, *Software—Practice and Experience*, 3:1 (January–March 1973) pp. 43–59, 7 refs. (6430171)

    The benchmark described in this article was constructed in order to compare two different versions of the same operating system (GEORGE 3), particularly those aspects of the system which affect a user on a console of a multi-terminal system.

    By observation of an existing system (the ELDON 2), a profile of typical users and user activities was developed and a set of commands was generated to simulate user activity at terminals.

    A parameter deemed critical to performance of a multi-terminal system is access to the file store. Much effort was devoted to a determination of the actual distribution of file sizes, and the distribution of files within the "seek-space" of the physical devices and within the index. Other parameters required were detailed timings of actual messages passing between users and the system. For the benchmark actual sequences of messages were used and all of the files to which these messages referred were incorporated into the file store.

    The appendix summarizes the findings from the benchmark run and attempts to define a distribution of "average" users and their activities at the computer system terminals. (JLW)

Category: 3.2

Key words: Artificial benchmarks; average user; computer systems; ELDON 2; England; GEORGE 3; operating systems; user profiles.

36. Ihrer, Fred C., Computer Performance Projected Through Simulation, *Computers and Automation*, 16:4 (April 1967) pp. 22, 25–27. (6430210)

    A discussion of simulation, its methodogy, and uses, oriented around SCERT. The author makes an interesting observation:

    The evaluation of computer hardware can no longer be realistically accomplished by comparing instruction execution times, peripheral speeds, memory sizes and other unrelated hardware performance data. Other approaches to performance prediction, such as benchmark processing and instruction mix statistical data, are inadequate for the evaluation of computers performing in a multiprogramming mode of operation. (Excerpt)

Category: 1.3

Key words: Computer performance evaluation; simulation.

37. Ishida, Haruhisa and Nobumasa Takahashi, Job Statistics at a 2000-User University Computer Center, *Performance Evaluation Review*, 2:2 (June 1973) pp. 2–13. (6430192)

    The Computer Center at the University of Tokyo (Todai) is one of 7 large university centers serving researchers throughout Japan; it processes the 120,000 jobs annually submitted by 2,000 academic users in various research institutions.

A FORTRAN program is used to analyze the job characteristics at the Todai Center. The article presents detailed data (in graphical form) on jobs submitted for the school year April 1969 to March 1970. Two reels of magnetic tape were necessary to record account data for 123,705 jobs. Data on the following parameters are reported for each job: use time, CPU time; core memory actually used; number of input cards; number of pages printed; number of cards punched; termination status; turnaround time; CPU busy time; and job size.

A table presents comparative data for 1966, prior to a scaling upwards of the system. An 8-fold increase in CPU speed of the new system (HITAC–5020E with 2 HITAC–5020 processors) resulted in only twice the former capacity because of the four-fold increase in job size. Plans are being made to upgrade the present system to a HITAC–8800 by the fall of 1972. (JLW)

Category: 2.2

Key words: Accounting data; computer systems; HITAC–5020; Japan; job statistics; Tokyo University; University computing center.

38. James, D. L., A Remote Terminal Emulator for Loading and Performance Measurement of On-Line Systems, Mitre Corp., Bedford, Mass., Rept. No. M72–83, March 1972, 32 pp. (Documentation of paper presented at SHARE XXXVIII meeting in San Francisco, March 1972) (6430206)

The paper (presented at SHARE XXXVIII) reported on experimental work at the Mitre Corporation on a remote terminal emulator system for use in measuring performance of large-scale, multi-terminal, online computer systems. The emulator consists of hardware and software separate from that of the system under test. In a live test demonstration for a given procurement, a suggested configuration would consist of the proposed central processor with its associated peripherals and the emulator attached to the communications interface, instead of to the remote terminals and part or all of the communications portion of the system.

The general characteristics of the remote terminal system are described in detail. This is followed by a discussion of an experimental model of a remote-terminal emulator which has been implemented and tested against an IBM–360/50 and an IBM–370/155. Implementation of the experimental model was on an Interdata Model 3 under VENUS, a microprogrammed operating system which can accommodate up to 11 simultaneous user jobs. The emulator operates as one job under VENUS. Sample scenarios are presented to illustrate and describe operation of the experimental model.

Analysis of a relatively small amount of data produced by the experimental model indicates that possibly sixteen teletypes might be supported or emulated by the model. However, considerable extrapolation is necessary in order to estimate total capacity of the emulator.

Current design efforts at Mitre have reached the initial implementation stage on a Data General NOVA–800 minicomputer. This design verification model will be used as a basis for specifications for purchase of a prototype system for delivery to the Air Force for evaluation under live test demonstration conditions. One of the most important features in the current design work is the modular concept which, theoretically, can emulate any number of terminals. An appendix provides and illustrates a sampling of scenario instruction types which are included in the current design. (Material in the Appendix was not presented at the SHARE meeting.) (JLW)

Category: 5.2

Key words: Computer systems; emulator; IBM–360/50; IBM–370/155; Interdata model 3; job loading; live test demonstration; NOVA–800; operating systems; remote terminal emulator; user simulation; user scenarios; VENUS; workload definition.

39. Johnson, R. R., Needed: A Measure for Measure, *Datamation*, 16:17 (December 15, 1970) pp. 22–30, 27 refs. (6430110)

Answers to typical questions posed today about computer capabilities can be provided by only one way and that is by programming the problem of interest and running it. This requires exhaustive benchmarks which are expensive, time consuming, and "provide little real knowledge that's extrapolable to other computers or different problems." There is a basic need for a "Theory of Computer Performance." Development of such a theory requires two kinds of information: information on the structure of programs during execution; and information on the response of computers to these program structures.

Lacking that, four parameters are in general use today for analyzing computer performance—capacity, throughput, run speed, and ease of use. The author defines the last parameter as that which describes the effort and time required to prepare a problem for solution by a computer plus the effort required to operate the computer and its peripherals.

These four performance parameters are used in four different ways which form four classes of measures.

Class I—Design Measures—needed by designers to evaluate trade-offs between alternative struc-

tures; generally do not exist; measures needed covering all four performance parameters.

Class II—Purchasing—Sales Measures—quantitative measures needed by customers and salesmen for comparative selection and competitive sales; have been developed and used widely; Adams charts, instruction mixes, and benchmarks represent three levels of escalating effort to relate run speed to throughput; ease of use considered a feature which might be available from certain vendors; capacity difficult to predict.

Class III—Configuration Measures—needed by users in deciding how many of what are actually required to do their job, once a given system has been decided on.

Class IV—Operating Measures—needed by computer center managers who need to optimize their systems' performance against known workloads; hardware and software monitors are being used; progress is being made in developing performance sensitivity criteria.

The author then discusses briefly some of the available measurement techniques and states that the only reliable and accurate one is the benchmark which he defines as a complete program selected from those that are to be run in the actual application. "As long as benchmarks are seen in the large and small mixes expected in the actual operation, a good indication can be obtained of both the large and small problem behavior of that system." However, the limitation of the benchmark is that its primary usefulness is as a Class II measure for marketing purposes. In addition to this, a large effort is required to program benchmarks and to prepare realistic data. "Benchmark performance can be misleading in terms of the capacity of a system unless complete benchmarks are selected to place a capacity load on that system. Performance predictions for other problems not benchmarked can be risky extrapolations."

The article concludes with discussions on: (1) techniques using computers for studying system performance simulation, monitoring, and analysis; (2) three forms of analysis used in the study of computer throughput—statistical analysis, graphical analysis, and algorithmic analysis. In the course of this, an excellent review of the literature is provided. (JLW)

Category: 1.2

Key words: Benchmarks; computer performance analysis; computer performance measurement; literature review; measurement parameters; simulation; system monitoring.

40. Joslin, Edward O., Application Benchmarks: The Key to Meaningful Computer Evaluations, in Association for Computing Machinery, *Proceedings of the National Conference*, 1965, pp. 27–37. (6430162)

The author considers the following question as the most important in the computer evaluation process: "How long will it take this system to process my workload?" He then states that application benchmarks are the best available timing techniques, and that the method used to set up application benchmarks will determine how truly representative the benchmarks will be. The article deals with the second of these factors.

An application benchmark is defined as: "A routine to be run on several different computer configurations to obtain comparative throughput performance figures regarding the abilities of the various configurations to handle the specific application." Three key characteristics emerge from this definition: (1) the benchmark is to be run on the configuration, not handtimed; (2) throughput, not processor time, is important; and (3) the run is aimed at a specific application, not just as a "general goodness" benchmark.

The method for preparing truly representative application benchmarks is then described. Total representativeness demands that each benchmark be representative of a class problems as to type of processing performed by its class, that it be representative of time required for processing, and of memory and peripheral equipment requirements for that class. Illustrative classes presented include: FORTRAN coded engineering problems; FORTRAN coded mathematical problems; and COBOL coded business problems.

The illustration is continued through the steps required for the development of an application benchmark:

—determination of timing requirements for each class of problems;
—determination of equipment requirements for each class of problems;
—determination of correct ratios between compile and compute times for each problem.

From an appropriate mix of the above factors, the full month's workload is developed. Next are the problems of reducing this to a "properly proportioned miniaturized version" of the workload, and of assigning proper queuing priorities to the problems used. Eight figures are presented to illustrate the mixing and matching that is essential to the development of a truly representative application benchmark. (JLW)

Category: 3.3

Key words: Application benchmarks; compile/compute ratios; timing requirements; workload construction; workload representation.

41. Joslin, E. O., Cost-Value Technique for Evaluation of Computer System Proposals, in AFIPS, *Proceedings of the Spring Joint Computer Conference,* 1964, pp. 367–381, 17 refs. (6430174)

The cost-value technique proposed here involves two major changes to existing evaluation techniques which could bring about a more understandable and realistic selection of computer equipment.

The first change is in methodology. The cost-value technique attempts to consider all items of value to a computer system, but to consider them only once and in the environment in which they belong. The categories scored are total system cost and "extras" which are defined as features like expansion potential, vendor support, or similar characteristics which are part of total system cost, but differentiating between vendors.

The second change is in scoring technique. The cost-value technique uses dollars rather than weighted points as the basis of comparison. This provides a more natural basis for comparison. It eliminates the need for "tradeoffs" and gives management deeper understanding of the total selection process.

Since the technique is a dynamic one, a number of improvements might be made. The use of debits, as well as of credits, could be made immediately and might make the technique a little more natural. Two other improvements, however, would require more work and understanding. These are quality determination and time dependent cost-value assignments, both of which are discussed briefly. (Modified author)

Category: 1.3

Key words: Cost-value; proposal evaluation; scoring.

42. Joslin, Edward O., Describing Workload for Acquiring ADP Equipment and Software, *Computers and Automation,* 18:6 (June 1, 1969) pp. 36–40. (6430225)

The author presents a detailed discussion on how to obtain the mix of representative programs to be used for benchmarking purposes. This mix of representative benchmark programs is the first necessary requirement for describing workload. The second requirement is a description of system growth in terms of a series of expected workload levels.

In selecting programs for the representative mix several considerations are important:

(1) the programs should be written in a standard, higher level language;
(2) the mix should be small enough to be processed during a single half-day benchmark demonstration;
(3) the programs are selected not to prove the worst case situation, but rather to test and demonstrate timing and capability for the normal situation.

Should it be necessary to assume and demonstrate capability to handle worst case situations, benchmark programs selected for that purpose should not be included in the representative mix. They should be treated separately as capability benchmarks.

A method for deriving a representative set of programs is discussed and illustrated in detail. Calculation of extension factors and growth projections is also discussed and illustrated. (JLW)

Category: 2.2

Key words: Application benchmarks; growth projections; job mix; system life projections; task mix; workload description.

43. Joslin, Edward O., Techniques of Selecting EDP Equipment, *Data Management,* 8:2 (February 1970) pp. 28–30. (6430143)

A brief discussion of the importance of using proven procedures in four areas of EDP equipment selection: preparation of specifications for competitive selection; workload representation; evaluation; and costing.

Workload description is part of the specifications, and important enough to merit its own place of importance in the procurement process. Description of the benchmark should consist of benchmark programs to be run on the bid system(s) for a determination of total time. For the benchmarks to be truly representative of the workload, it is essential that the programs selected be representative of the types of tasks to be processed, the time requirements, equipment and storage used, the language used, all in the required order or sequence. The extension factor must be determined; this can be described as the "total monthly time to perform the task set divided by the throughput time to run the representative program."

Other considerations relevant to selection of representative programs include:

(1) They must represent current and past workload data.
(2) The growth factor must consider current size and planned growth of the facility.
(3) They must be selected with an eye to future fiscal policies.
(4) They must reflect current and future manpower.
(5) They should cover some fixed system life period, normally four to six years.

This involves a large expenditure of labor which, however, pays large dividends in verification of

vendors' timing claims. A "single click-click with a stop watch" signals the end of a run and obviates any discussion, for: "It's put up or shut-up time, and everyone knows it and there is little point in arguing with the stop watch." (JLW)

Category: 2.2

Key words: Application benchmarks; computer selection procedures; system life projections; workload description; workload representation.

44. Joslin, Edward O. and John J. Aiken, The Validity of Basing Computer Selection on Benchmark Results, *Computers and Automation* 15:1 (January 1966) pp. 22–23. (6430196)

This elementary discussion of the validity of basing computer selections on the results of benchmark runs presents a "commonly accepted definition of a benchmark [as] a routine used to determine the speed performance of a computer system." If the routine used for the benchmark is truly representative of a workload, then the results of the benchmark run provide an excellent basis for selection.

An Air Force procurement in which actual problems were used as benchmarks, and some of the findings uncovered during the procurement are discussed. Compilation and execution times for four benchmark problems on four computer systems are tabulated. The relative ranking of each system varies from benchmark to benchmark, and this finding is used to make "clear that the measuring device (benchmark) used to determine the capability of a system *does* make a difference."

Examples from the tabular data are then used to illustrate the critical necessity of selecting benchmark problems to properly reflect the total workload which is to be processed. (JLW)

Category: 9

Key words: Application benchmarks; workload representation.

45. Karush, Arnold D., Benchmark Analysis of Time-Sharing Systems, System Development Corp., Santa Monica, Calif., Rept. No. SDC–SP–3347, June 1969, 40 pp. (AD–689 781)

The paper discusses the use of benchmarks to measure and understand the behavior of a general purpose time-shared system. This application of benchmarks is unusual in that: (1) there is no published literature on the application of benchmarks to time-shared systems; and (2) the benchmarks measure system functions rather than job tasks. After discussing the benchmark design concept, the paper describes the benchmark programs that were produced to measure System Development Corporation's **ADEPT** time-sharing system. Three types of calibrations were performed on ADEPT, together with numerous experiments. These are described together with some of the results. The paper concludes with a discussion of the problems and potential uses of the benchmark technique. (JLW)

Category: 3.2

Key words: ADEPT–50; Benchmarks; computer performance measurement; computer systems; experimentation; multiprogrammed computer systems.

46. Karush, Arnold D., Evaluating Time-sharing Systems Using the Benchmark Method, *Data Processing Magazine*, 12:5 (May 1970) pp 42–44, 2 refs. (6430104)

A summary article which discusses the benchmark programs which were used to measure the behavior of System Development Corporation's ADEPT–50 time-sharing system.

The techniques used focused on measuring the effects of functions basic to system operation, rather than on measuring system performance of predefined tasks. The functional variables included are: swap activity; compute activity; interactive activity; I/O activity; page activity; and resource allocation. Throughput (amount of work processed by the computer) and response time (delay between a user's request and system reply) served as metrics of behavior for these functional variables. Each of the seven benchmark programs developed provided one or more of the stimuli; thus all programs affected all of the functional variables, both individually and when run in reasonable combinations.

System performance was measured by the benchmarks in three different environments: (1) stand-alone; (2) benchmark; and (3) real-world. In the stand-alone environment only one program was run at a given time, thus testing system throughput and response time in a batch mode. The benchmark environment provided a measure of a "typical" set of demands upon the system. This was achieved by considering each benchmark program as a user and by running all seven programs simultaneously. Measures obtained from this experiment roughly paralleled those from the real-world environment. This environment simulated the behavior of a time-sharing system operating near its rated capacity. By considering one of the benchmark programs as a user with a constant and known demand for service, and running the program when the system has an almost full complement of real users, various metrics can be developed. Some of these include a measure for degradation in response time and throughput under varying user loads; load variances; and what scheduling

algorithms actually schedule. The technique may also be used to tune a system to the needs of its user population. (JLW)

Category: 3.2

Key words: ADEPT–50; benchmarks; compute activity; computer performance measurement; computer systems; interactive activity; I/O activity; multiprogrammed computer systems; page activity; resource allocation; response time; swap activity; throughput.

47. Karush, Arnold D., Two Approaches for Measuring the Performances of Time-Sharing Systems, *Software Age*, 4:3 (March 1970) pp. 10–13, 18 refs; Part Two. Stimulus Approach to Time-Sharing Measurement, *ibid.*, 4:4 (April 1970) pp. 26, 27, 40, 8 refs; Conclusion. A Comparison of Analytic and Stimulus Approach to Time-Sharing System Measurement, *ibid.*, 4:5 (May 1970) pp. 13, 14, 3 refs. (6430103)

    The two approaches for performance measurement which the author describes are: (1) the "stimulus" approach in which the system is considered as a black box to which a controlled set of stimuli is applied in order to activate the system's functions and then observing the results; and (2) the "analytic" approach in which probes are inserted into the black box in order to record any level of the system's behavior. Both approaches have been used to measure SDC's ADEPT time-sharing system.

    The author places benchmarks into the first category which is less costly to develop and generally requires less sophistication in the implementor. "The programming of benchmark programs is also less costly than the programming of instrumentation, measurement and recording routines. . . . Personnel with little experience can produce the benchmark programs. Testing can be done under time-sharing. Errors affect no one else."

    In benchmarking for the ADEPT system, 6 functional variables were selected; compute, interactive, high speed I/O, swapping, paging, and resource allocation. Seven benchmark programs, each incorporating stimuli of selected functional variables were written and run simultaneously in a time-sharing mode, thus simulating a "typical user population." The technique was also used to measure the effects of variable sizes of quantum-time upon different user demands as represented by the benchmarks.

    In discussing areas for further development, the author suggests the following relevant to benchmarks:

    Stimulus Approach:

    "(1) The conditions under which this approach is cost-effective must be defined. Although the analytic approach provides much more information, benchmark programs can still fill an important role due to their lower cost and the immediate utility of the information.

    (2) Standardized measures for describing and ranking the performance of time-sharing systems should be developed. If these measures could be expressed in terms of throughput and response time, perhaps standardized benchmark programs could be specified for inter-system comparison.

    (3) . . . . . .

    (4) The design of the benchmark programs should be refined so that a minimum system load and terminal time need be required to extract a maximum amount of information." (JLW)

Categories: 3.0; 1.2

Key words: ADEPT–50; benchmarks; compute activity; computer performance measurement; computer systems; experimentation; interactive activity; I/O activity; multiprogrammed computer systems; page activity; resource allocation; response time; swap activity; throughput; user simulation.

48. Kerner, H. and K. Kuemmerle, A Workload Model and Measures for Computer Performance Evaluation, George C. Marshall Space Flight Center, Alabama, NASA TN D–6873, October 1972, 26 pp., 8 refs. (6430151)

    A generalized workload definition is presented which constructs measurable workloads of unit size from workload elements, called elementary processes. An elementary process makes almost exclusive use of one of the processors. CPU, I/O controller, etc.. and is measured by the cost of its execution. Various kinds of user programs can be simulated by quantitative composition of elementary processes into a type. The character of the type is defined by the weights of its elementary processes and its structure by the amount and sequence of transitions between its elementary processes. A set of types is batched to a mix. Mixes of identical cost are considered as equivalent amounts of workload. These formalized descriptions of workloads allow investigators to compare the results of different studies quantitatively. Since workloads of different composition are assigned a unit of cost. these descriptions enable determination of cost effectiveness of different workloads on a machine. Subsequently performance parameters such as throughput rate, gain factor, internal and external delay factors are defined and used to demonstrate the effects of various workload attributes on the performance of a selected large scale computer system. (IBM)

Category: 2.2

Key words: Cost effectiveness; computer performance measurement; workload definition; workload specification.

49. Kernighan, B. W. and P. A. Hamilton, Synthetically Generated Performance Test Loads, in Association for Computing Machinery, *Proceedings of the SIGME Symposium*, February 1973, pp. 121–126, 6 refs. (6430148)

The paper describes the design of and experience with an automated benchmark-generation facility that involves two components. The first component is a simple, highly parameterized synthetic job or "program which uses precisely specified amounts of computing resources, but which does no 'useful' work." Any set of resource utilization parameters for the synthetic jobs specifies an executable program of known characteristics which can be used to model the behavior of another program. The second component of the facility is a generator program that converts a job stream specification into a complete, ready-to-run set of synthetic jobs that can then be used to exercise the system in a controlled and reproducible manner.

Many advantages are offered by this approach: (1) there is no need to find real jobs with the right properties; (2) creation of large test data bases is not necessary; (3) tests can be easily scaled up to test large systems; (4) incremental changes are easily made, as necessary; (5) transferability is simple, since only two programs are involved; (6) test generation and execution is entirely self-contained; and (7) all the benefits accruing from an automated procedure—not the least of which is the relative freedom from human error.

The environment for the experiments reported is a dual-processor Honeywell 6070 GECOS III with 256K of 36-bit words of storage. 15 million words of fixed-head storage, 75 million words of moving-head disk storage, and two DN–355 communications processors interfacing with "about half a dozen remote computers." The batch computing aspect, however, is most important in resource utilization, therefore a batch environment is implicit in the paper. However, the approach described is also valid for time-sharing and for mixes of time-sharing and batch: the system runs a large time-sharing system as a "permanent" batch job.

The simplest model of a synthetic job for such an environment has only 3 resource utilization parameters: core storage; CPU time; and I/O time. The authors have expanded on these basic parameters with sufficient detail to produce synthetic job streams that drive their system "essentially identically to specified real streams (agreement on most parameters within 10%)." These refinements, as well as others that could be added, involve a trivial increment to the basic structure of the synthetic programs and require only the addition of extra control cards.

Tabular data are presented on four experiments conducted: (1) matching a standard benchmark against a synthetic batch stream; (2) matching compiler job steps with the synthetic job steps; (3) comparison of two synthetic streams in a full-load test of the overall system; and (4) simulation of a "normal" user load.

In the experimentation there was much reliance on the detailed accounting information kept by the system for each step and on the metering information kept by the operating system.

A number of experiments are planned for the future: (1) measurement of major changes in system hardware; (2) fine tuning of system and measurement of effects of such changes as variations in scheduling and dispatching modules, memory management, location of system files, etc.; and (3) error-checking and catching of performance bugs. A profitable by-product of the experiments run to date was the detection of several system bugs and solution of a hardware problem as a result of running the benchmark job stream.

In conclusion, the authors state that:

> (1) synthetic jobs are easily constructed to match most real jobs; (2) the basic measures of memory, CPU, and I/O appear to be sufficient to represent real jobs; (3) system performance measures are relatively insensitive to the internal structure of the synthetic jobs comprising a test; (4) job streams representing special demands require relatively little refinement; (5) CPU and I/O time measurements are appropriate for matching job streams, since these are parameters of the jobs being matched; and (6) for experiments in transferability, synthetic jobs creating CPU and I/O transactions are to be preferred, since these measures will be independent of system hardware and software. (JLW)

Category: 3.2

Keywords: Computer systems: experimentation; Honeywell–6070; synthetic benchmarks; synthetic job parameters; synthetic jobs; synthetic job stream; synthetic job stream generator; transferability.

50. Kolence. Kenneth W., Experiments & Measurements in Computing, in Association for Computing Machinery, *Proceedings of the SIGME Symposium*, February 1973, pp. 69–72, 2 refs. (6430165)

The paper is intended as an initial step toward establishing an experimental discipline within

the field of software physics which "is concerned with understanding the laws governing the behavior of software units." Software units consist of arbitrarily large (or small) groupings of code whose variable observable properties constitute their behavior in contrast to their functional properties. In this context, several classes of computer performance measurement activity are reviewed, to provide perspective in the development of an experimental discipline for testing behavior of software.

Building of an experimental discipline requires some general agreement on how one proves or disproves a theoretical hypothesis. One is proposed from the physical sciences: Experiments performed to verify some hypothesis concerning the real-world behavior of software units can only be accepted as valid if direct measurement is used to obtain all data.

From this it is obvious that an experimental discipline must be based on measurement and not on other techniques such as simulation. Simulation offers prediction which must then be verified or countered by measurement.

The second step in establishing an experimental discipline is also suggested from the physical sciences and that is that results of important experiments be published easily. The author proposes that SIGME encourage the publication of measurement experiments and data in the computer field, for general constructive criticism and comparison against theoretical predictions. In the absence of theory applicable to problems of interest, the author suggests encouragement of membership to suggest procedures for developing empirical curves of interest. (JLW)

Category: 1.3

Key words: Measurement experiments; program behavior; software performance measurement; software physics; software testing; software units.

51. Kolence, Kenneth W., The Software Empiricist, *Performance Evaluation Review*, 2:2 (June 1973) pp. 31–36. (6430193)

    The article is an announcement as well as the first occurrence of "The Software Empiricist," a new feature of *Performance Evaluation Review*. Software engineering has as its goal "the design of systems to perform as we wish them to." This requires an understanding of the meanings of the variables that are measured, or "the rationale of what has been called software physics."

    The new feature will be devoted to the empirical development of such an understanding by providing a publication vehicle for empirical and experimental data and thus developing a solid body of knowledge.

    Correspondence is invited initially on "characterization of what types of empirical data are of value, what is meant by the term experimental data, and what constitutes an experiment in software physics." Empiricism, "the search for quantitative expressions of the behavior of some object of study *prior to* the existence of a formal predictive theory," in reference to software cannot be a "random search for any quantitative expression of some system's behavior. We have that situation today. Software empiricism is the search for *invariant* relationships between the variables of computer measurement."

    The author characterizes the people involved in performance improvement as "all empiricists in one way or another, . . . [and] software empiricism can and does proceed without formal theory to guide it." This must necessarily be so until good evidence is obtained of the existence of invariant relationships between measurable variables.

    The author invites submission for publication of evidence of such invariance, and conversely, evidence of relationships clearly *not* invariant.

    Comment is also invited on the form of data presentation for the new Section. The author takes the initiative by suggesting circular graphs, recommending that they be called "Kiviat plots" or "Kiviat graphs," and inviting experimentation with two samples appended to the article. One graph form is for the presentation of job step time usage, the other for workload characterization.

    Admitting that chances of any response are quite low, the author ends with a promise to "dig up examples" for the next few issues of the *Review*. (JLW)

Categories: 11; 1.3

Key words: Measurement experiments; software engineering; *Software Empiricist;* software performance measurement; software physics.

52. Kolence, Kenneth W., A Software View of Measurement Tools, *Datamation*, 17:1 (January 1, 1971) pp. 32–38. (6430176)

    In discussing the characteristics and role of software performance measurement tools, the author states that a usable set of measurements requires a combination of descriptive and quantitative variables which must be extracted from memory without altering significantly the run characteristics of the system. Three parameters are involved: CPU cycle requirements, I/O activity, and core usage, all of which must be measured for each program module or segment. Sampling techniques can be used to obtain data with an acceptably low CPU overhead rate. Separation of the function of data execution

from that of data analysis reduces further the danger and/or necessity of altering the system being measured.

Within this context the author discusses some of the performance measurement products on the market, with specific references to two Boole & Babbage products, the program evaluator (PPE) and the configuration evaluator (CUE). The key relationships in the performance equation are expressed as:

$$E_c = Sx$$

where S=throughput rate=software units per unit time

x=average CPU work per software unit=CPU cycles per software unit

E_c=configuration efficiency=efficiency of CPU cycles used for job mix represented by x=CPU cycles per unit time.

Solving the equation for S represents the throughput of a given job mix for a given configuration.

There are relatively few computer performance measurement tools on the market. Nearly all of them detect and measure CPU busy or non-available CPU wait time. The class of products called configuration evaluators (whether hardware or software) are used to detect sources and magnitudes of avoidable CPU wait time. In this connection, access properties of data devices appear to play a significant role in generating avoidable CPU waits. The author cites (but does not identify) an available product which uses data collected by a configuration evaluator to optimize the organization of data sets on disc in order to reduce average seek delay time.

Program evaluators measure CPU cycle requirements for each segment of a program and record those whose CPU cycle requirements are significant. Program evaluators are useful in identifying the top ten "CPU cycle using" programs which then need to be refined to reduce their requirements of CPU cycle times. The article concludes with a section on selection of measurement tools. (JLW)

Category: 1.2

Key words: Configuration evaluators; data optimizers; measurement tools; software evaluation; software monitors; software performance measurement.

53. Lambert, David W., Report on FIPS Task Group 13, Workload Definition and Benchmarking, paper prepared for the Fall 1973 Meeting of The Computer Performance Evaluation Users Group, held at National Bureau of Standards, Gaithersburg, Md., 4–7 December 1973, 5 pp. (6430259)

Benchmark testing, or benchmarking, one of several methods for measuring the performance of computer systems, is the method used in the selection of computer systems and services by the Federal Government. However, present benchmarking techniques not only have a number of known technical deficiencies, but they also represent a significant expense to both the Federal Government and the computer manufacturers involved. Federal Information Processing Standards Task Group 13 has been established to provide a forum and central information exchange on benchmark programs, data methodology, and problems. The program of work and preliminary findings of Task Group 13 are presented in this paper. The issue of application programs versus synthetic programs within the selection environment is discussed. Significant technical problem areas requiring continuing research and experimentation are identified.

These problem areas include: the lack of a common terminology and measurement parameters; difficulties in determination and representation of workloads; workload specification and generation for online systems; machine-dependence of synthetic benchmarks; and new computer system architectures. (Modified author)

Category: 3.0

Key words: Application benchmark program library; application benchmarks; benchmark program development; benchmarks; FIPS Task Group 13; synthetic benchmarks; workload definition; workload specification.

54. Lambert, David W. and John F. Wood, USDA Synthetic Benchmark Project, Informal Report, October 15, 1973, 7 pp. (6430204)

The report describes the steps, problems, and technical considerations in the development currently underway of synthetic benchmark programs for an equipment procurement in the near future. This particular procurement was initially for a single facility to accommodate the processing requirements of the Agricultural Stabilization and Conservation Service and its 3,000 field stations. Other Department of Agriculture requirements were later consolidated into a single procurement which, in turn, was further consolidated into a joint procurement with the General Services Administration.

Analysis of the combined total requirements resulted in the identification of approximately 600 online tasks, classified into 14 category types. Since application programs useful for benchmarking were nonexistent, it was decided to develop synthetic programs which could be ad-

justed as required to fit the 14 categories. Synthetic program sets have been designed and programmed, and the sizing and validation process is now underway.

The categorization of the workload was the initial step in the creation of a synthetic workload which is described in detail, together with technical problems encountered in the development process.

The 12 synthetic programs developed to date are written in FORTRAN or COBOL. For data management, assembly language programs were written to emulate the necessary functions of data generation, transaction sets, etc. Approximately one-half of the Department's benchmark development effort was devoted to these programs.

Once the validation process for the synthetic programs is complete, several additional steps still remain in the benchmarking process. These include specification of benchmark requirements for the solicitation process, vendor preparation and execution of the benchmark, re-run for validation purposes, and then the evaluation of the final results of the benchmark tests. (JLW)

Categories: 11; 3.2

Key words: Assembly language; COBOL; data generation; Department of Agriculture; FORTRAN; Synthetic benchmarks; synthetic program modules; task classification; transaction file generation; workload definition.

55. Leavitt, Don, Study Asks: Are Synthetic Benchmarks Possible? *Computerworld*, June 20, 1973, p. 13. (6430158)

A brief description of a study launched by a Department of Defense steering committee on the feasibility of developing a representative library of standard benchmark programs focusing on specific data processing tasks rather than on entire applications.

The benchmarks under study would be operational but synthetic programs composed of variable task-oriented instruction streams, procedures, and data files from which a user could choose to tailor programs sufficient to represent his workload.

The steering committee has identified nine basic data processing tasks that "probably" should be in the proposed benchmark library: modified sort; edit; sequential update; indexed sequential update; random update; report extract; compute; remote inquiry; and remote update. (JLW)

Category: 3.2

Key words: Department of Defense; standard benchmark program library; standard benchmarks; synthetic benchmark program library; synthetic benchmarks; synthetic program modules.

56. Larson, Chris, The Efficient Use of FORTRAN, *Datamation*, 17:15 (August 1, 1971) pp. 24–31. (6430178)

The author presents numerous and detailed suggestions for FORTRAN optimization that the user can manually do at the source code level to improve object code performance. Optimal object code for the target machine is the only concern when a FORTRAN program is in the production stage. Coding techniques which result in a fairly good code do not eliminate the need for optimizing compilers, because many machine-dependent features cannot be anticipated by a user program and because even the best source code can be improved and enhanced by an optimizing compiler.

An analysis of how FORTRAN compilers deal with source code constructions is presented under the following headings: input/output statements; subscripts; data types and conversions; expression evaluation; machine-dependent and miscellaneous optimizations for various instructions and statements.

Admittedly, many of the recommendations presented here defeat the purpose of a high-level language which is to free the programmer from the rigidity of computer languages. However, the author feels that optimal object code is important enough to the programmer to make him willing to share the optimization burden. Compilers, in their turn, could minimize this burden by providing optional feedback data regarding program execution. For FORTRAN, the author suggests an option in the form of an "execution report" based on compiler-inserted frequency collection code at every node of the object program and expansion of all STOP statements into calls to a FORTRAN subprogram which would merge the original source code with the frequency data. Such a report could conceivably be used as input to subsequent compilations for insertion in the most frequently executed areas of the program. (JLW)

Category: 3.5

Key words: FORTRAN optimization; program optimization.

57. Loughlin, John B., 360/370 Comparative Benchmark Analysis, in *Proceedings of SHARE XXXVII*, 1971, pp. 79–100. (6430150)

A series of eighteen programs, eleven in the scientific job stream and seven in the commercial job stream, were written initially in 1967 to assist Wichita State University in the evaluation of a third generation computing system. In the years which have followed, additional in-

terest was expressed upon the part of a number of computing vendors and other computing organizations to have this same job stream run through their respective systems. Since 1967 this benchmark job stream has been run over fifty times on some twenty-four different CPU's and sixteen different operating systems. Mainframes tested included—Burroughs 2500, 3500, 5500; Control Data 3100, 3200, 3300, 6400, 6600; General Electric 415, 425, 435, 625; IBM 1620, 1130, 360 models 25, 30, 40, 44, 50, 65, 67, 75, 145 and 155; UNIVAC 1106 and 1108.

The results of the first two years of testing were summarized and evaluated by McCullough. At that time, only gross occupancies were captured, and in many cases gross occupancy for the job step was not available. Lack of adequate and consistent job accounting information has posed many serious problems. In the benchmark tests in more recent years, every attempt was made to capture both gross occupancy and CPU time information at the job step level, and to identify interjob step gross occupancy overhead whenever possible. Naturally, one of the more difficult matters of concern has been the problem of comparing hardware and software systems which are not exactly alike with regard to peripheral equipment and operating system features. Unless otherwise noted, peripherals used were 800–1,000 cards per minute readers, 900–1,100 lines per minute printers with 48–64 character print sets, 120 KB tape drives, and 2314 or equivalent disk drives.

The report contains a short description and characteristics of each job in the job stream. Results from the various runs of the commercial and scientific job stream on IBM systems are reported as a series of unweighted gross occupancies. Job step gross occupancy and CPU times are shown whenever job accounting captured this information. Therefore, careful interpretation should be exercised in order to evaluate relative speed or throughput for any given system. Additional untimed runs were made in a multiprogrammed, multijob stream mode on a number of machines. The time from first card read to end of execution of last job and time of last line printed, as well as CPU time consumed were examined on several operating systems as the order of the jobs run was changed. Because of the wide variance in features, as well as options of generating an operating system, many differences, some difficult to repeat, were found and hopefully would provide the basis for subsequent analysis for an additional paper. (IBM)

Categories: 11; 3.3

Key words: Benchmark run analysis; benchmark testing; experimentation; job characteristics; job stream representation; multiprogrammed computer systems.

58. Lucas, Henry C., Jr., Synthetic Program Specifications for Performance Evaluation, in Association for Computing Machinery, *Proceedings of the National Conference*, 1972, pp. 1041–1058, 15 refs. (6430111)

Various techniques for performance evaluation have been reviewed. For the major purpose of selection evaluation, a proposal and preliminary specifications have been developed for a series of synthetic test modules. This proposal features a set of industry wide modules developed by a committee from the computer industry and refereed by a professional organization. These modules would be provided by each manufacturer in a variety of languages for all of his computers. An outline of some preliminary specifications for the modules has been presented, though the details will have to be worked out ultimately by the committee.

The evaluator represents his anticipated job load by selecting from among a series of synthetic modules to form different runs. He formulates a series of experiments in which different runs are executed using the various jobs. All of the modules are highly parameterized, making it easier to tailor the tests to the anticipated job load. It is maintained that synthetic modules are the most flexible method to test hardware, software, and their interaction for the purposes of evaluating system performance. Two examples were given to illustrate the use of synthetic program in performance evaluation. (Author)

Category: 3.2

Key words: Compiler evaluation; computer performance evaluation; operating system evaluation; program parameters; synthetic benchmarks; synthetic program modules; synthetic program specifications.

59. Meiners, Eugene E., A Machine-Independent Data Management System, *Datamation*, 19:6 (June 1973) pp. 92, 94–95, 2 refs. (6430195)

The article reports on a development that may help reduce the problems associated with transfer or changes in computer-based information systems.

A machine-independent data management system means that the DMS is readily transferable from one computer to another. This attribute is an advantage additional to those inherent in DMS packages. If a given DMS package and its associated files have been written in a programming language approved by the American National Standards Institute, then the DMS can be easily adapted for use as a benchmark.

Such a data management system is available today, and it is called "the Machine-Independent Data Management System" (MIDMS). This was developed by the Defense Intelligence Agency

(DIA) with contractual assistance from the General Electric Company.

Machine-independence was a design factor in MIDMS, since there were none on the market with that attribute in 1968. COBOL was chosen as the language for MIDMS which is modular in design with a dynamic overlay structure which permits execution with a minimum amount of core storage. Some additional attractive attributes are capability for accommodating both fixed and variable-length records; the records may contain fixed, periodic, and variable subsets of data; and may also contain unstructured information of unknown length in the variable sets. In addition, the standard MIDMS interface permits calls of COBOL, FORTRAN, or assembly language subroutines.

The system was programmed and is operational on the IBM–360 series. The transfer to the HIS G–635 is almost complete. Retrieval and output capabilities are operational on the Honeywell system, and file maintenance was expected to be available in June 1973.

The feasibility of machine-independence has been demonstrated by the fact that MIDMS is operational on two quite different systems. Keeping MIDMS viable on both systems simultaneously will give added proof of the practicality of the development. (JLW)

Category: 3.4

Key words: Data management systems; Defense Intelligence Agency; Machine Independent Data Management System; MIDMS; transferability; transferable benchmarks.

60. Morgan, D. E. and J. A. Campbell, An Answer to a User's Plea?, in Association for Computing Machinery, *Proceedings of the SIGME Symposium*, February 1973, pp. 112–120, 22 refs. (6430179)

The paper reports on criteria, tools and techniques which can be helpful to a computer user in deciding which system to procure or in determining how badly his current system is performing. Benchmarking appears to be a good way to provide the information necessary for evaluation of the relative merits of available computer systems.

The authors identify two types of conventional benchmarks: (1) general goodness benchmarks; and (2) programs selected from the job profile to be benchmarks.

A general goodness benchmark is a specially-constructed program that performs tasks common to many computer installations, represents a wide range of problems in general terms only, and serves as a computer industry standard in evaluating computer system performance.

The paper defines *system evaluation* as the "function of evaluating a system to improve its operation or for procurement," and *user evaluation* as the "function performed by a user in deciding which system to use." In system evaluation, choosing a system "tuned" to a particular set of programs is to be avoided, while in user evaluation, the objective is just that, i. e., choice of a system tuned to a particular set of programs.

Accurate results are produced by benchmarks, provided that the programs selected for the benchmark are, indeed, typical of the user's job profile. In the authors' opinion, the synthetic job, as described by Buchholz, obviates the danger of choosing a system tuned to a particular set of programs in user evaluation. Two concepts used by Buchholz, "nonsense" calculations and the cyclic nature of his synthetic program, are then used by the authors to synthesize a set of programs to aid in choosing a system tuned *to* a given set of programs. To do this, synthetic programs must (1) represent the user's job profile so accurately that relative costs remain the same, and (2) "be inexpensive to generate and use, relative to the savings realized by choosing to run on the least expensive system." Such synthetic programs are called *synthetic benchmarks* because they are used as conventional benchmarks. Two further considerations are presented: costs of synthetic programs can be significantly reduced because "properly structured files of garbage can be constructed inexpensively by a ten-line FORTRAN program," since data for the synthetic program is not used in the processing, therefore all that is required is that data be similar in structure to actual files. A further consideration is that synthetic benchmarks do not demand resources in the same order or amount as required by actual programs.

Two approaches to the generation of synthetic benchmarks are presented: (1) resource demand and (2) service demand. Both approaches are illustrated.

The resource demand approach is quite detailed and vulnerable to differences in system software, language translations, and even machine architecture. However, this approach lends itself to automation rather easily and produces "fairly accurate" results on different model computers of the same series, or on computers with identical operating systems.

The service demand approach measures at the language level the computing and I/O services required by the programs in the job profile. These service demands are then mapped by a translator into resource demands. By defining the synthetic benchmark in service terms, the results of the benchmark run reflect the efficiency of the system's software, the language translator,

and the computer architecture. The service demand approach is illustrated by a hypothetical job mix consisting of one FORTRAN program analyzed for services provided by it and a synthetic benchmark created from the service measurement.

The service approach was also tried with a nontrivial job profile of two COBOL programs. Four computer systems were evaluated: IBM 360/50 and 360/75 and two PDP–10's at two different installations. The costs of running actual and synthetic programs at the four installations are tabulated; costs for the synthetic runs were consistently lower at all four installations.

Measurement of services for the experiment was done manually by determining the number of times each statement is executed in a program and then determining the services demanded by each statement. The second step was written for statements which were executed only a few times. The Algol–W compiler could be used to automate the first step; automation of the second step, however, "is nearly equivalent to writing a compiler for the language."

In another experiment it was found that optimizing compilers such as FORTRAN H must be used with caution in synthetic benchmarks. The FORTRAN H compiler might omit a piece of code should it "become obvious" that that particular piece of code does nothing!

The authors conclude that both approaches have faults in user evaluation—and the faults of the resource demand approach are the more serious. Four advantages are claimed for the service approach over conventional evaluation techniques: (1) tailoring to user job profiles, thus more accurate results; (2) adjustable processing capability can reduce costs of evaluation; (3) data independence; (4) opportunity provided to user to learn how difficult it is to use a system, what is its performance, reliability, and availability, and where help is available. (JLW)

Category: 3.2

Key words: Benchmark costs; comparative computer performance measurement; computer systems; data independence; experimentation; IBM–360/50; IBM–360/75; optimizing compilers; PDP–10; standard benchmarks; synthetic benchmarks; test data.

61. Murphey, Jesse O. and Robert M. Wade, The IBM 360/195 In a World of Mixed Job Streams, *Datamation*, 16:4 (April 1970) pp. 72–79 (6430209)

    The article describes the method used for design verification and early estimates of performance of the IBM 360/195. For this, more than a hundred job samples were collected from the workloads of possible users of the system. The jobs were then dynamically traced and analyzed, and a small number of job steps were selected as a sample set for projections. For each selected job step, the one trace-output tape was located whose contents produced an instruction execution frequency mix most closely matched to that of the job step as a whole. In this way, 17 job step segments were established as criteria for comparative, internal system performance projections.

    To predict internal performance of the 360/195, a timer program was used to model CPU operations to the extent of drawing timing charts accounting for all computation progress during each machine clock cycle. Input to the timer program were the trace-output reels; summary reports produced by the timer tabulated data such as number of machine instructions executed, total CPU time per run, and processor and buffer storage fetches and stores. Accuracy of internal performance predictions based on this methodology has been found to be well within 10 percent. The three main application determinants of the 360/195 system's internal performance appeared to be the instruction mix, the operand addressing pattern, and the ordering of instruction codes in the program.

    A predictive methodology was also designed for measuring throughput potential, based on case studies to assess this potential in terms of user programs, operating system components, and I/O devices available in 1969. Three of these case studies are discussed.

    The methodology was based on the construction of an acceptable job stream which was then run on each of three different hardware systems: a 360/65 defined as the base for performance comparisons, a 360/85, and a 360/91, with each system having I/O and OS/360 components appropriate for each job stream. The results of these runs are presented in tabular form and are considered "impressive." However, as of October 1969, the job stream had not been fully optimized, nor had the case study been properly completed. Additional sources of performance improvement were discovered in the course of the study. (JLW)

Categories: 2.3; 11

Key words: Comparative computer performance measurement; computer performance measurement; computer systems; experimentation; IBM–360/65; IBM–360/85; IBM–360/91; IBM–360/195; job step execution frequency; job stream definition; software monitors; throughput.

62. Oliver, Paul, Review of Standard Benchmark Effort, Department of the Navy, Automatic Data Processing Equipment Selection Office, Wash-

ington, D.C., Memorandum Report, 31 July 1973, 9 pp., 9 refs. (6430163)

The thrust of current federal effort in "standard benchmarks" is to develop a measurement tool with certain desirable qualities. Certain characteristics of computers can be measured in the selection process: (1) *availability* of hardware and software, expressed in terms of reliability, ease of maintenance, etc.; (2) *work capacity*, which can be measured from various points of view—as can be seen from the terms used below; (3) *job time* or time required to execute a single job; (4) *system throughput* or how much work is done; a function of the job mix, the job load, and various system parameters; (5) *response time*—a measure of the quality of services rendered, and largely dependent on scheduling priorities. In the context of computer selection, it would be reasonable to limit the scope of effort of measuring throughput capacity, without losing sight of the considerable importance of response time in an on-line environment.

Benchmark study is very closely related to the subject of computer performance evaluation, since some combination of evaluation techniques will need to be used to develop "standard benchmarks." These evaluation techniques can be broadly classified into two categories. The first category of techniques is task-oriented and is concerned with system throughput capabilities with respect to a given workload. The techniques include (1) *instruction timing* that reduces workload to specific classes of instructions; (2) *instruction mixes* or representative samples of instruction sets designed to reflect usage in a given type of application; (3) *kernels* or small sequences of code which perform a single function, for example, a table search; (4) *"natural" benchmarks* that consist of a subset of a given workload; (5) *"hybrid" benchmarks* that consist of a subset (further modified) of a given workload; and (6) *"synthetic" programs* or a set of programs written specifically for the purpose of making comparative evaluation.

The second category of techniques is computer-oriented and emphasizes the system under evaluation rather than the workload to be processed by the system. These techniques include (1) *hardware monitors* which are relatively inexpensive, precise in measurements, non-disruptive, and insensitive to data-dependence; (2) *software monitors*—characteristically the exact opposite of hardware monitors; (3) *queueing models* which are convenient, imprecise, and shallow; and (4) *simulation models*, which are expensive, less imprecise, and usually present a credibility gap.

For some period of time now some form of benchmarks has been the accepted form of minimum performance measurement in computer selection throughout the federal marketplace. Natural or hybrid benchmarks have an advantage over synthetics in that they deal with a real system, and thus avoid a "semi-real" job mix as well as the credibility problem inherent in simulation. Some of the more serious problems associated with benchmarks are: (1) inaccurate reflection of a given job mix; (2) system-dependence; (3) frequently do not run correctly, even on their native systems; (4) require too unduly long execution times; (5) require unreasonable file volumes; (6) require inconsistent measurement procedures; and (7) costly—both in time and dollar resources—to buyers as well as vendors.

The impetus for "standard benchmarks" is resolution of the problems associated with natural and hybrid benchmarks. Within the DoD additional impetus has come from: (1) "Task Group Report on Methods for Reducing Time and Cost Required for ADPE Procurement"; (2) "Contributors' Symposium on Standard Benchmarks"; and (3) suggestions in the literature on possibility of synthetic programs.

Use of synthetic programs in performance evaluation is not new. This technique has problems, too, all of them related to modeling the job mix. Some of these problems are: (1) their highly stylized form makes them very vulnerable to optimizing compilers; (2) use of analytic techniques to characterize a job mix is known to be grossly imprecise and the techniques themselves tend to become limiting factors; (3) use of software monitors for data collection frequently results in the creation of the Hawthorne effect; (4) a multitude of parameters is involved in a mix of programs; (5) there are no clear techniques for matching job parameters to mix parameters; and (6) workload modeling is dependent on component-oriented techniques cited above, and suffers from the same weaknesses.

Several current, complementary efforts in the Federal Government are underway, aimed at designing representative benchmarks.

1. *U.S. Army System Support and Evaluation Command*

Recently issued a solicitation for a "Standard Benchmark Study." Objectives of the contract are: "(a) the definition of all tasks and measurable functions performed by a computer in executing business-type applications: and (b) development of a method or technique of identifying and measuring the occurrence of each function or parameter in each task for the purpose of profiling computer workloads." The solicitation is the result of a study by a DoD Joint Steering Committee which has defined a

preliminary set of application tasks and task parameters for benchmark purposes.

2. *Department of Agriculture*

Has constructed a comprehensive set of benchmark programs which include such functions as transaction processing and data base management. This package should be studied in any effort geared toward designing a library of standard benchmark programs.

3. *Department of Labor*

Has under development a job simulation model with actual-use statistics as control parameters. Standard benchmarks are not the goal of this effort—but there is potential spinoff.

4. *U.S. Marine Corps*

Is involved in a project similar to that of the DoL, except that hardware monitors are used to provide data for creation of synthetic jobs.

5. *ADPESO*

The Navy's ADP Equipment Selection office has several activites under way. These include:

(1) the development of a small (5–7 program) library of synthetic programs in joint support of the DoD Steering Committee and their own in-house effort. The following assumptions underlie this effort: (a) relatively few parameters control the behavior of synthetic programs; (b) behavior of the programs relative to changes in parameters is predictable; (c) a workload can be specified based on parameters implicitly defined by the synthetic programs; and (d) these parameters can be set so as to reflect the workload. Such a set of programs could be used to enhance existing natural benchmarks, and for relatively simple systems such a mix could be the entire benchmark. The programs are currently in the testing phase of development.

(2) "Sanitization" of natural benchmarks through: (a) use of correct programs which run on at least their native machines; (b) standard code—or identification of non-standard in cases where standard is impossible; (c) translation of routines for conversion of machine-independent modules to machine-dependent form, as needed. Some indication of the merit of this activity is expected by the end of calendar 1973.

(3) Investigation of machine-independent basic procedures for benchmarks: duration, file volumes; code standardization; and allowable configurations.

The library of synthetic programs consists of the following: (1) sequential file processing; (2) indexed sequential file processing; (3) relative I/O processing; (4) COBOL sort; and (5) computation—a program to exercise arithmetic processing capabilities. The Army's Selection Office is writing Program Edit and Report Extract modules to add to the library. As each program is completed, it will be exercised in order to determine execution under varying parameter settings. This phase is also scheduled for completion in calendar 1973.

A final effort is planned to relate program parameters to installation workload parameters—to institute the "acceptance" phase of the effort. By second quarter of 1974, some indication should be apparent as to the usefulness of this synthetic programs approach. (JLW)

Category: 3.0

Key words: Department of Defense; Department of the Navy; Standard Benchmark Study; standard benchmarks; survey; synthetic benchmark program library; synthetic benchmarks; synthetic program modules; transferable benchmarks.

63. Parupudi, Murty and Joseph Winograd, Interactive Task Behavior in a Time-Sharing Environment, in Association for Computing Machinery, *Proceedings of the National Conference,* 1972, pp. 680–692, 12 refs. (6430182)

Continuous Software Monitoring System (COSMOS) is a measurement tool developed for observing and analyzing user behavior and operating system performance under the UNIVAC Series 70 Virtual Memory Operating System (VMOS), an interrupt-driven, time-shared, demand paging operating system.

This paper reports empirical data obtained by using COSMOS to observe a large number of interactions in which a wide class of programs were being executed interactively. Distributions of think time, compute time, page fault behavior, and I/O frequency are presented. (Author)

Category: 1.2

Key words: Compute time; computer performance measurement; demand paging; interactive activity; I/O activity; operating system measurement; operating systems; page faults; software monitors; software performance measurement; think time; Virtual Memory Operating Systems (VMOS); virtual memory systems.

64. Pearlman, Jack M., Richard Snyder and Richard Caplan, A Communications Environment Emulator, in AFIPS, *Proceedings of the Spring Joint Computer Conference,* 1969, pp. 505–512.

The Honeywell Communications Environment Emulator (HCEE) is a communications network simulator whose prime purpose is to aid in the checkout and debugging of communication software. It will simulate up to 63 lines with up to 8

terminals per line, and can generate at least 70,000 messages of 700 characters each per hour. During execution HCEE generates, transmits, and logs queries; receives and analyzes error codes; and logs system responses. A number of parameters describing the system under test, the terminal, and user characteristics, and the reporting are all parameterized and under control of the operator. It is interesting to note that the query generated by HCEE is chosen from a query vocabulary list which is part of its data base. The actual query itself is a random combination of words from that list. (MDA)

Category: 3.5

Key words: Communications network simulators; Honeywell Communications Environment Emulator; measurement driver; query generator; simulators; software performance analysis; user characteristics; user simulation; workload generator.

65. Robinson, Louis, Computer Systems Performance Evaluation (and Bibliography), IBM Corp., November 1972, 35 pp. (6430157)

    An overview of the state of the art, techniques and tools in use for measuring and evaluating computer systems performance. The bibliography consists of 365 citations, each annotated with keywords. (JLW)

Category: 1.5

Key words: Bibliography; computer performance evaluation.

66. Ruth, Stephen R., Using Business Concepts To Evaluate Large Multi-Level Business Systems—Some Applied Techniques, in Association for Computing Machinery, *Proceedings of the SIGME Symposium*, February 1973, pp. 73–77. (6430147)

    The author proposes dollar costs as a basic element of any device involving system measurement and evaluation. Three examples are provided of the application of marginal analysis for the purpose of selection of the best mix of measurement and evaluation techniques. The examples present "before and after" data for cost tradeoffs in: (1) evaluation of a vendor-supplied linkage-checking routine; (2) evaluation of the use of system resources by a program involving data manipulation primarily; and (3) evaluation of compiler efficiency.

Category: 1.3

Key words: Compiler efficiency; compiler evaluation; compiler optimization; program analysis; software evaluation.

67. Scherr, Allan L., Time Sharing Measurement, *Datamation*, 12:4 (April 1966) pp. 22–26. (6430184)

    The article describes measurements made of the performance of the MAC system during the 3-month period from December 1964 through February 1965. The system at that time consisted of an IBM–7094(I) with two 32K memories, IBM–1301–2 discs, an IBM–7320A drum and an IBM–7750 connected to model 35 teletypes and to IBM–1050 terminals. The community of users consisted of nearly 300 people who were characterized by the computational load they placed on the system.

    Users, or the programs serving them, were considered to be in one of six states, each of which is defined. These six states are: dead, command wait, working, input wait, output wait, and dormant.

    The basic unit of work in a time-sharing system is considered to be the interaction, or "the following sequence of events: user thinks, types input, waits for a response from the system, and finally watches the response being printed." Thus the user may be in either of two states, the *working* while he is waiting for the system to execute a program, or *command wait* while the system is waiting for the user. An interaction, then, can be defined as the activity that occurs between two successive exits from either *working* or *command exit* states.

    Data were gathered by a program which ran as part of the scheduling algorithm which recorded the sequence and timing of the events comprising typical interactions. Approximately 80,000 commands of five types were monitored: file manipulation, source program input and editing, program execution and debugging, compilation and assembly, and miscellaneous commands such as save and resume core images, programs to generate commands, etc. Data from the measurements are presented in graphs which show "think" time per interaction, program size distribution, processed time per interaction, and interactions per command. Other graphs show a typical response time distribution measured from a simulation of MAC under a constant load of 25 interacting users, and simulation results of response time versus processor time per interaction. These last two parameters were derived from simulation in order to eliminate from the measurements results of a constantly changing load. The author feels that simulation models can be easily derived from these data of accurate performance predictions. These predictions have been confirmed by comparing them with actual performance data from the MAC system.

Category: 1.2

Key words: Computer performance measurement; computer performance prediction; computer systems; IBM–7094; MAC system; man-computer interaction;

multiprogrammed computer systems; system monitoring; think time; user characteristics; user simulation; work unit.

68. Schwemm, Richard E., Experience Gained in the Development and Use of TSS, in AFIPS, *Proceedings of the Spring Joint Computer Conference,* 1972, pp. 559–569, 10 refs. (6430207)

    The author classifies the experience gained under four major areas: system structure; system performance analysis; software development tools; and management of software development. The second of these areas is of concern to benchmarking.

    In the course of the development of TSS/360, a comprehensive scheme evolved for dealing with system performance. Three components comprised the scheme: establishment of performance objectives; creation of external (to the computer) performance measurement tools; and creation of internal recording tools and data reduction facilities. Since TSS/360 is designed for three modes of operation (batch, conversational, and a mixture of the two), performance was defined for each mode of use. However, only conversational performance is discussed in the paper.

    Conversational performance is defined as the maximum number of tasks which the system will support with acceptable response time. Specifically, a benchmark terminal session was defined by dividing the interactions into three classes—trivial, non-trivial and data-dependent. Acceptable response times for each of these classes were further defined as follows:

command	*example*	*response time*
trivial	text entry	4.0 seconds
non-trivial	data set creation	7.5 seconds
data-dependent	compilation	undefined

    Since the above benchmark terminal session was not typical of any user's conversational workload, most users specified their own benchmarks. However, the consensus is that the above definition of performance is adequate. The initial performance objective was to support 40 tasks with one CPU with 512K memory, 1 drum and 1 disk channel.

    In order to measure a load imposed by live users on TSS/360, a measurement driver was created to simulate the live environment under controlled and reproducible conditions. A schematic diagram of the TSS/360 measurement driver is presented. To measure conversational performance, a series of driver runs is executed, with varying numbers of tasks for each run. A curve is then drawn representing response time as a function of the number of tasks. The paper includes such a curve for TSS/360 release 6.0 for 2 different system configurations.

    The measurement driver discussed here runs on the IBM–360/40 and has been used by some installations to evaluate system performance on their own benchmarks. A second measurement driver is mentioned, this one developed by Carnegie-Mellon University for the IBM–360/67 with the version of TSS under study in this paper. The Carnegie-Mellon Simulator (SLIN) is compatible in script and timing characteristics with IBM's and produces comparable output.

    The paper concludes with a discussion of debugging aids and internal recording tools such as the Systems Performance Activity Recorder (SPAR), the Systems Internal Performance Evaluation (SIPE), and the Instruction Trace Monitor (ITM). (JLW)

Category: 5.2

Key words: Benchmark terminal session; computer systems; debugging aids; hardware monitors; IBM–360/40; IBM–360/67; man-computer interaction; measurement driver; multiprogrammed computer systems; operating systems; response time; software monitors; TSS/360; user scripts; user simulation; workload generator.

69. Schwetman, H. D. and J. C. Brown, An Experimental Study of Computer System Performance, in Association for Computing Machinery, *Proceedings of the National Conference,* 1972, pp. 693–703, 17 refs. (6430181)

    This paper describes an experimental study of the performance of a large multiprogrammed computer system (the UT–1/CDC 6600 system at the University of Texas at Austin) under systematic variation of available resources and resource allocation algorithms. The experiments were carried out in a controlled and reproducible environment provided by a synthetic job stream generator. The experimental data were recorded by an event-driven software monitor which recorded a complete trace of system activities at the level of system defined events. The study relates resource utilization and queuing patterns to the metric of job completion rate. The experiments undertaken in these studies are single factor experiments. Compensatory reactions by this complex system to variation of individual resources are nonetheless revealed. The experiments also demonstrate the criticality of optimal scheduling of bottleneck resources and offer comparisons of the performance of multi-drive disk units under different conditions of availability and space assignment. The data gathering facility was also run on the production environment to determine base lines for comparison to the experiments as well as for an understanding of the production mode of operation of the system. (Author)

Category: 1.2

Key words: CDC–6600; computer performance analysis; computer performance measurement; computer systems; data gathering; experimentation; job completion ratio, queuing patterns; resource allocation; software monitors; synthetic job stream; synthetic job stream generator; Texas University.

70. Shope, W. L., K. L. Kashmarak, J. W. Inghram and W. F. Decker, System Performance Study, in *Proceedings of SHARE XXXIV*, Vol. 1, 1970, pp. 568–659. (6430106)

Directors and managers of computing facilities are faced each day with questions such as—how fast is our workload growing? Can current machinery contain future growth and for how long? What do new hardware and software developments mean to this installation?

In order to obtain answers to these questions, the University of Iowa Computer Center began in the spring of 1969 to perform benchmark analyses of both hardware and software. It was apparent at that time and is still apparent today that very little valuable information exists to help guide the installation in making an evaluation. Prior to any analysis, many important questions regarding benchmarking techniques need to be resolved. What kind of data are needed? How can the data be obtained? Should software or hardware measurements be made?

One of the prime requisites for a meaningful benchmark analysis is a job stream representative of a given environment. At UCC, the decision was made to assemble a set of jobs from the real job stream which reflected normal operations. Fifty-two jobs were selected through the use of a sample procedure involving the distribution of jobs run versus time of day to generate random times of day at which samples were extracted. The samples consisted of jobs in execution at these times; selection, in turn was on the basis of first-in-first-out and on job classification (which involved amount of core storage and type of processor required). The effect of this procedure was to weight selection in favor of hours of heavy usage and distribution of jobs by classes similar to the normal job stream. The sampling period lasted three days. For testing purposes, the jobs were organized in order of time of selection to construct a job stream that required approximately 55 minutes of execution time on the University's IBM–360/65. In retrospect it was learned that average CPU utilization was approximately 10 percent higher in the test job stream than in the actual job stream. The test job stream was adjusted somewhat after the first set of tests; a comparison of actual versus test job streams (presented in the Appendix) revealed that the latter were representative.

Other material presented in the Appendix presents data on the 16 benchmark users, and details on the jobs in the test job stream. A summary sheet describes changes instituted at the University Computer Center and an estimate of of values returned as a result of the tests. (JLW)

Categories: 3.3; 10

Key words: Benchmark run analysis; computer systems; cost-value; experimentation; hardware monitors; IBM–360/65; Iowa University; job characteristics; job classification; University Computing Center; workload construction.

71. Smith, J. Meredith, A Review and Comparison of Certain Methods of Computer Performance Evaluation, *The Computer Bulletin*, 12:1 (May 1968) pp. 13–18, 5 refs. (6430190)

The article presents a discussion of various types and relative merits of instruction mixes and benchmarks. For an estimate of the relative power of different computers over a wide range of applications, some means must be found for combining the run results of the benchmarks and mixes. A calculation of such a measure of relative performance is suggested and illustrated. The work unit, used by the British GPO for evaluating performance, is described briefly. (JLW)

Category: 3.0

Key words: Benchmark run analysis; benchmarks; instruction mixes; work unit.

72. Sreenivasan, K. and A. Kleinman, On the Construction of Representative Synthetic Workloads, The Mitre Corp., Bedford, Mass., Rept. No. MTP–143 March 1973, 31 pp., 9 refs. (6430170)

The evaluation of computer systems is usually conducted for the purposes of improving the present performance, predicting the effects of changes in either the existing system or the workload, or comparing different systems. The evaluation may use analytical modeling, simulation, or experiments with the existing system. In all these cases there is a need for a drive workload that imitates the real workload with reasonable fidelity, but in an abbreviated form.

This paper describes a method of constructing the drive workload using synthetic programs. The real workload is characterized by the magnitude of the demands placed on the various system resources; for example, the CPU time, number of I/O activities initiated, core used, and the usage of unit record devices. These are obtained from the system accounting data. The representative drive workload is constructed by matching the joint frequency distribution of the selected characteristics. The drive workload is

realized by using a synthetic program that contains many parameters. By adjusting these parameters, any desired combination of workload characteristics can be obtained. Using this procedure a synthetic workload with 78 jobs is constructed to represent a month's workload (for an IBM–370/155) consisting of about 7000 jobs. (Author)

Categories: 2.2; 3.2

Key words: Accounting data; computer systems; IBM–370/155; synthetic workload construction; workload characteristics; workload representation.

73. Sreenivasan, K. and A. J. Kleinman, On the Construction of a Representative Synthetic Workload, *Comm. ACM*, 17:3 (March 1974) pp. 127–133, 13 refs. (6430278)

A revised and later (July 1973) version of Mitre Rept. No. MTP–143. The method described was applied to the construction of a synthetic workload of 88 jobs, representing a month's workload consisting of about 6000 jobs. (JLW)

Categories: 2.2; 3.2

Key words: Accounting data; computer systems; IBM–370/155; synthetic workload construction; workload characteristics; workload representation.

74. Stanley, W. I., Measurement of System Operational Statistics, *IBM Systems Journal*, 8:4 (1969) pp. 299–308, 6 refs. (6430100)

The paper describes design factors and data gathered by JAS, the job accounting system developed for continuously and automatically monitoring a real-time operating system. Use of JAS provides a variety of statistics at optional levels of detail, at little cost in computing time or time lost in collecting unwanted information. All monitored information is stored in a data base for post-processing which is flexible so that reports tailored to particular needs may be produced.

Operational statistics gathered by JAS are used for performance evaluation in two ways: to optimize the operating system performance and to characterize workloads in simulating system operation. Experience with the real-time environment indicates that few statistics are needed to characterize the workload, given suitably parameterized models calibrated with detailed, measured performance data. These data include types of jobs and job steps (assembly, linkage–editing, etc.) and some information about the equipment (main storage capacity, CPU speed, etc.). Use of operational statistics in preference to hypothesizing workloads was found to enhance the accuracy of system simulation. (JLW)

Categories: 1.2; 2.2

Key words: Accounting data; job accounting system (JAS); simulation; statistical analysis; statistical models; system monitoring; workload characteristics.

75. Statland, N., R. Proctor, J. Zelnick, R. Getis and J. Anderson, An Approach to Computer Installation Performance Effectiveness Evaluation, Auerbach Corp., Philadelphia, Pa., Rept. No. 1243–TR–2, June 1965, 164 pp. (ESD–TR–65–276; AD–617 613)

The process provides objective measures of performance efficiency based on both quantitative and qualitative data, and provides standards for measuring installation effectiveness. Specifications and characteristics are collected via questionnaires, once and only once, in four categories—computer hardware, extended machine (hardware/software interaction), software evaluation and problem specification. An extension of this measurement of computer system performance provides a rating for the performance of a given software package on a given piece of hardware by comparing the time derived from the hand tailored coding to the timing resulting from the object program produced by the software. This ratio measures the efficiency of the software on the specific hardware configuration.

The aggregate ratios for all the individual performance criteria are used to derive a performance standard for a software system. Algorithms are used to summarize the raw data elements and a computer program will select data elements, make simple arithmetic combinations of these elements into composites, and prepare the data for entry into a statistical analysis. Stepwise multiple regression analysis is utilized to determine the relative significance of various data elements and to calculate their relative weights. (Author)

Category: 1.3

Key words: Computer performance evaluation; computer performance measurement; program timing; questionnaires; software performance measurement; statistical analysis.

76. Steffes, Sylvester P., How the Air Force Selects Computers, *Business Automation*, 14:8 (August 1967) pp. 30–35. (6430161)

The article reports on an interview with Col. Steffes in which he describes the system analysis that preceded the formulation of the bid package prepared by the Electronic Data Processing Equipment Office, the Air Force's centralized agency for competitive evaluation and selection of commercial computer systems. The RFP was for the Air Force's Phase II Base Level Data Automation Standardization program, and involved the requisition of about 135 computer

systems. After identifying all systems which would be replaced, the second step in the analysis involved the gathering of workload statistics.

Analysis of workload statistics resulted in a 4-level classification of workloads, A through D. After this, more than 1,000 applications were defined and simulated in order to gather data on the capability of current equipment to perform the processing tasks required. Like processing tasks were then consolidated into an RFP workload which consisted of approximately 250 applications for each workload level. The RFP specified that this workload had to be accomplished in 200 hours of operational use time per month. The specification for the "D" level workload required further that a live test demonstration be given of the equipment prepared.

Four suggestions are presented for users in preparation for solicitation of vendor proposals: (1) expend as much effort and time as possible in analyzing and developing specifications: (2) develop evaluation criteria based upon the specifications released to industry prior to the issuance of the RFP; (3) determine the techniques to be used for validating vendors' proposals, e.g., benchmarks, simulation, etc., (4) establish firm dates for accomplishing all defined tasks, e.g., proposal submission, benchmark, equipment installation, etc. (JLW)

Categories: 1.3; 11

Key words: Application benchmarks; computer selection procedures: Department of the Air Force; workload analysis; workload characteristics; workload specification.

77. Strauss, J. C., A Benchmark Study, in AFIPS, *Proceedings of the Fall Joint Computer Conference,* 1972, pp. 1225–1233, 4 refs. (6430109)

The paper presents a case study of benchmarking for third generation, multiprogrammed system selection which was conducted by the Washington University at St. Louis to aid in selection of a replacement for their IBM–360/50. The new system would have to be able to process the existing IBM–360/50-oriented workload with a minimum of conversion. In addition, it was desirable that the replacement system provide sufficient excess capacity to support new applications. It became clear from these and other considerations that the selection decision had to be strongly influenced by the relative performance of prospective new systems on the existing workload. This was the basis for the decision to conduct a benchmark study of the relative performance of various competing systems on a representative sample of the current batch-oriented workload. Selection of systems for the study was based on combinations of past experience, subjective opinions of suitability, and satisfactory relations with local representatives of vendors.

The objectives of the study and the design of a comprehensive benchmark to achieve these objectives are presented in detail for four prospective systems, Burroughs B–6714, IBM–370/155, Univac–1108, and the XDS Sigma 9, together with comparative data for the University's IBM–360/50.

The principal objective of the study was "to determine the relative thruput of similarly priced and similarly configured computers on a well defined workload." Secondary objectives included a determination of: (1) a measure of conversion difficulty, based on conversion effort required by the various vendors; (2) the extent and usefulness of standard accounting data collected by the various operating systems; (3) sensitivity of system performance to tuning of the hardware/software configuration; and (4) relative performance of the systems on certain benchmark characteristics (relative CPU times for COBOL and FORTRAN compilation and execution, for example).

The text of the benchmark description that was supplied to the vendors is given in the Appendix to the paper. The text includes detailed run procedures and specifications of expected output. Operational rules for the benchmark runs and a summary of output expected from each vendor are presented in tabular form.

The benchmark itself consisted of 25 jobs: 6 COBOL (IBM COBOL F) and 13 FORTRAN (IBM FORTRAN G) compile and execute runs; and 6 WATFIV jobs. One COBOL job tested compiler diagnostics, but all the other jobs were executable. In addition, the benchmark included one job whose sole function was to expend any available CPU time not required by other jobs or by the operating system, and thus gain some measure of extra capacity.

The results of the benchmark series are summarized and detail data are presented in tabular form for all systems. Summary data for thruput performance are given in terms of elapsed time for the execution of the benchmark series. Some interesting observations are added on the corrections that had to be made on the raw data in order to arrive at comparable thruput measures in the cases where vendors chose to interpret the rules "somewhat differently." The paper ends with brief discussions of individual system performance.

Two "obvious and important" conclusions were drawn from the study:

(1) That the general benchmarks described here could be run with relatively little effort

on such a wide diversity of machines speaks well for the standardization of COBOL, FORTRAN and general operational procedures.

(2) When establishing rules for benchmark operation, it is imperative that vendors fully understand the meaning and importance of each constraint.

No system was chosen, as a result of all of the above. "In light of today's troubled economics picture, [this] is probably not surprising." Also, in the process of the benchmark study and analysis, "it became obvious that the concept of a single central computer had to be carefully reviewed in light of the rapidly developing technology currently surfacing in the small machine area." (JLW)

Categories: 2; 11

Key words: Accounting data; benchmark description; benchmark design; benchmark run analysis; benchmark specifications; Burroughs B–6714; computer systems; IBM–360/50; IBM–370/155; multiprogrammed computer systems; system tuning; transferability; UNIVAC–1108; university computing center; XDS Sigma 9/Washington University.

78. Timmreck, E. M., Computer Selection Methodology, *Computing Surveys*, 5:4 (December 1973) pp. 199–222, 88 refs. (6430261)

The process of selecting a computer is very complex, involving much technical detail and much economic approximation. Because of the substantial investment which many organizations have in computers, it is of significant benefit to use selection procedures which minimize computer costs as much as possible within the numerous difficult constraints always present in computer selections.

The issues involved in such selections are so complex, the needs and economics are so variable, and the trade-offs which must be made are so qualitative that it is not meaningful to suggest a single, universal approach to selection. Instead of attempting such a recommendation, this paper provides a framework and a basis from which the user can confidently determine a selection approach which is appropriate to his own circumstances.

An exposition of the basic considerations underlying selection is presented, followed by a detailed analysis of the steps in the selection process, including alternative approaches to carrying out these steps. An account of procedures used by various organizations is given. Finally, a bibliography, organized by subject matter, is included to facilitate deeper investigation of the issues raised. An appendix contains an attempt at comparison of some major computers.

Three types of benchmarks are discussed and defined. "A live benchmark is a representative set of programs chosen directly from the user's workload. . . An artificial benchmark is a program which models a live benchmark. . . A standard benchmark may be considered a more sophisticated kernel, and sometimes it is difficult to distinguish between the two."

The artificial benchmark has a significant advantage over the live benchmark in that it can easily be converted to run on many different systems. However, the artificial benchmark is a much less accurate model than the live benchmark. Another disadvantage of the artificial benchmark is the difficulty inherent in exercising a machine and proving what was accomplished. The standard benchmark suffers as a selection tool because it is not drawn from a user's workload. The literature indicates substantial agreement that the well-composed live benchmark is the only tool accurate enough for selection purposes. The problem of "tailing" in live benchmarks is seldom discussed in the literature. This happens when all jobs in the benchmark are finished except one which may require two additional hours, and thus processing time for the whole benchmark must include these two hours. (Modified author)

Category: 1.3

Key words: Artificial benchmarks; benchmark practices; computer performance evaluation; computer selection procedures; costs; cost-value; live benchmarks; simulation; standard benchmarks: synthetic benchmarks; tailing.

79. Totaro, J. Burt. Real Time Processing Power: A Standardized Evaluation, *Computers and Automation*, 16:4 (April 1967) pp. 16–19. (6430211)

The article describes Auerbach's standardized estimating procedures for measuring computer performance in locating and updating randomly addressed records in real-time applications. Like the generalized file processing benchmark problem (described by Hillegass), the random-access benchmark represents a typical inventory control application. Unlike the first problem, however, the random-access benchmark problem uses master files stored on random-access devices, accepts detail transactions in random order, and writes the reports of file activity on auxiliary storage devices.

Results of the benchmark problem are presented in timing "kernels" keyed to the problem flowchart, so that a systems analyst can obtain performance estimates for specific processing jobs by assembling timing kernels for program segments unique to his applications. (JLW)

Category: 3.0

Key words: Application benchmarks; standard benchmarks.

80. Waldbaum, Gerald, Evaluating Computing System Changes by Means of Regression Models, in Association for Computing Machinery, *Proceedings of the SIGME Symposium,* February 1973, pp. 127–135, 10 refs. (6430183)

This paper describes how a regression model of the Research Center's APL system was constructed and used for evaluating system changes. The regression methodology that is generally used for performing such an analysis is believed to have been expanded by the introduction of multiple regression equations for estimating the cumulative distribution function of a performance variable.

A model is developed, which shows that the size of the workspace, but not the number of workspaces in core, significantly affects the performance of the Research Center's system. In particular, the model indicates that increasing the workspace size from 5 to 7 tracks causes 11 percent of all requests to be degraded by at least 0.4 seconds under mean load conditions. (Author)

Category: 1.2

Key words: Computer performance analysis; statistical models; workspace size.

81. Watson, R., Computer Performance Analysis: Applications of Accounting Data, the RAND Corp., Santa Monica, Calif., Rept. No. R–573–NASA/PR, May 1971, 61 pp., 2 refs. (6430188)

Virtually all Air Force and NASA computer installations collect and record accounting data (which, in computer systems, are an account of computer resources used by each job processed). However, seldom is any use made of these data except at those installations that use accounting data to charge for computer services. This report suggests that the analysis of computer system accounting data can be a valuable tool in computer performance analysis. The report describes the types of accounting data generally available at most computer installations—discussed in the context of the accounting data collcted by RAND's IBM–360/65 computer system. Techniques for conditioning and reducing the data are then discussed, along with various reports that can be generated from such data. The balance of the report concerns specific applications of accounting data analysis in computer performance analysis.

The report also discusses such other applications of accounting data analysis as validation of system performance measurements taken by hardware or software monitors, and use in either developing and testing a new computer charging scheme or in updating an installation's current charging scheme in the face of changing workloads or changes in the system.

Accounting data can be used to verify that a typical workload of jobs was run during a monitored period. This is determined by comparing workload and performance characteristics during the monitored period with workload and performance characteristics during previously monitored periods.

A scan of the computer resources used by each job step is useful in: (1) checking the CPU-boundness of jobs by observing (a) the number of CPU seconds used by each job step, and (b) the ratio of CPU seconds to total time on the computer; (2) checking the I/O-boundness of jobs by observing the number of I/O's used by each job step; (3) checking to see if a job unusually dominated the computer system over the monitored interval.

Summary figures for the monitored interval are useful in comparing the workload and performance characteristics with previously monitored periods. Performance measures that proved useful at RAND include: (1) average CPU utilization; (2) average job I/O's processed per second; (3) average job steps processed per hour; (4) average number of jobs in core; and (5) average revenue produced per hour. Workload measures that proved useful at RAND include: (1) mean and frequency distribution of CPU seconds used per job step; (2) mean and frequency distribution of I/O's used per job step; (3) mean and frequency distribution of core memory requested per job step.

With these simple measures, it was often determined that impressive results were due to shifts in job stream characteristics rather than improvements in the system. For example, one change (making some supervisor programs core-resident instead of storing them on disk) seemed to result in a doubling of CPU activity when a software monitor was used to measure performance under actual operation. However, analysis of the accounting data for the same time period indicated that an abnormally heavy load of CPU-bound jobs had caused the shift. Had the accounting data not been checked, unwarranted conclusions would have been the result. (Modified author)

Category: 1.2

Key words: Accounting data; computer performance analysis; computer performance measurement; computer systems; IBM–360/65; job step execution frequency; measurement tools; resource utilization; validation; workload characteristics.

82. Weihrich, W. Fred, Computer Selection, *Data Management,* 8:2 (February 1970) pp. 31–33. (6430144)

A good summary description of the computer selection procedure followed by the Navy De-

partment's ADPE Selection Office, a staff function reporting directly to the Special Assistant to the Secretary of the Navy.

An indispensable feature of all major acquisitions is the requirement for an actual benchmark demonstration. This is the "only way to assure that the hardware and software required do indeed exist and that together they will accomplish what the supplier promises in an acceptable time."

Equally imperative is a thorough review of the workload. The review must result in the assurance that the benchmarks are truly representative of the nature of the workload and that realistic factors are applied to extrapolation of the benchmarks into a realistic projection of the total workload as well as to projections of overall system growth and life.

Benchmarks are characterized as "merely nothing more than typical operational program samples." (JLW)

Category: 1.3

Key words: Application benchmarks; benchmark demonstration; computer selection procedures; Department of the Navy; proposal validation; workload definition.

83. Williams, Quincy N., *et al.*, A Methodology for Computer Selection Studies, *Computers and Automation*, 12:5 (May 1963) pp. 18–22. (6430218)

The article describes the method used at Smith Kline & French Laboratories to arrive at figures of merit for ranking vendor responses to their RFP for upgrading their computer installation.

Five factors were considered important: equipment specification; benchmarks; software; support; and cost. Each of these was given a weight of 20; the components which comprise such factors, and the rating assigned to each component are presented in chart form.

The benchmark chosen consisted of four problems: marketing information retrieval and mailing list preparation; billing and sales analysis; scientific information retrieval; and statistical analysis involving matrix inversion. The 20 percentage points allocated to the benchmark factor were distributed to the individual problems in accordance with their relative importance in the company's workload. (JLW)

Category: 1.3

Key words: Application benchmarks; computer selection procedures; proposal evaluation.

84. Wiorkowski, Gabrielle K. and John J. Wiorkowski, A Cost Allocation Model, *Datamation*, 19:8 (August 1973) pp. 60–65. (6430217)

The cost allocation model presented fulfills the criteria of equitable, reproducible, and realistic charges. The model establishes the relative value of resources for comparison on a common basis and encourages users in practices conducive to establishing lower operating costs. Data necessary for the implementation and maintenance of the cost allocation model provide a measurement of overall system performance. Although cost allocation in a multijob processing, teleprocessing and virtual storage system is a complex task, it is a necessary task, and can result in numerous benefits to the installation and its users. (Excerpt)

Category: 1.3

Key words: Computer performance measurement; cost allocation; cost allocation model; multiprogrammed computer systems; virtual memory systems.

85. Wood, David C. and Ernest H. Forman. Throughout Measurement Using a Synthetic Job Stream, in AFIPS, *Proceedings of the Fall Joint Computer Conference*, 1971, pp. 51–55, 3 refs. (6430223)

The paper describes experience in defining workload characteristics and then translating these into a synthetic job stream. A job stream is defined here as "a collection of independent jobs which can be used to determine the relative throughput of a multiprogramming system based on the time taken to execute all the jobs."

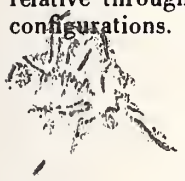
A job stream can be assembled from actual jobs. However, the following difficulties have been experienced in using for test purposes a job stream consisting of actual jobs: (1) reluctance of users to supply programs, data bases, and operating instructions; (2) security requirements prevent the inclusion of many jobs; (3) characteristics of each job are fixed, therefore many jobs are needed to closely match the overall characteristics of the job stream; (4) duplication of large data bases is extravagant; (5) difficulty in keeping complex jobs viable in the face of changes in operating systems and catalogued procedures.

To characterize a workload, the parameters considered important were: CPU utilization, I/O channel activity, core requirements, printer output, and tape and disk requirements. These parameters of actual jobs can be used as specifications for creating a synthetic job.

The synthetic job used in this paper is based on the type of program suggested by Buchholz, and written in PL/I. A listing of the program is supplied, along with a brief description of what it does, its parameters, plus a brief discussion on running time and the order of jobs in the job stream.

The validity of using a synthetic job stream to measure throughput was tested by comparison with a representative job stream composed of actual jobs selected from the workload so as to reflect the workload characteristics previously defined. Activities reported by a hardware monitor for the two job streams are discussed.

Subsequent use of a synthetic job stream was for the purpose of obtaining a measure of the relative throughput of three different IBM–360 configurations. Conclusions reached from the experimentation reported confirm the practicality of the synthetic job stream approach to performance measurement. (JLW)



Categories: 3.2; 2.3

Key words: Actual job stream; computer systems; experimentation; hardware monitors; IBM–360/50; IBM–360/65; multiprogrammed computer systems; synthetic jobs; synthetic job stream; synthetic job stream generator; synthetic program (PL/I); throughput; workload characteristics.

Category Index

Preliminary Classification Scheme

Key Word Index

The assistance of Mrs. Dolly Downs who typed the manuscript is gratefully acknowledged.

NBS-114A (REV. 7-73)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS Spec. Pub. 405	2. Gov't Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Benchmarking and Workload Definition: A Selected Bibliography with Abstracts

5. Publication Date: November 1974

6. Performing Organization Code

7. AUTHOR(S)

Josephine L. Walkowicz

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

10. Project/Task/Work Unit No. *640.1135*

11. Contract/Grant No.

12. Sponsoring Organization Name and Complete Address *(Street, City, State, ZIP)*

Same as No. 9

13. Type of Report & Period Covered

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Card Catalog Number: 74-17210

16. 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 85 citations to the literature of benchmarking and workload definition were selected from a longer list of documents, encompassing a somewhat broader scope, that was submitted to Federal Information Processing Standards (FIPS) Task Group 13 in response to a request made to attendees of the Task Group's Planning Session held on July 12, 1973, at the National Bureau of Standards. One of the topics discussed at the Planning Session was the collection of a selected bibliography on workload definition and benchmarking. The bibliographic effort was to be directed not so much toward exhaustiveness as toward the development of a bibliography that the attendees had found useful and would, therefore, recommend to other workers in the field. Of the approximately 250 citations submitted to the Task Group, these 85 were selected on the basis of two criteria: (1) the item dealt primarily with benchmarking or workload definition; and (2) hard copy was available at the Institute for Computer Sciences and Technology. The citations are arranged alphabetically by last names of the first authors. Each citation has an abstract, a classification category assignment, and a list of keywords. The category assignments are made from a classification scheme that was developed for the collection and that is used here as a Category Index to the Bibliography. A Keyword Index is also provided.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Benchmarking; bibliography; computer performance measurement; computer procurement; workload definition.

18. AVAILABILITY

[XX] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office Washington, D.C. 20402, SD Cat. No. C13.10:405

[ ] Order From National Technical Information Service (NTIS) Springfield, Virginia 22151

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PAGES
UNCLASSIFIED	45
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	$1.05

USCOMM-DC 29042-P74

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