U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology

NISTIR 89-4128

National Computer Systems Laboratory

Processing Rate Sensitivities of a Heterogeneous Multiprocessor

G.E. Lyon

U. S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
National Computer Systems Laboratory
Advanced Systems Division
Gaithersburg, MD 20899

CMRF

COMPUTER MEASUREMENT
RESEARCH FACILITY
FOR HIGH PERFORMANCE
PARALLEL COMPUTATION

August 1989

Partially sponsored by the
Defense Advanced Research Projects Agency
and the
Department of Energy.

Processing Rate Sensitivities of a Heterogeneous Multiprocessor

Gordon Lyon

Advanced Systems Division
National Computer Systems Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899

Partially sponsored by
- Defense Advanced Research Projects Agency
- Department of Energy

U.S. Department of Commerce, Robert A. Mosbacher, Secretary

National Institute of Standards and Technology
Raymond G. Kammer, Acting Director

August 1989

TABLE OF CONTENTS

Processing Rate Sensitivities of a Heterogeneous Multiprocessor

Gordon Lyon

Key words: Application signature; architecture; capacities; models; performance; sensitivities.

Simple performance characterizations of multiprocessors show that such models, freed of specialized detail, apply easily and widely. They can convey insight about parallel efficiency [4] or scheduling [6]. Processing rate sensitivities are also amenable to a simple approach. Consider a parallel program whose code is represented via *resource demands,* $\alpha=\{\alpha_i\}$. The coefficients, which sum to unity, reflect how the program's total computation divides among a system's *disjoint* computational modes. Modes typically differentiate on type (say scalar or vector) or number (single vector unit or chained) or both. Workload fractions for a vector program might be {0.3, 0.7}, for demands of scalar-mode (1-processor) and vector-mode (1-processor), respectively [2]. Mode *capacities* determine a second set of coefficients, $r=\{r_i\}$. These are rates of satisfying demand. A typical set of scalar-mode/vector-mode capacities is {10, 110}, measured in millions of floating-point operations per second (Mflops). Program performance as an average processing rate R is estimated by equating "time=time" [1, 2, 7]:

$$\frac{1}{R} = \left[\frac{\alpha_1}{r_1} + \frac{\alpha_2}{r_2} + \ldots.\right] \qquad (i)$$

Combining the example coefficient sets, $R = [0.3/10 + 0.7/110]^{-1} = 27.5$ Mflops. Coefficients for *(i)* also contribute, along with scheduling, to program rate sensitivities.

Sensitivity studies of capacity on otherwise fixed structures (architectures) are common in engineering disciplines, e.g., [8]. Analogous programming studies emphasize where an application class (represented by signature α) benefits from component (capacity) changes on a host computer architecture. However, set α is not generally independent of capacity changes, which can upset processor load balances. The redistribution of α is modeled via best and worst cases of rescheduling, which together establish accuracy limits of estimates [5]. While the net influence of workload redistribution may be small for some systems, an object lesson can be made of an *n*-processor

---

 Partially sponsored by the Defense Advanced Research Projects Agency, ARPA Task No. 9157, and the Department of Energy, DoE Order No. DE-AI05-87ER25046.

MIMD shared-memory machine for which demand changes do matter. The context is that of individually exchanging regular processors for faster ones to accelerate Quicksort execution: It happens that upgrading many of the processors may give a poor return on investment.

Upgrading k of n Processors

Imagine a multiprocessor with *n* identical processor units. In this case, modes can differentiate solely on the *number* of active processors. Let *i* denote a mode of *computing with i processors*, the other *n-i* being idle. Capacities $\{r_i\}$ then summarize average system rates for multiprocessing levels, e.g., r_{12} is the average collective rate for 12 active processors. The r_i have been normalized such that

$$R_{original} = \left[\sum_{i=1}^{n} \frac{\alpha_i}{r_i} \right]^{-1} = 1 \qquad (comparison\ basis)$$

$$constrained\ by \sum_{j=1}^{n} \alpha_j = 1 \qquad (full\ workload)$$

Suppose that *k* of the *n* processors can each be boosted to an improved processing rate $1+\Delta$ faster than the original rate retained by all others. Overall improved performance then hinges upon rebalancing disparities in processor loading. Two example scheduling strategies are examined, one that uses the added power and another that ignores it except for α_1, the serial mode.

Scheduling I: Ideal Balancing within a Mode. One "best case" assumes a flexible system with fine-grained load balancing characteristics. The system redistributes computing *within a mode* with no noticeable overhead. This ensures that all *k* improved processors are engaged to the degree *that an application permits*, and is denoted $R_{(k)}$. A shared-memory design might approach this ideal improvement, whereas a loosely-coupled system probably cannot unless computation grain is coarse enough to mask communication latencies. It is assumed that algorithmic constraints prevent coefficients α_j from changing. However, any workload portion that runs with j parallel processors, $j > k$, nonetheless speeds up to the extent offered by *k* faster processors. This assumption appears in the new collective rate for mode j, which is $r_j(1+[k/j]\Delta)$. (Only k of the j are improved.) It may require almost magical reassignments among faster and slower processors to keep various portions from getting ahead and starving, which would sacrifice some workload to a lesser mode. The following holds:

$$R_{(k)} = \left[\sum_{i=1}^{n} \frac{\alpha_i}{r_i \left(1 + \frac{min(k,\, i)}{i} \Delta \right)} \right]^{-1}$$

Scheduling II: Only Serial Speedup. Let $R_{(0)}$ denote the worst scheduling, which systematically excludes fast processors whenever possible. An intuitive argument for this redistribution predicts no improvement except when all n processors have been upgraded. Otherwise there is always some slow processor that delays completion. While a detailed argument is possible that accounts for details of the coefficients (see the appendix), its predictions hardly differ in usual circumstances from the intuitive view. Thus $R_{(0)} \approx R_{original} = 1$, even with k of n processors faster by $1+\Delta$.

Strategy $R_{(0)}$ is obviously deficient for reasonable cases. Improvement $R_{(1)}$ lets the scheduler dispatch all serial workload (mode 1) on the k faster processors. The effort to do this is usually minimal, there being no other processes to interfere. An exception would be exceptionally brief serial sections spread scattershot across all processors; here, process relocation overhead might be high. The example in the next section does not have this problem. It has around 20 serial epochs, each averaging 90 ms duration. $R_{(1)}$, profitable because most programs have some region of serial bottleneck, is assumed henceforth to be the worst case scheduling. This can be expressed as

$$\frac{1}{R_{(1)}} = \frac{1}{R_{(0)}} - \frac{\alpha_1}{r_1} + \frac{\alpha_1}{r_1(1+\Delta)} \approx 1 - \frac{\alpha_1}{r_1} + \frac{\alpha_1}{r_1(1+\Delta)}$$

Example: Actual Sort Workload

Scheduling performances really depend upon coefficient sets. For this example, the machine and its normalized capacities are modeled as a shared memory system without much memory or bus contention. Each added processor diminishes overall performance only a half percent, to 99.5% what it would otherwise be. With 16 processors, this limits efficiency to 92.8% of the sum of individual processors. A parallel Quicksort illustrates, for data sets of 31000 values, some consequences of a divide-and-conquer paradigm. On a 16-processor computer with shared memory, Quicksort demand coefficients α_i are typically small except for the 16-processor mode, α_{16}. All demand coefficients (column 3, below) are actual values measured via special low-perturbation methods [3, esp. Fig. 4].

Number of Processors, Mode i	Normalized Capacity, r_i	Resource Demand, α_i
		20 trial ave.
01	.111	.026
02	.222	.023
04	.440	.036
08	.862	.057
10	1.067	.001
12	1.268	.001
14	1.465	.002
16	1.657	.854

Capacity-and-Use Profile--Parallel Quicksort

Estimates are made for improvement in one, two, and four of the processors. Processor speedup ranges from three to nine. Best and worst scheduling cases ($\pm x$ in table below) are established via $R_{(k)}$ and $R_{(1)}$, respectively. These scheduling tolerances should be assessed relative to other sources of variation. For instance, demand fluctuates from trial to trial, the greatest change being when workload is exchanged between α_{16} and α_1 (max. and min. processing rates). Since $\alpha_1+\alpha_{16} = 0.88$ and $R_{original} = 1$

$$\frac{\partial R_{original}}{\partial \alpha_{16}} = -R_{original}^{-2} \frac{\partial}{\partial \alpha_{16}} \left[\frac{0.88 - \alpha_{16}}{r_1} + \frac{\alpha_{16}}{r_{16}} \right] = \frac{1}{r_1} - \frac{1}{r_{16}} = 8.4$$

Looking at *relative* changes, $(\partial R/\partial \alpha_{16})(\alpha_{16}/R) = (8.4)(0.854/1) = 7.2$. Consequently, even minor coefficient fluctuations among trials pose large performance uncertainties; actual rangings of α_{16} over 20 trials were +3%, -2% about its mean. In this light, scheduling tolerances are acceptable.

16 Processors	Improved, Faster Processor		
Improved: Base	*3x (Δ=2)*	*5x (Δ=4)*	*9x (Δ=8)*
4:12	1.55 ± 24%	1.99 ± 38%	2.79 ± 55%
2:14	1.41 ± 16%	1.67 ± 27%	2.13 ± 41%
1:15	1.31 ± 10%	1.47 ± 17%	1.73 ± 27%

Relative Improvements, Sort Workload

Minimal performance gain for any configuration is determined by serial speedup in the worst case scheduling, since $R_{(1)}$ assumes faster processors only help mode 1. Consequently, each *column* has the same minimal performance. More than one faster processor is generally no help. Substituting a 3x faster processor adds the equivalent of 2 processors, or 2/16=12.5%, to the original configuration. This plus alternate substitutions of 5x and 9x in the 1:15 configuration (additions of 12.5, 25, and 50%

overall) produce guaranteed gains of 18, 23 and 26%, respectively. Clearly, an economical improvement for minimal scheduling performance is to substitute *one moderately faster processor.*

Maximum performance does benefit from numerous faster processors. Each super-, main, and sub-diagonal of the table matrix identifies configurations that have equivalent processor power. That is, four processors boosted to 3x (extra capacity of 2x each) equals two processors 5x faster (extra capacity of 4x each). But yield along a diagonal is unequal for two reasons: (i) coefficients $\{\alpha_i\}$ are unequal, and (ii) $R_{(k)}$ does not change demand coefficients and distribute work among more processors. A single faster processor can (by earlier assumption) help serial or parallel modes, whereas y processors each improved to a lesser degree cannot help a mode $j, j < y$, *as much.*

The modest processing rate model raises issues and invites discussion. For instance, the example schedulings do bias performance somewhat toward substitution of one faster processor. The model probably exaggerates the efficacy of load balancing mechanisms for shared-memory machines. An opposing factor that has been completely ignored is the nonlinear cost of upgrading processors. A 9x faster processor is likely significantly more expensive than two 5x units, this very fact being a strong economic motivation behind parallel architectures.

Acknowledgment. Thanks to D. Dimmick for measuring the Quicksort performance on our group's specially instrumented, shared-memory machine, and to R. Snelick for supplying various codes. R. Carpenter and A. Mink questioned numerous points and made suggestions.

Citations

[1] G.M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proc., AFIPS Spring Joint Computer Conference 1967*, Atlantic City, NJ, April, 1967, 483-485.

[2] I.Y. Bucher, "The computational speed of supercomputers," Report LA-UR-84-740, Los Alamos National Laboratory, Los Alamos, New Mexico, 1984, 15pp.

[3] R.J. Carpenter, "Performance measurement instrumentation for multiprocessor computers," in *High Performance Computer Systems*, E. Gelenbe (ed.), North-Holland, 1988, 81-92.

[4] D.L. Eager, J. Zahorjan, and E.D. Lazowska, "Speedup versus efficiency in parallel systems," *IEEE Trans. on Computers 38*, 3(March 1989), 408-423.

[5] G.E. Lyon, "Capacity-and-use trees for estimating computer performance variations," *Proc., Int. Conf. on Computers and Inf, ICCI'89,* Vol. 2, Toronto, May, 1989, 309-313. Extended version, "Hybrid structures for simple computer performance

estimates," Report NISTIR 89-4063, NIST, Gaithersburg, Md. 20899, 1989, 24pp.

[6] K.C. Sevcik, "Characterizations of parallelism in applications and their use in scheduling," *Proc., 1989 SIGMETRICS Conf.*, Berkeley, Ca., May 1989, 171-180.

[7] W.H. Ware, "The ultimate computer," *IEEE Spectrum*, 84-91, (March 1972).

[8] R.G.S. White, "Rating scale estimates automobile drag coefficient," *SAE Journal 77*, 6(June, 1969), 52-53.

Appendix: Lower Bound for Worst Case Scheduling

The "worst case", $R_{(0)}$, can be refined considerably via a lower bound. As before, there are *k* faster processors and *n-k* of the originals. Let P_j be a *partition* of the workload actually handled by j active processors, i.e., all demand serviced in mode j. Argument begins with original *workload partitions* P_1 through P_{n-k-1} unchanged and assumes that these do not involve improved processors. (Remember that $R_{(0)}$ excludes faster processors whenever it can.) Partition P_{n-k} is special; it is the partition of largest index that does not have to account directly for faster processors, although workload will be added to it. In contrast, partitions P_{n-k+1} to P_n suffer degenerate circumstances that cause parts of their workload to fall into other partitions (due to idle processors). Because some processors handling workload in these latter partitions must be faster versions (there are but *n-k* originals), there will be gaps in available load. $R_{(0)}$ does not try to rebalance these pauses, so there is idling. Consequently, some workload in original partition P_j no longer belongs there, but instead is assigned to a partition whose lesser index signifies fewer active processors. For calculation convenience this residual portion is clumped at the end of P_j's processing, as if all improved processors idle simultaneously. Computing for this residual is attributed to partition P_{n-k}. The time free of idling can be expressed as $\alpha_j/(r_j(1+\Delta))$, *i.e.*, one assumes that *all* j processors run faster and thereby completely shorten the partition processing time. In reality this is but a convenience for calculation. It isolates that fraction of workload *not* processed in mode j:

$$\alpha_j \left[\frac{\Delta}{1+\Delta}\right] \left[\frac{n-k}{j}\right]$$

The left set of parentheses express a residual fraction of dimension proportional to rate. Faster processors of P_j finish sooner, leaving this fraction for the slower processors to complete in mode n-k. The right set of parentheses expresses another dimension, that fraction of processors identified with another partition because they are not idle. These are *n-k* in number--all the slow processors--of the j processors overall.

Assume that contention and other factors of marginal efficiency render $r_i / i \geq r_{i+1} / (i+1)$. Then the worst possible per unit processing rate for the dropout load from P_j is that from the previous partition (j-1). This is scaled up for n-k processors of P_{n-k}. Being pessimistic (for a lower bound), the processing rate for the partition dropout is:

$$\frac{r_{j-1}(n-k)}{j-1} \quad \textit{which is} \leq r_{n-k}$$

Terms of relative time for the higher-numbered partitions are now expressed as:

$$\sum_{j=n-k+1}^{n} \frac{\alpha_j}{r_j\,(1+\Delta)} + \sum_{j=n-k+1}^{n} \alpha_j \left[\frac{\Delta}{1+\Delta}\right] \left[\frac{n-k}{j}\right] \left[\frac{j-1}{r_{j-1}(n-k)}\right]$$

This establishes a lower bound for the relative rate. Add terms for the first n-k partitions and invert,

$$\textit{assuming that } \frac{r_i}{i} \geq \frac{r_{i+1}}{i+1} \text{ and } k < n,$$

$$R_{(0)} \geq \left[\sum_{i=1}^{n-k} \frac{\alpha_i}{r_i} + \sum_{j=n-k+1}^{n} \frac{\alpha_j}{1+\Delta} \frac{j\,r_{j-1} + (j-1)\,\Delta\,r_j}{j\,r_{j-1}\,r_j} \right]^{-1} \geq R_{original}$$

Note that whenever

$$\frac{r_i}{i} = \frac{r_{i+1}}{i+1}$$

for any $1 \leq i < n$, (scale-up with no processing loss), then $R_{(0)} = R_{original}$, the intuitive "worst case" of the text. Also, whenever $n = k$, there are no new opportunities for either strategy, since all processors are improved models, and $R_{(k)} = R_{(0)}$. As a cautionary note,

$$\textit{some conditions of } \frac{r_i}{i} > \frac{r_{i+1}}{i+1} \textit{ determine that } R_{(0)} > R_{(k)}.$$

Such is the curious efficacy of strategy $R_{(0)}$! It restricts amounts of parallelism. Whenever added processors are too parasitic, they do not pay their way, and the viewpoint of $R_{(0)}$ is productive.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NISTIR 89-4128	2. Performing Organ. Report No.	3. Publication Date AUGUST 1989

4. TITLE AND SUBTITLE

Processing Rate Sensitivities of a Heterogeneous Multiprocessor

5. AUTHOR(S)

Gordon Lyon

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Defense Advanced Research Projects Agency, Arlington, VA 22209

Department of Energy, Washington, DC 20545

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

A recent trend in multiprocessor evaluation has been to seek fundamental but easily parameterized performance characterizations. Freed of specialized detail, simplified models can convey good insight while applying easily and widely.

A simple performance estimator and alternate scheduling schemes can, from very modest effort, highlight some first-order improvement tradeoffs in multiprocessors. An application example of a quicksort is used to demonstrate the approach.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

application signature; architecture; capacities; models; performance; sensitivities

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

11

15. Price

$9.95

USCOMM-DC 6043-P80