NASA CONTRACTOR REPORT CR177342

Vector Computer Memory Bank Contention

David H. Bailey

CONTRACT NAS2–11555
February 1985

NASA

NASA CONTRACTOR REPORT CR177342

Vector Computer Memory Bank Contention

David H. Bailey
Informatics General Corp.
Palo Alto, California

Prepared for
NASA Ames Research Center
Contract NAS2-11555

National Aeronautics and
Space Administration

Ames Research Center
Moffett Field, California 94035

Vector Computer Memory Bank Contention

David H. Bailey

February 20, 1985

Abstract

A number of recent vector supercomputer designs have featured main memories with very large capacities, and presumably even larger memories are planned for future generations. While the memory chips used in these computers can store much larger amounts of data than before, their operation speeds are rather slow when compared with the significantly faster CPU (central processing unit) circuitry in new supercomputer designs. A consequence of this speed disparity between CPUs and main memory is that memory access times and memory bank reservation times (in CPU ticks) are sharply increased from previous generations.

While it has been recognized that these longer memory operation times would reduce scalar performance, it has not been generally realized that vector performance could suffer as well, due to a sharp increase in memory bank contention. This paper will examine this phenomenon using both a Markov chain mathematical model and a Monte Carlo simulation program. The potential for performance reduction will be described and techniques for ameliorating this reduction will be proposed.

One significant conclusion of this analysis is that the number of independent memory banks necessary to preserve a constant level of memory efficiency is (approximately) proportional to the number of CPUs times the square of the memory bank reservation time (in ticks). As a result, it appears that future generations of supercomputers will either have to employ memory chips with significantly faster operation speeds or else feature much larger numbers of independent memory banks.

The author is an employee of Informatics General Corp., under contract to the NAS (Numerical Aerodynamic Simulation) program at NASA Ames Research Center. This work was performed under contract NAS 2-11555.

Introduction

In recent years advances in fields such as computational fluid dynamics and plasma physics have outstripped the main memory capacity of even the largest scientific computer systems. Furthermore, users have found that using disk drives or other external storage media for temporary data storage in these large problems is seldom satisfactory, as it often increases their wall clock run time by several orders of magnitude. Thus many scientific programmers are now clamoring for vector supercomputers with vastly increased main memory.

Fortunately for such users, the semiconductor industry has been remarkably successful in recent years in producing memory chips with burgeoning capacity. 256 kilobit chips are now readily available from suppliers, and prototypes of one megabit chips have recently been displayed. Thus it is not too surprising that a number of recently announced supercomputers have featured main memories as large as 256 million 64-bit words, and presumably even larger memories are in the works for the next generation.

While the emphasis in the development of memory chips has been increased capacity, the emphasis in the design of supercomputer CPU circuitry has been increased speed. CPU clock "ticks" of ten nanoseconds or so are now commonplace, and supercomputers with four nanosecond or even one nanosecond CPU cycle times are on the horizon. This disparity in speed between CPU circuitry and memory bank circuitry means that the memory bank reservation time and the memory access time, as measured in CPU clock ticks, are sharply increased for new supercomputers. While it has been recognized for some time that these long operation times would lower the scalar performance of supercomputers, it is only recently that the potential for vector performance reduction has come to light.

The reason for this potential reduction in vector performance is memory bank contention – that is, delays encountered when a CPU attempts to access a bank of main memory that has been reserved from a previous access by another (or even the same) CPU. This article will analyze the phenomenon of memory bank contention and discuss both the potential for performance reduction and techniques for ameliorating this reduction.

A Markov Chain Model for Memory Bank Contention

The memory bank operation of a multiprocessor vector computer system may be approximately modeled using a relatively simple Markov chain model. While such a model cannot precisely describe the phenomenon of memory bank contention in a real vector computer, it does serve as a good introduction to the problem, and in fact some quantitative conclusions can be drawn from this simple model that do carry over to a more realistic model.

In order to facilitate analysis, certain simplifying assumptions will be made. It will be assumed that the computer system being modeled has m CPUs and n banks of interleaved memory (i.e., successive data words are in successive memory banks). Whenever one of the CPUs accesses a word of memory (either to store or recall), a reservation of t ticks is

placed on the bank containing that word. This means that for the next t system ticks, any CPU wishing to access a word in that bank of memory must wait before it may complete its access. At every system clock tick, it is assumed that each CPU that is not waiting tosses a coin with probability of heads equal to r, and attempts to access memory (from a memory bank chosen at random) if the coin turns up heads. It will be assumed that when a CPU attempts to access to a bank that is busy from a prior reservation, its remaining reservation is uniformly distributed between 1 and t. The infrequent case where more than one CPU is waiting to access a single reserved bank will be ignored for the time being. A final approximating assumption is that the fraction of memory banks that are busy at any time is approximately a constant x. Such an assumption may be made assuming that the process has achieved a steady state.

It should be mentioned that in real vector computer operation, a CPU is typically either attempting to access memory cells every tick, as part of a long vector fetch or store, or else it is "crunching" and not attempting to access memory at all. Further, most memory accesses are from consecutive memory banks, instead of from randomly chosen memory banks. This is the most serious deviation from a real vector computer system in the model, but it appears to be necessary to make such an assumption in order to render the model tractable for exact solution.

The operation of each CPU may now be approximately modeled by a Markov chain on the $t+1$ states $s_0, s_1, s_2, \ldots, s_t$. Here s_0 denotes the free state and s_k denotes the state of waiting for a bank that has a reservation of k ticks remaining. Let T denote the Markov transition matrix T for this model (i.e., T_{ij} is the probability that the next state is j, given that the current state is i). Then T may be written as

$$T = \begin{pmatrix} 1-rx & rx/t & rx/t & \cdots & rx/t & rx/t \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}$$

It may easily be verified that the Markov chain described by this transition matrix is a regular (ergodic) process. This means that the *a priori* probability of any state is equal to the limiting frequency of appearance of that state (for almost every sample sequence). Let $p = (p_0, p_1, p_2, \ldots, p_t)$ denote the vector of *a priori* probabilities of the $t+1$ states. These probabilities may be determined from the relationship $pT = p$ (see [11], p. 72). This equivalence yields the linear system of equations

$$
\begin{array}{lcll}
p_0(1-rx) & + & p_1 & = p_0 \\
p_0 rx/t & + & p_2 & = p_1 \\
p_0 rx/t & + & p_3 & = p_2 \\
 & & \vdots & \\
p_0 rx/t & + & p_t & = p_{t-1} \\
p_0 rx/t & & & = p_t
\end{array}
$$

When combined with the fact that the probabilities p_k must sum to one, the solution is easily found to be

$$
\begin{aligned}
p_0 &= \frac{1}{1+rx(t+1)/2} \\
p_1 &= \frac{rx}{1+rx(t+1)/2} \\
p_2 &= \frac{rx(t-1)}{t[1+rx(t+1)/2]} \\
&\vdots \\
p_{t-1} &= \frac{2rx}{t[1+rx(t+1)/2]} \\
p_t &= \frac{rx}{t[1+rx(t+1)/2]}
\end{aligned}
$$

Since it was assumed that the fraction x of banks that are in a reservation cycle is constant, the expected number of banks initially reserved at any instant must equal the number whose reservation expires at that instant. This can be expressed by the relation

$$rmp_0 = nx/t$$

where it is assumed that at each time $1/t$ of the busy banks are freed. This relation combined with the above yields the solution

$$x = \frac{\sqrt{1+2mr^2t(t+1)/n} - 1}{r(t+1)}$$

so that

$$p_0 = \frac{2}{1 + \sqrt{1 + 2mr^2t(t+1)/n}}$$

The remaining p_k can be similarly calculated.

Now that the probability vector p has been found, a memory efficiency statistic may be calculated. Let E denote the ratio of the expected number of memory accesses divided by the sum of this figure and the expected time spent in wait states. This efficiency statistic can be written as

$$E = \frac{rp_0}{rp_0 + (1 - p_0)}$$
$$= \frac{2r}{2r - 1 + \sqrt{1 + 2mr^2t(t+1)/n}}$$

Implications of the Markov Chain Model

The efficiency statistic derived in the previous section from the Markov chain model does not, unfortunately, agree closely with most actual vector supercomputer operation. The main problem appears to be, as mentioned above, that most vector computer memory accesses are from consecutive banks (or at least from banks differing by some constant stride) instead of from randomly chosen banks. Even so, some conclusions can be drawn from this Markov chain result that do carry over to the more realistic model described below.

First of all, one can conclude from the formula above for the memory efficiency that if the number of processors m is increased by a factor q, then the number of banks n must also be increased by a factor q to preserve the same level of efficiency. Secondly, if the bank reservation time t is increased by a factor q, then the number of banks must be increased by a factor of about q^2 to maintain the same memory efficiency. As it turns out, these two relations also hold (to a good approximation) in the following more detailed model.

Monte Carlo Simulations of Memory Bank Contention

A more sophisticated (and realistic) model of memory bank contention will now be presented. Above it was assumed that each free CPU tosses a coin with a certain probability and attempts to access a single randomly chosen memory bank if the coin turns up heads. It will now be assumed that each free CPU instead initiates a vector access (fetch or store) of a certain length if its coin turns up heads. The starting bank number for this vector access is assumed chosen at random, but thereafter the bank number advances with some constant stride through the duration of the vector access. The length of the vector access is assumed chosen at random according to a distribution that is uniform on the set $\{1, 2, \ldots, V\}$, except that a specified larger fraction v of the vector lengths have the

maximum value V. Similarly, the memory stride is assumed to be chosen from a uniform distribution on the set $\{1, 2, \ldots, n\}$, except that a certain specified larger fraction s of the strides are 1.

It should be noted that strides greater than n do not need to be considered, because such strides are equivalent for our purposes to their remainder when divided by n. It should also be noted that the mean restart time R between vector accesses is merely the reciprocal of the coin toss probability r, a fact that can be easily demonstrated from elementary probability theory. Unlike the Markov chain model, this model will not ignore the case where two or more CPUs are waiting to access the same memory bank – it will be assumed that the CPUs merely take turns until all accesses have been completed.

This model is not intended to exactly mimic the memory operation of any actual supercomputer. Instead it is intended to enable the general problem of memory bank contention to be simulated and analyzed. However, variations of this model have been shown to quite closely mimic a number of real computers. For example, the Cray-2 memory has been successfully analyzed [1] by using this basic model with an enhancement that mimics the operation of the Cray-2 quadrants. The Cray X-MP/48 memory has similarly been studied using this model with enhancements that handle the multiple memory ports from each CPU.

Unfortunately, it is does not seem possible to analyze this model with the elementary Markov chain techniques of the previous section. It is possible, however, to run Monte Carlo simulations based on such a model. Such a simulation program has been written, and numerous runs with it have been made on the Cray X-MP/12 belonging to the NAS (Numerical Aerodynamic Simulation) program at the NASA Ames Research Center. Each separate assumption of the above parameters was run for one million ticks, enough for the resulting empirical efficiency figures to be reliable to within a percent or two.

Results of the Monte Carlo Simulation Runs

Several plots displaying important simulation results are shown in the pages following the end of the article. Except where indicated otherwise, these results are for the case $n = 256$, $m = 4$, $V = 128$, $R = 100$, $t = 40$, $v = 0.75$, $s = 0.75$. These parameters were chosen for a "generic" vector computer, roughly a composite of a number of current and projected supercomputers.

Figure 1 shows how the memory efficiency E decreases as the reservation time t increases. The four separate curves represent results for various numbers of CPUs. Figure 2 shows how efficiency increases as the fraction s of unit stride varies from zero to one. Each curve in this figure represents results for different reservation times. Figure 3 shows how efficiency decreases with large numbers of processors. The four curves on this figure are for different numbers of banks. Figures 4 and 5 present a different slant on the problem: with other parameters held fixed, the number of banks necessary to preserve a constant level of memory efficiency (75%) is shown as a function of increasing reservation time (figure 4) and as a function of increasing numbers of processors (figure 5). In figure 4 the separate

curves represent results for different numbers of banks, and in figure 5 each curve gives results for different reservation times.

Several definite trends can be quickly identified from these plots. First of all, from figure 5 it is clear that the relationship between banks and processors is exceedingly close to linear – in fact the number of banks necessary to compensate for an increasing number of processors appears to be very closely proportional to the number of processors minus 1. This relation, except for the minus 1, matches the relation found in the Markov chain analysis above. Secondly, although it is not immediately clear from figure 4, logarithmic regression of the simulation results shows that the number of banks necessary to compensate for an increase in the bank reservation time t is proportional to approximately $t^{1.85}$. The corresponding relation from the Markov chain analysis is $t(t+1)$, which is equivalent to approximately $t^{1.06}$ over the range of the data in question. Relationships quite close to these were also found in other cases that were run with the simulator program.

The reduction of memory efficiency whenever the fraction of strides that are equal to one is not 100% (see figure 2) presents a dilemma of sorts to designers of supercomputers. It is clear that significantly less memory bank contention would result by designing hardware that does not allow strides other than one on most vector memory accesses. This approach has been taken, for example, by CDC in its Cyber 205 design. However, such a restriction reduces the ability of a computer to efficiently process Fortran data arrays by other than the first dimension. As a result most supercomputer users, particularly those who run codes with large multidimensional arrays, feel that a variable memory stride is a definite advantage in a vector computer design. Nevertheless, it clear from these simulation results that memory efficiency will be lower with a variable stride architecture.

Conclusions

The analysis of the phenomenon of memory bank contention does indeed indicate the potential for substantial reductions in performance in new generations of supercomputers. For example, suppose a supercomputer were to be designed with eight central processing units and a two nanosecond clock. A number of the current technology DRAM (dynamic random access memory) chips now in production dictate a bank reservation time of roughly 160 nanoseconds, or 80 ticks. According to simulation runs based on the generic vector computer model described above, more than 10000 memory banks would be necessary to achieve an average efficiency of roughly 75%. This number is considerably greater than the 64 or 128 that characterize current designs. Thus it appears that future generations of vector computers must either be designed with memory chips substantially faster than those available today, or else they must feature much larger numbers of independent banks of memory. Failure to address this problem will result in catastrophic performance reductions.

In the future, it is likely that memory chips significantly faster than today's typical DRAM chips will be available for supercomputer memories. For example, a number of supercomputers feature static RAM chips with faster operation speeds than dynamic RAM chips. However, such fast chips cost considerably more and have only a fraction of the

capacity of equivalent generation DRAM chips. This pattern can be expected to continue for the foreseeable future. Thus it seems probable that future designers and purchasers of supercomputers will have to make painful tradeoffs between performance and memory size – computer systems may be available with either a smaller memory of faster chips and minor memory contention, or with a much larger memory of slower chips and substantial memory contention.

Increased memory size does of course have a number of advantages. In addition to the ease of programming large-scale scientific application codes, larger memories reduce the amount of time a supercomputer must spend transferring jobs in and out or waiting for I/O requests to be handled. Even with these advantages, though, it seems clear that computational throughput will generally be degraded with the larger (slower) memories.

One possible solution to this dilemma is to design vector supercomputers with large main memories of the slower chips, but with a "cache" of much faster memory private to each CPU. With such a scheme some programs could still access very large data arrays, while other programs not requiring a large amount of data storage could use the cache memory, thus lessening the competition for main memory banks. However, problems arise with this design also. For one thing, swapping jobs in and out of a CPU would be much more time consuming, due to the extra time required to save the entire cache memory. In addition, nonstandard constructs may be required for the Fortran programmer to control which type of memory is assigned to his or her data arrays. Finally, unless a fairly large amount of cache memory is provided, most programs will be unable to perform a significant amount of their required computation using cache memory.

In any event, it is clear that both manufacturers and potential users of supercomputers must pay close attention to the problem of memory bank contention. It is thus hoped that the techniques described in this paper will be of assistance to such persons and will help prevent unacceptable reductions in supercomputer performance.

References

1. Bailey, D. H., and Barton, J. T., *Cray-2 Memory Performance Analysis*, NAS Technical Memorandum 101-JTB, October 15, 1984.
2. Baskett, F., and Smith, A. J., "Interference in Multiprocessor Computer Systems with Interleaved Memory", *Communications of the ACM*, vol. 19, no. 6 (Jun. 1976), p. 327-334.
3. Bhandarkar, D. P., "Analysis of Memory Interference in Multiprocessors", *IEEE Transactions on Computers*, vol. C-24, no. 9 (Sep. 1975), p. 897-908.
4. Briggs, F. A., and Davidson, E. S., "Organization of Semiconductor Memories for Parallel-Pipelined Processors", *IEEE Transactions on Computers*, vol. C-26, no. 2 (Feb. 1977), p. 162-169.
5. Briggs, F. A., and Dubois, M., "Effectiveness of Private Caches in Multiprocessor Systems with Parallel-Pipelined Memories", *IEEE Transactions on Computers*, vol. C-32, no. 1 (Jan. 1983), p. 48-59.
6. Burnett, G. J., and Coffman, E. G., "Analysis of Interleaved Memory Using Blockage Buffers", *Communications of the ACM*, vol. 18, no. 2 (Feb. 1975), p. 91-95.
7. Chang, D. Y., Kuck, D. J., and Lawrie, D. H., "On the Effective Bandwidth of Parallel Memories", *IEEE Transactions on Computers*, vol. C-26, no. 5 (May 1977), p. 480-490.
8. Chung, K. L., *Markov Chains*, Springer-Verlag, New York, 1967.
9. Flores, I., "Derivation of a Waiting Time Factor for a Multiple Bank Memory", *Journal of the ACM*, vol. 11, no. 3 (Jul. 1964), p. 265-282.
10. Hoogendoorn, C. H., "A General Model for Memory Interference in Multiprocessors", *IEEE Transactions on Computers*, vol. C-26, no. 10 (Oct. 1977), p. 998-1005.
11. Kemeny, J. G., and Snell, J. L., *Finite Markov Chains*, D. Van Nostrand Co., Princeton, NJ, 1963.
12. Knuth, D. E., and Rao, G. S., "Activity in an Interleaved Memory", *IEEE Transactions on Computers*, vol. C-24, no. 9 (Sep. 1975), p. 943-944.
13. Patel, J. H., "Analysis of Multiprocessors with Private Cache Memories", *IEEE Transactions on computers*, vol. C-31, no. 4 (Apr. 1982), p. 296-304.
14. Ramamoorthy, C. V., and Wah, B. W., "An Optimal Algorithm for Scheduling Requests on Interleaved Memories for a Pipelined Processor", *IEEE Transactions on Computers*, vol. C-30, no. 10 (Oct. 1981), p. 787-800.
15. Rau, B. R., "Program Behavior and the Performance of Interleaved Memories", *IEEE Transactions on Computers*, vol. C-28, no. 3 (Mar. 1979), p. 191-199.
16. Ravi, C. V., "On the Bandwidth and Interference in Interleaved Memory Systems", *IEEE Transactions on Computers*, vol. C-21, no. 8 (Aug. 1972), p. 899-901.

17. Sastry, K. V., and Kain, R. Y., "On the Performance of Certain Multiprocessor Computer Organizations", *IEEE Transactions on Computers*, vol. C-24, no. 11 (Nov. 1975), p. 1066-1074.

18. Smith, A. J., "Multiprocessor Memory Organization and Memory Interference", *Communications of the ACM*, vol. 20, no. 10 (Oct. 1977), p. 754-761.

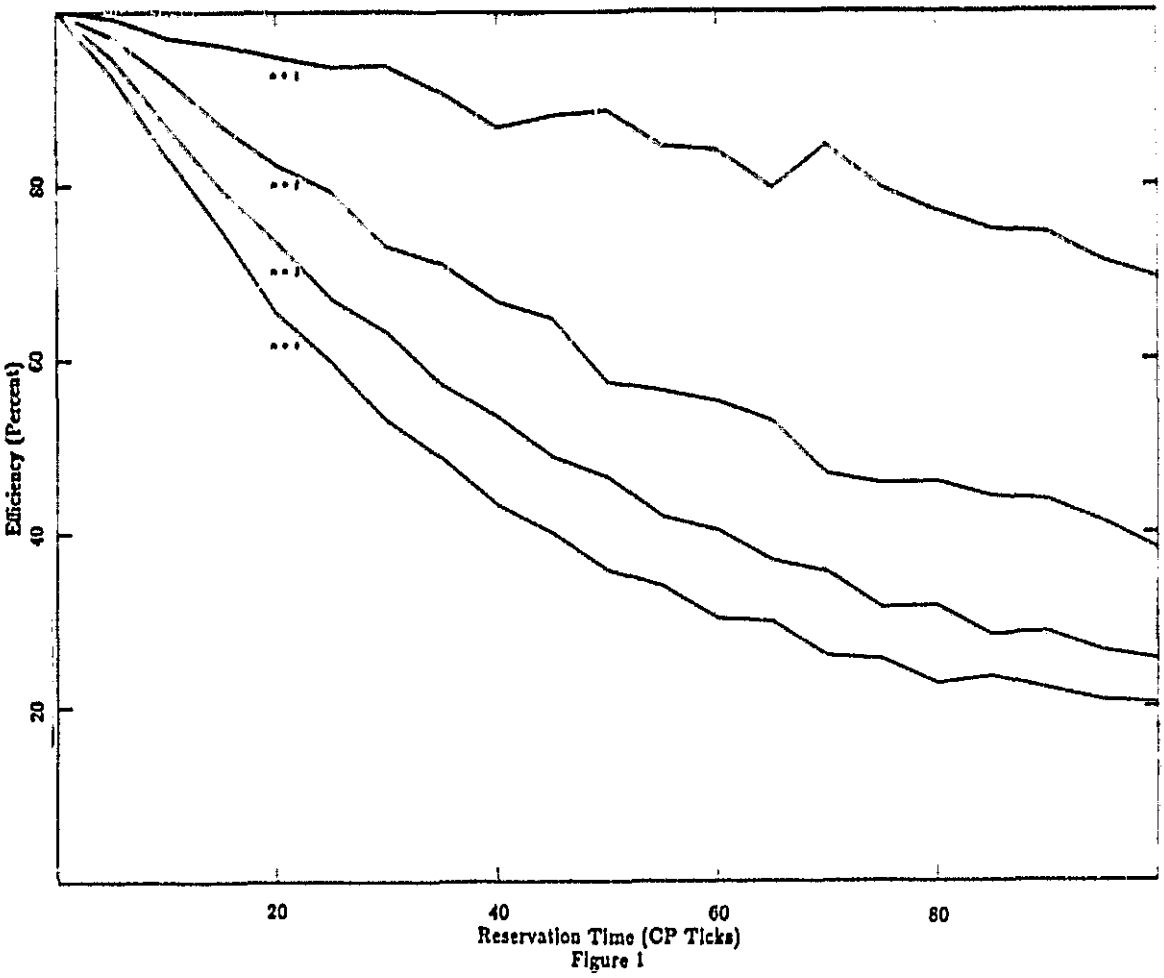

Figure 1

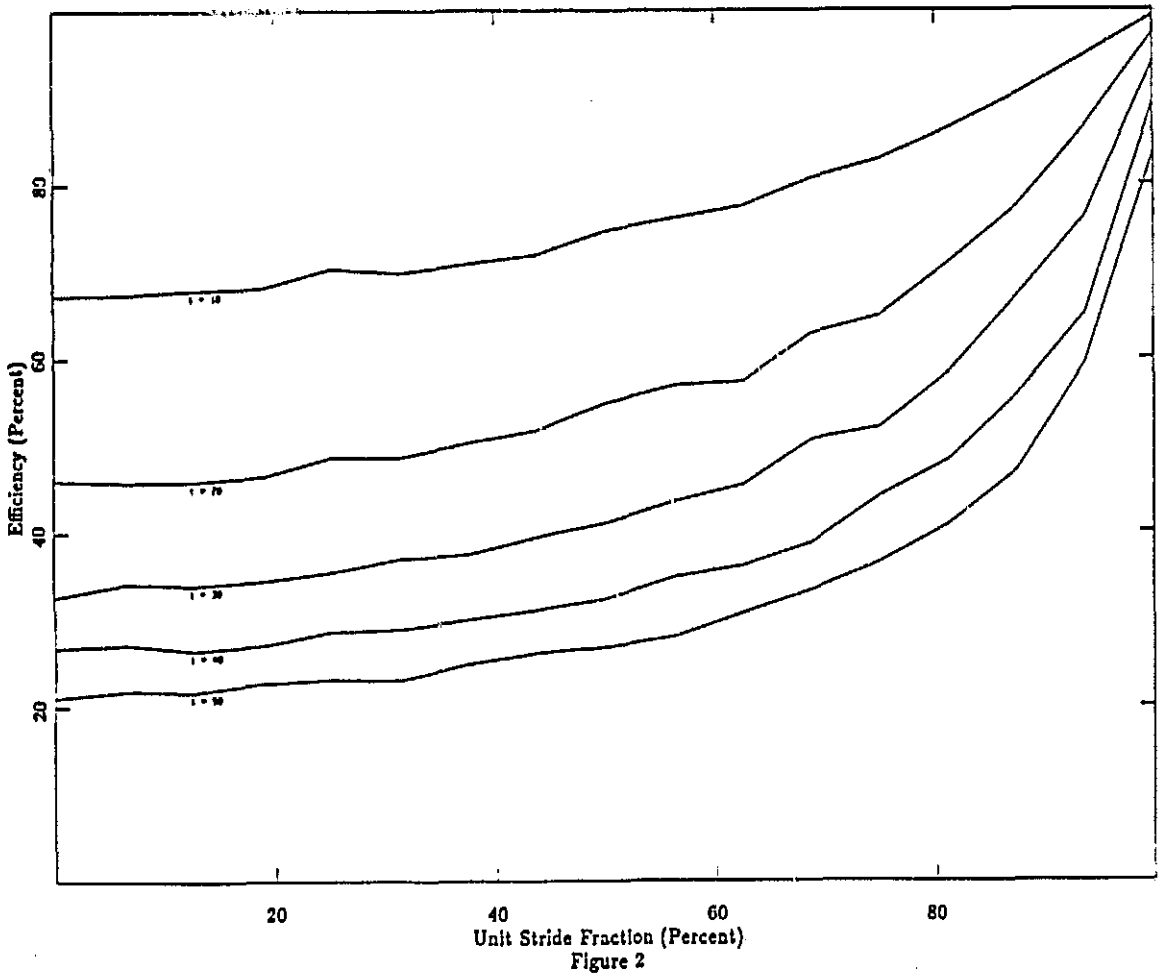

Figure 2

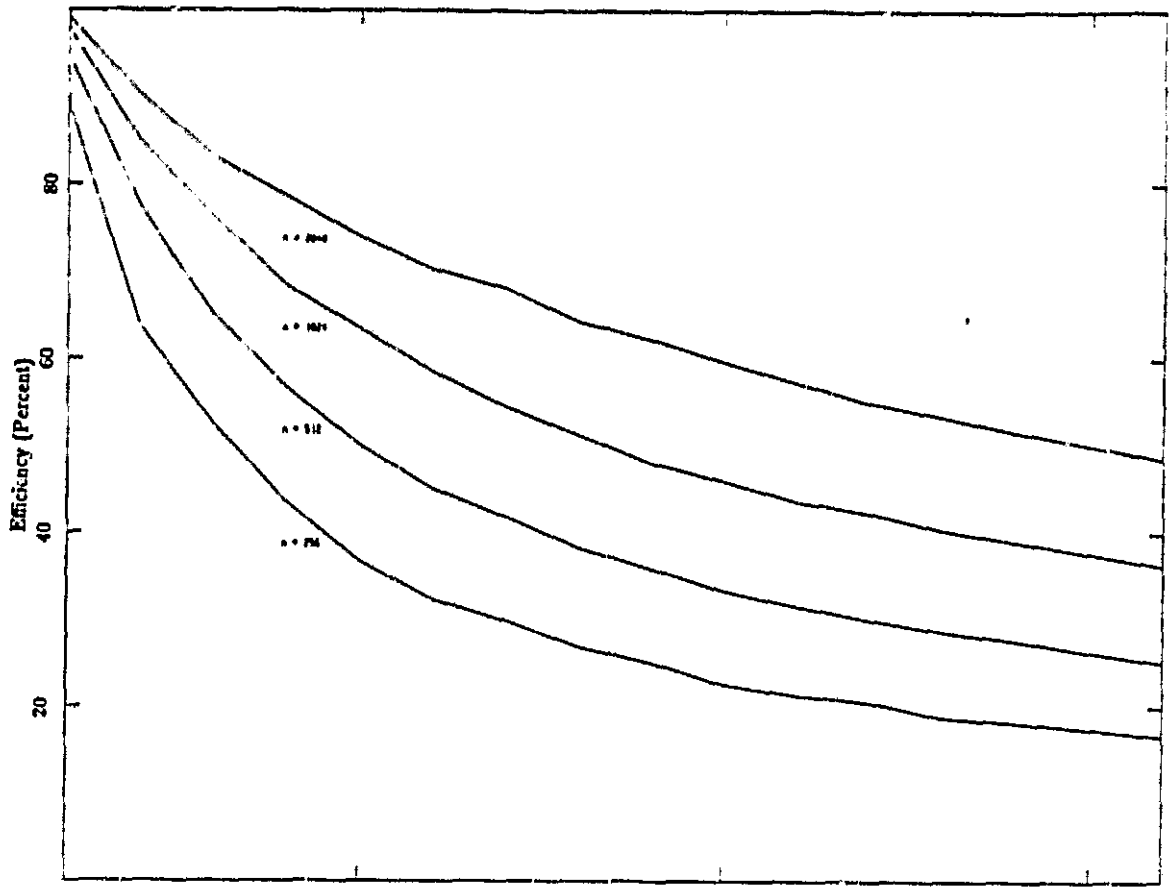

Figure 3

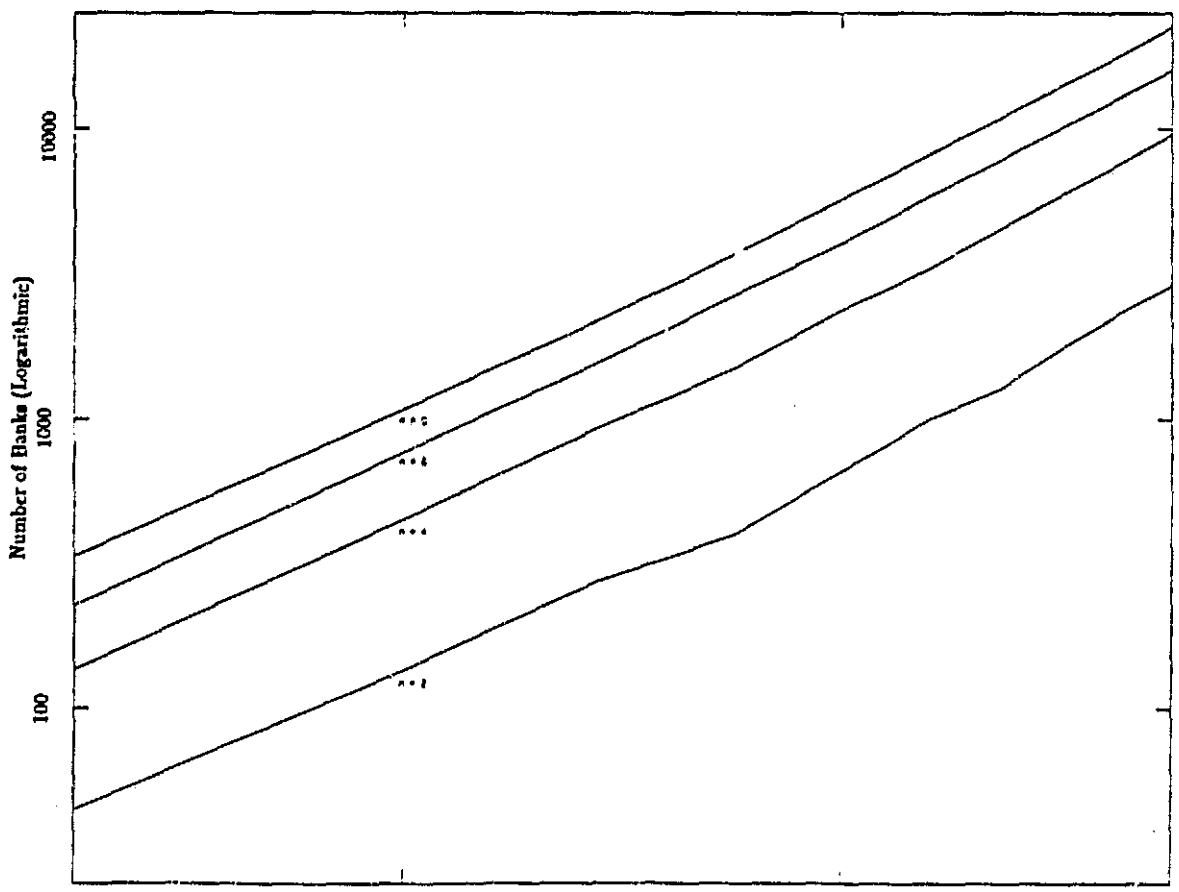

Figure 4

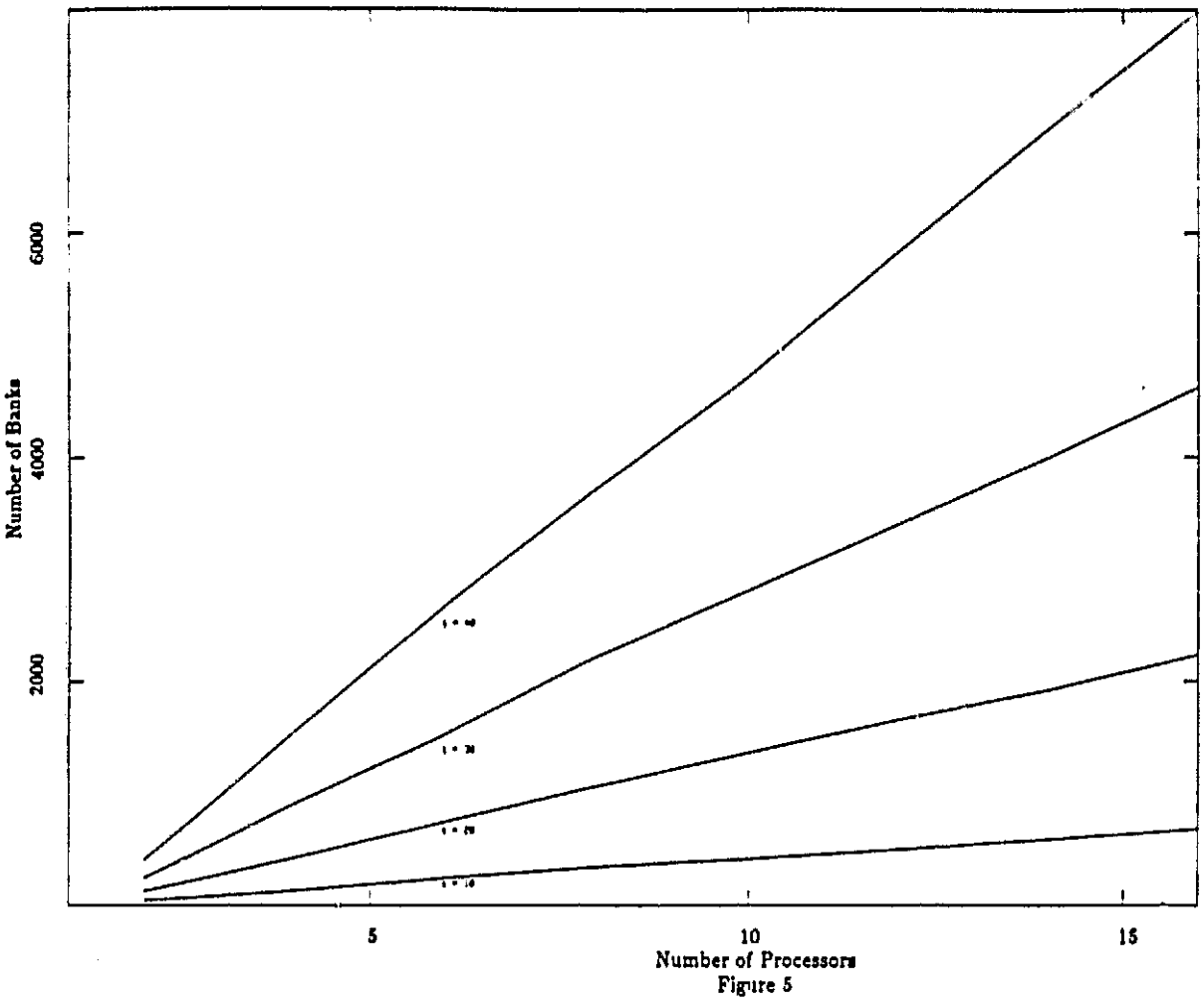

Figure 5