Scalar Memory References in Pipelined Multiprocessors: A Performance Study

Ravi Ganesan and Shlomo Weiss, *Member, IEEE*

Abstract—**Interleaved memories are essential in pipelined computers to attain high memory bandwidth. As a memory bank is accessed, a reservation is placed on the bank for the duration of the memory cycle, which is often considerably longer than the processor cycle time. This additional parameter, namely, the bank reservation time or the bank busy time, adds to the complexity of the memory model. For Markov models, exact solutions are not feasible even without this additional parameter due to the very large state space of the Markov chain. In this paper we develop a Markov model which explicitly tracks the bank reservation time. Because we only model one processor and the requested bank, the transition probabilities are not known and have to be approximated. The performance predicted by the model is in close agreement with simulation results.**

Index Terms— **Pipelined computers, supercomputers, interleaved memory, Markov chains, memory conflicts.**

I. Introduction

INTERLEAVED memories are commonly used in pipelined processors to increase the memory bandwidth beyond the bandwidth of a single memory module (or bank). The performance of interleaved memory systems depends on the number of banks, the bank cycle time, the number of processors, and the pattern of requests generated. The design of such systems involves a number of trade-offs which are dominated by the nature of the interrelationships between these factors. Performance analysis of interleaved memory systems yields insights into these interrelationships, thus enabling the designer to study cost/performance trade-offs and determine system parameters.

A large amount of useful work has been done in this area, and we briefly survey the various approaches and their implications. Of special interest is Bailey's work [1] on memory contention in vector computers. While in [1] the focus is on vector memory references, Bailey also introduced a Markov model, which is the starting point of our research. Our primary thrust is to develop a more accurate Markov model by relaxing the restrictive condition that at most one processor can be blocked on a busy bank.

Stochastic models are commonly used to study the performance of parallel memories. A basic assumption in these models is that memory references are randomly distributed. This is clearly not the case in vector processing, where each reference is at a constant distance from the previous reference. Hence the model developed in this paper is not intended for vector processors. One reasonable area of application for the model is a transaction processing environment, which is characterized by a large number of independent transactions executing concurrently. The model developed in this paper can be also applied to pseudorandomly interleaved memories [2]–[8]. In a pseudorandomly interleaved memory the address sequence presented to the memory system is a pseudorandom sequence which is generated by "scrambling" the address pattern produced by the processor. Randomizing memory schemes have been implemented in the IBM RP3 [3] and Cydrome Cydra 5 [7] memory systems.

The remainder of this paper is organized as follows. In Section II we look at prior work on memory performance modeling. We point out that assumptions commonly made in many multiprocessor memory analyses are not applicable in the context of pipelined computers. We summarize Bailey's model in Section III. Comparing it with simulation results (Section IV), we conclude that Bailey's model often overestimates performance, primarily because of the restriction that at most one processor can be blocked on a busy bank. We develop a Markov model in Sections V and VI and present results predicted by it in Section VIII. Finally, in Section IX we discuss the accuracy of our model.

II. Survey of Memory Performance Modeling Techniques

The contention problem in memory systems is a well-known problem that has received considerable attention in the literature. Cheung and Smith [9] analyze the memory system of the CRAY X-MP by considering the various ways in which memory references interact with other references from the same vector stream or from other vector streams. This deterministic approach, taken also by Oed and Lange [10], is most appropriate for the memory system of a vector multiprocessor, since vector references are clearly not random. Cheung and Smith point out, however, that when several vector streams are active simultaneously, and each may have a different stride, the number of different stride combinations is too large to analyze each case separately.

The more common approach to analyze contention in parallel memories is the use of probabilistic models. In an early work, Hellerman [11] proposed a model in which he assumes that in each cycle a sequence of b requests is presented to a parallel memory system with b banks. In this model, Hellerman

Manuscript received June 21, 1991; revised August 29, 1991. Recommended by E. Gelenbe.

R. Ganesan is with Bell Atlantic, Beltsville, MD 20705.

S. Weiss is with the Department of Computer Science, University of Maryland, Baltimore, and the Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

IEEE Log Number 9104758.

assumes that memory requests are independent and randomly distributed. If multiple requests arrive at the same bank, one is serviced and the others are rejected. Rejected requests are dropped and b new independent requests are submitted in the next cycle to randomly selected banks. The stream of b requests is inspected in the arrival order in one cycle. Requests arriving prior to a conflict are forwarded to memory within the same cycle. The rest of the b requests are rejected.

Based on this model, Hellerman derived a formula for the average number of busy banks and showed that $b^{0.56}$ is an approximation of this formula to within 4%, for $b \leq 45$. Knuth and Rao [12] showed that Hellerman's expression is a well-studied function with known asymptotic behavior. Ravi [13] changed Hellerman's model to allow all requests that arrive to distinct banks, within the same cycle, to be forwarded to memory. That is, in a stream of m requests arriving in a given cycle (where $m \leq b$, the number of banks), a conflict does not block other requests that are directed to different banks. Ravi showed that removing this constraint in Hellerman's model significantly enhances the predicted memory bandwidth.

Ravi's memory model with m processors and b banks is equivalent to a combinatorial problem with m balls and b partitions, in which m balls are randomly distributed among the b partitions in each cycle. Also, in each cycle, one ball is removed from each partition, which corresponds to forwarding a request to a memory bank. In Ravi's model, unserviced requests are simply dropped at the end of the cycle. Chang, *et al.* [14] point out that this assumption makes Ravi's model somewhat optimistic, because unserviced requests can cause additional conflicts in later cycles.

A memory model similar to Ravi's was analyzed by Bhandarkar [15] using Markov chain techniques. The state of this Markov chain is defined by a b-tuple $(k_1, k_2, \cdots, k_b)$, where b is the number of banks, and k_i, $1 \leq i \leq b$ is the number of memory requests issued to bank i in a given cycle. With m processors, $0 \leq k_i \leq m$ and $\sum_{i=1}^{b} k_i = m$. Even for modest size systems (i.e., $b = 16$ and $m = 16$), this Markov chain leads to a very large number of states. By making the assumption that the processors are identical, Bhandarkar was able to obtain and solve a system with fewer states.

Following are the assumptions in Ravi's and Bhandarkar's model:

A) The processors are statistically identical.
B) The memory requests are independent and randomly distributed in the range $1 \ldots b$.
C) Each of the m processors generates one memory reference per cycle with probability 1; that is, m requests are submitted to the memory system in every cycle.
D) There are not conflicts in the processor–memory interconnect (i.e., a crossbar switch).
E) The memory cycle time is constant.
F) A processor blocks while its request is being serviced. This assumption implies that either the processor is bound by the speed of the memory, or the processor cycle time is equal to the memory cycle time.
G) Rejected requests are dropped and a new set of independent requests is submitted in the next cycle.

While assumption F may be realistic in SIMD parallel processors (e.g., the Burroughs Scientific Processor—BSP) or in some MIMD machines, it is clearly incorrect in the context of pipelined supercomputers (e.g., CRAY computers). A common characteristic of pipelined processors is the short clock period, which leads to efficient pipeline operation. The memory system cannot match the speed of the processor, and one memory cycle corresponds to multiple processor cycles. Furthermore, a processor does not block while its request is being serviced by a memory bank; rather, one processor can have multiple requests simultaneously in service.

Assumption G is also problematic. If a processor is waiting for a bank whose remaining reservation is 10 cycles, for example, then in the next 10 cycles that processor will gain access to memory with probability 0. The processor has a much higher chance to succeed in accessing memory if it is permitted to select a bank, at random, in each of the following 10 cycles. Clearly, this assumption leads to a model that overestimates memory bandwidth, particularly in pipelined vector multiprocessors (e.g., the CRAY-2), where the memory is characterized by long bank busy times.

Both assumptions F and G are common in most of the multiprocessor memory analyses reported in the literature [16]–[26]. Notable exceptions are [27], [28], and [1]. Mudge and Al-Sadoun [27] make assumption G, but not F. Their Markov chain model is primarily targeted to applications where the connection times between processors and memories is variable (e.g., variable size data blocks transferred over a bus).

Queueing models represent an alternative approach to the models discussed above. Bucher and Calahan [29] derived a scaling relation, which indicates that memory access delays depend quadratically on the bank reservation time for light memory load. For heavy load, delays scale linearly with the ratio of processors to the number of memory banks. Bucher and Calahan also developed an open and closed queueing model of memory conflicts, and validated the model by simulation. A heuristic extension of the open model provides results chose to simulation for scalar and long vector references.

III. Bailey's Model

In a recent paper [1], Bailey introduced a simple Markov chain model for memory bank contention in vector computers. The important characteristic of this model, and the reason for choosing it as the starting point of our analysis, is that it does not make assumption F. That is, the memory cycle time is an integer multiple of the processor cycle time, and, if the requested bank is free, the processor does *not* block while its request is being serviced. We briefly summarize Bailey's model below.

The system consists of m processors and a shared interleaved memory with b banks. The bank cycle time is c. In any cycle, a processor may make a request with probability r (request rate). To keep track of the state of a bank, a reservation of c cycles is placed on a bank when it is accessed. In each

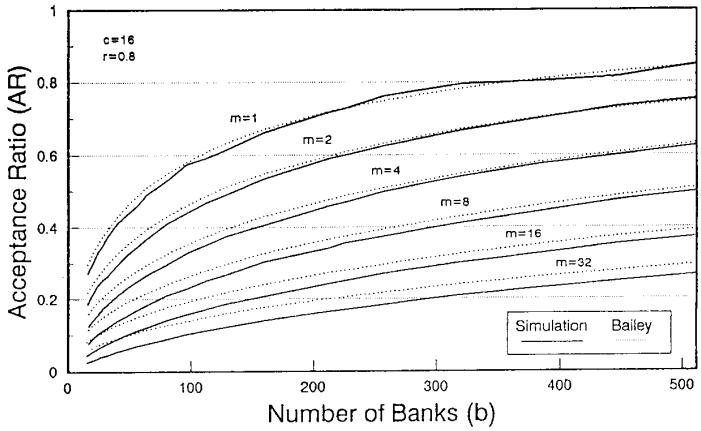


Fig. 1. Bailey's model normally overestimates the memory acceptance ratio. The acceptance ration (AR) is the ratio of accepted requests to total requests.

subsequent cycle the remaining reservation on the bank is decremented by one until it reaches 0, at which time the bank becomes free.

Assuming that processors are statistically identical, it is sufficient to model a single processor. A processor is in state 0 (the free state) if it is not blocked. Otherwise, the processor is in state i, where $1 \leq i \leq c$, in which case the processor is blocked on a bank that has i cycles remaining to process a previous request. In each cycle, a blocked processor performs a transition with probability 1 from state i to state $i-1$ until the requested bank becomes free.

Let x be the probability that a bank is busy. Then rx is the probability that a free processor makes a request to a busy bank. Assuming that the reservation on a busy bank is uniformly distributed in the range $1 \ldots c$, the transition probability from state 0 to state i is given by

$$p_{0,i} = \frac{rx}{c}, \qquad 1 \leq i \leq c.$$

Based on this approximation, Bailey solved the Markov chain in terms of x. He obtained a second equation from the following consideration. The mp_0 free processors request rmp_0 banks. In the steady state the number of requested banks equals the number of banks that are freed in that cycle, which is bx/c. Hence

$$rmp_0 = \frac{bx}{c}. \tag{1}$$

Substituting the value of x from this equation, Bailey solved the Markov chain in terms of the system parameters m, r, c, and b.

IV. Analysis of Bailey's Model

In this section we compare figures obtained using Bailey's model with simulation results, and comment on the accuracy of the model. The following two basic assumptions in Bailey's model have a significant impact on its accuracy:

Assumption 1

At most one processor may be blocked on a busy bank.

Assumption 2

The remaining reservation on a busy bank is uniformly distributed in the range $1 \ldots c$.

Assumption 1 considerably simplifies the Markov chain and limits the number of states to c. As this assumption ignores memory conflicts caused by multiple processors contending for the same bank, we expect Bailey's model to overestimate the memory performance. This effect is indeed confirmed by simulation (see Fig. 1), and becomes especially prominent as the number of processors is increased.

Assumption 2 allows the approximation of transition probabilities. There are, however, situations in which this assumption does not hold, as demonstrated by the following pathological case with $m = 1$, $b = 1$, $c = 4$, and $r = 1.0$ (i.e., one processor, one bank, and the processor makes a request on every cycle it is not blocked). On the first cycle the processor will make a request that is accepted, and on the next the processor's request will hit the bank when it has 3 cycles left. This leaves the processor blocked. The processor will get unblocked after three cycles when its initial request is complete and its second request is accepted. Since the request rate r is 1.0, the processor will immediately make a request which will hit the bank when it has 3 cycles left. From then on every request the processor makes will go to the bank when it has 3 cycles left.

Hence the transition probabilities are $p_{0,1} = p_{0,2} = 0$, and $p_{0,3} = 1$. Interestingly enough, $p_{0,4}$ is also 0, since the only way the processor may request a bank with four reservations remaining is that another processor has requested, and obtained access to, the same bank in the same cycle. This is not possible in the above example, however, since $m = 1$.

Modeling a single processor drastically reduces the size of the Markov state space, but results in loss of information, which manifests itself in the inability to determine exact transition probabilities. Hence the need to approximate

transition probabilities, which leads to inaccuracies in the model, as indicated above. Markov chains that explicitly model each memory bank permit the calculation of exact transition probabilities. However, as noted earlier, such chains suffer from an explosion in the size of their state space.

V. Markov Chain Model

We have seen in the previous section that Bailey's model tends to overestimate the memory performance, because of the assumption that at most one processor can be blocked on a busy bank. This may be also regarded as a model in which a queue is attached to each bank to hold incoming requests, but the size of the queue is restricted to one request. Even in this relatively simple model the transition probabilities are not known and have to be approximated.

Attaching unbounded[1] queues to each bank greatly complicates attempts to obtain reasonable approximations for the transition probabilities. Further, the state space of the chain also grows. Hence our approach was to incrementally increase the size of the queue (i.e., allow for a queue size of 2, 3, ...), approximate the transition probabilities, and solve the corresponding Markov chain until the performance results predicted by the Markov model were within a few percentage points of the simulation figures. A somewhat surprising result was that at the queue size of 2, our model was predicting accurate figures for a large range of system parameter values. Hence we restrict our attention to the case where the maximum length of the queue attached to a bank is 2.

The model must be able to keep track of two quantities: (a) the number of reservation cycles remaining on the requested bank if it is busy, and (b) the number of pending requests in the queue of a busy bank. We define a Markov chain as follows. Let the random variable X_t be 0 if the processor is free, of i if the processor is blocked on a busy bank that is reserved for i additional cycles. Let the random variable Y_t be 0 if the requested bank is free, or j if the bank is busy and there are j processors blocked on this bank. The processor currently serviced by the bank is not blocked and therefore is not included in the count of the j blocked processors. Then $\{X_t, Y_t, t \geq 1\}$ is a Markov chain with the transition probabilities:

$$P_{ij,kl} = \text{Prob}[X_{t+1} = k, Y_{t+1} = l | X_t = i, Y_t = j\,].$$

As before, we assume that there are m processors, b banks, and when a bank accepts a request it becomes reserved (busy) for c cycles. In each cycle a processor issues a request with probability r, and x is the probability that a bank is busy. Hence the probability that a free processor makes a request to a busy bank is rx. We make the following two assumptions regarding the status of the busy bank requested by the processor.

1. There are no processors blocked on the busy bank (again, the processor whose request is being processed by the busy bank is *not* blocked).
2. The remaining reservation on the busy bank is uniformly distributed in the range, $1 \ldots c$.

These assumptions are needed to approximate the transition probabilities from the free state (00) to a blocked state $(i1)$. Using the notation,

$$\alpha = \frac{rx}{c} \tag{2}$$

we get:

$$P_{00,i1} = \alpha, \qquad 1 \leq i \leq c.$$

When the system is in a blocked state $(i1)$, $1 \leq i \leq c$, a new request from a different processor may arrive at the same bank. To determine the probability of such a request arriving, we need to know the number of free processors. This number depends on P_{00}, the probability that a processor is free. Therefore the transition probabilities from states $(i1)$, $1 \leq i \leq c$, depend on the probability that the system is in state (00). But in a Markov chain the history of the process does not go beyond the last state. Hence the need to make a third approximating assumption. We assume that approximately half of the processors $(m/2)$ are free to generate requests, and therefore the request rate in the entire system is $mr/2$. Since these requests are uniformly distributed across b banks, the probability of a new request arriving at the bank is

$$\beta = \frac{mr}{2b} \tag{3}$$

and the transition probabilities are approximately given by

$$P_{i1,(i-1)2} = \beta, \qquad 2 \leq i \leq c.$$

The Markov chain is shown in Fig. 2. The remaining transition probabilities are derived as follows. When the system is in state (12), the bank is reserved for one more cycle, and is about to become free in the next cycle. With two requests pending in the queue, the request accepted by the bank is chosen at random. If the accepted request belongs to the processor being modeled, the transition is to state (00), indicating that the processor became free. Otherwise, the transition is to state $(c1)$, indicating that the processor is now waiting for a bank that has just accepted a request and is reserved for c more cycles. Hence

$$P_{12,00} = \frac{1}{2}$$
$$P_{12,c1} = \frac{1}{2}.$$

If the system is in state (11) just one cycle before the bank becomes free and a new request arrives (with probability β), there are again two requests, and one is accepted at random. If the new incoming request is accepted (with probability 1/2), the transition is to state $(c1)$, since a reservation of c cycles has just been placed on the bank. Therefore,

$$P_{11,c1} = \frac{\beta}{2}.$$

[1] If we assume that a processor blocks until its request is accepted for service, than the maximum number of requests in a bank's queue will be bounded by m, the number of processors.

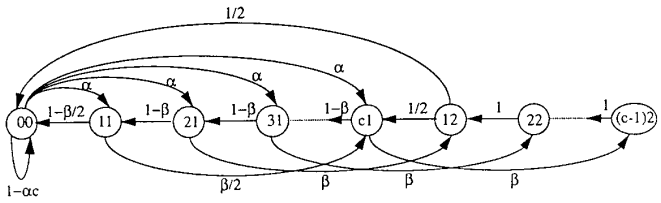


Fig. 2. Markov model. An accepted request places a reservation of c cycles on the bank, and the reservation is decremented by one in each subsequent cycle. Up to two requests may be blocked on the bank.

Otherwise, the processor becomes free and the transition probability is

$$P_{11,00} = 1 - \frac{\beta}{2}.$$

VI. Derivation of State Probabilities

The balance equations for the Markov chain are given by:

$$\begin{aligned}
P_{00} &= (1-\alpha c)P_{00} + \left(1 - \frac{\beta}{2}\right)P_{11} + \frac{1}{2}P_{12} \\
P_{i1} &= \alpha P_{00} + (1-\beta)P_{(i+1)1}, \qquad 1 \le i \le c-1 \\
P_{c1} &= \alpha P_{00} + \frac{\beta}{2}P_{11} + \frac{1}{2}P_{12} \\
P_{i2} &= \beta P_{(i+1)1} + P_{(i+1)2}, \qquad 1 \le i \le c-2 \\
P_{(c-1)2} &= \beta P_{c1}.
\end{aligned}$$

The state probabilities can be obtained in terms of P_{00}:

$$P_{i1} = \alpha P_{00}\left[\frac{k}{(1-\beta)^{i-1}} - \sum_{j=1}^{i-1}\frac{1}{(1-\beta)^j}\right], \qquad 1 \le i \le c$$

where

$$k = \frac{\left(\sum_{j=1}^{c-1}\frac{1}{(1-\beta)^j}\right) + c + 1}{\frac{1}{(1-\beta)^{c-1}} - \beta + 1}$$

and

$$P_{i2} = \alpha P_{00}\left[k' - \beta k\sum_{j=1}^{i-1}\frac{1}{(1-\beta)^j} + \beta\sum_{j=1}^{i-1}\frac{(i-j)}{(1-\beta)^j}\right], \qquad 1 \le i \le c-1$$

where

$$k' = 2\left[c - k\left(1 - \frac{\beta}{2}\right)\right].$$

Define:

$$S = \sum_{i=1}^{c}\left[\frac{k}{(1-\beta)^{i-1}} - \sum_{j=1}^{i-1}\frac{1}{(1-\beta)^j}\right] + \sum_{i=1}^{i=c-1}\left[k' - \beta k\sum_{j=1}^{i-1}\frac{1}{(1-\beta)^j} + \beta\sum_{j=1}^{i-1}\frac{(i-j)}{(1-\beta)^j}\right]. \tag{4}$$

Since the state probabilities sum to one, we obtain:

$$P_{00} + \sum_{i=1}^{i=c}P_{i1} + \sum_{i=1}^{i=c-1}P_{i2} = 1.$$

But since,

$$\sum_{i=1}^{i=c}P_{i1} + \sum_{i=1}^{i=c-1}P_{i2} = \alpha P_{00}S$$

we get:

$$P_{00} = \frac{1}{1+\alpha S}. \tag{5}$$

Equation (1) can be rewritten in the context of the Markov chain as:

$$rmP_{00} = \frac{bx}{c}$$

which combined with (2) yields:

$$\alpha = \frac{mr^2P_{00}}{b}.$$

Substituting this expression of α into (5), we obtain:

$$P_{00} = \frac{\sqrt{1 + 4mr^2S/b} - 1}{2mr^2S/b} \tag{6}$$

where S is given in (4).

VII. Performance Measures

Several measures have been used in the literature to quantify the performance of the memory system. The most common one is the effective bandwidth (BW), defined to be the expected number of requests accepted by the memory in one cycle (see [26], for example). Bailey [1] introduced a different measure, called *memory efficiency,* which is similar to Cheung and Smith's *acceptance ratio* (AR) [9]. Let A be the number of accepted requests in a period of k cycles, and R be the number of rejected requests in the same time interval. A request rejected by the memory on the first attempt will be resubmitted in each subsequent cycle until accepted. The AR is the ratio of accepted requests to total number of requests:

$$\mathrm{AR} = \frac{A}{A+R}. \tag{7}$$

P_{00} is the probability that a processor is free, and r the probability that a free processor makes a memory request. In k

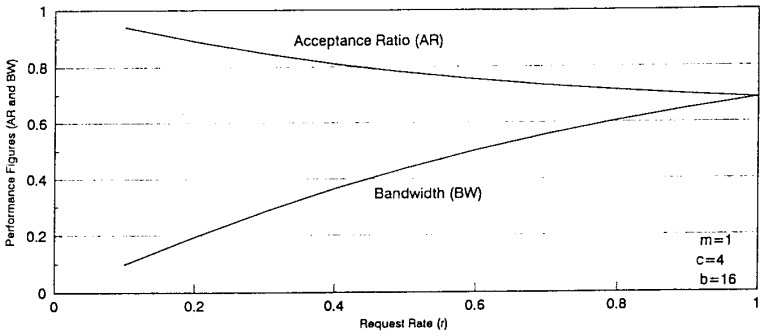


Fig. 3. Acceptance ratio and bandwidth for $m = 1$, $b = 16$, $c = 4$, and variable request rate r.

cycles, a processor issues $rP_{00}k$ requests. Of these, j requests are rejected. Assume that a rejected request is resubmitted an average of π times until accepted. Hence πj attempts lead to $(\pi - 1)j$ rejected requests and j accepted requests. We have:

$$\begin{aligned} A &= \text{requests accepted on the first attempt} \\ &\quad + \text{the last (and successful) attempt of requests} \\ &\quad \text{submitted multiple times} \\ &= [rP_{00}k - j] + j \\ &= rP_{00}k \\ R &= \text{the first atempt of rejected requests} \\ &\quad + \text{subsequent attempts, with the exception of the} \\ &\quad \text{last one} \\ &= j + (\pi - 1)j \\ &= \pi j. \end{aligned}$$

We get:

$$AR = \frac{rP_{00}k}{rP_{00}k + \pi j}.$$

In a time interval of k cycles, the processor is blocked for $(1 - P_{00})k$ cycles. πj is the number of resubmissions, but since in a blocked state the processor resubmits a pending request on every cycle, this is also the total cycles in which the processor is blocked. Hence $\pi j = (1 - P_{00})k$, and we obtain:

$$\text{AR} = \frac{rP_{00}}{rP_{00} + (1 - P_{00})}. \tag{8}$$

The bandwidth is the total accepted requests for the m processors:

$$\text{BW} = mrP_{00}. \tag{9}$$

To compare these two measures, namely, BW and AR, consider a system with $m = 1$, $b = 16$, and $c = 4$. We chose these parameters as an example to correspond to the CRAY-1 memory system. Fig. 3 shows the AR and BW predicted by our model as a function of r, the probability that a processor makes a request on a given cycle (request rate). A higher request rate corresponds to a higher load on the memory system, and, as we would expect, a higher load leads to more conflicts and lower performance. This is clearly indicated by the acceptance ratio, which changes from 0.94 to 0.69 as r is varied in the range of 0.10 to 1.00. In the same range, however, the bandwidth increases from 0.10 to 0.69 accepted requests per cycle, because the extra bank conflicts are masked by the higher request rate. In the remainder of this paper we use the acceptance ratio to measure the memory performance.

Another performance measure that arises naturally in queueing models is the *delay* D. This is the measure used by Bucher and Calahan [29]. The delay D can be related to P_{00} by the following observation. In k cycles, a processor is free for $P_{00}k$ cycles and issues $rP_{00}k$ requests. Each request is delayed by D cycles. Hence the total delay is $rP_{00}kD$, during which the processor is busy waiting for requests to be serviced. Hence:

$$\begin{aligned} P_{00} &= (\text{free cycles})/(\text{free cycles} + \text{busy cycles}) \\ &= P_{00}k/(P_{00}k + rP_{00}kD) \\ &= 1/(1 + rD). \end{aligned}$$

VIII. Perormance Results

The memory performance predicted by the Markov model, as measured by the acceptance ratio, can be calculated by substituting the value P_{00} from (6) into (8). Figs. 4–6 show performance results for the Markov model. In Fig. 4 we illustrate the relationship between the AR and the number of banks for a different number of processors. We fixed the request rate and bank cycle time to $r = 0.8$ and $c = 16$, respectively. As the graph shows for a given processor size, there is an increase in the AR until a point where further increase does not yield increases in the same proportion. For a given m, c, and r, choosing a value of b at the knee of the graph would provide a good cost–benefit ratio. This is, however, not true when the number of processors is high, since then the relationship appears to be fairly linear in the range shown, without an observable knee.

Fig. 5 illustrates the variations in AR caused by changes in the number of processors m. Again, we present the data for the case where $r = 0.8$ and $c = 16$. This graph contains information similar to Fig. 4, although from a different perspective. It is instructive to note that even for a relatively small number

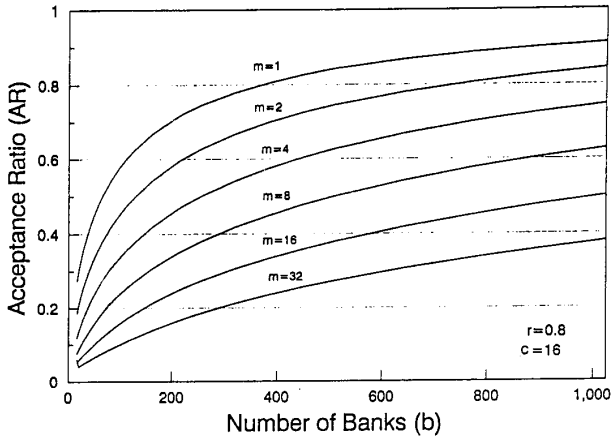


Fig. 4. Memory performance (acceptance ratio) versus number of banks.

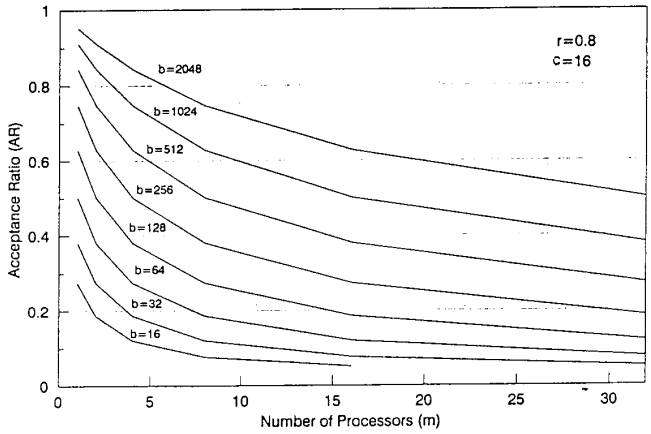


Fig. 5. Memory performance (acceptance ratio) versus number of processors.

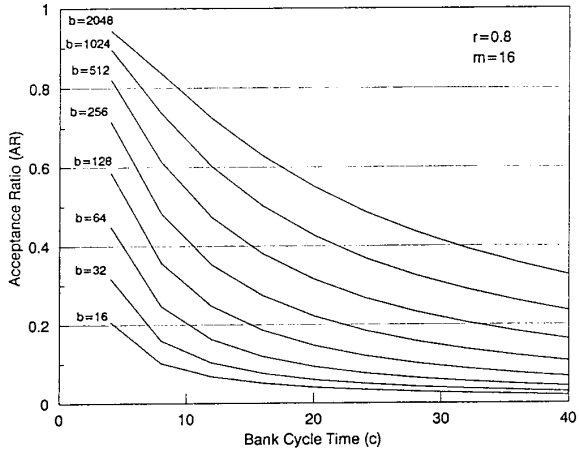


Fig. 6. Memory performance (acceptance ratio) versus bank cycle time.

of processors (e.g., 4), the number of banks required to obtain a reasonable acceptance ratio (e.g., 0.8) is very high.

In Fig. 6 we consider the relationship between the AR and changes in the bank cycle time c. Here, $r = 0.8$ and $m = 16$. The sharp drop shown in the acceptance ratio for increases in the bank cycle time are at the crux of a growing practical problem. Since the clock speeds of processors have been falling at sharper rates than the speeds of memories (especially high capacity–low cost dynamic memories), there is a steady increase in the bank cycle times. The increased efficiency of processors will not transfer into a corresponding improvement in system performance if memory becomes the bottleneck.

IX. Discussion and Conclusions

Most of the memory performance models studied in the literature assume that: (a) the memory cycle time is identical to the processor cycle time, or (b) the memory cycle time may be longer than the processor cycle time, and then a processor whose request has been accepted blocks for the duration of the memory cycle. Neither of these two assumptions hold in pipelined computers. In a pipelined machine, the operation of the memory itself is pipelined and a processor is free to perform computations or submit other memory requests while

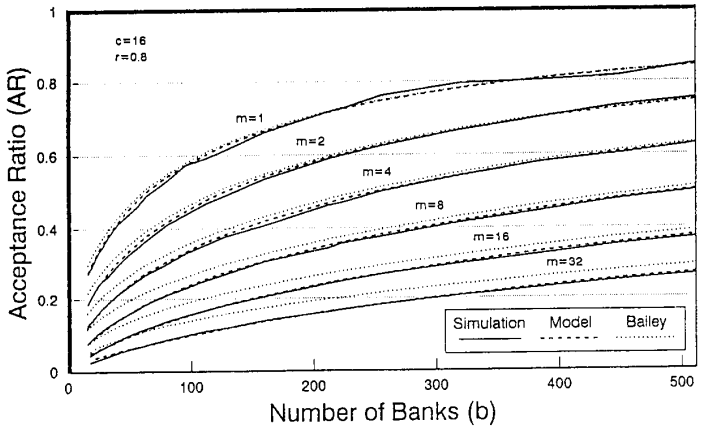


Fig. 7. Comparison with Bailey's model.

one or more of its previous requests are still in the "memory pipeline." Hence a processor does not block for the duration of the memory cycle if its request has been accepted. A reservation, though, is placed on the requested bank for the length of the memory cycle. A processor can have multiple requests simultaneously pending in the memory system as long as different banks are involved.

To capture this behavior a performance model has to keep track of the status of each memory bank (free or busy: if busy, the length of the remaining reservation). Even without this additional information, Markov memory models with a practical number of banks have a very large state space and direct solutions are not feasible. Bailey's approach to this problem is to assume that all processors are statistically identical and to model a single processor and the requested memory bank if the bank is reserved by a previous request. The model explicitly keeps track of the remaining reservation on the busy bank.

To approximate the transition probabilities, which are unfortunately unknown in this model, Bailey made a number of simplifying assumptions. These include the restriction that at most one processor may be blocked on a busy bank. By ignoring contention between processors for the same bank, Bailey's model overestimates memory performance unless the memory system is lightly loaded.

The primary objective of this research was to improve Bailey's model by allowing multiple requests to be blocked on the same memory bank. The resulting Markov chain still has a reasonably small state space, but again, the transition probabilities are not known. Faced with the task of approximating transition probabilities, we proceeded in a step-wise fashion, by first allowing at most two processors to be blocked on a busy bank (this corresponds to a queue length of two). Even with this restriction, the model is close to the simulation, as shown in Fig. 7. This conclusion, namely, that a queue length of two is sufficient in most circumstances, is in close agreement with Bucher and Calahan's work [29]. They point out that, for the range of b, m, and c common in today's computers, the average queue lengths are always quite short.

In summary, the Markov model developed in this paper approximates the behavior of interleaved memories by modeling one processor and the requested memory bank. The model explicitly maintains the remaining length of the reservation placed on a busy bank. Inaccuracies are introduced in the model by the restriction that at most two processors may be blocked on the same bank, and by the approximation of transition probabilities. For large m, the absolute difference in the acceptance ratio between the model and simulation is within 3%. This deviation increases for $m \leq 4$; the largest difference that we have observed being 10% for $m = 1$. Since the model is intended for multiprocessors, we feel that this is a reasonable limitation.

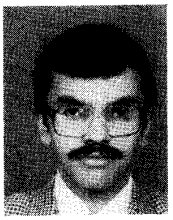
ACKNOWLEDGMENT

The authors would like to thank D. Bailey and I. Bucher for their comments and suggestions.

REFERENCES

[1] D. H. Bailey, "Vector computer memory bank contention," *IEEE Trans. Computers*, vol. C-36, pp. 293–298, Mar. 1987.
[2] J. M. Frailong, W. Jalby, and J. Lenfant, "XOR-schemes: a flexible data organization in parallel memories," in *Proc. 1985 Int. Conf. Parallel Process.*, Aug. 1985, pp. 276–283.
[3] A. Norton and E. Melton, "A class of boolean linear transformations for conflict-free power-of-two access," in *Proc. 1987 Int. Conf. Parallel Process.*, Aug. 1987, pp. 247–254.
[4] M. Balakrishnan, R. Jain, and C. S. Raghavendra, "On array storage for conflict-free memory access for parallel processors," in *Proc. 1988 Int. Conf. Parallel Process.*, Aug. 1988, pp. 103–107.
[5] D. Lee, "Scrambled storage for parallel memory systems," in *Proc. 15th Ann. Int. Symp. Computer Arch.*, May 1988.
[6] G. S. Sohi, "High-bandwidth interleaved memories for vector processors—a simulation study," Computer Sci. Dept., Univ. Wisconsin-Madison, Tech. Rep. 790, Sept. 1988.
[7] B. R. Rau, M. S. Schlansker, and D. W. L. Yen, "The Cydra 5 stride-insensitive memory system," in *Proc. 1989 Int. Conf. Parallel Process.*, Aug. 1989, pp. 242–246.
[8] R. Raghavan and J. P. Hayes, "On randomly interleaved memories," in *Proc. Supercomputing '90 Conf.* (New York), Nov. 1990.
[9] T. Cheung and J. E. Smith, "A simulation study of the CRAY X-MP memory system," *IEEE Trans. Computers*, vol. C-35, pp. 613–622, July 1986.
[10] W. Oed and O. Lange, "On the effective bandwidth of interleaved memories in vector processor systems," *IEEE Trans. Computers*, vol. C-34 pp. 949–957, Oct. 1985.

[11] H. Hellerman, *Digital Computer Systems,* 2nd ed. New York: McGraw-Hill, 1973.
[12] D. E. Knuth and G. S. Rao, "Activity in interleaved memory," *IEEE Trans. Computers,* vol. C-24, pp. 943–944, Sept. 1975.
[13] C. V. Ravi, "On the bandwidth and interference in interleaved memory systems," *IEEE Trans. Computers,* vol. C-21, pp. 899–901, Aug. 1972.
[14] D. Y. Chang, D. J. Kuck, and D. H. Lawrie, "On the effective bandwidth of parallel memories," *IEEE Trans. Computers,* vol. C-26, pp. 480–489, May 1977.
[15] D. P. Bhandarkar, "Analysis of memory interference in multiprocessors," *IEEE Trans. Computers,* vol. C-24, pp. 897–908, Sept. 1975.
[16] F. Baskett and A. J. Smith, "Interference in multiprocessor computer systems with interleaved memory," *Commun. ACM,* vol. 19, pp. 327–334, June 1976.
[17] G. Chiola, M. A. Marsan, and G. Balbo, "Product-form solutions techniques for the performance analysis of multiple-bus multiprocessor systems with nonuniform memory references," *IEEE Trans. Computers,* vol. C-37, pp. 532–540, May 1988.
[18] M. A. Holliday and M. K. Vernon, "Exact performance estimates for multiprocessor memory and bus interferences," *IEEE Trans. Computers,* vol. C-36, pp. 76–85, Jan. 1987.
[19] C. H. Hoogendoorn, "A general model for memory interferences in multiprocessors," *IEEE Trans. Computers,* vol. C-26, pp. 998–1005, Oct. 1977.
[20] K. B. Irani and I. H. Onyuksel, "A closed-form solution for the performance analysis of multiple-bus multiprocessor systems," *IEEE Trans. Computers,* vol. C-33, pp. 1004–1012, Nov. 1984.
[21] J. H. Patel, "Processor–memory interconnections for multiprocessors," *IEEE Trans. Computers,* vol. C-30, pp. 771–780, Oct. 1981.
[22] B. R. Rau, "Interleaved memory bandwidth in a model of multiprocessors," *IEEE Trans. Computers,* vol. C-28, pp. 678–681, Sept. 1979.
[23] A. S. Sethi and N. Deo, "Interference in multiprocessor memory systems with localized memory access probabilities," *IEEE Trans. Computers,* vol. C-28, pp. 157–163, Feb. 1979.
[24] C. E. Skinner and J. R. Asher, "Effects of storage contention on system performance," *IBM Syst. J.,* vol. 8, pp. 319–333, 1969.
[25] D. Towsley, "Approximate models of multiple bus multiprocessor systems," *IEEE Trans. Computers,* vol. C-35, pp. 220–228, Mar. 1986.
[26] D. W. L. Yen, J. H. Patel, and E. S. Davidson, "Memory interference in synchronous multiprocessor systems," *IEEE Trans. Computers,* vol. C-31, pp. 1116–1121, Nov. 1982.
[27] T. N. Mudge and H. B. Al-Sadoun, "Memory interference models with variable connection time," *IEEE Trans. Computers,* vol. C-33, pp. 1033–1038, Nov. 1984.
[28] F. A. Briggs and E. S. Davidson, "Organization of semiconductor memories for parallel-pipelined processors," *IEEE Trans. Computers,* vol. C-26, pp. 162–169, Feb. 1977.
[29] I. Y. Bucher and D. A. Calahan, "Access conflicts in multiprocessor memories queueing models and simulation studies," in *Proc. 1990 Int. Conf. Supercomput.* (Amsterdam, The Netherlands), June 1990, pp. 428–438.



Ravi Ganesan was born in Bombay, India, in 1966. He received the B.E. computer science and engineering degree in 1989 from Anna University, Madras. He then received the University of Maryland's Graduate Merit Fellowship and completed the M.S. degree in computer science in 1990 from the University of Maryland, Baltimore County.

Since August 1990 he has been with the Technology Research and Transfer group in the Information Systems and Operations Department of Bell Atlantic, Beltsville, MD, where, in addition to pursuing his research interests, he is responsible for defining the security architectures and protocols for the corporate computing environment. His current research interests are in the heuristic search in artificial intelligence, cryptology, and high-performance computing.

Mr. Ganesan is a member of the IEEE Computer Society, the Association of Computing Machinery, and the Phi Kappa Phi honor society.

Shlomo Weiss (S'84–M'84) received the B.S. degree in electrical engineering in 1973 from the Technion, and the Ph.D. degree in computer science in 1984 from the University of Wisconsin–Madison.

He has been an Assistant Professor of Computer Science at the University of Maryland at Baltimore since 1987. From 1985 to 1987 he was a member of the technical staff of the Microelectronics and Computer Technology Corporation (MCC). His research interests focus on high-performance computer architecture and parallel processing.

Dr. Weiss is a member of the IEEE Computer Society.