TITLE MEASUREMENT OF MEMORY ACCESS CONTENTIONS IN
MULTIPLE VECTOR PROCESSOR SYSTEMS

AUTHORS: Ingrid Y. Bucher
Margaret L. Simmons

SUBMITTED TO Supercomputing '91
November 18-22, 1991
Albuquerque, New Mexico

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Los Alamos National Laboratory, Los Alamos, New Mexico 87545. This report is available to the public through the National Technical Information Service, Springfield, Virginia 22161.

For sale by the National Technical Information Service, Springfield, Virginia 22161. Price: \$12.00. Order form is available from the National Technical Information Service.

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MEASUREMENT OF MEMORY ACCESS CONTENTIONS IN MULTIPLE VECTOR PROCESSOR SYSTEMS

by

Ingrid Y. Bucher and Margaret L. Simmons

Computer Research Group
Los Alamos National Laboratory
Los Alamos, NM 87545

1. Introduction

The purpose of this paper is to study delays encountered when several vector processors access a common memory. Such delays may be caused by access conflicts at the memory or by contentions within the network connecting processors to common memory. Because such delays are potentially the limiting performance bottleneck of common memory architectures, it is important to understand how they depend on design parameters, in particular the number of processors serviced by the common memory, the degree of memory interleaving, and the speed of the memory chips. Such understanding will make it possible to assess the performance of systems with increasing number of processors and new memory technologies. It is essential for any assessment of the future potential of common memory systems.

There are three possible approaches to the study of this problem, analytical models, simulations, and measurements on existing machines. All three approaches are necessary because each has advantages and limitations. Analytical models owe their effectiveness to simplifying assumptions that need to be checked by the other two methods. In particular, stochastic methods and queueing theory offer no techniques to deal with regular memory access patterns that are common in scientific and engineering applications. Simulations can generally approximate reality more closely, however, even simulations do not replace real measurements. They are often very slow and contain assumptions that need to be tested. This paper presents a set of measurements of memory access delays performed on several computers of the Cray family under carefully controlled conditions. We will compare the results with previously published modeling work [1,2].

We will outline in the next section essential features of the hardware and software architectures that are needed to understand the rationale of our approach and the interpretation of results. Section 3 is a description of measurement techniques and necessary precautions to make results reproducible. Section 4 presents results and a discussion and comparison to previously published models. We outline our conclusions in Section 5.

2. Characteristics of Hardware and Software

The measurements described in this paper were performed on three types of Cray computers with differing memory architectures. The systems are a CRAY X-MP with 4 processors, a CRAY Y-MP with 8 processors, and two CRAY-2 systems with 4 processors, all with common memory. Some essential system parameters are listed in Table 1.

Cray processors like all current vector architectures are register-to-register computers. Vector arithmetic units get their operands from and deposit their results into vector registers, which on Crays contain sixty-four 64-bit words. On the CRAY X-MP and Y-MP there are three access paths per processor connecting vector registers with main memory: two for load operations and one for store operations. Thus, for the vector operation under study in this paper, $V = V + V$, three memory streams are active simultaneously: two operand loads and the storing of the result vector proceed concurrently with the add operation only slightly offset in time. The CRAY-2, in contrast, has only one memory access path per processor. Therefore, load and store operations cannot proceed concurrently.

TABLE I. DESIGN PARAMETERS OF CRAY COMPUTERS

Model	Number of Processors	Clock Cycle Time (NS)	Number of Memory Banks	Bank Reservation Time (CP)
CRAY X-MP	4	8.5	64	4
CRAY Y-MP	8	6	256	5
CRAY-2S	4	4	128	13
CRAY-2D	4	4	128	45 ^a 17 ^b

^aLong Reservation Time

^bShort reservation Time

Vector operations of most high-performance vector computers access memory directly without an intermediate cache. To increase bandwidth, memory in such systems is subdivided into many parallel independent units, commonly called memory banks. Each bank, after being accessed successfully, cannot be accessed again for a period called the bank reservation time. It is distinct from the memory access time, the time between the issue of a memory access request and the arrival of the requested data item at the processor register. Bank reservation times are in general much longer than processor cycle times (see Table 1). To enable vector processors to access arrays in main memory at rates of one per processor clock cycle, adjacent array elements are stored in different memory banks. Usually, the low-order bits of the memory address determine the bank address. Thus, a vector access stream to contiguous locations does not encounter memory bank conflicts. However, several such access streams may have requests issued to a particular bank less than the bank reservation time apart, leading to a bank conflict and delays.

The memories of the computers under study differ in the number of memory banks and the bank reservation times, as shown in Table 1. The structure of the CRAY-2 memory with dynamic memory chips is slightly more complicated than described above. Each of its 128 memory banks is subdivided into two pseudo banks. After a successful access to a location within a pseudo bank, that particular pseudo bank is inaccessible for the period of the long bank reservation time τ_L (45 cycles in the system tested). However, its companion pseudo bank is also inaccessible for the period of the short bank reservation time τ_S (17 cycles in the tested system). For memory traffic distributed randomly, both companion pseudo banks are accessed with equal probability. Therefore, the memory should behave very much like a system with 128 banks and a bank reservation time of $(\tau_L + \tau_S)/2$ (31 cycles in our case).

In addition to memory bank conflicts, access conflicts within the network connecting processors to main memory may slow down memory access. The architectures of the interconnection networks are quite different for the three Cray computers under study. While the CRAY X-MP is equipped with a bus type network [3], which is prone to access conflicts, the CRAY Y-MP provides an independent access path for each processor to each bank [4]. Here conflicts between the three paths for each processor are resolved within the network. The CRAY-2 network differs considerably from the two others and is highly buffered. Major access delays occur when these buffers fill up [5]. Of course the number of access requests waiting in network buffers will be strongly dependent on the number of bank conflicts at the memory.

In addition to differences in the hardware architecture the three machines exhibit different strategies of conflict handling. Operands have to arrive in sequence on the CRAY X-MP and Y-MP. Therefore each memory access conflict causes a whole access stream to halt. If the arrival of a single vector element is delayed all subsequent elements are delayed by at least that amount. Thus, the delay in the arrival of the 64 word content of a vector register is the sum of all individual delays. On the CRAY-2 vector elements do not have to arrive in sequence. The access stream will halt only if the interconnection network buffers are full. Thus, the total delay for the arrival of 64 words will be less than the sum of individual delays for light memory traffic.

The CRI CFT77 compiler running on all four systems practices vector loop unrolling [6] to hide vector strip startup times at least partially. The level of unrolling for simple vector loops is two for the CRAY X-MP and Y-MP, and four for the CRAY-2. Vector loop unrolling is responsible for the unusual dependence of vector execution times on vector length seen in the measurements of this paper and is discussed in more detail in a companion paper [7].

3. Measurement Technique

We have measured execution times of simple vector operations as a function of vector length and the number of "interfering processes." While one processor was running the instrumented "test process," the other processors of the system were either kept idle or running an "interfering process." We will report results of only one "test process" in this paper, $V = V + V$, the addition of a contiguous vector to another contiguous vector. This operation involves two vector load operations, a vector add, and the subsequent store of the contiguous result vector. Results obtained for other test operations are very similar to those reported for $V = V + V$ and will therefore not be discussed. The interfering processes were vector add operations involving very long vectors. Operands and results of interfering processes were either in contiguous memory locations, randomly scattered across memory, or stored in nonadjacent locations with stride 23. We did not access memory with stride equal to powers of two that can cause severe access conflicts within a single vector access stream. We feel that such access conflicts can generally be avoided.

All measurements were performed on dedicated systems by necessity. However, a dedicated system was frequently not sufficient to assure correctness or reproducibility of results. In one case, the idle loop of the operating system accessed memory at rates comparable to those of our interfering processes. We therefore found it necessary to either disable idle processors by turning the hardware off or to run on them a loop written in assembly language that spent its time accessing registers only. With these precautions our timing results without interfering processes were reproducible with very few exceptions. The exceptions were values far out of their range and could be identified and discarded easily. With increasing numbers of interfering processes the noise in the collected data increased. However, we could still readily identify far out values.

The hardware cycle counter was used for the measurements reported here. This timing call constitutes a single machine instruction on Cray computers. We timed single vector operations and corrected the results for the duration of the timing operations. As noted in [8], execution times for short vectors may depend on the way times are measured. We timed each operation several times and eliminated outlying values as discussed above.

All measurements reported here were performed on standard fortran code compiled by various versions of the Cray Research, Inc., CFT77 compiler. Special precautions are necessary with optimizing compilers that like to eliminate code considered to be unnecessary. We have used a variety of tricks to outwit compilers. Examples are putting vectors in common blocks, printing some results, and constructing infinite loops with if tests. In any case, we considered it necessary to inspect the code produced by the compiler to make sure we were timing the operations we intended to measure.

Even with only a single processor active, execution times of vector operations may depend on the relative location of operands and results in memory as shown in [8], with differences as high as 50% of the average execution times. This is due to access conflicts between access operations originating from a single processor. Even on the CRAY-2 with only a single access path to memory for each processor, the start of a vector access operation may interfere with the tail end of the previous one. To eliminate these effects we have timed our test codes for all possible relative locations of operands and result vectors. Assuming that all relative locations are equally likely to occur, we are presenting averages of these measurements. For computer systems with m memory banks and the test operation $V = V + V$, m^2 measurements were averaged for a single data point.

The above precautions are necessary to obtain meaningful data not only on Cray computers but are equally applicable to similar studies on other machines.

4. Results

Plots of measured vector execution times as a function of vector length are presented in Figs. 1 through 4 for the four systems under study, the CRAY X-MP, the CRAY Y-MP, and two versions of the CRAY-2, one with static and one with dynamic memory. The lowest curve in each plot represents data collected with only the test process running on a single processor, the remaining curves represent results with one or more interfering processes running in addition to the test process on the multiprocessor system, up to seven on the CRAY Y-MP and up to three on the other systems. The interfering processes in each case are the vector addition of two very long vectors. While the test process always involves stride-1 operations, i.e. vectors stored in contiguous memory locations, interfering processes do not all follow this regular memory access pattern. In parts *a* of all figures, interfering processes A proceed with stride 1. In parts *b* of all figures, one operand of interfering processes B is stored at random memory locations. In addition to the gather operation, which accesses memory in a random fashion, and the fetch of the other operand and store of the result, which access memory with stride 1, the index vector has to be loaded, again a stride-1 operation. Therefore, the processing of each strip involves a total of 4 vector memory access operations. With only three memory access ports available on the CRAY X-MP and CRAY Y-MP, not all these operations can proceed concurrently. The attempted memory access rate of interfering process B is therefore less than that of process A. The predicted factor is 2/3, we measured 0.58. The attempted access rate is unchanged for the CRAY-2 with only one access port per processor. In spite of the lighter interfering memory traffic, performance is degraded more severely by interfering process B with one random memory access out of four than by interfering process A with all regular access operations. In parts *c* of all figures interfering processes C access memory with stride 23. The attempted memory access rate is 0.96 of that of process A for CRAY X-MPs and Y-MPs. Performance degradation is very serious, especially for the CRAY X-MP and CRAY-2 systems. We took measurements for other prime strides, including mixed strides, with similar results. For the CRAY-2, attempted memory access rates are the same for all three interfering processes. All four systems tested show severe performance degradations due to memory access conflicts. The figures indicate that heavy vector traffic can cause slowdowns by more than a factor of three. In normal machine operation memory traffic is generally less intense and therefore affects performance to a lesser extent.

While Figs. 1 through 4 represent raw measured data, we have subtracted the execution time of the undisturbed test process from the times measured with one or more interfering processes. Figures 5 through 8 therefore represent the measured delays as functions of vector length for one or more interfering processes. The most striking observation from these plots is the relatively modest performance degradation for interfering processes A (with stride 1) in parts *a* of all figures as compared to the very severe degradation due to strided access operations in parts *c* of all figures, or even interfering processes B with only one of two operands accessed at random locations. For the CRAY-2 the ratio can be as high as 12. But all systems measured show the same trend.

While the delays for random and strided access (parts *b* and *c* of figures) are approximated well by a linear function of vector length, the delays for interfering process A (contiguous access) are not. They show a structure similar to that of the execution time of the test process with marked discontinuities at integer multiples of the 64, the vector register length. In addition, the slope of the curves decreases for long vector lengths.

We interpret these differences as follows. While interfering processes B and C interfere with memory access operations of the test process roughly at random, interfering process A and the test process will synchronize their memory access operations in such a way that after an initial startup delay very few collisions will occur during the processing of a strip. The curves will therefore show discontinuities at integer multiples of the vector register length and saturation for very long vectors. The overall disturbance rate will be very low.

We have fitted straight lines to the data points of parts *b* and *c* of Figs. 5 through 8 and computed the initial slopes of the curves in parts *a*. These slopes represent the average delays per computed result and are plotted in Figs. 9 through 12 as functions of the number of interfering processes for all measured systems. The dashed curves represent results obtained from a queuing model of memory access [1,2]. Agreement between the measured data and the queuing results is not expected to be close because the model represents delays due to memory bank conflicts only and not delays within the access networks. It should therefore set a lower limit for delays achievable for a very well designed interconnection network. Figure 10 indicates that the CRAY Y-MP comes close to this ideal. The measured delays for random and strided access (curves B and C) are close to those predicted by the

model that assumes random memory access patterns. The model predicts zero delays for regular access operations (curve A). It does not handle startup delays. It is interesting to note that the slope of the curves A for stride-1 access operations is initially close to zero and increases with increasing number of processes. We ascribe this to the fact that with increasing memory traffic and increasing number of delays, memory access for interfering process A becomes less regular. Therefore, curve A should approach the delays for random access for very heavy memory traffic (large numbers of interfering processes).

The CRAY X-MP shows delays that are considerably higher than those predicted by the model. We therefore conclude that the network contributes considerably to these delays.

Because vector elements do not have to arrive in sequence at the vector registers of the CRAY-2, memory access delays due to bank conflicts should be considerably less than those predicted by the simple model, unless the network contributes significantly to the delays. Because calculated and measured values are of similar magnitude, we conclude that the contribution of the network is considerable. The ratios of delays for the two CRAY-2 systems measured is roughly equal to the ratio of their bank reservation times.

Our measurements yield memory access delays of roughly the same magnitude as those reported previously by Calahan and Bailey [9] under similar test conditions.

5. Conclusions

We can draw the following conclusions from our experiments:

- 1) Memory access conflicts can significantly degrade the performance of common memory multiprocessor systems. We have seen slowdowns by more than a factor of 3. Good network design is crucial as demonstrated by our results for the CRAY Y-MP network with an array of crossbar switches at the memory banks. In spite of its greater number of processors and higher processor speed, the performance of its memory is superior to that of its older siblings, the CRAY X-MP and CRAY-2. Hopefully, this design can be extended to larger numbers of processors. Careful modeling of interconnection networks will be essential for the design of future systems.
- 2) Delays are much worse (approximately one order of magnitude) for random access operations than for access with uniform stride at moderate memory traffic. Access operations with mixed strides are also badly degraded. Therefore, storage schemes designed to avoid single vector stream access conflicts for operations with stride divisible by certain powers of two may avoid these conflicts. However, because these schemes scramble stride-1 access operations, they will seriously degrade performance of multiprocessor systems. Because stride-1 access is the predominant mode of accessing vectors in most algorithms and because these operations run smoothly with current storage schemes, as shown by our measurements, this feature should not be sacrificed to the solution of a problem which is due to the design of some access networks. Instead, network design should avoid these conflicts.
- 3) In spite of existing problems, common memory multiprocessors are not at the end of their path. There is a lot of space for improvement to avoid memory access conflicts: better network design, caches for scalar data to avoid disturbing regular vector access operations, increasing the number of memory banks, and last but not least, latency hiding techniques.

Acknowledgements

We would like to acknowledge the generous assistance given to us by people from Cray Research, Inc., especially, Ken Lord, Vito Buongiorno, Bob Numrich, and Dave Foster. Vito and Ken, in particular, gave service above and beyond the call of duty, for which we are very grateful. Bob Numrich shared his extensive knowledge about the Cray-2 memory system, and Dave Foster ran interference for us. We would also like to thank Becky Koskela for her help with some of the measurements.

References

1. I. Y. Bucher and D. A. Calahan, "Access Conflicts in Multiprocessor Memories: Queueing Models and Simulation Studies," Proc. 1990 International Conf. on Supercomputing, pp. 428-438 (ACM Press, 1987).
2. I. Y. Bucher and D. A. Calahan, "Models of Access Delays in Multiprocessor Memories," submitted for publication to IEEE Transactions on Parallel and Distributed Computing.
3. Cray Research, Inc., CRAY X-MP Computer Systems Hardware Manual, "Four Processor Mainframe Reference Manual," #HR-0097 (1985).
4. Cray Research, Inc., CRAY Y-MP Hardware Manual, "CRAY Y-MP System Programmer Reference Manual," #CSM-0400-000 (1988).
5. Robert Numrich, "The Cray-2 Common Memory," Cray Research Inc. internal report, #HN-2043, (1988).
6. J. J. Dongarra and S. C. Eisenstadt, "Squeezing the Most Out of an Algorithm in Cray Fortran," ACM Trans. Math. Software 10, pp. 221-230 (1984).
7. I. Y. Bucher and M. L. Simmons, "How Hardware and Software Design Shape Vector Performance," submitted for publication to Supercomputing '91.
8. I. Y. Bucher and M. L. Simmons, "A Close Look at Vector Performance of Register-to-Register Vector Computers and a New Model," Proceedings of the 1987 ACM SIGMETRICS conference on Measurement and Modeling of Computer Systems," p. 39 (ACM Press, 1987).
9. D. A. Calahan and D. H. Bailey, "Measurement and Analysis of Memory Conflicts in Vector Multiprocessors," in *Performance Evaluation of Supercomputers*, J. L. Martin, Ed., (Elsevier, 1988).

Fig. 1. CRAY X-MP data.

Fig. 2. CRAY Y-MP data.

Measured vector execution times as functions of vector length for $V = V + V$ with stride-1 vectors and with zero to $p - 1$ interfering processes. Data points are connected for visual guidance only. The lowest curve in each plot is for the undisturbed test process, the uppermost for the maximum number equal to the number of processors in the system minus one. Part *a* of each figure represents data with stride-1 interfering processes, part *b* with processes involving one gather operation, and part *c* with all vectors of interfering processes proceeding with stride 23.

Fig. 3. CRAY 2 data for static memory.

Fig. 4. CRAY 2 data for dynamic memory.

Measured vector execution times as functions of vector length for $V = V + V$ with stride-1 vectors and with zero to $p - 1$ interfering processes. Data points are connected for visual guidance only. The lowest curve in each plot is for the undisturbed test process, the uppermost for the maximum number equal to the number of processors in the system minus one. Part *a* of each figure represents data with stride-1 interfering processes, part *b* with processes involving one gather operation, and part *c* with all vectors of interfering processes preceding with stride 23.

Fig. 5. CRAY X-MP data.

Fig. 6. CRAY Y-MP data.

Average delays caused by memory access contentions of one to $p - 1$ interfering processes as functions of vector length for the contiguous vector operation $V = V + V$. Data points are connected for visual guidance only. Part *a* of each figure represents data with stride-1 interfering processes, part *b* with processes involving one gather operation, and part *c* with all vectors of interfering processes proceeding with stride 23.

Fig. 7. CRAY 2 data for static memory.

Fig. 8. CRAY 2 data for dynamic memory.

Average delays caused by memory access contentions of one to $p - 1$ interfering processes as functions of vector length for the contiguous vector operation $V = V + V$. Data points are connected for visual guidance only. Part a of each figure represents data with stride-1 interfering processes, part b with processes involving one gather operation, and part c with all vectors of interfering processes proceeding with stride 23.

Fig. 9 Average delay per computed result as a function of the number of interfering processes for the CRAY X-MP. Curve A is for interfering processes with contiguous memory access, curve B for processes involving one gather operation, and curve C is for processes with stride 23. The dashed lines represent values computed from a queuing model for memory bank conflicts, the upper curve for the memory access rate of curve C, the lower one for the rate of curve B.

Fig. 10 Average delay per computed result as a function of the number of interfering processes for the CRAY Y-MP. Curve A is for interfering processes with contiguous memory access, curve B for processes involving one gather operation, and curve C is for processes with stride 23. The dashed lines represent values computed from a queuing model for memory bank conflicts, the upper curve for the memory access rate of curve C, the lower one for the rate of curve B.

Fig. 11. Average delay per memory access as a function of the number of interfering processes for the CRAY 2 with static memory. Curve A is for interfering processes with contiguous memory access and curve C is for processes with stride 23. The dashed line represents values computed from a queuing model for memory bank conflicts for random access.

Fig. 12. Average delay per memory access as a function of the number of interfering processes for the CRAY 2 with dynamic memory. Curve A is for interfering processes with contiguous memory access, curve B for processes involving one gather operation, and curve C is for processes with stride 23. The dashed line represents values computed from a queuing model for memory bank conflicts for random access.