CCHIME:
A Cache Coherent Hybrid Interconnected Memory Extension

Matthew Farrens
Division of Computer Science
University of California
Davis, CA 95616
tel: (916) 752-9678
fax: (916) 752-4767
email: farrens@cs.ucdavis.edu

Arvin Park
Division of Computer Science
University of California
Davis, CA 95616
tel: (916) 752-5183
fax: (916) 752-4767
email: park@cs.ucdavis.edu

Allison Woodruff
Division of Computer Science
University of California
Davis, CA 95616
tel: (916) 752-8878
fax: (916) 752-4767
email: woodruff@cs.ucdavis.edu

Abstract

This paper presents a hybrid shared memory architecture which combines the scalability of a multistage interconnection network with the contention reduction benefits of coherent caches. We achieve this by replacing the memory modules and final stages of a multistage interconnection network with clusters of coherent caches. The performance of Cache Coherent Hybrid Interconnected Memory Extension (CCHIME) is evaluated by analyzing the results of extensive simulations of the network and coherent cache clusters. These results indicate that the CCHIME architecture can achieve lower memory access latencies and higher throughputs than typical multistage interconnection networks.

1. Introduction

In order to achieve the goal of teraflop computing speeds by the end of the century, the number of processing nodes in a multiprocessor will have to increase substantially. However, as the number of processors grows, so does the impact of two problems inherent to multiprocessing in a shared memory environment: how can memory and processors be efficiently connected, and how can the side effects of memory contention be controlled?

A number of architectures are commonly used to interconnect a large number of processors through shared memory. A bus-based shared memory architecture using coherent caches is the least expensive, but is inappropriate for systems with a large number of processors because the memory traffic is limited by the capacity of a single bus. Another popular shared memory architecture is a multistage interconnection network (MIN) which connects N processors through $\log_j N$ levels of $j \times j$ switches to N memory modules. Unlike bus-based shared memory architectures, MINs can scale to provide sufficient bandwidth for large numbers of processors. However, MIN architectures can suffer significant performance limiting effects arising from contention.

Contention for memory modules in MINs leads to an undesirable phenomenon known as *hot spots*. Hot spots are nodes within the network, memory modules, or specific addresses which receive a disproportionately large share of memory traffic. This concentration of traffic can be caused by events such as references to shared or global variables, block transfers, or a coincidental concentration of requests to memory modules or internal switches. When nonuniform traffic occurs, queues at the hot spots become full and can not accept further requests; requests which are blocked will remain in the switches feeding the hot spot. The queues at these switches can also fill, and the effects will propagate backwards through the network forming a tree of nodes whose queues are full. This condition is called *tree saturation* [PfNo85].

A number of methods of improving MIN performance in the presence of hot spots and tree saturation have been studied. In this paper, a hybrid of shared bus and multistage interconnection networks called CCHIME (Cache Coherent Hybrid Interconnected Memory Extension) is proposed. Instead of switching nodes, the last levels of the network are replaced by small clusters of coherent caches, where each cluster of caches connects to a single shared memory module as is illustrated in Figure 1. Note that coherence is only maintained within cache clusters; maintaining coherence globally across clusters is not necessary. Each coherent cache cluster along with its associated memory module can be thought of as functioning as a single multi-ported memory. If the cache clusters are small enough, and the cache hit rates are high enough, cache accesses should suffer little performance degradation due to cache miss delays.

1.1. Previous Approaches

The use of caches at the processors to mitigate traffic in a multistage interconnection network has been previously suggested. However, while caching at this level improves the latency of cold requests, it does not alleviate memory contention and tree saturation caused by

This work was supported by the National Science Foundation under Grants CCR-90-11535, CCR-90-122800, by the Hewlett-Packard Corporation through gift number 17-01080, and by the University of California under MICRO Grants 91-118 and 91-033.

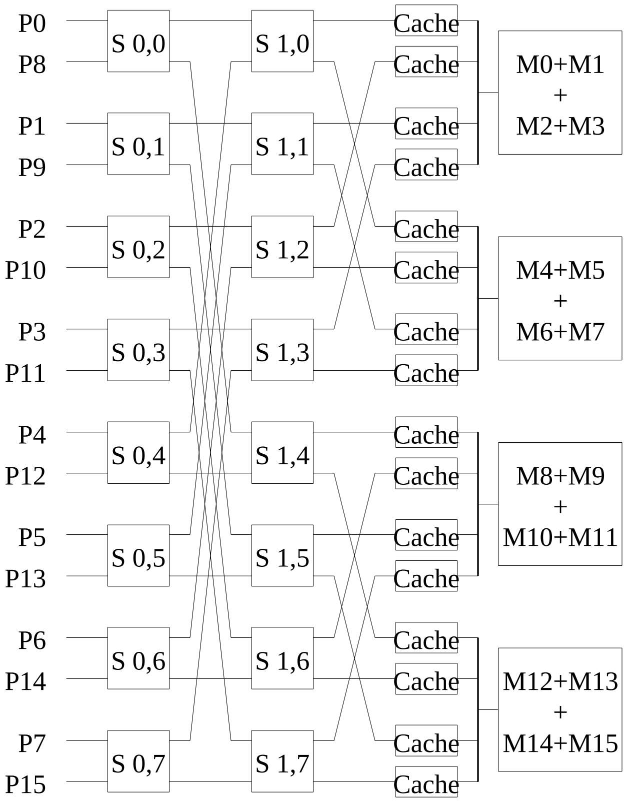

Figure 1.
CCHIME Topology with Cache Cluster Size = 4

hot spots [PfNo85]. Systems with such caches must also deal with complex cache coherency issues, some of which are addressed in Stenstrom [Sten89].

The incorporation of caches in the switching nodes of the network has also been discussed [MaTi90, MBLZ89]. However, simulations have indicated that the substantial complexity and expense of maintaining caches and cache coherence at all nodes internal to the network is not warranted by the limited increase in throughput.

A hybrid network combining a MIN and a collection of shared busses has been proposed by Harper and Jump [HaJu90]. In this architecture the final stages of a multistage interconnection network are replaced by busses which connect directly to memory modules. This architecture can be thought of as a compromise between bus-based and MIN based architectures. The Harper Jump hybrid provides greater throughput than a bus, but less than a MIN. It also is more expensive to implement than a bus, but less expensive than a MIN.

The CCHIME architecture can be viewed as a modification of the Harper and Jump architecture in which the shared busses that connect to memory modules are equipped with coherent cache clusters. These coherent caches clusters allow the CCHIME architecture to actually achieve performance that is superior to an unmodified MIN. (the incorporation of caches at the memory modules is orthogonal to other methods of controlling memory contention and tree saturation, such as *combining* [GGKM83, LeK86, PfNo85]).

2. The CCHIME Architecture

CCHIME is a hybrid of the bus-based cache-coherent and multistage interconnection network architectures. In the CCHIME architecture the function of the multi-stage interconnection network is **not** to connect processors and memory modules, but to connect processors and caches (see Figure 1). As can be seen in Figure 1, groups of memory modules are combined into larger memory modules, and caches take the place of the original smaller memory modules. These caches are grouped in coherent clusters, and all caches in a cluster are connected by a bus to a single memory module. Since the memory module is now much larger, covering more addresses, the number of network stages required has been reduced. In the case of figure 1, four memory modules have been combined and replaced by four caches, and the number of stages in the interconnection network has been shorted by two.

Bus-based coherence schemes are not scalable past a certain point due to bandwidth limitations of the coherence bus. However, they can be effective for limited cluster sizes [ArBa86]. Therefore, CCHIME only provides cache coherence across **small** clusters of caches attached to a single memory module.

This approach can reduce both the amount of memory contention and the latency of memory requests in a number of ways. For example, concurrent read requests to a single memory location can be processed simultaneously, since the memory location can be replicated across multiple caches in a cluster. Caches also typically have faster access times than memory modules, which reduces latency for cache hits. In addition, due to the smaller number of levels in the network, requests pass through fewer switches. By decreasing both the number of levels in the network and the degree of memory contention, the amount of tree saturation which occurs in non-uniform traffic conditions is reduced, thereby lowering average request latency.

3. Simulations

3.1. The Network Model

In order to test the effectiveness of this architecture, a number of modifications to the MIN simulator used in [FaWW91, ScSo90] were made and a range of simulations were performed. The topology of the network used in the CCHIME architecture is that of a data manipulator, which has been shown to be isomorphic with the flip, omega, banyan, and indirect binary *n*-cube networks. The processors and caches are connected by a network with

2×2 crossbar blocking switches. The number of levels in the network vary as a function of the number of processors (and caches), N, and the size of a cache cluster, K. Each time K increases by a power of 2, the new caches replace one additional level in the network. Each processor can reach exactly one of the caches in each of the M/K clusters via the network.

Each switching node is capable of receiving one request from each node input per cycle, and will place that request in a FIFO queue before the next network cycle; queues can accept up to two inputs per cycle. Each queue feeds the next stage of the network, and can hold up to 4 elements. The queue that lies between the final stage of the network and the caches can hold up to 8 elements.

The model also provides two complete interconnection networks, one to propagate requests from the processors to the caches, and one for the reverse trip. Each processor has a network interface which interacts with both the forward and the reverse networks and is responsible for requests being released or received. Each processor may make at most one request per network cycle via its processor network interface, and there is no limit on the number of requests a processor may have outstanding.

3.2. The Coherent Cache Cluster Model

Caches are grouped in coherent clusters, and all caches in a cluster are connected by a bus to a single memory module. If the coherence bus is busy processing a cache miss when a new cache miss occurs, the new miss queues up for access to the cache bus. On writes to shared data, coherence is maintained within each cluster using a *write−invalidate* policy for updating the other caches. Simulations were performed assuming that each cache miss requires either three or five processor cycles (six or ten network cycles) to service. We expect a cache miss to require about three processor cycles to service, but additional simulations with the more pessimistic five processor cycles per cache miss were done to examine the impact of a slower cache miss service time on performance.

Cache hit rates of 95%, 97%, and 99% were simulated. These cache hit rates are meant to reflect a mixture of instruction and data references for a relatively large cache, (256K to 1M bytes). Although data hit rates will probably be lower than 95%, the instruction hit rates should be quite high. Because instructions account for the majority of cache accesses, the high instruction hit rates will result in high combined instruction/data hit rates.

3.3. The Workload Model

Our workload model assumes a uniform distribution of memory requests with a small user-selectable percentage of requests directed to a designated "hot" memory module. The model does not differentiate between time spent in hot spot versus uniform traffic conditions. It has been noted that hot spots are transient [LeK86], but the amount of time spent in different states (both in terms of the degree of nonuniformity and the percentage of hot processors) has not been fully explored in the literature, and is not examined here. We assume a steady-state representation. Further, the model is somewhat optimistic in forcing only one hot spot at a time. It will allow more than one hot spot to arise, but this will only happen if the random distribution of uniform requests leads to an additional hot spot. In practice there may be nonuniform requests to several memory modules simultaneously, although Pfister and Norton suggest that typically there are only one or two hot memory modules at one time [PfNo85].

3.4. The Simulations

The simulator was run under a variety of conditions, varying parameters such as number of caches in each cluster, hot rate, cache access time, cache hit rate, etc.

Each processor will attempt to offer a request during each processor cycle time, P. If the network can not accept the request because the first-stage queue is full, the processor must wait to resubmit. Simulations were run with processor cycle time $P = 2$. A processor cycle time of 2 indicates that a processor cycle is twice as long as a network cycle. We expect the processor cycle time to be longer than a network cycle time because the network nodes perform much simpler operations during a network cycle than a processor does during a processor cycle.

For each processor, the effective maximum bandwidth is calculated by the following formula:

Issue Rate = (Total Number of Requests) / (TIME x N)

where *TIME* is the simulated number of network cycles elapsed, accumulated from the first cycle in which a request was generated. The maximum bandwidth of a MIN can only approach unity, even if there are no hot spots, because collisions in both the nodes and the memory modules occur even under uniform traffic [DiJu81].

3.5. Performance of CCHIME

Our simulations measured the performance of the CCHIME architecture in terms of issue rate and round trip memory latency. In the simulation results presented in this paper, the number of processors was set to 256, the number of network cycles per processor cycle was set to two, and the cache access time was set equal to the processor cycle time. As stated earlier, processors generate requests to random memory modules on every processor cycle, and there is no limit to the number of requests a processor can have outstanding.

These parameters were chosen to maximize the number of requests the network must deal with, in order to observe the performance of the network under the most pessimistic conditions.

3.5.1. Simulation Results

Figure 2 shows the issue rates of requests for simulations with a hot rate of 2%, five different memory configurations (cache clusters K of size 0, 2, 4, 8, and 16), the hit rate of the caches set to 95%, 97%, and 99%, and the time to service a cache miss set to 10 network cycles (5 Processor Cycles).

The memory modules are assumed to be pipelined - a cache miss at time t will be serviced at time t + Cache Miss Service Time. However, the model does take into account coherence bus collisions by forcing multiple misses within a coherent cache cluster to wait extra cycles until the bus becomes clear. (Note that the zero cache case is simply an unmodified MIN network without caches.)

Figure 2 shows how CCHIME can dramatically increase the issue rate. Examining the figure we see that when all the processors are making hot requests, configurations with caches uniformly provide a higher issue rate than the zero cache configuration. In fact, the configuration with K=4 has an issue rate that is usually over twice as high as that of a configuration with no cache clusters, and in many cases is over 2.5 times as high.

Figure 2 also shows that the CCHIME architecture is relatively sensitive to cache hit rates for larger cache cluster sizes. This is not unexpected, since a larger number of caches in a cluster will cause more contention for the coherence bus and memory (just as in a bus-based multiprocessor the bus begins to saturate as the number of processors increases). These effects can be seen quite clearly by looking at figures 2b and 2c; the larger cluster sizes (8, 16) start to show serious performance degradation when the cache hit rate falls to 95%. However, it is also interesting to note that even under these conditions, when all processors are making requests to the hot module (at the 2% rate), the configuration with cache cluster size of 16 still outperforms the unmodified MIN network.

CCHIME has an even more dramatic impact on round trip latency. (Round trip latency is defined as the amount of time from the point at which a request successfully enters the network until its return to the issuing processor.) Round trip latencies do not increase dramatically for cache cluster sizes of 4 or more as the number of processors making requests to the hot module increases. In fact, for cache cluster sizes of 8 and 16, the round-trip times stay virtually constant. However, the sensitivity to cache hit rate does show up again; the round trip latency of K=16 is uniformly higher than other configurations.

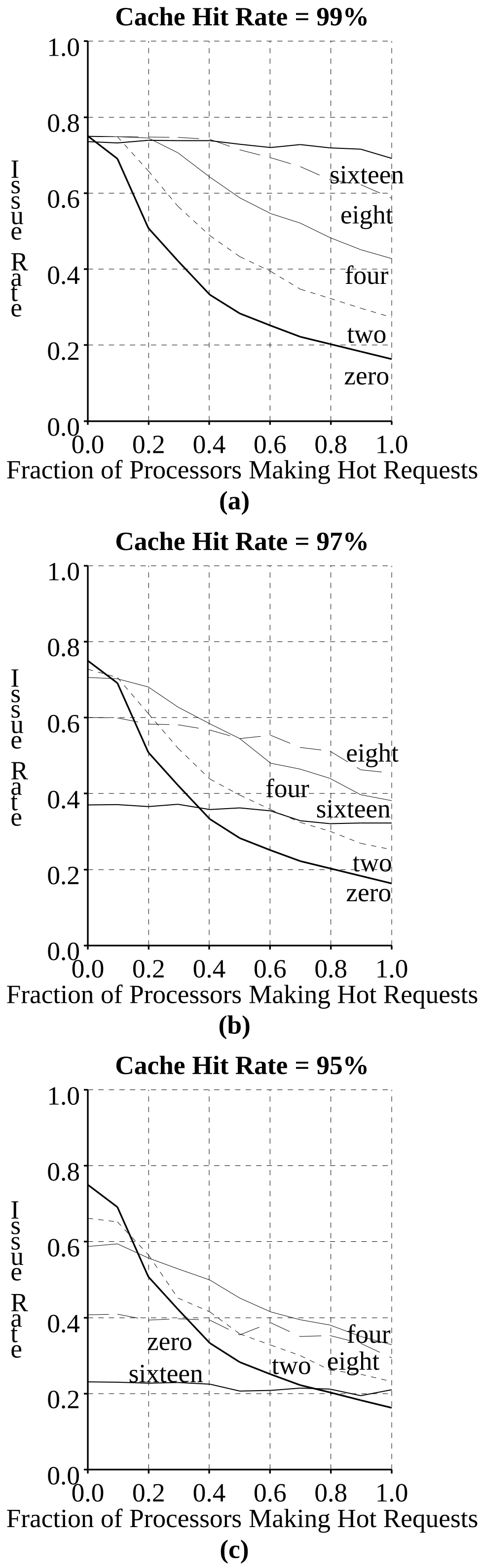

Figure 3.

Issue Rate of a CCHIME Network, Hot Rate = 2%
Cache Miss Service Time = 10 cycles

A cache cluster size of K=4 consistently outperforms the unmodified MIN network. When all processors are issuing hot requests the K=4 configuration round trip latency is 40% to 50% that of the zero cache configuration, and in all cases the K=4 configuration appears to be most insensitive to cache hit rates.

4. Conclusions

This paper has presented a hybrid shared memory architecture which combines the advantages of multistage interconnection networks and coherent caches. Simulation results show that this architecture decreases both the throughput limiting effect of hot spots as well as memory reference latency. According to simulation results, the optimal size for a cache cluster appears to be 4. This value shows the best resistance to coherence bus saturation effects, increasing the issue rate by a factor of 2 while providing a round trip latency 40%-50% that of an unmodified MIN network. Since CCHIME is a direct extension of existing MINs and cache coherent architectures, its implementation should be relatively straightforward.

The simulations performed so far present a worst-case scenario, with requests being issued every clock by all processors. Simulations currently underway will more closely model the actual performance of a CCHIME architecture. In addition, the characteristics of memory-side caches which service multiple processors must be examined in order to discover the optimal cache coherency protocol. Traces of actual parallel programs may also be useful in more accurately determining the behavior of caches in a CCHIME architecture.

5. Acknowledgements

We would like to thank Steve Scott and Guri Sohi for providing a copy of their network simulator.

References

[ArBa86] J. Archibald and J. Baer, ‘‘Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model’’, *ACM Transactions on Computer Systems,* vol. 4:4 (November 1986), pp. 273-298.

[DiJu81] D. M. Dias and J. R. Jump, ‘‘Analysis and Simulation of Buffered Delta Networks’’, *IEEE Transactions on Computers,* vol. C-30:4 (April 1981), pp. 273-282.

[FaWW91] M. K. Farrens, B. R. Wetmore and A. G. Woodruff, ‘‘Alleviation of Tree Saturation in Multistage Interconnection Networks’’, *Proceedings of Supercomputing ’91*, Albuquerque, New Mexico (November 18-20, 1991), pp. 400-409.

[GGKM83] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Randolph and M. Snir, ‘‘The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer’’, *IEEE Transactions on Computers,* vol. C-34:10 (February 1983), pp. 175-189.

[HaJu90] D. T. Harper and J. R. Jump, ‘‘Evaluation of Reduced Bandwidth Multistage Networks’’, *Journal of Parallel and Distributed Computing,* vol. 9 (July 1990), pp. 304-311.

[LeK86] G. Lee, C. P. Kruskal and D. J. Kuck, ‘‘The Effectiveness of Combining in Shared Memory Parallel Computers in the Presence of ‘Hot Spots’’’, *International Conference on Parallel Processing*(1986), pp. 35-41.

[MaTi90] S. M. Mahmud and V. Tiruveedhula, ‘‘Hit Ratio and Communication Cost of Shared Data in a Cache-Based System with Multistage Interconnection Network’’, *International Conference on Parallel Processing*(1990), pp. I-173-I-176.

[MBLZ89] H. E. Mizrahi, J. Baer, E. D. Lazowska and J. Zahorjan, ‘‘Introducing Memory into the Switch Elements of Multiprocessor Interconnection Networks’’, *Proceedings of the Sixteenth Annual International Symposium on Computer Architecture*(1989), pp. 158-166.

[PfNo85] G. F. Pfister and V. A. Norton, ‘‘‘Hot Spot’ Contention and Combining in Multistage Interconnection Networks’’, *IEEE Transactions on Computers,* vol. C-34:10 (October 1985), pp. 943-948.

[ScSo90] S. L. Scott and G. S. Sohi, ‘‘The Use of Feedback in Multiprocessors and Its Application to Tree Saturation Control’’, *IEEE Transactions on Parallel and Distributed Systems,* vol. 1:4 (October 1990), pp. 385-399.

[Sten89] P. Stenstrom, ‘‘A Cache Consistency Protocol for Multiprocessors with Multistage Networks’’, *Proceedings of the Sixteenth Annual International Symposium on Computer Architecture*(1989), pp. 407-415.