

Design and Performance of Directory Caches for Scalable Shared Memory Multiprocessors

Maged M. Michael Ashwini K. Nanda

IBM Research
Thomas J. Watson Research Center
Yorktown Heights, NY 10598
{michael,ashwini}@watson.ibm.com

Abstract

Recent research shows that the occupancy of the coherence controllers is a major performance bottleneck for distributed cache coherent shared memory multiprocessors. A significant part of the occupancy is due to the latency of accessing the directory, which is usually kept in DRAM memory. Most coherence controller designs that use protocol processors for executing the coherence protocol handlers use the data cache of the protocol processor for caching directory entries along with protocol handler data. Analogously, a fast Directory Cache (DC) can also be used by the hardwired coherence controller designs in order to minimize directory access time. However, the existing hardwired controllers do not use a directory cache. Moreover, the performance impact of caching directory entries has not been studied in the literature before.

This paper studies the performance of directory caches using parallel applications from the SPLASH-2 suite. We demonstrate that using a directory cache can result in 40% or more improvement in the execution time of applications that are communication intensive. We also investigate in detail the various directory cache design parameters: cache size, cache line size, and associativity. Our experimental results show that the directory cache size requirements grow sub-linearly with the increase in the application's data set size. The results also show the performance advantage of multi-entry directory cache lines, as a result of spatial locality and the absence of sharing of directories. The impact of the associativity of the directory caches on performance is less than that of the size and the line size.

Also, we find a clear linear relation between the directory cache miss ratio and the coherence controller occupancy, and between both measures and the execution time of the applications, which can help system architects evaluate the impact of directory cache (or coherence controller) designs on overall system performance.

1 Introduction

Previous research has shown convincingly that scalable shared-memory performance can be achieved on directory-based cache-coherent multiprocessors such as the Stanford DASH [5] and MIT Alewife [1] machines. A key component of these machines is the coherence controller on each node that provides cache coherent access to memory that is distributed among the nodes of the multiprocessor. Each coherence controller is responsible for keeping track of the state and location of memory lines in a part of the

global shared memory, typically the part residing in the same node as the coherence controller. The state information necessary for such a task is stored in the *directory*, which is usually implemented in DRAM memory either separately or along with the local node's main memory.

Designs for coherence controllers fall in two broad categories: designs which use programmable protocol processors to implement the coherence protocols and designs which hardwire the coherence protocol in custom hardware finite state machines. The former designs such as in the Stanford FLASH [3] and Wisconsin Typhoon [10] architectures, use the data cache of the protocol processor to cache directory entries in order to reduce directory access latency. The Sequent NUMA-Q [6] architecture uses a programmable engine, but includes only a small Tag Cache used only for transactions in progress.

The custom hardware designs of the Stanford DASH [5], MIT Alewife [1], Sun S3.mp [9], and SGI Origin [4] use DRAM directories and do not include DCs. The HAL S1 [11] architecture uses a sparse directory implemented in SRAM and therefore is in less need of directory caching. However, almost all of the other recent commercial distributed shared memory architectures use full directories, which are too large to be implemented in SRAM. It seems that these architecture would benefit from reducing the directory access time by using a directory cache (DC).

Recent research results [2, 7] show that the occupancy of the coherence controller can be the performance bottleneck for applications with high communication requirements. Motivated by these results and the fact that a major part of coherence controller occupancy is due to directory access latency, we study in this paper the performance impact of using DCs on distributed shared memory multiprocessors. The performance of caching directory entries has not been studied before in the literature.

In addition to determining the performance impact of DCs, we examine in detail the interaction between the main DC parameters: size, line size, and associativity, and system and application parameters that would influence their performance such as the data set size and the coherence unit size. We base our experimental evaluation of DC performance on realistic hardware parameters of system components. We simulate applications from the SPLASH-2 benchmark suite [12] to identify the common trends in DC performance.

We find that DCs have significant impact on the performance of applications with high communication requirements. We also find that DC size requirements grow sub-linearly with data set size. We show that DC line size is a major factor in DC performance, with multi-directory-entry DC lines benefitting from spatial locality. We

also show the effect of DC size and coherence unit size on the choice of the DC line size.

DC associativity plays a minor role in DC performance relative to DC size and DC line size, and the largest gain in performance with respect to DC associativity is from 1-way to 2-way associativity. Also, a 4-way set associative DC almost shadows the performance of a fully associative DC.

By relating DC miss ratio to coherence controller occupancy and execution time of the applications, we find a clear linear relation between these measures. This result emphasizes the importance of DCs, and also provides a helpful tool for systems designers in relating coherence controller performance, including the DC, to overall system performance.

The main contributions of this paper are: (1) it demonstrates the significant impact of DCs on the performance of distributed shared memory multiprocessors, (2) it identifies and characterizes the most important DC parameters and their synergy with each other and the impact of other system parameters on them, and (3) it determines an important relation between coherence controller occupancy and execution time.

The rest of this paper is organized as follows. Section 2 presents the multiprocessor system and details coherence controller and DC design alternatives and parameters. Section 3 describes our experimental methodology and presents the experimental results and their implications. Finally, Section 4 presents our conclusions and recommendations for DC designs in future architectures.

2 System Description

The CC-NUMA multiprocessor system model used in this paper is composed of 16 SMP nodes connected by a 16 byte-wide fast switch. Each SMP node includes four 400 MHz PowerPC processors with 32 KB L1 and 1 MB L2 4-way-associative LRU caches. We vary the cache line size in our experiments. The SMP bus is a 133 MHz 16 byte-wide fully-pipelined split-transaction bus. The memory is interleaved and the memory controller is a separate bus agent from the coherence controller. Figure 1 shows a block diagram of an SMP node.

In this paper we consider a custom hardware coherence controller (CC) design as shown in Figure 2. Duplicate directories are used to allow fast response to common requests on the pipelined SMP bus (one directory lookup per 2 bus cycles). The bus-side copy is abbreviated (2-bit state per cache line) and uses fast SRAM memory. Since the controller-side copy of the directory is full-bit-map, SRAM memory is too expensive, therefore a dedicated DRAM directory memory is used.

A directory cache (DC) is used for reducing the directory read latency. We use a DC on the CC chip, except for one set of experiments where we also consider an off-chip DC implementation using SRAM with an optimized data path. We vary the DC size, line size, and associativity. We use the directory entry as the unit of DC size and DC line size. A directory entry consists of a presence bit vector, a modified bit, and a pending bit. The first time a directory entry is accessed by the protocol engine, the DC line containing that entry is cached in the DC. Any changes made to the cached copy of the directory is written through only to the bus side two-bit copy of the directory. A DC line continues to be in the DC until it gets replaced by another DC line due to DC capacity or conflict misses. Note that, since there is only a single writer to a particular directory entry, there

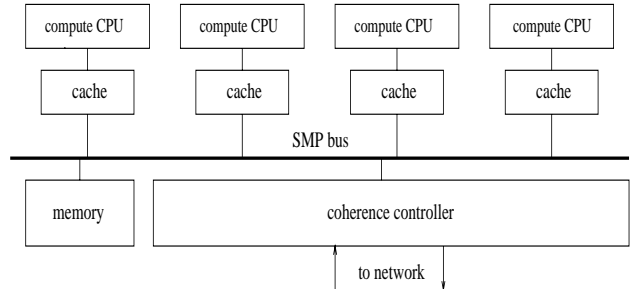


Figure 1: A node in a SMP-based CC-NUMA system.

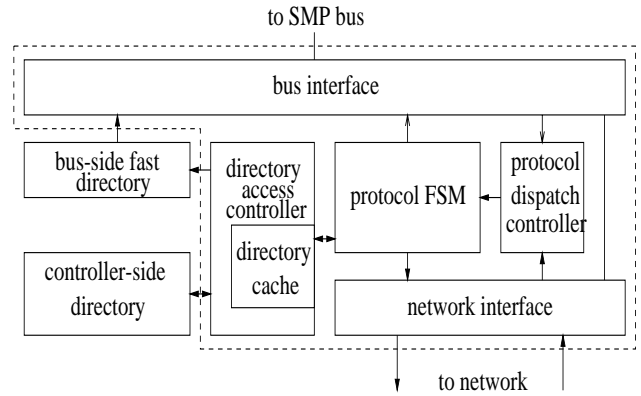


Figure 2: A custom-hardware-based coherence controller (CC) design with directory cache (DC).

are no sharing misses in the DC. The CC keeps using the locally cached copy until the line gets replaced. The controller side DRAM directory is updated only when a DC line gets replaced. For the 16 node system used in this paper, each directory entry will be 18 bits wide (presence vector + modified + pending). For example, a DC containing 4K directory entries requires 9KB (plus tag storage) of memory to store the DC lines.

The CC design includes a custom protocol dispatch controller for arbitration between the request queues from the local bus and the network. The CC runs at 133 MHz. Table 1 shows the no-contention latencies of key system and CC events.

3 Experimental Results

In this section, we present simulation results of the effect of various system and DC parameters on performance. First we demonstrate the impact of DCs on application performance. Then we investigate the main factors in DC design. We conclude this section with investigating the relation between DC performance and overall system performance. We start with the experimental methodology.

3.1 Experimental Methodology

We use execution-driven simulation (based on a version of the Augmint simulation toolkit [8] that runs on the PowerPC architecture) to evaluate DC performance. Our simulator includes detailed contention models for SMP buses, memory controllers, interleaved memory banks, protocol engines, directory DRAM, and external point contention for the interconnection network, in addition to accurate timing of the interaction between the coherence controller

Event	Latency
L1 hit	1
L2 hit (L1 miss)	13
L2 miss to address strobe on bus	6
Bus address strobe to bus response	21
Bus address strobe to start of cache-to-cache data response	27
Bus address strobe to next address strobe	6
Bus address strobe to start of data transfer from memory	30
Network point-to-point	54
CC issue request to bus	3
CC detect response from bus	3
CC issue network message	3
CC read special bus interface associative registers	6
CC write special bus interface registers	3
CC directory read (on-chip DC hit)	3
CC directory read (off-chip SRAM DC hit)	6
DRAM directory read	30
CC directory write	3
CC handler dispatch	3
CC condition	3
CC loop (per iteration)	3

Table 1: No-contention latencies in processor cycles (2.5 ns.).

Application	Type	Problem size
FFT	FFT computation	256K complex doubles
Ocean	Study of ocean movements	258×258 ocean grid
Radix	Radix sort	1M integer keys, radix 1K
Water-Nsquared	$O(n^2)$ study of forces and potentials in water molecules	512 molecules
Water-Spatial	Study of forces and potentials of water molecules in a 3-D grid	512 molecules

Table 2: Benchmark types and data sets.

and the SMP bus, memory, directory, and network interface.

We use five benchmarks from the SPLASH-2 suite [12], to evaluate the performance of directory caches. Unless otherwise mentioned, the data set sizes for the applications are as in Table 2. These applications have high communication requirements except for Water-Spatial which is included as an example of applications with low communication requirements. All the benchmarks are written in C and compiled using the IBM XLC C compiler with optimization level -O2. All experimental results reported in this paper are for the parallel phase only of these applications. We use a round-robin page placement policy. We ran all the applications with data sizes and systems sizes for which they achieve acceptable speedups.

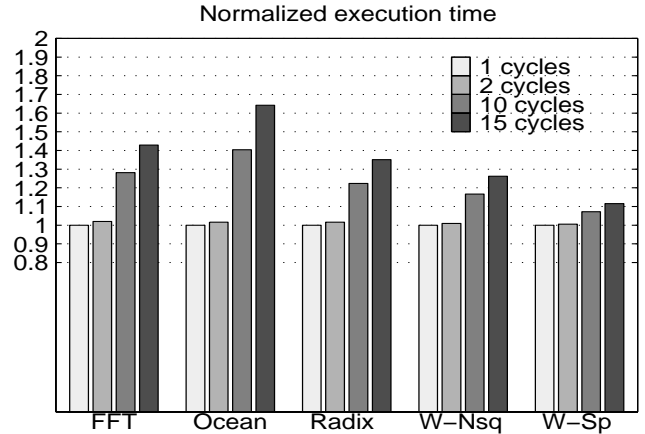


Figure 3: Effect of DC on execution time.

3.2 Performance Results

First, we determine the impact of DCs on overall system performance, then we investigate the design of each of the main DC parameters: size, line size, and associativity, and their synergy with each other and with other communication-related parameters such as the coherence unit size (L2 cache line size) and the data set size.

3.2.1 Impact of Directory Caches

Figure 3 shows the execution times for four different designs: (1) perfect (100% hit) on-chip DC with 1 cycle (7.5 ns.) access time, (2) perfect off-chip SRAM DC with 2 cycle access time, (3) DRAM directory without DC with 10 cycle (75 ns.) access time, and (4) a slower DRAM directory without DC with 15 cycle access time. The execution times are normalized to that of the first design. The L2 cache line size is 128 bytes.

We notice a significant increase (up to more than 40%) in execution time for the case with 10 cycle DRAM directory without DC over a perfect on-chip DC, indicating the possible significant performance gains of using DCs. We also notice more significant increase (up to more than 63%) in the relative execution time for the case with 15 cycle DRAM directory without DC, emphasizing the importance of using DCs in future systems as the gap between DRAM speed and processor, bus, and network speeds increases.

We also observe a minor increase in execution time for the 2 cycles SRAM DC relative to the on-chip DC (less than 2%), indicating the minor performance loss of using custom off-chip DC in cases where a large DC is needed, but cannot be accommodated on-chip.

3.2.2 Directory Cache Size

We examine the main factors in determining the effect of DC size on DC performance. We consider data set size and L2 line size (coherence unit size). For these experiments, we use 4-way set associative DCs and 4 directory entries per DC line.

Effect of Data Set Size

Figure 4 shows the DC miss ratios for a range of DC sizes with different ratios of data set size to DC size for each application. The L2 line size is 128 bytes. For FFT, Ocean, and Radix, we vary the data set size. For Water-Nsquared and Water-Spatial we

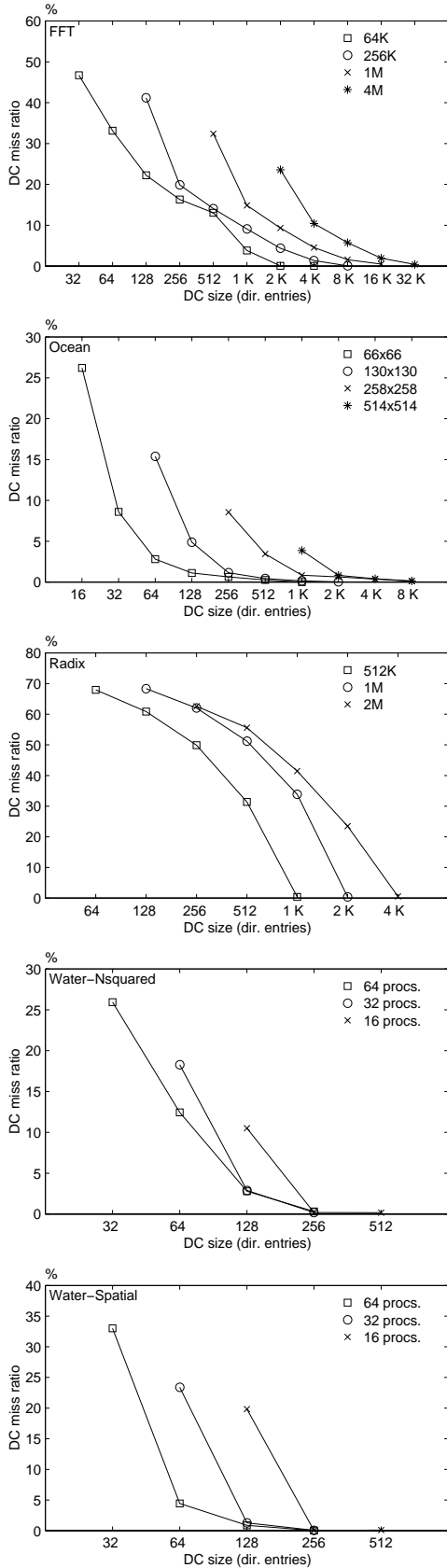


Figure 4: Effect of DC size on DC miss ratio with different data set sizes.

varied the data set size to DC size ratio by using the same data set size (512 molecules) on systems with 64 processors (16 nodes \times 4 processors), 32 processors (8 nodes), and 16 processors (4 nodes). This was forced by the prohibitively large simulation time of the next larger data set size that can be evenly distributed on 64 processors.

From the graphs we observe a clear trend of decrease in DC size requirements relative to the data set size with the increase in the data set size for the same level of DC performance. This implies the potential effectiveness and sufficiency of feasible directory cache sizes (e.g. 64KB on-chip or 4MB off-chip) for the requirements of real systems with large data sets. This is due to the sub-linear growth of the working sets of these applications with the data set size, which holds true for most applications [12].

We also observe that for all applications except Radix, the DC miss ratio drops rapidly (super-linearly) with the increase in DC size because the primary working sets for these applications are significantly smaller than the data set, while Radix has a relatively large primary working set [12].

Effect of Coherence Unit Size

Figure 5 shows the DC miss ratios for a range of DC sizes with different L2 cache line sizes (coherence unit size). As the L2 cache line size increases, the same DC size covers more memory, thus a proportional decrease in the DC size requirement is expected.

From the graphs we observe that with keeping the amount of memory covered by the DC constant, the performance of the DC drops with the increase in L2 line size, with varying degrees from one application to another. That is, if we double the L2 cache line size, cutting the DC size in half, we do not maintain the same DC performance. This is primarily due to the facts that increasing the L2 line size reduces the communication traffic sub-linearly, and that reducing the DC size increases the possibility of capacity misses, even when covering the same amount of memory. However, if the DC size remains constant, as expected, increasing the L2 line size improves the performance of the DC.

This observation is not intended for influencing the choice of the coherence unit size, which undoubtedly has greater impact on performance than the DC, but to help designers adapt DC size to changes in L2 line size in different generations of a system.

The inferior performance of small DC sizes with the larger L2 lines in the case of Radix is because larger L2 lines (128 and 256 bytes) cause more false sharing than 64 byte L2 lines, and accordingly causes more demand on the directory, yielding higher miss ratios. This effect is less evident for larger DC sizes as the higher capacity of the DC balances the effect of the higher demand.

3.2.3 Directory Cache Line Size

By increasing the DC line size (i.e. the number of directory entries per DC line), the DC can benefit more from the spatial locality of directory access without concern for the conventional communication drawbacks of large cache lines due to sharing, as directories are not shared. However, if increasing DC line size implies reducing the number of DC sets (i.e rows), large DC lines can cause more DC fragmentation (more capacity and conflict misses). In this case, we expect the balance between these two opposite factors to be mostly affected by DC size and L2 line size. We also consider the case where the number of sets remains constant with various DC line sizes in order to study the case where the number of sets in the DC

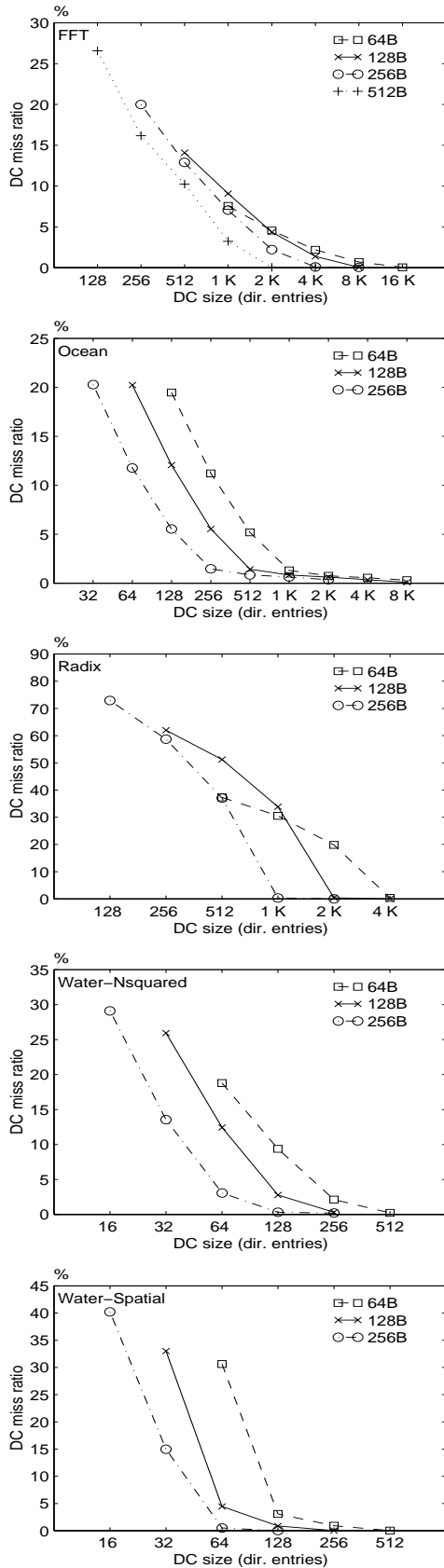


Figure 5: Effect of DC size on DC miss ratio with different L2 cache line sizes.

is more of a design constraint than the total DC size, for example by requiring one-cycle DC access latency, and to separate the effect of spatial locality from that of DC fragmentation.

Effect of Directory Cache Size

As in the previous subsection, we assume 4-way set associative DCs. Figure 6 shows the DC miss ratios for a range of DC line sizes with different DC sizes (total number of directory entries). The L2 line size is 128 bytes.

From the graphs we observe a significant impact of DC line size on DC miss ratio for all applications except Radix, and a wide range of optimal DC line sizes for different applications.

With larger DC sizes, we expect better performance for larger DC lines, as the DC becomes more tolerant to fragmentation and benefits more from spatial locality. We observe this effect with FFT and Ocean. For the other applications, DC size almost has no effect on the optimal DC line size.

Effect of Number of Directory Cache Sets

Figure 7 shows the DC miss ratios for a range of DC line sizes with different DC set numbers (number of rows in the DC). The L2 line size is 128 bytes. From the graphs we observe that larger DC lines clearly improve DC performance for all applications due to spatial locality, and that, in general, the benefits appear to diminish as the DC line sizes increase. A DC line size of 4 to 8 directory entries appears to be at the knee of the size-performance curve for DC lines.

Effect of Coherence Unit Size

Figure 8 shows the DC miss ratios for a range of DC line sizes with different L2 cache line sizes. The DC size is chosen for each application to be at the knee of the size-performance curve for 256 byte L2 lines.

From the graphs we observe a significant impact of DC line size on DC miss ratio for all applications, and a variation in optimal DC line sizes from one application to another. The variation of optimal DC line sizes with different L2 line sizes is not as large as their variation with different DC sizes. In general a DC line size of 4 or 8 directory entries appears to be the range with best performance in general.

3.2.4 Directory Cache Associativity

We examine the effect of DC associativity on DC performance. We show the results with various DC sizes and DC line sizes to evaluate the importance of DC associativity in comparison to the other DC parameters. The L2 line size is set to 128 bytes.

Figure 9 shows the DC miss ratios for a range of DC associativities with different DC sizes and a DC line size of 4 directory entries. Figure 10 shows the same relation with different DC line sizes, in order to determine the relative importance to DC performance of DC associativity vs. DC line size.

From the graphs we observe that, in general, DC associativity plays a minor role in DC performance in comparison to DC size and DC line size, and that it has more impact on performance with smaller DC sizes, as the chances of conflict misses are larger. We also observe that the largest differences in DC performance, with respect to associativity, are between 1-way and 2-way DCs, and that a 4-way DC almost shadows the performance of a fully-associative DC.

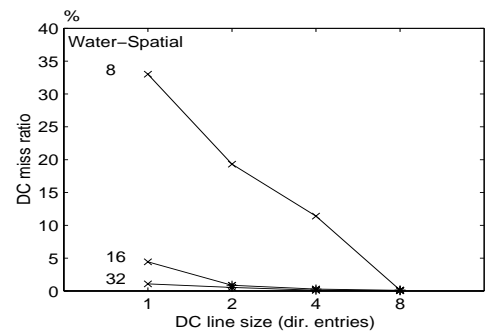
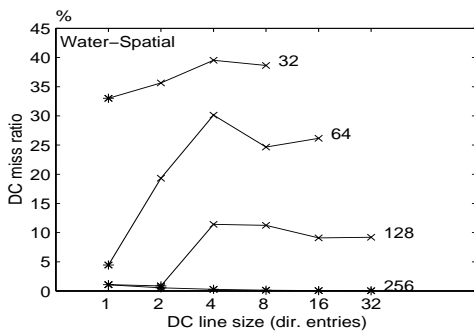
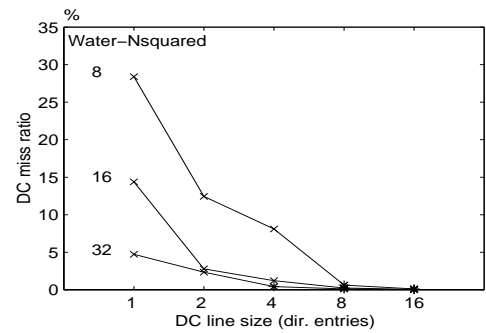
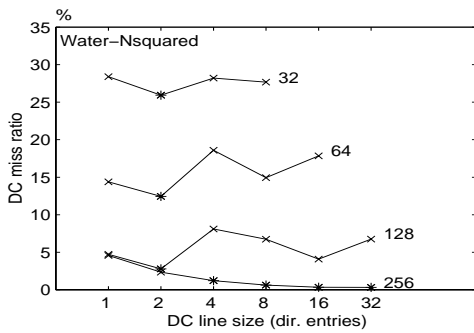
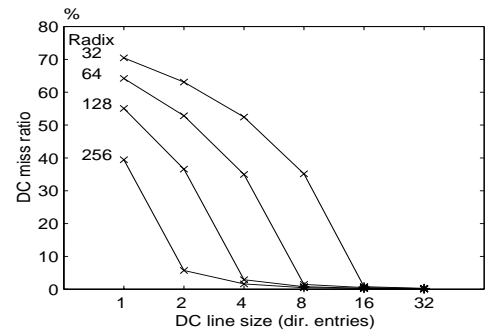
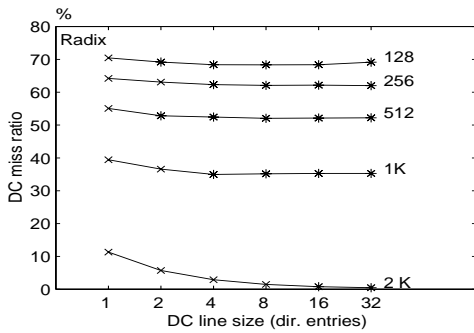
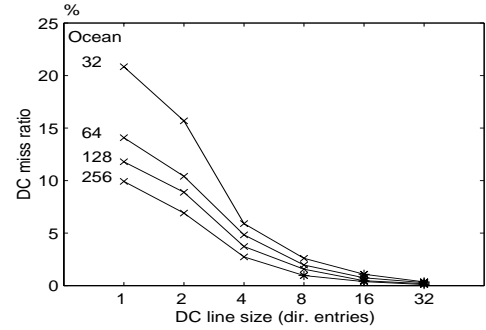
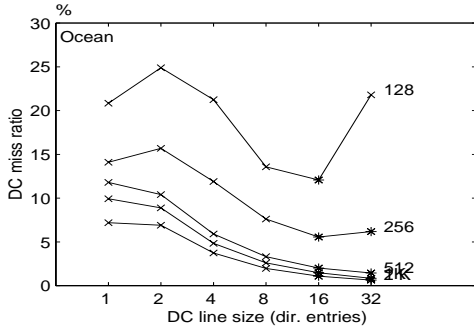
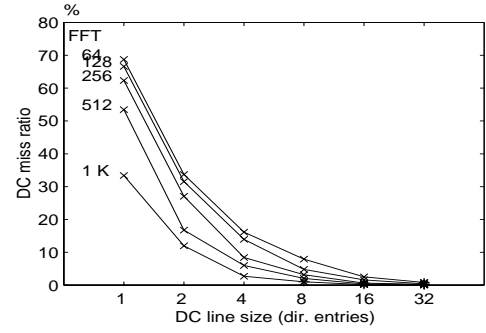
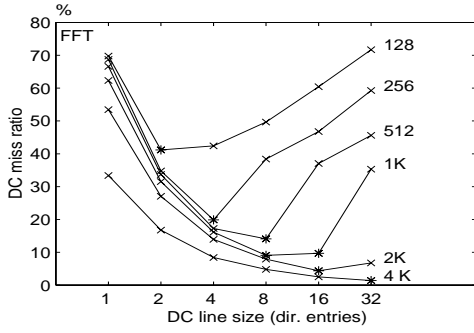


Figure 6: Effect of DC line size on DC miss ratio with different DC sizes (noted to the right of each curve).

Figure 7: Effect of DC line size on DC miss ratio with different DC set numbers (noted to the left of each curve).

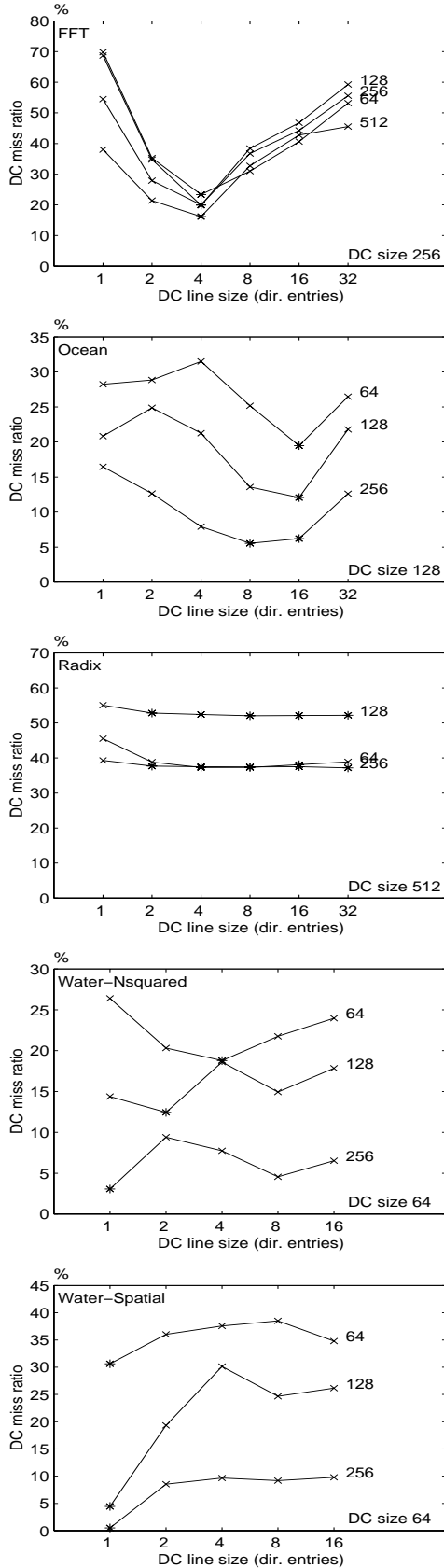


Figure 8: Effect of DC line size on DC miss ratio with different L2 cache line sizes (noted to the right of each curve).

In comparison to DC line size, DC associativity appears to be less critical for performance. Therefore, in cases where the width of DC sets has to be limited in order to simplify the CC chip design, DC set width is better used in widening DC lines rather than in increasing associativity from 2-way to 4-way, for example.

3.3 Relations between Performance Measures

In order to gain more insight into quantifying the impact of DC performance on coherence controller (CC) performance and the whole multiprocessor system, we investigate the relation between three performance measures: (1) DC miss ratio, which is the main performance measure for a DC, (2) CC occupancy which is the most important CC parameter in affecting overall system performance [2, 7], and (3) execution time, which is the primary performance measure for a computing system.

Figure 11 shows the DC miss ratio versus the increase in CC occupancy relative to the case of a CC with a perfect (100% hit) DC¹ for the points with 128 byte L2 lines.

The graphs show a clear linear relationship between the two measures implying that any improvement in DC miss ratio translates directly into an improvement in CC occupancy. The slopes in the graphs for the different applications are more or less equal, since we are using the same coherence protocol and the same CC parameters. The minor variations in the slopes is due to the variations in sharing patterns between the applications creating different mixes of protocol handler frequencies, which result in differences in directory access patterns.

Figure 12 shows the relation between the increase in CC occupancy relative to the case of a CC with a perfect DC and the increase in execution time relative to the case of a CC with a perfect DC, for the points with 128 byte L2 cache lines. The graphs show a clear linear relationship² between CC occupancy and execution time, for all the applications with high communication requirements. The slopes vary among applications, depending on their inherent communication characteristics as well as other system and application parameters such as the data set size.

The linear relationship not only emphasizes the impact of CC occupancy on multiprocessor systems performance as shown in previous research [2, 7], but also means that any improvement in CC occupancy counts directly into improving the overall system performance. Besides supporting the use of DCs, this result also supports any reduction in CC occupancy, for example by pipelining the CC, favoring custom-hardware over slower programmable protocol engines, or using multiple protocol engines that can execute multiple protocol handlers concurrently [7, 9].

The linear relationship between CC occupancy and execution time is attributed to the fact that the coherence controller is the bottleneck for applications with high communication requirements. This is supported by the slightly super-linear (or the increase in slope) for Water-Spatial, the application with lower communication requirements. Therefore, the relation between the two measures, most likely, starts super-linear (or with a low slope) until the CC becomes the bottleneck. At which point, the relation grows linearly with a higher slope.

¹The increase in CC occupancy relative to the case with perfect DC remains a pure representation of CC occupancy. We are just dividing it by a constant and subtracting a constant from it, for the sake of producing a quantity that can be related across applications and system configurations than absolute CC occupancy. The same argument applies to the increase in execution time relative to the case with perfect DC.

²We find a clear linear relationship between DC miss ratio and execution time. We do not include the graphs in this paper as they are derivatives of Figures 11 and 12.

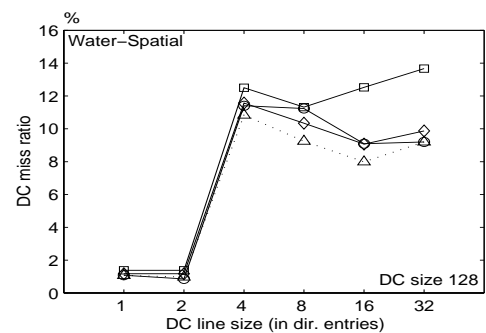
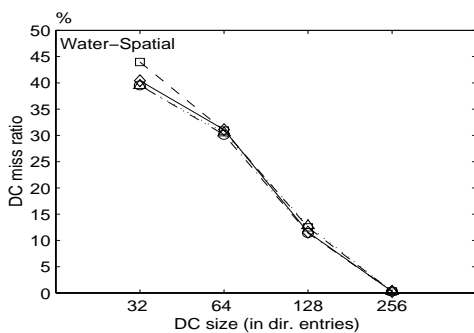
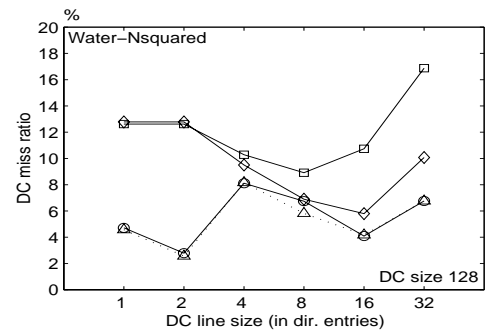
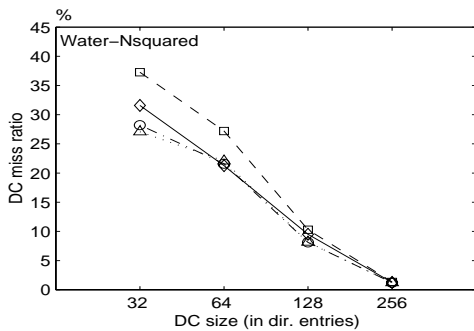
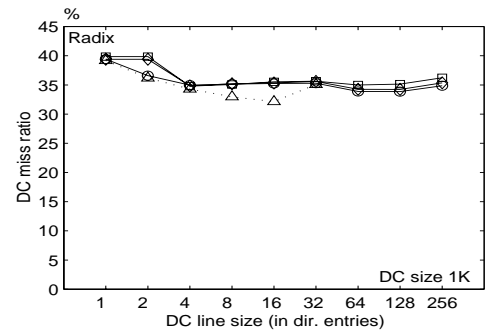
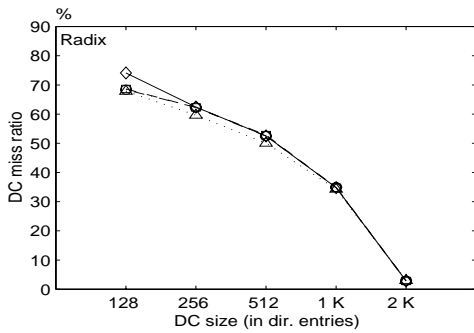
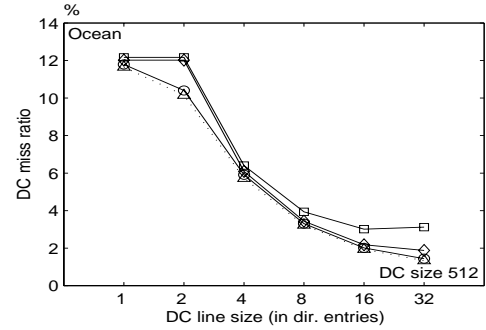
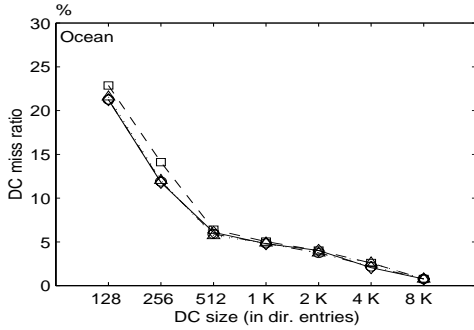
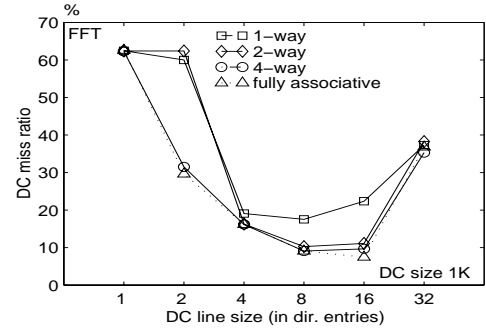
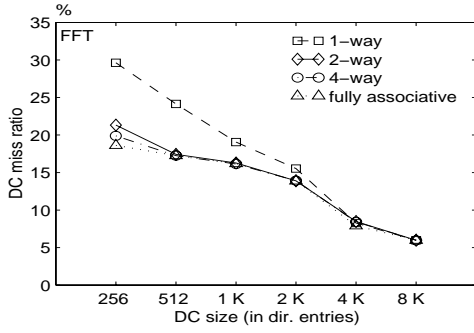


Figure 9: Effect of DC associativity on DC miss ratio with different DC sizes.

Figure 10: Effect of DC associativity on DC miss ratio with different DC line sizes.

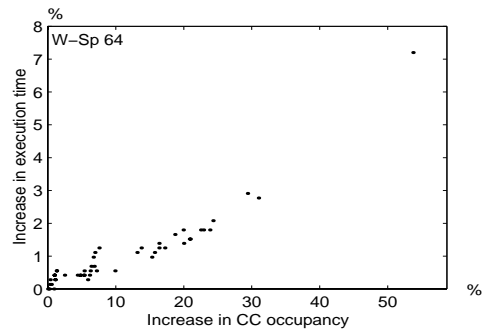
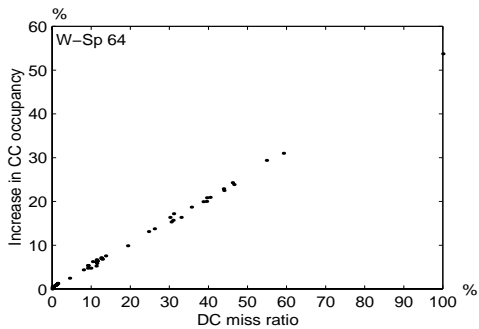
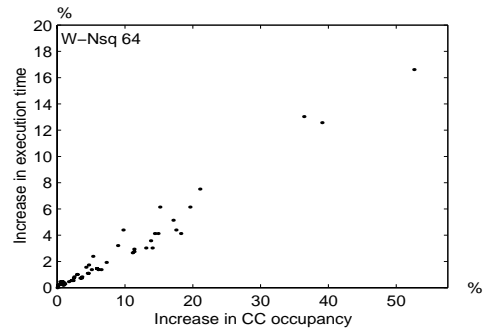
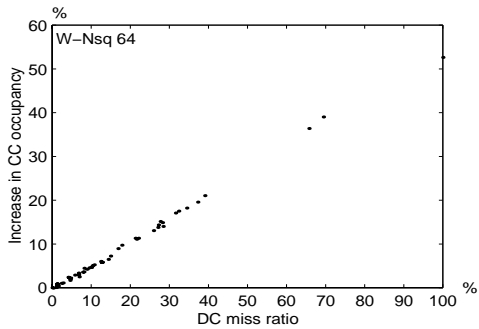
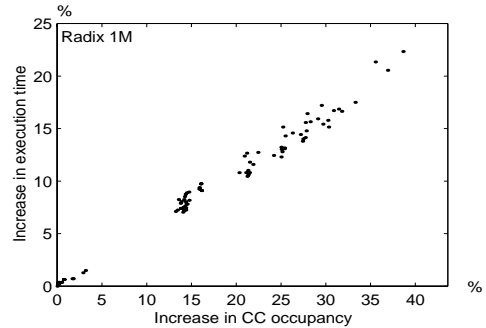
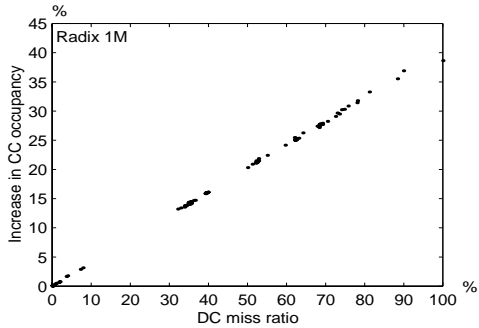
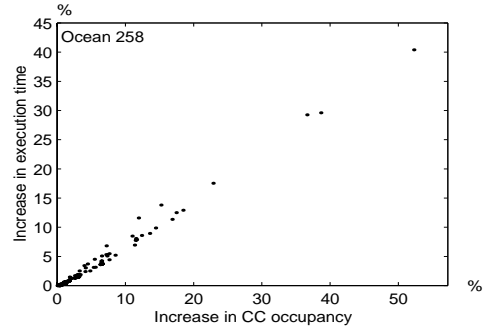
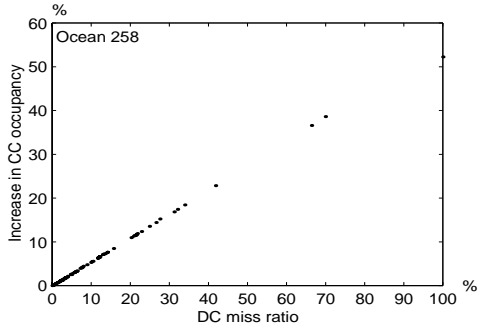
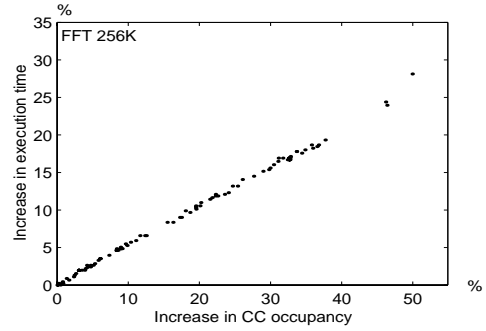
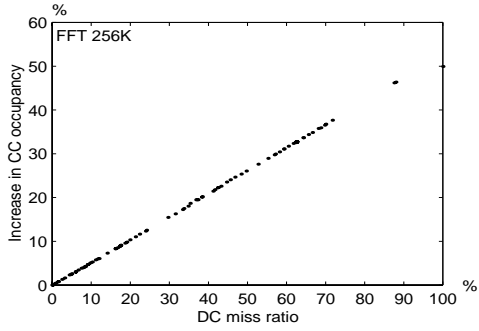


Figure 11: Relation between DC miss ratio and change in coherence controller occupancy relative to that with a perfect DC (0% miss ratio).

Figure 12: Relation between change in coherence controller occupancy vs. change in execution time relative to those with a perfect DC.

4 Conclusions

The main focus of this paper is on characterizing the performance impact of directory caches on distributed shared memory multiprocessors, and characterizing the DC parameters with the most impact on system performance and the system and application variables with the most influence on the choices for these parameters. We find that for applications with high communication requirements, such as FFT, Ocean, Radix, and Water-Nsquared, using a DC has significant impact on reducing the execution time. Moreover, using an off-chip SRAM directory if an on-chip DC is infeasible still yields most of the performance gains of a DC on-chip. Therefore, we recommend using on-chip DCs in future coherence controllers, and if necessary they can be implemented in off-chip SRAM.

Our results also show that DC size requirements grow sub-linearly with the increase in the data set of the application, implying the potential performance benefits of using a DC on real systems with large workloads. We also show that DC line size has significant effect on DC performance and that the optimal DC line size varies from one application to another and with different DC sizes and L2 line sizes. However, a DC line size of 8 or 4 directory entries appears to be the best choice in general.

Associativity has a relatively small impact on the performance of DCs. The largest differences in DC performance, with respect to associativity, are between 1-way and 2-way DCs, and that a 4-way DC almost shadows the performance of a fully-associative one. Also we find that if a choice arises between increasing DC line size and DC associativity due to limited DC set width, then the former parameter is of greater importance to performance than the latter.

We also investigate the relation between the miss ratio of the DC, the occupancy of the coherence controller, and execution time. We find a clear linear relationship between these three measures, indicating a clear and direct performance benefit for the whole system performance with any reduction in directory access time, by using a DC, and any reduction in coherence controller occupancy. This result can help system architects evaluate the expected performance gains (or losses) as a result of design variations affecting coherence controller occupancy.

In summary, the results of our research imply the importance of using DCs in future coherence controller designs, and help designers select DC parameters and evaluate the impact of these choices on overall system performance.

References

- [1] A. Agarwal, R. Bianchini, D. Chaiken, K. Johnson, D. Kranz, J. Kubiawicz, B.-H. Lim, K. Mackenzie, and D. Yeung. The MIT Alewife Machine: Architecture and Performance. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 2–13, June 1995.
- [2] C. Holt, M. Heinrich, J. P. Singh, E. Rothberg, and J. Hennessy. The Effects of Latency, Occupance, and Bandwidth in Distributed Shared Memory Multiprocessors. Technical report, Stanford University, January 1995.
- [3] J. Kuskin, D. Ofelt, M. Heinrich, J. Heinlein, R. Simoni, K. Gharachorloo, J. Chapin, D. Nakahira, J. Baxter, M. Horowitz, A. Gupta, M. Rosenblum, and J. Hennessy. The FLASH Multiprocessor. In *Proceedings of 21st International Symposium on Computer Architecture*, pages 302–313, April 1994.
- [4] J. Laudon and D. Lenoski. The SGI Origin: A ccNUMA Highly Scalable Server. In *Proceedings of the 24rd International Symposium on Computer Architecture*, pages 241–251, June 1997.
- [5] D. Lenoski, J. Laudon, K. Gharachorloo, W.-D. Weber, A. Gupta, J. Hennessy, M. Horowitz, and M. Lam. The Stanford DASH Multiprocessor. *IEEE Computer*, pages 63–79, March 1992.
- [6] T. Lovett and R. Clapp. STiNG: A CC-NUMA Computer System for the Commercial Marketplace. In *Proceedings of the 23rd International Symposium on Computer Architecture*, pages 308–317, May 1996.
- [7] M. M. Michael, A. K. Nanda, B.-H. Lim, and M. L. Scott. Coherence Controller Architectures for SMP-Based CC-NUMA Multiprocessors. In *Proceedings of the 24rd International Symposium on Computer Architecture*, pages 219–228, June 1997.
- [8] A.-T. Nguyen, M. M. Michael, A. D. Sharma, and J. Torrellas. The Augmint Multiprocessor Simulation Toolkit for Intel x86 Architectures. In *Proceedings of the 1996 IEEE International Conference on Computer Design*, pages 486–490, October 1996.
- [9] A. Nowatzyk, G. Aybay, M. Browne, E. Kelly, M. Parkin, B. Radke, and S. Vishin. The S3.mp Scalable Shared Memory Multiprocessor. In *Proceedings of 1995 International Conference on Parallel Processing*, August 1995.
- [10] S. K. Reinhardt, J. R. Larus, and D. A. Wood. Tempest and Typhoon: User-Level Shared Memory. In *Proceedings of the 21st International Symposium on Computer Architecture*, pages 325–336, April 1994.
- [11] W.-D. Weber, S. Gold, P. Helland, T. Shimizu, T. Wicki, and W. Wilcke. The Mercury Interconnect Architecture: A Cost-Effective Infrastructure for High-Performance Servers. In *Proceedings of the 24rd International Symposium on Computer Architecture*, pages 98–107, June 1997.
- [12] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, pages 24–36, June 1995.