Memory Organization in Multi-Channel Optical Networks: NUMA and COMA Revisited

YanYang Xiao and John K. Bennett

Computer Systems Laboratory
Department of Electrical and Computer Engineering
Rice University, P.O. Box 1892
Houston, Texas 77251-1892
email: {yyx, jkb}@rice.edu

Abstract

Multi-channel optical networks, although common in telecommunication applications, have only recently found application in computer systems. Multi-channel optical networks offer the potential for high performance interconnects for both local computer networks and multiprocessor systems. In addition to providing high bandwidth, multi-channel optical networks exhibit the capability for *scalable broadcast*. The absence of scalable broadcast in conventional networks has governed the choice of memory system configuration for many systems, and has in particular favored cache-coherent non-uniform memory access (CC-NUMA) over cache-only memory access (COMA) architecture. This paper examines the choice of memory system architecture in the presence of high bandwidth and scalable broadcast. Using simulation, we compare the performance of CC-NUMA and COMA memory architectures in a multi-channel optical network multiprocessor system. Seven well-known parallel benchmarks are used in the study. Our results indicate that COMA consistently and significantly outperforms CC-NUMA in systems that support scalable broadcast, and confirm previous results, that in the absence of broadcast, CC-NUMA outperforms COMA for those applications exhibiting significant sharing.

1 Introduction

Fiber optical networks feature high communication bandwidth and low bit error rate. However, the large communication bandwidth (about 25,000 GHz) can not be well utilized using a single optical channel because the electronic components coupled to the optical networks are not fast enough to keep up with the large optical bandwidth. The optical fibers are therefore often divided into multiple channels using either wavelength division or time division multiplexing. Each channel is operated at the speed that can be managed by electronic components (several GHz), while multiple channels are used to achieve large aggregated bandwidth.

Besides providing large bandwidth, multi-channel optical networks allow concurrent transmission and processing of data packets between different pairs of communicating parties [29]. If more channels are available, even simultaneous communication from the same node can be overlapped. The most relevant property of optical networks to this study is the scalable broadcast capability. Unlike electronic high-bandwidth, multi-stage interconnection networks where broadcasts are normally expensive, in multi-channel optical networks, signals (light pulses) from each source transmitter can be observed by receivers at all nodes, achieving scalable broadcast.

Fiber optical networks are becoming increasingly feasible for both telecommunication and local area computer networks as new opto-electronic technologies become more available and cost-effective [23, 7, 12]. One recent project demonstrated an all-optical network connecting up to 32 nodes that employs wavelength division multiplexing (WDM), and has an aggregated bandwidth of 32 gigabits/sec [15]. An optical time division multiplexed (OTDM) network consisting of 250, 1 gigabit/sec channels has also been demonstrated [21]. With further technological advances in optical fiber and opto-electronic devices, these networks will become increasingly feasible for many communication intensive applications, including data communication in tightly-coupled parallel computer systems [2, 8, 9, 21, 29].

When used as the interconnection networks for multiprocessor systems, multi-channel optical networks offer both high bandwidth and scalable broadcast capability, particularly in a form of WDM known as broadcast-and-select. The absence of scalable broadcast in conventional networks has governed the choice of memory system configuration for many systems, e.g., [18, 1, 17, 6, 5], and has in particular favored cache-coherent non-uniform memory access (CC-NUMA, or simply NUMA) over cache-only memory access (COMA) architecture [27]. In this paper, we investigate the impact of the availability of scalable broadcast on these two memory organization alternatives. Using simulation, we compare the performance of CC-NUMA and COMA memory architectures in a multi-channel optical network multiprocessor system. Seven well-known parallel benchmarks (Barnes, CG, FFT3D, LU, MP3D, Radix and Water) are

used in the study. Our results indicate that COMA consistently and significantly outperforms CC-NUMA in systems that support scalable broadcast, and confirm previous results, that in the absence of broadcast, CC-NUMA outperforms COMA for those applications exhibiting significant sharing.

The rest of the paper is organized as follows. In Section 2 and 3 we describe the implementation of NUMA and COMA memory systems using point-to-point and multi-channel optical networks. In Section 4, we describe how the performance of these systems is evaluated, and present experimental results. In Section 5 we discuss related work, and in Section 6 we summarize our results.

2 COMA and NUMA in Conventional Interconnection Networks

2.1 Network Model

Current interconnection networks range from bus-based to multi-level packet-switched interconnection networks. Buses provide convenient broadcast but are bandwidth-limited, and cannot support the communication needs of large parallel systems. Many research and commercial multiprocessor systems use some kind of high-bandwidth, packet-switched network, such as two- or three-dimensional meshes or fat-tree connections [18, 1, 17, 6, 5]. These networks allow high-speed point-to-point communication, but one-to-all broadcast normally takes multiple point-to-point messages.

2.2 NUMA in Conventional Interconnection Networks

In NUMA systems, each memory block is assigned a home-node according to a memory placement policy. A home-node provides a convenient reference point when a cache miss occurs and a remote memory access is needed. Each home-node maintains consistency in all caches for the portion of memory it governs, maintains information in a directory, and initiates appropriate coherence messages across the network.

NUMA is particularly appropriate for point-to-point interconnection networks. When a cache miss occurs, causing a remote memory access, the memory address is used to determine the home-node for the data. Only a single message is needed from the requester to the home-node. In the case of an invalid memory line in the home-node, the home-node forwards the request to the node with the most current copy. The directory maintained on the home-node provides information about the valid node. The response requires two messages, one to send the data from the valid node to the home-node, and the other from the home-node to the requester. This multi-hop coherence traffic increases memory access latency as well as network loads. The latency can be avoided in multi-stage networks if the paths from the responder to the requester and the home-node are disjoint. Broadcast, if available, reduces the request latency by eliminating forwarding and the response message from the valid node to the home-node.

Cache miss latency in NUMA systems can be very sensitive to the memory placement policy. Data replication and migration is limited to the size of processor caches, which normally are not large enough to cover all of the application's working sets [24]. Without compiler or runtime system support, static memory placement is unlikely to be effective. Moreover, these problems are exacerbated with increase in system size, since the proportion of accesses that will be remote memory accesses increases with system size.

2.3 COMA in Conventional Interconnection Networks

COMA systems do not require the home-node found in NUMA systems. Each memory line is free to be cached and replicated in all nodes' memory, sometimes referred to as the COMA-cache, since memory at each node acts more as a cache than as conventional memory.

The COMA-cache provides a large depository cache to the processor, in addition to the first- and second-level processor caches. It captures a larger application working set than the processor cache alone, and therefore can reduce remote memory accesses. Memory lines are automatically cached and coherency is maintained by cache-like memory controller.

The main difficulty with COMA in a point-to-point network is the lack of an efficient tracking mechanism for cache misses. Unlike NUMA, where a request can be directed to the home-node in one network transaction, COMA may require multiple network transactions to locate a valid copy. The directory in COMA is often organized into a tree structure to reduce this seek time. Stenström et al. showed that the tree directory has adverse impact on COMA's performance when a large number of tree traversals are needed [27]. We have observed that the large capacity of COMA-cache increases cache line residence time, which can result in COMA-cache coherence misses that do not exist in processor caches (see Section 4). In certain applications (MP3D and Radix), this also has a large impact on COMA's performance.

3 COMA and NUMA in Multi-channel Optical Networks

3.1 Network Model

The multi-channel optical network evaluated here is a single-hop, broadcast-and-select optical WDM network configured in a passive star topology [12, 8, 20]. Each node is equipped with a transmitter that is tunable over the entire range of wavelengths used in the system, and a receiver that de-multiplexes optical signals and selects each in turn. Though an N-channel network, where N is the number of nodes in the system, has the desired property of being contention free (no contention among accesses from different nodes), it is neither necessary nor cost-effective for the purpose of providing adequate bandwidth for data communication in most parallel applications [29]. This study focuses on a four-channel network in a 32-node system, where network contention is in the range of low to medium. Each transmitter is therefore required to tune rapidly to any of the 4 wavelengths, and the receiver de-multiplexes input wavelengths into separate channels. After opto-electronic conversion at the receiver ends, the electronic signals are buffered and processed.

In addition to fast tuning transmitters and receivers, an efficient medium access control protocol is essential in coordinating transmissions between various nodes to achieve high throughput in the network. This protocol needs to resolve possible contention from more than one transmitter sending at the same wavelength, and more than one transmitter sending to the same receiver. In this study, we assume token-passing protocols to resolve transmission contention within each channel. The receiver de-multiplexes all input optical signals concurrently. We assume an infinite receiver buffer to avoid packet loss and retransmission caused by receiving multiple packets simultaneously.

3.2 COMA and NUMA in Multi-Channel Optical Networks

Because of the home-node directory mechanism, NUMA systems only require point-to-point communication. A broadcast mechanism can eliminate the potential need to forward a request from the home-node, as well as the need to respond to the home-node first. We evaluate both point-to-point and broadcast NUMA architectures in Section 4, and show that broadcast offers little benefit to NUMA for the applications studied.

In contrast to NUMA, a broadcast mechanism is very beneficial to COMA systems. The broadcast capability provided in multi-channel optical networks eliminates the tree traversal during COMA-cache misses seen in previous COMA systems [27]. Instead, COMA-cache misses can be satisfied in two network transactions. By checking the information kept locally in each node, only one node will respond to the request, while others will ignore the request. A possible drawback of a broadcast implementation is the packet processing pressure seen by the network receivers. The difference between the processing latency in NUMA and COMA is the additional tag-lookup of the memory address in the COMA-cache, which is normally much less than protocol processing overhead. An additional owner bit field (described below) can also help to offload processing pressure. For this study, we assume tags and owner bit fields are duplicated, so that access contention to tags and owner bit fields is negligible. We also ignore tag-lookup overhead in the simulation.

Network contention is otherwise accurately modeled. In networks with realistic bandwidth, reducing the number of network messages is as important as reducing cache miss latencies. A COMA-cache not only reduces overall memory access latencies but also network traffic.

3.3 Cache Coherence Schemes

A write-invalidate protocol is used as the default protocol for maintaining cache coherence in both COMA and NUMA systems. In addition, an owner bit field is attached to each memory line in the COMA implementation. Read requests broadcast in the network are only responded to by the owner; nodes that find they are not the owner of the memory line simply discard the message without further processing. Invalidation packets are processed regardless of owner field value. The owner field of a cache line is initially set by the first processor that references it, and is set subsequently to the last processor requesting an exclusive copy. The owner field does not change during reads. The owner field is used to prevent more than one node from responding to the same read request. This information may also be duplicated to reduce access contention.

In this implementation, each COMA-cache only maintains local information, in contrast to point-to-point COMA implementations where information about the memory lines of all nodes in the subtree is maintained in the directory of a tree node. In point-to-point COMA, any node that needs any information about that particular line is forwarded to that directory, while in broadcast COMA, each node only keeps information about the local COMA-cache; all read and invalidation requests are broadcast system-wide.

4 Performance Evaluation

4.1 Performance Prediction

The performance improvement (or degradation) of COMA over NUMA is determined by several factors, including the applications' processor cache miss rate, the nature of these misses, the number of write-backs, NUMA memory placement policy, and the number of COMA-cache coherence misses. In this section we first qualitatively discuss the importance of these factors, and then demonstrate the effect of these factors using simulation.

Cache misses are commonly classified into three categories: cold misses, capacity misses (including conflict misses caused by limited set associativity), and coherence misses (also called invalidation misses) [10]. The miss types have different effects in NUMA and COMA systems.

Cold Misses: Cold miss counts and latencies remain the same in both systems under the same cache configuration, i.e., the same cache line size, cache size, and set associativity.

Coherence Misses: Coherence miss counts remain the same under NUMA and COMA architectures. Miss latencies in COMA depend upon the COMA implementation. In a tree directory implementation, as in the KSR-1 [16], coherence misses incur a large overhead of multiple network transactions due to tree traversals. Large coherence miss overhead can result in inferior performance for COMA implementations [27]. A flat COMA implementation is often used in machines implementing COMA systems in point-to-point networks [27]. At a coherence misses, at most three network transactions are needed, at the cost of a more complex cache coherence scheme and additional network traffic between home-nodes and master nodes of the data [13]. The alternative COMA implementation proposed in Section 3.2 utilizes the scalable broadcast capability of multi-channel optical networks so that coherence misses require only two network transactions. In this study, in addition to point-to-point NUMA, we also simulated broadcast NUMA. In broadcast NUMA implementation, all remote memory accesses can be satisfied in two network transactions. Thus coherence misses are the same for both broadcast COMA and NUMA implementations.

Capacity Misses: Capacity miss counts are again the same for both NUMA and COMA systems. Misses that can be satisfied in COMA-caches incur only bus access overhead,

while NUMA capacity misses may incur network access overhead if remote accesses are needed. Since the COMA-cache capacity is typically at least an order of magnitude larger than processor cache capacity, we follow [27] and assume infinite COMA-caches, i.e., all processor cache capacity misses (subtracting the portion of capacity misses that encounter COMA-cache coherence misses as explained below) can be satisfied by the COMA-cache. COMA improves total capacity miss latency, therefore, by an amount equivalent to the number of remote capacity misses that can be satisfied in the COMA-cache times the network access latency.

COMA-cache Coherence Misses: COMA-cache cold misses are the same as processor cache cold misses provided that the line sizes are the same in the two caches. For the assumed infinite COMA-cache, there is no capacity misses. Coherence misses, however, are higher than that in processor cache. This is because a remote invalidation to a line that has been written back to the COMA-cache will invalidate the line in the COMA-cache but not the processor cache. Subsequent access to the line will encounter capacity miss in the processor cache, but will encounter coherence miss in the COMA-cache. This type of processor cache capacity misses, therefore, can not be satisfied by COMA-caches.

Write-backs Another potential benefit of COMA is reduced write-back latency. The write-backs increase network traffic in NUMA systems if remote write-backs are required, while write-backs in COMA only require bus accesses to the local COMA-cache.

Cache misses, including write-backs, are sensitive to application working set size [24, 26]. Capacity misses are determined by the processor cache size and the working set size of the application. These relative sizes also determine the write-back rate seen by the application. In this study, we have scaled both application problem sizes and system cache sizes in order to appropriately model actual problem sizes on "real" machines. Increasing processor cache capacity reduces processor cache capacity misses, which diminishes the performance benefit of COMA. However, application problem size is usually limited by memory size and application data sets are unlikely to fit entirely in the processor caches. In these situations, COMA provides a substantial reduction in remote access rate.

4.2 Simulated System Architecture

We simulated a 32-processor system with 4KB, 64-byte line size, four-way set associative processor caches. Processors are assumed to be RISC, and run at 100 MHz. Buses are 64-bits wide and support split transactions. A bus cycle requires 20 ns.

All networks are multi-channeled, operating at 1 gigabit/sec for each channel. Our results focus on the performance for four-channel networks. Four-channel networks exhibit low-to-medium contention [29]. We also present results for one- and sixteen-channel networks, representing high and low network contention, respectively. Bus and network contentions are accurately simulated. Where multiple channels are used, channels are allocated dynamically, i.e., each node is allowed to use the first available channel [29]. Processors communicate using point-to-point messages or by broadcast, as appropriate.

Memory can either be sequentially or release consistent. We compared the performance of COMA and NUMA for both consistency models, but we present results for release consistent systems only. The results for sequential consistent systems were similar to that for release consistent. Write buffers contain sixteen entries.

Two memory placement policies were evaluated for NUMA. One was a page-cyclic placement policy, where 1024-byte pages are allocated to memory in a round-robin fashion. The second placement policy places each *line* of 64-bytes on the node that references the line first. This policy is impractical to implement, but performs better for statically partitioned applications. Initial memory loads are not simulated (i.e., all lines are assumed to be in at least one node's memory for both NUMA and COMA systems). The default cache coherence protocol is a standard ownership-based write-invalidate protocol [11] for both COMA and NUMA systems. Read latencies, in the absence of bus and network contention, are summarized in Table 1.

4.3 Application Characteristics

Seven parallel applications were used: an N-body gravitational force computation (Barnes-Hut), Conjugate Gradient (CG), 3D Fast Fourier Transformation (FFT3D), LU Decomposition (LU), a Monte-Carlo particle simulation (MP3D), an integer sort (Radix), and a molecular dynamic simulation (Water). CG and FFT3D are from the NAS benchmark suite [3], and have been rewritten in the C language. MP3D is from the SPLASH suite [25]. Barnes, LU, and Water are from SPLASH-2 [28]. All applications are compiled with `gcc -O2` version 2.5.8. Table 2 lists problem and important working set sizes for each application.

The smaller working set (WS1) was found for each application to be the size of the most frequently referenced data objects [24, 26]. For example, WS1 is the size of the tree data for one body in Barnes, the size of one row of the matrix in FFT3D, the size of one block in LU, the size of one particle plus one cell in MP3D, and the size of one molecule in Water. The listed WS1 size takes into account the 64-byte cache line size in the simulated architecture, and is rounded to the closest power-of-two. The larger working set (WS2) is the partition of shared data of each processor assuming a 32-processor system, except for CG and MP3D. WS2 is the largest working set that each processor requires. CG and MP3D are different because some shared variables are accessed by all processors (as explained below). Our 4KB processor caches are chosen to be representative of realistic sizes of application and cache, where processor cache sizes are large enough to hold WS1 (except for Radix, which does not have a well-defined WS1), but small enough to not hold WS2.

MP3D: The most important shared variables are particles and cells. The large working set WS2 includes the processor's partition of particles and cell space. The number of cells accessed is unknown until runtime. If particles are spread out uniformly, the entire cell space will be referenced. The particle partition is 34KB, and the cell space is 396KB, resulting in a WS2 of 420KB.

CG: CG is a sparse matrix application that uses conjugate gradient to find the smallest eigenvalue and correspond-

Types	*Time (proc. cycles)*
Cache Hits	1
Cache Misses Filled from Local COMA-cache or Local NUMA Memory	39
Cache Misses Filled from Remote COMA-cache or Remote NUMA Memory	167

Table 1: Read Latencies.

Application	*Problem Size*	*WS1 (KB)*	*WS2 (KB)*
Barnes	2048 bodies in 4 time steps	1KB	50KB
CG	1400 by 1400 matrix, 78148 non-zeros in 4 main iterations and 4 CG iterations	0.5KB	38KB
FFT3D	32 by 32 by 32 points in 3 iterations	0.5KB	58KB
LU	192 by 192 matrix, block size 6	0.5 KB	10KB
MP3D	40000 particles, 23 by 33 by 7 cells in 5 time steps	0.25KB	420KB
Radix	256K keys, radix 1024, maximum key 512 K	32KB	32KB
Water	343 molecules in 2 time steps	1KB	8KB

Table 2: Applications Problem and Working Set Sizes.

ing eigenstate of a matrix. The main computation involves a matrix-vector multiplication. The results of the multiplication are placed in a gradient vector. The partition of the gradient vector is WS1 in CG, which is 350 bytes for each processor (total of 32 processors). WS2 is the partition of the matrix plus the entire dense vector that is used in the multiplication. The dense vector is about 12KB, and represents 78% of the total misses with a 4KB cache.

4.4 Simulation Results

Figure 1 depict the performance of each of the seven applications for six different system configurations: page-cyclic point-to-point NUMA (PC-PTP-NUMA), page-cyclic broadcast NUMA (PC-BR-NUMA), first-reference point-to-point NUMA (FR-PTP-NUMA), first-reference broadcast NUMA (FR-BR-NUMA), point-to-point COMA (PTP-COMA) and broadcast COMA (BR-COMA). The results shown in Figure 1 are for the four-channel configuration.

Each bar in the figure represents the execution time of the labeled configuration normalized to the execution time of PC-PTP-NUMA (the first bar) in each application. Execution time is broken down into processor busy time (Busy), processor read latency (Read), processor write latency (Write) and synchronization time (Synch). For each application, the processor busy time remains unchanged across the six configurations. The write latency is insignificant except in two applications (FFT3D and Radix), because with the release consistent memory model, the write latency is largely hidden from the processor. There is also little change in synchronization latency across the six configurations. The read latency, varies significantly across the six configurations, however. Read latency represents the largest proportion of execution time for all applications except Water, which has a high computation-to-communication ratio.

Similar experiments were performed for one- and sixteen-channel network configurations. For these two network configurations, Table 3 presents the normalized execution time for the two NUMA performance extremes (PC-PTP-NUMA and FR-BR-NUMA), as well as PTP-COMA and BR-COMA.

Table 3 is divided into two parts. The first four columns depict execution times for the one-channel network configuration, and the last four columns are execution times for the sixteen-channel network configuration. The execution time is again normalized to the execution time of PC-PTP-NUMA in each channel configuration. For example, for Barnes, the execution time of FR-BR-NUMA is 97 percent of the execution time of PC-PTP-NUMA in the one-channel configuration, and the execution time of PTP-COMA and BR-COMA are 39 percent and 14 percent, respectively, of the execution time of PC-PTP-NUMA.

4.5 Results Summary

Our results are as follows: (1). While point-to-point COMA outperforms page-cyclic point-to-point NUMA in Barnes, CG, FFT3D and Radix, it performs poorly in LU, MP3D and Water. This is consistent with the results in [27]. (2). NUMA does not benefit significantly from scalable broadcast in either memory placement policy. The improvement is only significant for the one application (MP3D) that exhibits appreciable forwarding at read misses. For the rest of the applications, the improvements are less than 5%. (3). COMA, on the other hand, benefits significantly from broadcast in all cases. The poorer performance of point-to-point COMA in LU, MP3D and Water caused by directory traversals is eliminated using broadcast. Broadcast COMA consistently outperforms all NUMA cases and point-to-point COMA cases for all applications. (4). A comparison between the best NUMA performance and the best COMA performance shows that attempts to optimize memory placement for NUMA succeeded only for LU, FFT3D, MP3D, and Radix, but not for Barnes, CG and Water. (5). The performance benefit of COMA is more significant in higher contention networks, and less significant when network contention is low.

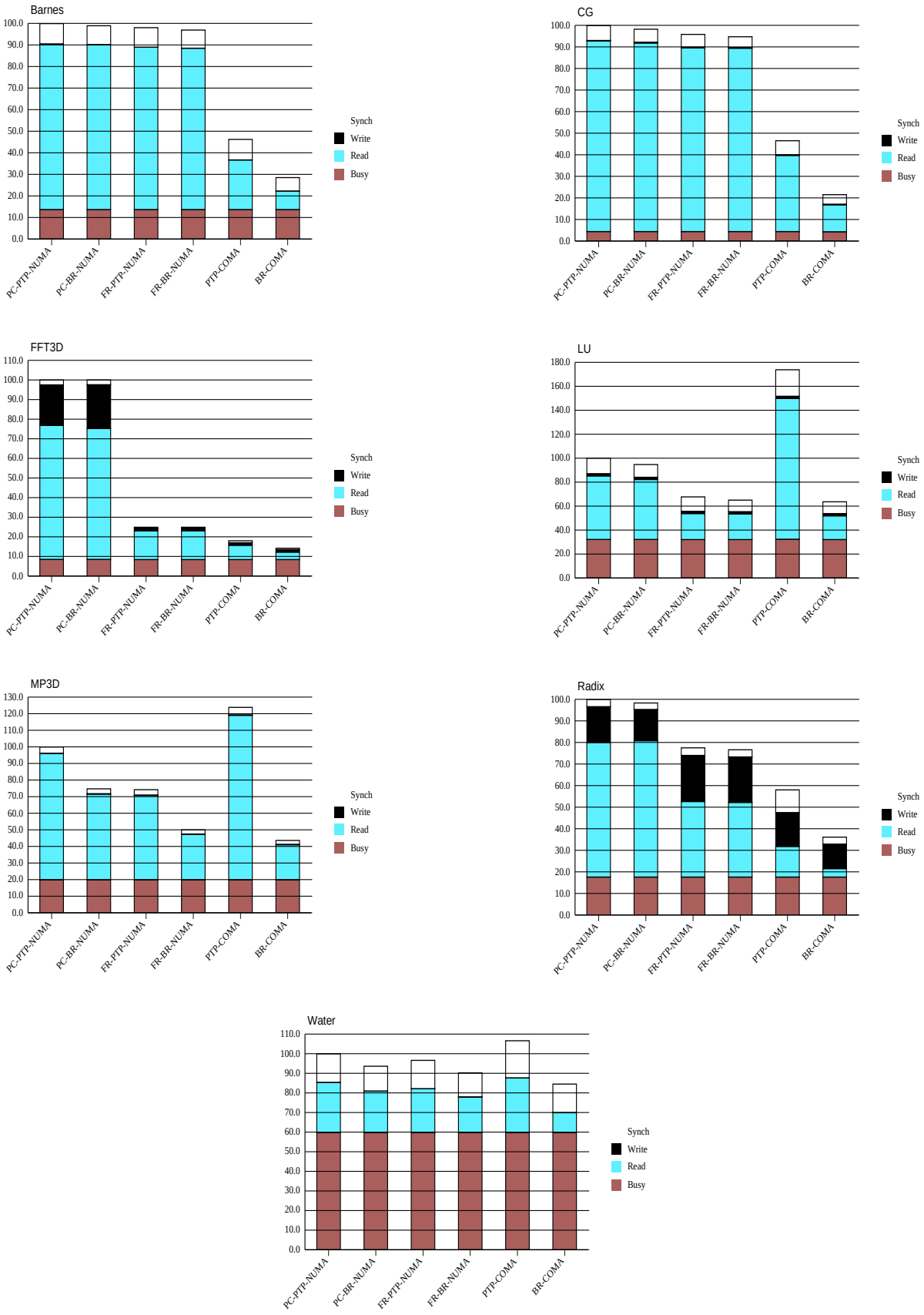


Figure 1:COMA and NUMA Execution Time.

Application	*Normalized Execution Time*							
	1-channel				*16-channel*			
	NUMA		*COMA*		*NUMA*		*COMA*	
	PC-PTP	FR-BR	PTP	BR	PC-PTP	FR-BR	PTP	BR
Barnes	100	97	39	14	100	98	83	69
CG	100	95	39	14	100	93	76	52
FFT3D	100	16	9	5	100	48	46	42
LU	100	58	179	56	100	89	121	89
MP3D	100	49	124	42	100	83	111	81
Radix	100	73	53	28	100	86	76	63
Water	100	79	69	41	100	98	102	94

Table 3: Normalized Execution Time.

4.6 Quantitative Comparison

In the following discussion, we concentrate on broadcast COMA and the best NUMA (the first-reference broadcast NUMA), unless otherwise mentioned. Also, to simplify the comparison, we ignore write latency since it is only significant in two applications (FFT3D and Radix), and is reduced in a similar way as read latency.

The expected performance difference between NUMA and COMA can be quantified as follows. The difference in write latency is ignored in the formula.

$$\begin{aligned}\text{Difference} &= \text{NUMA - COMA} \\ &= (\text{ Coherence Misses }) \\ &\quad \times (\text{ NUMA Latency - COMA Latency }) \\ &\quad + (\text{ Remote Capacity Misses} \\ &\qquad - \text{ COMA Coherence Misses }) \\ &\quad \times (\text{ Network Read Latency }) \end{aligned} \tag{1}$$

We will analyze this relationship using CG and MP3D as examples. In CG, 88% of processor cache misses are capacity misses. These capacity misses mainly result from accesses to the dense vector, vector p, since the vector (12KB) does not fit in the processor caches. There are few coherence misses and write-backs in CG, and few COMA-cache coherence misses as well. Because each processor needs to access the entire vector of p, and only 1/32nd of the vector is in each node's local memory, 31/32nd of these capacity misses result in remote memory accesses. The improvement, therefore, is the large number of capacity misses times network read latency. With no contention in the network, the network read latency is 167 processor cycles. The average waiting time in the network queue is, however, 330 cycles. So the total improvement of COMA over NUMA is the remote capacity misses times 497 cycles, which amounts to about 75% of the NUMA execution time. This network latency is eliminated in COMA, hence, COMA achieves a 4-fold improvement over NUMA.

In contrast to CG, coherence misses dominate in MP3D. Sixty-four percent of the total number of cache misses are coherence misses and 20% are capacity misses. The large number of coherence misses causes the point-to-point COMA to perform more poorly than NUMA. In broadcast COMA, coherence misses have the same latency as with broadcast NUMA, so coherence misses do not cause as much problem as in point-to-point COMA. The remote capacity misses are low for the particle objects, because particle objects are partitioned statically among processors. Most references to particle objects by each processor are to the ones that were initially touched by that processor, which have been placed in that processor's local memory. The reduction in remote accesses to particle objects is hence very small. Most of remote capacity misses in MP3D are caused by accesses to the cell objects. Remote capacity miss latencies to cell objects are not reduced in COMA, however, because a large number of these capacity misses are reflected in the COMA-cache as coherence misses. These capacity misses can not be satisfied by the COMA-cache. Only 4% improvement is achieved due to the reduced capacity miss latency. The remaining improvement in COMA is achieved by reducing the overall network load caused by writing back to local COMA-caches.

In essence, COMA achieves a more "active" sharing among processors. It allows a larger working set of data to be cached locally in each processor's COMA-cache than just in the processor cache. Active sharing is not always necessary if the access requires an exclusive copy. In this situation, COMA may have more coherence misses in the COMA-caches than that in the processor caches. The performance improvement afforded by COMA depends on the relative weight of these two factors, as well as the amount of network loading caused by factors such as the frequency of remote write-backs.

5 Related Work

The Kendall Square Research KSR-1 employed a COMA memory system [16]. The KSR-1 used a hierarchical token-ring bus network to connect up to 1088 nodes. A hierarchical directory was used to keep track of the states of the COMA-caches. The directory was a natural result of the hierarchical connection. A similar architecture was implemented in the Data Diffusion Machine [14].

Stenström et al. compared COMA and NUMA memory systems for a hierarchical directory with a mesh interconnection network. Bus and network contention were not considered. A "flat-COMA" memory was proposed, which resembled a combination of a COMA memory with a NUMA home-directory. Our implementation of COMA differs from flat-COMA in that no central directory is required, since broadcast is available.

Network caches have been previously studied [19, 4]. Network caches can be viewed as a combination of both NUMA and COMA. The large network cache allows a larger working set to be captured locally, while memory is organized like NUMA, and a home-node directory is consulted at local cache misses. Compared with a network cache architecture, COMA eliminates the need to manage a large shared cache *and* memory at the same time, simplifying the design. COMA also simplifies other design decisions, such as how to partition the total available memory hardware between network cache and memory, and how to achieve locality through memory placement policy and/or migration support.

A wide range of medium access control protocols exists that explore design trade-offs between hardware requirement and protocol complexity in multi-channel optical networks. For example, multi-hop protocols may only need one fixed transmitter and receiver at each node, while single-hop protocols, in general require more complex hardware. A survey and review of medium access control protocols can be found in [20, 22]. While token passing is not suitable for long-haul communication, because of long propagation delays, delays in a tightly-coupled parallel systems are small compared with transmission time. For example, with a system span of 10 m, the propagation delay is 5 ns. With a data rate of 1 gigabit/s, the packet transmission delay for a packet length of 64 bytes is 512 ns, two orders of magnitude greater than the propagation delay. Hence, the concern associated with long propagation delays that is an issue in long-haul communication is less relevant to a computer system.

6 Conclusion

The interaction of memory and network subsystems in a parallel system is complex, and requires careful examination in order to achieve desired performance. In a network that supports efficient broadcast, COMA memory systems offer significant reduced remote memory accesses, without incurring appreciable coherence overhead. Current systems, designed with a multi-dimensional mesh-connection or other multi-stage interconnect often lack a viable broadcast or multi-cast capability. These systems usually rely on point-to-point communication to maintain coherence using a directory-based protocol. Supporting COMA in these systems is costly due to the lack of knowledge of the communication destination during COMA-cache misses.

Network bandwidth is another important factor in comparing the overall system performance of COMA and NUMA. The cost of a COMA implementation can only be justified when superior performance can be achieved. In a high-contention network, the significant reduction of remote memory accesses in COMA systems greatly reduce network contention. Because the network is the most important performance bottleneck in these systems, this reduction of network traffic results in superior performance for COMA systems. However, as network bandwidth is increased, the importance of memory access latency decreases, and the overall performance of NUMA systems approaches that of COMA systems.

In this paper we have demonstrated the impact of both scalable broadcast and the number of channels in the performance of memory organization alternatives. We observed that COMA consistently outperforms NUMA if broadcast is available. These performance gains vary depending on the number of channels available in the network. The higher the network contention, the more benefit is seen from reduction of remote memory accesses.

References

[1] A. Agarwal, R. Bianchini, D. Chaiken, K. Johnson, D. Kranz, J. Kubiatowicz, B. H. Lim, K. MacKenzie, and D. Yeung. The MIT Alewife machine: Architecture and performance. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 2–13, 1995.

[2] E. Arthurs, J. M. Cooper, M. S. Goodman, H. Kobrinski, M. Tur, and M. P. Vecchi. Multiwavelength optical crossconnect for parallel-processing computers. *Electronics Letters*, 24(2):119–120, 1988.

[3] D. Bailey, J. Barton, T. Lasinski, and H. Simon. The NAS parallel benchmarks. Technical Report RNR-91-002, NASA Ames, 1991.

[4] J. K. Bennett, K. E. Fletcher, and W. E. Speight. The performance value of shared network caches in clustered multiprocessor workstations. Technical Report TR 9511, Rice University, 1995.

[5] Intel Corporation. Paragon XP/S product overview, 1991.

[6] Thinking Machine Corporation. The Connection Machine CM-5 technical summary, 1991.

[7] N. R. Dono, P. E. Green, K. Liu, R. Ramaswami, and F. F Tong. A wavelength division multiple access network for computer communication. *IEEE Journal on Selected Areas in Communications*, 8(6):983–994, 1990.

[8] P. W. Dowd. High performance interprocessor communication through optical wavelength division multiple access channels. In *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pages 96–105, 1991.

[9] P. W. Dowd and I. S. Hwang. Memory and network architecture interaction in an optically interconnected distributed shared memory system. *Journal of Parallel and Distributed Computing*, 25:144–161, 1995.

[10] M. Dubois, J. Skeppstedt, L. Ricciulli, K. Ramamurthy, and P. Stenström. The detection and elimination of useless misses in multiprocessors. In *Proceedings of the 20th Annual International Symposium on Computer Architecture*, pages 88–97, 1992.

[11] S. Eggers and R. H. Katz. Evaluating the performance of four snooping cache coherence protocols. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 2–15, 1989.

[12] M. S. Goodman, H. Kobrinski, M. P. Vecchi, R. M. Bulley, and J. L. Gimlett. The LAMBDANET multiwavelength network: Architecture, applications, and demonstrations. *IEEE Journal on Selected Areas in Communications*, 8(6):995–1004, 1990.

[13] A. Gupta, T. Joe, and P. Stenström. Performance limitations of cache-coherent NUMA and hierarchical COMA multiprocessors and the flat-COMA solution. Technical Report CSL-TR-92-524, Stanford University, 1992.

[14] E. Hagersten, S. Haridi, and D. H. D. Warren. The cache-coherent protocol of the Data Diffusion Machine. In M. Dubois and S. Thakkar, editors, *Cache and Interconnect Architectures in Multiprocessors*. Kluwer Academic Publishers, 1990.

[15] E. Hall, J. Kravitz, R. Ramaswami, M. Halvorson, S. Tenbrink, and R. Thomsen. The Rainbow-II gigabit optical network. *To appear in IEEE Journal on Selected Areas in Communications*, June, 1996.

[16] H. Burkhardt III, S. Frank, B. Knobe, and J. Rothnie. Overview of the KSR1 computer system. Technical Report KSR-TR-9202001, Kendall Square Research, Boston, 1992.

[17] Cray Research Inc. Cray T3D system architecture overview manual, HR-04033, 1993.

[18] 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 Stanford FLASH multiprocessor. In *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pages 302–313, 1994.

[19] S. Mori, H. Saito, M. Goshima, M. Yanaghihara, T. Tanaka, D. Fraser, K. Joe, H. Nitta, and S. Tomita. A distributed shared memory multiprocessors: ASURA – Memory and cache architecture. In *Proceedings of the 1994 International Conference on Supercomputing*, pages 740–749, 1994.

[20] B. Mukherjee. WDM-based local lightwave networks Part I: Single-hop systems. *IEEE Network*, 6(5):12–27, 1992.

[21] A. G. Nowatzyk and P. R. Prucnal. Are crossbars really dead? The case for optical multiprocessor interconnect systems. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 106–115, 1995.

[22] R. Ramaswami. Multiwavelength lightwave networks for computer communication. *IEEE Communications Magazine*, 31(2):78–88, 1993.

[23] E. G. Rawson and R. M. Metcalfe. Fibernet: Multimode optical fibers for local computer networks. *IEEE Transactions on Communications*, COM-26(7):983–990, 1978.

[24] E. Rothberg, J. P. Singh, and A. Gupta. Working sets, cache sizes, and node granularity issues for large-scale multiprocessors. In *Proceedings of the 20rd Annual International Symposium on Computer Architecture*, pages 14–25, 1993.

[25] J. P. Singh, W. D. Weber, and A. Gupta. SPLASH: Stanford parallel applications for shared-memory. Technical Report CSL-TR-91-469, Stanford University, 1991.

[26] W. E. Speight, K. E. Fletcher, and J. K. Bennett. Working set requirements and performance of network caches in cluster-based multiprocessors. Technical Report TR 9412, Rice University, 1994.

[27] P. Stenström, T. Joe, and A. Gupta. Comparative performance evaluation of cache-coherent NUMA and COMA architectures. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pages 80–91, 1992.

[28] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. Methodological considerations and characterization of the SPLASH-2 parallel application suite. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 24–36, 1995.

[29] Y. Y. Xiao and J. K. Bennett. Performance of multichannel networks in clustered multiprocessors. Technical Report TR 9510, Rice University, 1995.