Effects of Architectural and Technological Advances on the HP/Convex Exemplar’s Memory and Communication Performance

Gheith A. Abandah  Edward S. Davidson
Advanced Computer Architecture Laboratory, University of Michigan, Ann Arbor
{gabandah,davidson}@eecs.umich.edu

Abstract

Advances in microarchitecture, packaging, and manufacturing processes enable designers to build new systems with higher performance and scalability. Using microbenchmark techniques, we contrast the memory and communication performance of two generations of the HP/Convex Exemplar scalable parallel processing system. The SPP1000 and SPP2000 have significant architectural and implementation differences, but maintain upward binary compatibility. The SPP2000 employs manufacturing and packaging advances to obtain shorter system interconnects with wider data paths and improved functionality, thereby reducing the latency and increasing the bandwidth of remote communication. Although the memory latency is not significantly improved, newer out-of-order execution processors coupled with nonblocking caches achieve much higher memory bandwidth. The SPP2000 employs a modern, superscalar processor that features out-of-order execution and nonblocking caches. Additionally, the SPP2000 has larger nodes, a richer interconnection topology, better packaging and manufacturing technology, and more optimized protocols to improve its memory latency and bandwidth.

This paper evaluates the effects of these advances on the memory and communication performance. We use a suite of microbenchmarks that is designed to characterize the performance of DSM systems [2]. Similar to other multiprocessor microbenchmarks [11, 19, 12], these microbenchmarks measure the latency and bandwidth of memory accessing. However, they also offer wide coverage of the DSM memory and communication performance, including characterizations of local and shared accesses as functions of access pattern and distance, and overheads due to cache coherence and concurrent accessing. We execute these microbenchmarks on the two Exemplar systems to identify their strengths and weaknesses. The performance data presented in this paper is also useful for compiler and application development, and for making next generation design decisions.

The following section describes the architecture of the two systems and outlines their relevant differences. Section 3 describes the microbenchmarks used in this paper. Section 4 compares the local memory performance of the two systems. Section 5 evaluates the various aspects of the communication performance when accessing shared memory. Section 6 evaluates the aggregate achievable bandwidth and the effects of contention. Section 7 evaluates an aspect of the synchronization overhead. Finally, Section 8 draws some conclusions and suggests future directions.

1. Introduction

Distributed Shared Memory is becoming the prevailing approach for building scalable parallel computers. DSM systems use high-bandwidth, low-latency interconnection networks to connect powerful processing nodes that contain processors and memory. The distributed memory is shared through a global address space, thus providing a natural and convenient programming model [17, 10].
2. SPP2000 vs. SPP1000 – System Overview

The SPP1000 and the SPP2000 connect multiple nodes using multiple-ring interconnects. An SPP1000 node has 4 pairs of processors and 4 memory boards (with 2 banks each) connected by a crossbar; an SPP2000 node has 8 processor pairs and 8 memory boards (with 4 banks each) connected by a crossbar. Thus, an SPP2000 node has twice the processors and four times the memory banks of the SPP1000. Each processor pair (in both systems) has an agent that connects it to a crossbar port; each memory board has a memory controller that does likewise (Figure 1). Each SPP2000 node can be configured with up to 16 Gbytes of SDRAM; each SPP1000 node with up to only 2 Gbytes of DRAM. Yet, due to improved packaging and device density, a 2-node SPP2000 tower is physically the same size as a 2-node SPP1000 tower.

The SPP1000 uses the Hewlett-Packard PA 7100 [3], a two-way superscalar processor, running at 100 MHz. The SPP2000 uses the PA 8000 [13], a four-way superscalar processor, running at 180 MHz. In addition to the PA 8000’s higher frequency and larger number of functional units, it supports out-of-order instruction execution, and memory latency overlapping and hiding. On HP workstations that use these processors,1 the PA 8000 achieves 4.7x floating-point performance and 3.6x integer performance on SPEC95, relative to the PA 7100 [21].

Each processor has two off-chip data and instruction caches that are virtually addressed. The data caches within one node are kept coherent using directory-based cache coherence protocols. The SPP1000 uses a three-state protocol, and the SPP2000 uses a four-state protocol [20]. Each memory line has an associated coherence tag to keep track of which processors have cached copies of this line.

In the SPP1000, each memory controller has a ring interface that connects it to one ring. Thus, a multi-node SPP1000 system has 4 rings. Each ring carries the remote memory accesses to the corresponding two memory banks of each other node in the system. The SPP1000 interconnects up to 16 nodes using these 4 rings. Similarly, the SPP2000 has one ring interface per memory board, and two or three node systems simply use 8 rings. However, rather than one port, each ring interface has two ring ports which allow a two-dimensional ring interconnection topology. For example, four SPP2000 nodes can be interconnected in a 2 by 2 configuration using a total of 32 rings (Figure 2). The largest SPP2000 configuration is 4 by 8, comprised of 32 nodes interconnected with 96 rings. The two-dimensional topology provides lower latency and higher bisection bandwidth.

The two systems use variants of the IEEE Scalable Coherent Interface [14] to achieve internode cache coherence. The SCI protocol uses a distributed doubly-linked list to keep track of which nodes share each memory line. Each node that has a copy of a memory line, maintains pointers to the next forward and next backward nodes in that line’s sharing list.

Most of the glue logic that coherently interconnects the distributed processors and memory banks is in custom-designed gate arrays [4]. The SPP1000 uses 250-K gate gallium arsenide technology, which provided the speed to pump a 16-bit flit onto the ring each 3.33 nanoseconds. The SPP2000 uses 0.35-μ CMOS technology, which has evolved to provide competitive speeds in addition to its lower power consumption and higher integration. The SPP2000’s ASICs are implemented using Fujitsu’s 1.1-M gate arrays clocked at 120 MHz. However, the higher integration of these chips provided wider data paths and richer functionality. Each port of the SPP2000 ring interface can pump a 32-bit flit per 8.33 nanoseconds.

1The HP 9000 Model 735/99 runs a PA 7100 at 99 MHz with 256-KB data and instruction caches. The HP 9000 Model C180-XP runs a PA 8000 at 180 MHz with 1-MB data and instruction caches.
3. Evaluation Methodology

Although some applications are latency limited, e.g., those that do frequent pointer chasing, and some others are bandwidth limited, e.g., those that frequently move large data blocks, many applications benefit from both low latency and high bandwidth.

In this evaluation, we have used simple microbenchmark programs that are designed to characterize the performance of a DSM system’s memory, communication, and synchronization. The building blocks of the memory and communication microbenchmarks are memory kernels that are called many times to measure the minimum and average call times. Each kernel accesses the $l$-byte elements of an array of size $w$ bytes with stride $s$ bytes. Thus, one call time corresponds to the $w/s$ accesses in a kernel call.

The memory kernels need to be carefully designed to accurately measure the latency and bandwidth of modern processors [18, 2, 12]. A kernel that measures latency must expose the access latency and cannot overlap or hide the latencies of individual memory accesses; a kernel that measures bandwidth must maximize access overlap to achieve the highest possible bandwidth. In this paper, $l = 4$ for latency kernels, and $l = 8$ for bandwidth kernels.

Load latency is measured by the load-use kernel, which uses the loaded value to find the address of the next load. The data dependency in the load-use kernel serializes the load accesses. The store-load-use kernel measures the store latency. In each iteration, this kernel performs a store to one element followed by a load from an adjacent element. The loaded value is used to find the address of the next store and establish a data dependency. The runtime of the store-load-use kernel is some number of store latencies plus other known times. The load kernel and the store kernel measure bandwidth by simply generating a sequence of loads or stores, respectively, without any data dependencies.

These four kernels are the building blocks of the SPP2000 experiments. However, since the PA 7100 does not overlap cache misses, the load and store kernels are used to measure bandwidth as well as latency in the SPP1000 experiments. The benchmarked SPP1000 and SPP2000 systems have 4 nodes each. The SPP2000 nodes are interconnected in a 2 by 2 configuration, while the SPP1000 nodes lie along one dimension.

The following performance aspects of the two systems are evaluated in the following sections:

Local memory: the latency and bandwidth of accessing a private array that is allocated in the local memory. The array size is varied to characterize cases where the array fits in the processor data cache through cases where all accesses are satisfied from the local memory. The access stride is varied to expose processor and system bottlenecks.

Remote memory: the latency of accessing a shared array that is allocated in a remote node. The array size is varied to characterize the performance of the interconnect cache, which is described in Section 5.1.

Producer-consumer communication: the latencies when two processors access a shared array in a producer-consumer pattern. The locations of the two processors are varied to characterize intranode and internode communication.

Effect of distance: the latency as a function of the distance between two processors engaged in an internode producer-consumer communication.

Effect of home location: the latency as a function of the node location where the shared array is allocated (home node) in the internode producer-consumer communication.

Coherence overhead: the overhead of maintaining cache coherence as a function of the number of processors engaged in producer-consumer communications.

Concurrent traffic effects: the aggregate bandwidth when multiple processors are active in performing memory accesses.

Barrier synchronization time: the time to perform a barrier synchronization as a function of the number of processors when the processors reach the barrier simultaneously.

4. Local Memory

This section evaluates the local-memory performance when only one processor is accessing the local memory in its node. This performance is a function of the performance of the processor, processor cache, processor bus, processor agent, crossbar, memory controller, and memory banks.

4.1. Local Memory Latency

Figure 3 shows the average load latency of the two systems as a function of array size for the access strides $s = 8, 16, 32, \ldots$ bytes. For both systems, the load latency with $s = 64$ bytes equals the latency with $s = 32$ bytes, indicating that the cache line size is 32 bytes. The hit region ends at $w = 1024$ KB and the transition region ends at $w = 2048$ KB, indicating that the data cache is a direct-mapped 1024-KB cache [2]. In the hit region, every access fits in the cache, while in the miss region ($w \geq 2048$ KB), every access to a new cache line is a miss.
Figure 3. Average load latency as a function of array size $w$. The three regime regions are: hit region ($0 < w \leq 1024$ KB), transition region ($1024 < w < 2048$ KB), and miss region ($w \geq 2048$ KB).

Figure 4. Average store latency.

The SPP2000 load miss latency is 0.52 $\mu$sec, which is slightly lower than 0.55 $\mu$sec of the SPP1000. However, when considering clock cycles, the SPP2000 latency of 93 cycles is 69% more than the SPP1000 latency. The slight time advantage of the SPP2000 may be attributed to the slight speed advantage of the SDRAM over the DRAM and the lower processor bus and crossbar transfer times.

It is worthwhile to compare this latency to a comparable uniprocessor system. The HP 9000 Model C160 workstation runs a PA 8000 with a 160-MHz clock and has a load miss latency of 0.32 $\mu$sec. It seems that the cost of connecting 16 processors in a coherent SPP2000 node adds an additional 0.20 $\mu$sec latency, which penalizes serial applications.

Figure 4 shows the average store latency of the two systems. The store miss latency, measured as the latency from issuing the store to completing the allocation in the cache of the missed line, is slightly higher than the load miss latency due to the overhead of flushing the dirty replaced line.

In the SPP2000, the load-use and store-load-use kernels have 3-cycle access latency in the hit region. This latency is composed of 1 cycle for calculating the address plus 2 cycles cache latency, which is twice the SPP1000 cache latency.

### 4.2. Local Memory Bandwidth

When repeating the above experiments using the bandwidth-measuring kernels, the two systems exhibit very different performance. In the hit region, the SPP1000 can perform 1 load every cycle or 1 store every 2 cycles. For $s = 8$ bytes, the SPP2000 uses the two PA 8000 cache ports to perform 2 loads per cycle or 1 store per cycle. Thus, the SPP1000 maximum cache transfer rate is 800 MB/s for loads and 400 MB/s for stores, when accessing double-word elements with a 1-element stride. The SPP2000 maximum transfer rates are $(2 \times 186/100)$ times larger. However, with larger strides, the SPP2000 performs only 1 access per cycle because the cache has one bank for the even double-words and one bank for the odd double-words and each cache port accesses an associated bank. Thus, with $s = 16, 32, \ldots$ bytes, only one port is utilized.

Figure 5 shows the memory transfer rates as a function of the access stride for the two systems in the miss region. The transfer rate is found as the cache line size divided by the average access time per line. Note that for strides larger than

---

3The SPP1000 load latency changes slightly from one experiment to another: 0.55 $\mu$sec is the smallest latency measured in the miss region.
Table 1. Accessed SPP2000 memory banks for strides 8 through 1024 bytes. There are 8 memory boards (0 through 7) each has four banks (i, ii, iii, and iv).

<table>
<thead>
<tr>
<th>s</th>
<th>Accessed banks</th>
</tr>
</thead>
<tbody>
<tr>
<td>8–32</td>
<td>All banks</td>
</tr>
<tr>
<td>64</td>
<td>The four banks of boards 0, 2, 4, and 6.</td>
</tr>
<tr>
<td>128</td>
<td>The four banks of boards 0, and 4.</td>
</tr>
<tr>
<td>256</td>
<td>The four banks of board 0.</td>
</tr>
<tr>
<td>512</td>
<td>Banks i and iii of board 0.</td>
</tr>
<tr>
<td>1024</td>
<td>Bank i of board 0.</td>
</tr>
</tbody>
</table>

The four banks of boards 0 and 4. The four banks of board 0.

one element, some of the transferred bytes are not used. The
store transfer rate is smaller than the load transfer rate due
to the additional traffic generated to write back the replaced
dirty lines.

The best-case memory transfer rate in the SPP2000 is
about 10 times that of the SPP1000. The SPP1000’s trans-
fer rate of about 50 MB/s is limited by the latency of the
individual misses because the PA 7100 allows only one out-
standing miss. This small transfer rate does not stress the
memory system and is independent of the access stride.
Whereas, the PA 8000’s ability to have up to 10 outstanding
misses enables it to achieve much higher transfer rates,
and exposes a number of bottlenecks as apparent from the
variation in transfer rate as the access stride changes.

The first bottleneck at $s = 8$ bytes is due to the PA 8000’s
instruction reorder buffer size, which has up to 28 entries for
loads, stores, and some other instructions. With this stride,
every fourth access is to a new cache line, therefore, the
PA 8000 can have at most 28/4 dispatched instructions that
generate cache misses. For $s = 8$ bytes, the degree of miss
latency overlapping in the load kernel is about 6 (10 ×
290/500, assuming that there are 10 outstanding misses at
the peak bandwidth).

The other bottlenecks that occur with $s \geq 64$ bytes are
external to the PA 8000. In the SPP2000, the memory is
32-way interleaved; consecutive 32-byte memory lines are
first interleaved across the 8 memory boards then across the
4 banks of each board. Table 1 shows which memory banks
are accessed for strides 8 through 1024 bytes. The mem-
ory transfer rate declines as fewer memory boards or fewer
memory banks are utilized. From $s = 256$ bytes, we see
that a single memory board supports up to 295 MB/s to one
processor; from $s = 1024$ bytes, we see that a single mem-
ory bank supports up to 139 MB/s.

The store transfer rate for strides 16 and 32 bytes is lim-
ited by a bus bandwidth bottleneck. Recall that the store
kernel transfers two lines for each miss, a fetched line and a
dirty replaced line. The SPP2000 uses the Runway bus [8],
which is an 8-byte wide bus clocked at 120 MHz. This
bus multiplexes data and addresses with a 4:2 ratio. Thus,
the available bandwidth is 640 MB/s; the store kernel
achieves 92% of this limit.

5. Shared Memory Communication

The memory kernels are also used here to evaluate the
performance of accessing shared memory. The accessed ar-
ray is allocated in shared memory and each experiment uses
multiple processors. This section characterizes produc-
consumer performance, including the effects of communica-
tion distance, home node, and degree of sharing. Since the
interconnect cache affects the latency of accessing re-
move memory, we start with an evaluation of this cache.

5.1. Interconnect Cache

The interconnect cache is a dedicated section of each
node’s memory. The IC in each node exploits locality of
reference for the remote shared-memory data (shared data
with a home memory location in some other node). When
a remote shared-memory access misses in both the proces-
sor’s data cache and the node’s IC, a memory line is re-
trieved over the ring interconnect through its home node.
This line is then stored in the local IC as well as in the pro-
cessor’s data cache. Hence, subsequent references to this
line miss in one of the data caches of this node can be
satisfied locally from the IC, until such time as this line is
replaced in the IC or invalidated due to store by a remote
node. The IC size is selectable by the system administrator,
and is usually selected to achieve the best performance for
frequently executed applications.

The microbenchmark that evaluates IC performance uses
two processors from distinct nodes. The first processor al-
llocates a local shared array and initializes it. The second
processor accesses the array repetitively using the load-
use kernel (the load kernel in the SPP1000 case). Fig-
ure 6 shows the average load time of the second processor
as a function of the array size.

The SPP1000 node’s IC is set to 128 MB and the
SPP2000 node’s is 512 MB. For arrays that fit in the IC, the
load time is similar to the local-memory load time shown in
Figure 3.
In the SPP1000, the transition region is 128 MB wide, which indicates that the IC is direct mapped. For array sizes larger than 256 MB, no part of the array remains in the IC between accesses to it in two successive kernel calls, therefore, the accesses are satisfied from the remote memory. Although the maximum local-memory latency in the SPP1000 is reached with \( s = 32 \) bytes (1 processor cache line), the remote load latency almost doubles as \( s \) increases from 32 to 64 bytes, and continues to increase to reach a maximum at \( s = 256 \) bytes. In contrast, the SPP2000 remote load latency does not change for \( s \geq 32 \) bytes.

In the SPP1000, the memory line size is 64 bytes, twice the size of the processor cache line. The 64-byte memory line is the base unit for maintaining internode coherence in the SPP1000. Therefore, for \( s \geq 64 \) bytes in the miss region, every access is an IC miss, while for \( s = 32 \) bytes, every two processor cache misses generate one IC miss. The miss latency continues to grow as \( s \) increases from 64 to 256 bytes due to ring congestion, which increases as fewer rings are used to serve the missed lines that are interleaved across the memory banks; all 4 rings are used with \( s \leq 64 \) bytes, 2 rings with \( s = 128 \) bytes, and only one ring with \( s = 256 \) bytes. A ring apparently remains busy after fetching a line from the remote node (in order to collapse the sharing list of the replaced IC line).

Unlike the general trend toward wider caches, the IC line is shorter in the SPP2000 (32 bytes, as in the processor cache). Since the coherence tags attached to each memory line require 8 bytes in both systems, the shorter line results in higher overhead for coherence tags relative to user-available memory (1:4 in the SPP2000, and 1:8 in the SPP1000), and they provide less prefetching for applications with spatial locality.

However, shorter lines enabled the SPP2000 to achieve lower remote latency because it is faster to extract a 32-byte line from the memory and transfer it over the ring interconnect, the coherence protocol is simpler, and the data transfer is more streamlined. Furthermore, the PA 8000 allows multiple outstanding misses, which reduces the prefetching disadvantage of short lines.

As \( s \) increases and fewer rings are used to satisfy remote misses, the SPP2000’s latency, unlike the SPP1000, remains constant, implying no ring congestion even when only one ring is utilized with \( s \geq 256 \) bytes. This improved performance is achieved by supporting up to 32 outstanding requests in each ring interface, rather than only one [4]. Therefore, the SPP2000 ring controller can process a line request concurrently with collapsing the sharing list of a replaced line.

Notice that the SPP2000’s curve shape differs slightly from the curve of the 512-MB direct-mapped cache model shown [2]. In fact, it changes from one experiment to another according to the OS mapping of the array’s virtual pages to the physical memory pages. The curve shown indicates that some of the array’s physical pages conflict in the IC for \( w = 512 \) MB, resulting in an average latency that is higher than the IC hit latency. The SPP2000 data was obtained by running this microbenchmark on a recently booted system. The number of IC conflicts generally increases with time as virtual to physical mapping becomes almost random. The IC addressing scheme in the SPP1000 is different, and never revealed any IC conflicts in our experiments.

5.2. Communication Latency

Figures 7 and 8 show the latencies of the two phases of shared-memory producer-consumer communication as a function of the access stride. Two processors repetitively take turns accessing a shared array, Processor 0 executes the store-load-use kernel (produces), then after synchronization, Processor 1 executes the load-use kernel (consumes), then after synchronization, this process is iterated. The time that Processor 0 spends in the store-load-use kernel is used to find the average write-after-read (WAR) latency, and Processor 1’s time is used to find the read-after-write (RAW) latency. Latencies are shown for the two cases of processor allocation: near, when both processors are in the same node, and far when they are from distinct nodes.

The array size is 1 MB, which fits in the processor data cache. In the near case, this array is allocated in the local shared memory; whereas, it is interleaved across the shared memory of all nodes in the far case.

In the SPP1000, the far latency increases as the stride increases due to the increase in misses per reference (up to \( s = 64 \) bytes), and then the decrease in the number of rings used. In contrast, as \( s \) is increased beyond 32 bytes, the SPP2000 far latency generally does not increase, indicating no ring congestion. WAR generates more ring traffic than RAW because in addition to joining at the head of the line’s sharing list, the WAR sends invalidation signals to the other
Figure 7. Write-after-read access latency as a function of the access stride. Far WAR for the case when the two processors are from distinct nodes, and near WAR when the two processors are from the same node.

Figure 8. Read-after-write access latency.

sharing nodes. Thus, SPP1000 far WAR latency increases faster than RAW latency as fewer rings are used.

Far latency is higher than the near latency due to the overheads of the internode coherence protocol and the added latency of the ring interconnect. As with IC miss latency, the SPP1000 far latency at $s \geq 64$ bytes is nearly double the SPP2000 far latency.

However, the SPP2000 near WAR latency is higher than that of the SPP1000. In the SPP2000 intranode coherence protocol, the consumer gets an exclusive copy of the loaded line from the producer’s cache, and the producer loses its copy. Therefore, when the producer subsequently updates this line, it must wait to check whether the consumer’s copy has been modified. This check lengthens the near WAR latency, and seems to have some residual effect that slightly increases the latency when one bank is utilized at $s = 1024$ bytes. In the SPP1000, the consumer always gets a shared copy, which is invalidated concurrently when the memory controller returns a writable copy to the producer, without waiting to check the consumer’s copy status.

In the SPP2000, although giving an exclusive copy of a modified line hurts repetitive near producer-consumer communication, it is profitable for migratory lines (lines that are accessed largely by one processor at a time [22]). When a processor gets exclusive ownership of a migratory line, it does not need to request ownership when it subsequently updates this line.

When the consumer performs a RAW access, the valid copy of the accessed line is in the producer’s cache. Although, Section 4.1 showed that the two systems have similar local-memory latency, the SPP1000 has about 33% higher latency than the SPP2000 in near RAW when accessing the copy in the producer’s cache.

5.3. Effect of Distance

In the SPP1000, each remote access has a request path and a reply path, which collectively circle one ring. The remote latency is not affected by where the remote node is positioned on the ring—it depends only on the number of system nodes. However, the SPP2000 remote latency does depend on the remote node location.

Consider a far RAW access to a line that is valid in the memory of the producer’s node (the producer’s node is the home of this line). This simple access is satisfied by one request and one response. On the SPP2000, this takes 1.58 μsec between two adjacent nodes, and 1.87 μsec when the two nodes are opposite corners of the 2 by 2 configuration. The additional 0.29 μsec is needed to change
dimension and circle a second ring.

For a 4 by 2 SPP2000 system, it still takes 1.58 μsec between two nodes directly connected by 2-node rings, but 1.73 μsec between two nodes directly connected by 4-node rings, and 2.04 μsec otherwise. A simple model that is consistent with these observations is that this RAW access takes $1.43 \mu\text{sec} + 0.075 \mu\text{sec}$ per node to node link traversal of a ring + 0.15 μsec if two rings are traversed.

For the largest SPP2000 configuration, 4 by 8, the latency of this access is calculated as $1.73 \mu\text{sec}$ for two nodes directly connected by 4-node rings, $2.03 \mu\text{sec}$ for two nodes connected by 8-node rings, and $1.43 + 0.075 \times 12 + 0.15 = 2.48 \mu\text{sec}$ for nodes not on the same ring. Had the SPP2000 used a one-dimensional topology, the latency of this access on a 32-node system with these parameters would always be $1.43 + 0.075 \times 32 = 3.83 \mu\text{sec}$.

5.4. Effect of Home Location

The shared array used in characterizing the far producer-consumer communication in Section 5.2 is interleaved among the memory banks of the four nodes. Thus, the measured far WAR (or far RAW) latency is the average of four cases according to the location of the home node. The way that the internode coherence protocol satisfies a cache miss does in fact depend on the home node of the missed line [5, 14]; the three main cases are:

**Local home**: the line’s home node is the local node.

**Remote home**: the line’s home node is the remote node.

**Third-party home**: the line’s home node is a third node other than the local and remote nodes (in a 4-node system, there are two third-party nodes).

Table 2 shows the WAR and RAW latencies of the three cases of home node locations in the SPP2000 with $s = 32$ bytes. When measuring the latency of the first two cases, the two nodes involved are adjacent. For the third-party home case, the remote node is adjacent to the local node in the $x$ direction and the home node is adjacent to the local node in the $y$ direction. The table shows the latencies of cached and uncached WAR and RAW accesses. For the cached accesses, the shared array size is selected to fit in the processor cache; thus, a WAR or RAW access occurs when the remote processor has a copy of the line (the valid copy in RAW). For the uncached accesses, the array size is twice the processor cache size; thus, a WAR or RAW access occurs when the remote memory or the remote IC has a copy and the remote processor does not.

WAR latency is higher than RAW latency because WAR involves getting exclusive ownership and invalidating other copies. The latency of cached accesses is equal or higher than that of uncached accesses.

<table>
<thead>
<tr>
<th>Type</th>
<th>Cached</th>
<th>Local</th>
<th>Remote</th>
<th>Third-party</th>
</tr>
</thead>
<tbody>
<tr>
<td>WAR</td>
<td>No</td>
<td>2.08</td>
<td>2.91</td>
<td>3.57</td>
</tr>
<tr>
<td>RAW</td>
<td>No</td>
<td>2.07</td>
<td>1.58</td>
<td>2.26</td>
</tr>
</tbody>
</table>

WAR latency generally increases as we move from local to remote to third-party homes. Uncached WAR latencies with remote and third-party homes are identical to those of cached WAR because, as the processor does not inform the directory when it replaces a clean line, invalidations are sent to the remote processor in both uncached and cached WAR accesses. However, the latency of the cached local-home WAR is high due to complications of the SCI coherence protocol. In uncached WAR, the line’s sharing list has one entry that points to the remote node, and the WAR is performed by sending an invalidation to the remote node. In cached WAR, the line’s sharing list has two nodes; the remote node at the head and the local node at the tail. In order to modify this line, the SCI protocol requires that the local node leaves the sharing list, rejoins at the sharing list head, and invalidates the other node, which sums up to a latency of 3.37 μsec (only the head of the list can invalidate copies in other nodes).

Cached RAW latency is higher than that of uncached RAW because of the additional latency of retrieving the modified copy from the remote producer’s cache after checking the coherence tag in the remote node. This retrieval is not needed in uncached RAW where the remote IC or remote memory has the valid copy. SCI complications also increase the gap between cached and uncached remote-home RAW latencies. Uncached remote-home RAW latency is the lowest latency among remote accesses (only 1.58 μsec), as it is simply satisfied through a request to the home node and a reply with data from the home node’s memory. However, cached remote-home RAW is more expensive because it comes after the expensive cached local-home WAR described above. When this RAW occurs, the line’s sharing list has the home node at the head. The SCI protocol handles this RAW similar to the third-party home case where the requester node first joins the sharing list at the head, then gets the modified line from the old head.

5.5. Coherence Overhead

This section evaluates latency as a function of the number of processors, $p$, involved in shared-memory producer-consumer communications in order to characterize the coherence overhead. In the microbenchmark used, Processor 0 uses the store-load-use kernel to access a shared
array that is interleaved among all nodes, and the other \( p - 1 \) processors take turns accessing it using the load-use kernel. The WAR time of Processor 0 is the invalidation time, shown in Figure 9 as a function of \( p - 1 \). The times for the other processors depend on the order in which they read. Figure 10 shows the incremental read times for the \( p - 1 \) reader processors from experiments using 24 and 64 processors on the SPP1000 and SPP2000, respectively. In these experiments, the processors read in the order of their IDs. The accumulative sum of the invalidation time and the \( p - 1 \) read times is the time of one iteration of this microbenchmark.

The invalidation time increases as the access stride increases, for the same reasons outlined earlier. This time also increases, in steps, as the number of processors increases. When all processors are in the same node, the invalidation time remains constant due to the crossbar’s broadcast capability. There is a large step when the new processor is from the second node, and other smaller steps as each subsequent node is added. For the SPP2000, with \( p = 2 \), the invalidation time is higher than that with \( p = 3 \) or more within one node. When there is only one reader, the invalidation time is longer in order to check the status of the exclusive copy in the reader’s cache; with \( p > 2 \) there is no exclusive copy when invalidation is being done.

In the SPP1000, as suggested by Figure 10, Processor 1 reads from the writer’s cache causing the memory to be updated as a side effect. The read time of Processors 2 through 7 is thus less since they are satisfied from the local memory that they share. The read time is higher for Processor 8 since the data is not in its node and must be provided remotely. When Processors 9 through 15 read, they find the data in their node, and their read time is similar to Processors 2 through 7. This sequence repeats for each node.

The SPP2000 read time is similar with three differences: (i) the number of processors per node is 16, (ii) the second reader also sees a high latency in order to check the status of the exclusive copy in the first reader’s cache, and (iii) the read time changes from one node to another because of varying numbers of IC conflicts.

6. Concurrent Traffic Effects

In this section, we use the bandwidth-measuring kernels to evaluate the aggregate bandwidth when multiple processors are active accessing memory.

Figure 11 shows the aggregate load bandwidth from local memory when \( p \) processors from one node use the load
kernel to access a 2p-MB array, simultaneously. Each processor accesses a disjoint 2-MB segment with stride s. The segment size is selected to evaluate the local-memory performance through the crossbar in the miss region.

As the memory bandwidth of one SPP1000 processor is limited by the latency of the individual cache misses, one processor does not stress the memory system, and even the aggregate bandwidth is nearly proportional to p for s ≤ 128 bytes. The maximum local memory load bandwidth for 8 processors is 420 MB/s when uniformly accessing all the node’s memory banks. However, the aggregate bandwidth is significantly lower, 320 MB/s, when accessing one memory board (s = 256 bytes), and 200 MB/s when accessing one bank (s = 512 bytes).

As discussed in Section 4.2, due to the high bandwidth requirements of the PA 8000, the bandwidth to one SPP2000 processor drops when the access stride skips some of the memory banks. Additionally, the local memory cannot provide enough bandwidth to allow linear scaling of the aggregate bandwidth as the number of active processors increases. The aggregate bandwidth falls off even faster when the stride begins to skip memory banks. The maximum local-memory load bandwidth for 16 processors is 4020 MB/s when uniformly accessing all the node’s memory banks, 560 MB/s when accessing one memory board, and 300 MB/s when accessing one bank.

Figure 12 shows the aggregate remote RAW bandwidth when p processors from one node use the store kernel to update disjoint 2-MB segments of a 2p-MB shared array, simultaneously. Then p processors from another node use the load kernel to access these array segments, simultaneously. The load kernel time is used to find the aggregate remote RAW bandwidth through the rings.

As the number of processors increases, the remote RAW bandwidth scales worse than the local load bandwidth. The SPP1000 remote bandwidth is adversely affected by the limitation that each ring interface allows only one outstanding request. This limitation makes communication through the rings a bottleneck, which becomes more severe as s increases and fewer rings are utilized.

The SPP2000 remote bandwidth is much higher since its ring interfaces each allow multiple outstanding requests. This high aggregate remote bandwidth can only be achieved by overlapping the remote latencies. The remote bandwidth scalability limitation here is mainly due to the complexity of the internode coherence protocol. The SCI requires at least four memory accesses for each RAW access: one to check the local IC, a second to retrieve the line from the remote
memory, a third to update the remote memory coherence tag, and the fourth to put the line in the local IC.

In the SPP1000, the aggregate remote RAW bandwidth with 8 processors is 69 MB/s when all four rings are utilized, and falls to 31 MB/s with one ring. In the SPP2000, with 16 processors, this bandwidth is 510 MB/s when all memory banks are utilized, and falls to 80 MB/s with one memory bank.

7. Synchronization Time

The SPP2000 has new primitives to improve synchronization performance. Figure 13 shows the average time spent in the WAIT_BARRIER library subroutine when all the processors enter the barrier simultaneously.

As visible from the measured data and the curve fits, the SPP2000 has much improved performance. The SPP2000 uses the new coherent_increment primitive to reduce ring traffic and to lower barrier times [7]. In the SPP1000, when a thread reaches the barrier, it increments a counter and waits on a flag. When the last thread reaches the barrier, it updates the flag to signal to the other threads to go on. This update invalidates the cached flag copies by destroying the sharing list. The next access of each thread acquires the updated flag value and rebuilds a new sharing list, which generates a lot of traffic and contention. In the SPP2000, this traffic and latency is avoided by the coherent_increment primitive which uses the sharing list to update the flag copies in the sharing nodes, without destroying the sharing list.

8. Conclusions

Exploiting advances in microarchitecture, integrated circuit technology, and packaging, the SPP2000 achieves much better performance and scalability than its predecessor, the SPP1000. Although the memory latency, in processor cycles, has not improved, the newer processor overlaps multiple outstanding cache misses to allow up to 10x local memory bandwidth. However, the PA 8000 processor’s maximum bandwidth is only achievable when memory load accesses are uniformly distributed over the memory banks and not hindered by accesses from other processors. The 32 memory banks in one SPP2000 node can actually sustain 50% of the maximum bandwidth of all 16 processors in the node, even without intervening cache hits. In the SPP1000, the PA 7100’s lower maximum bandwidth allows the 8 memory banks in a node to sustain all 8 node processors at maximum bandwidth.

The latency of local communication through shared memory accesses has changed. The latency of loading a cache line that is dirty in the cache of another local processor has improved. The new four-state intranode coherence protocol reduces communication for private and migratory access patterns, at the expense of repetitive producer-consumer communication.

The two-dimensional ring interconnection topology of the SPP2000 provides much lower internode latency and higher bisection bandwidth than the SPP1000’s single-dimension rings. The SPP2000 shrinks the gap between the remote and local latency further by adopting a smaller line size for internode communication, which now matches the line size of the processor. The SPP2000 remote latency is about 0.5x that of the SPP1000, and its remote bandwidth is about 10 times higher for a single active processor, thanks to the ability of the processor and the ring interface to overlap many memory requests. The SPP1000 ring interface, which serves outstanding requests serially, reduces the one processor maximum remote bandwidth by 42% when all 8 processors in a node are actively utilizing the rings. Since it requires multiple memory accesses per remote access, the SPP2000 internode coherence protocol reduces this bandwidth by 76% for 16 processors.

The SPP2000 decision to use custom-designed crossbar and controllers to support the needed aggregate bandwidth allowed it to have 16 processors per node, which greatly helps medium-scale parallel programs. The addition of this glue logic, however, raises the local memory and communication latencies, which penalizes sequential and small-scale parallel programs. Many modern processors are incorporating interfaces for small-scale multiprocessor buses, which can be exploited to build economic small nodes with low local latency [16]. However, for a system based on small nodes to succeed with large-scale parallel programs, the remote latency should not increase prohibitively.

Since the Exemplar does not strictly adhere to the SCI standard and does not use second-party SCI-standard components, we do not see a justification for using this protocol in such a high-performance parallel system, other than to avoid developing a new internode coherence protocol. This study has exposed several problems related to SCI efficiency and performance. The interconnect cache, which assists internode accesses, did take 12.5% of the physical memory space of the benchmarked systems, not including the 25% overhead of the coherence tags in the SPP2000. The distributed linked-list nature of the SCI directories unnecessarily increases the latencies of some remote accesses like cached WAR of local-home lines and cached RAW of remote-home lines. Additionally, the invalidation time is linear with the number of sharing nodes. Although some of these issues are addressed in research work such as [15], linked-list protocols fundamentally require more directory accesses than other protocols [1, 9, 10]. Since the SPP2000 directory is implemented in memory, the high directory access rate adversely affects remote memory latency and bandwidth.
Acknowledgments

We would like to thank University of Michigan’s Center for Parallel Computing, and Caltech’s Center for Advanced Computing Research for providing time on the SPP1000 and SPP2000, and our anonymous reviewers for their useful comments. This research was funded in part by the National Science Foundation under Grant No. ACI–9619020.

References


