

Can High Performance Software DSM Systems Designed With InfiniBand Features Benefit from PCI-Express? *

Ranjit Noronha and Dhabaleswar K. Panda

Dept. of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210

{noronha, panda}@cse.ohio-state.edu

Abstract

The performance of software distributed shared memory systems has traditionally been lower than other programming models primarily because of overhead in the coherency protocol, communication bottlenecks and slow networks. Software DSMs have benefited from interconnection technologies like Myrinet, InfiniBand and Quadrics which offer low latency and high bandwidth communication. Additionally, some of their features like RDMA and atomic operations have been used to implement some portions of the software DSM protocols directly, further reducing overhead. Such network aware protocols are dependent on characteristics of these networks. The performance of the networks is in turn dependent on the system architecture, specially the I/O bus. PCI-Express the successor to the PCI-X architecture, offers improved latency and bandwidth characteristics. In this paper, we evaluate the impact of using an improved bus technology like PCI-Express, on the performance of software DSM protocols which use the network features of InfiniBand. We can see a reduction in application execution time of up to 13% at four nodes, when PCI-Express is used instead of PCI-X.

Keywords: *DSM Systems, Cache coherency protocol, InfiniBand, System Area Networks*

1 Introduction

Software Distributed Shared Memory (SDSM) systems allow developers to easily program parallel applications, without having to worry about explicit communication of shared data items. The penalty for this is the higher overhead from false sharing, which leads to an increase in communication. The cost of communication was significant on older generations of networks like Gigabit Ethernet. Ad-

ditionally, most SDSM systems used the request-response model shown in Figure 1. In this model, a node might need to access a particular datum for processing. The node initiates a request to the owner of the datum. The owner processes this datum and sends a response back to the requestor. The receiver might need to process a large number of requests from different nodes, increasing wait times for the SDSM system. The high overhead of network communication, coupled with a heavy processing load at the receiver were some of the factors, making it hard to achieve improved performance from SDSM's.

Modern network like InfiniBand and Myrinet [1, 6] offer improved bandwidth and latency characteristics. For example, InfiniBand 4X can deliver small message latencies of a few micro-seconds, and bandwidth for large messages up to 900 MegaBytes/sec. They also allow low-overhead user-level communication. These developments have shifted the ratio of processor to network speed. The network capacity is improving at a rate greater than the ability of the processor to keep it filled to capacity. Additionally, these networks also allow the user to use advanced networking features such as Remote Direct Memory Access (RDMA), atomic operations, hardware multicast and QoS.

With these advances in networking technologies, there have been some investigations into designing SDSM protocols using network features [13]. Some of these systems like NEWGENDSM [13] attempt to reduce the need for the asynchronous handler through the use of RDMA and atomic operations in InfiniBand. These systems have demonstrated improvements in performance, both in terms of execution time and CPU utilization. With these improvements, NEWGENDSM potentially becomes more sensitive to network parameters like latency, bandwidth as well as congestion in the network. Improvements in network technology might benefit these network protocols. Advances in networking technology might not be available to applications because of system bottlenecks such as a slow

*This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506, and National Science Foundation's grants #CCR-0204429 and #CCR-0311542.

bus. Current generation systems employ the PCI-X architecture, which can provide bandwidth up to 1 GigaByte/sec. InfiniBand 4X which can deliver an aggregate bandwidth of upto 2 GigaBytes/sec is thus limited by the PCI-X architecture. Additionally, the shared architecture of PCI-X further impinges on the performance of InfiniBand 4X.

PCI-Express the successor to PCI-X employs serial point-to-point links. Each link is made of x1, x2, x4, x8, x12, x16 or x32 lanes. Each lane can transmit signals in both directions. The links can sustain upto 2.5

GigaBits/sec/lane/direction (0.5 GigaBytes/s aggregate bandwidth per lane). InfiniBand 4X can be fully utilized using PCI-Express x4 and higher interfaces.

In this paper, we evaluate the impact of improved bus technology on the performance of the network based protocol NEWGENDSM. This protocol NEWGENDSM, is potentially sensitive to communication latency and bandwidth. Bus technologies like PCI-Express allow the InfiniBand 4X to achieve higher bandwidth and lower latency. In section 2, we discuss software distributed shared memory, networking and bus technologies. In section 3, we provide an overview of the protocol NEWGENDSM. In section 4, we discuss the impact of PCI-Express on NEWGENDSM both in terms of micro-benchmarks and applications. Finally, in section 6, we present our conclusions and discuss future work.

2 Background

In this section, we discuss some of issues relating to SDSM's, networks and bus-architectures. In particular, in section 2.1, we discuss a popular SDSM package HLRC. In section 2.2, modern networks are discussed. In section 2.3, the bus architectures PCI-X and PCI-Express are discussed.

2.1 Overview of Software DSM

Software Distributed Shared Memory systems provide an easy programming environment to the application developer. However, the penalty for this ease of programmability is lower performance from false sharing and increased communication. SDSM can mainly be categorized into those which implement the non-home based lazy release consistency model like TreadMarks [9] and those that implement the home based model.

HLRC [20] is a popular software DSM which runs without any modification to the operating system kernel. HLRC

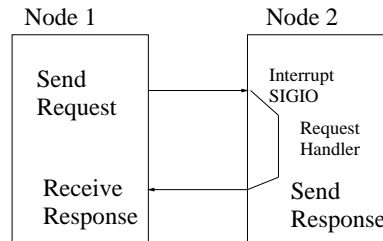


Figure 1. DSM client server communication model.

implements the *home based lazy release consistency* protocol and relies on user-level memory management techniques to provide coherency among participating processes. The communication protocol used is the virtual interface architecture or VIA [7]. The VIA implementation of HLRC has been ported to work with InfiniBand VAPI. The basic design of HLRC as well as its successors with InfiniBand network features is described in Section 3. In the next section, we discuss the networking technology InfiniBand.

2.2 Overview of Network Technologies

In the high performance computing arena, there are three popular technologies, namely Myrinet, InfiniBand and Quadrics. These networks allow low latency and high-bandwidth communication. They also allow the application to use features like Remote Direct Memory Access (RDMA), atomic operations and hardware multicast.

InfiniBand [1] uses a switched, channel-based interconnection fabric, which allows for higher bandwidth, more reliability and better QoS support. The Mellanox implementation of the InfiniBand Verbs API (VAPI) supports the basic send-receive model and Remote Direct Memory Access (RDMA) operations. The communication architecture used is the work-queue, completion-queue model. In this model, the sender posts work request descriptors to a queue. The HCA, in turn, processes these requests and inserts result descriptors in the completion queue. The sender may check the status of his request by polling the completion queue.

RDMA Write allows the user to write data elements directly into the memory space of remote nodes. RDMA Read allows an application to directly read data from the memory space of remote nodes. There is also support for atomic operations and multicast. The current generation of adapters from Mellanox is 4X. In InfiniBand 4X, small message latency is of the order of a few micro-seconds, while uni-directional bandwidth for large messages is up to 1 Giga-Byte/sec.

2.3 Overview of Bus Technologies

PCI [2] has been the dominant standard, interconnecting I/O and memory modules of a computer system for almost a decade. All other computer system components besides PCI such as CPU, memory, etc. have been scaling in some form or the other with time. The low bandwidth (132 MegaBytes/sec) of PCI posed a serious bottleneck. PCI-X, at 64 bits/133 MHz enhanced PCI, provides aggregate bandwidth up to 1 GigaBytes/sec. The shared nature of the PCI architecture, where several I/O devices could share a single bus, however still has limited scalability. As shown in Table 1, the PCI architecture could scale in terms of bandwidth, but at the expense at the number of devices it could serve. The introduction of PCI-Express alleviated some of these limitations. PCI-Express [3] employed serial point-to-point links between memory and I/O devices as shown

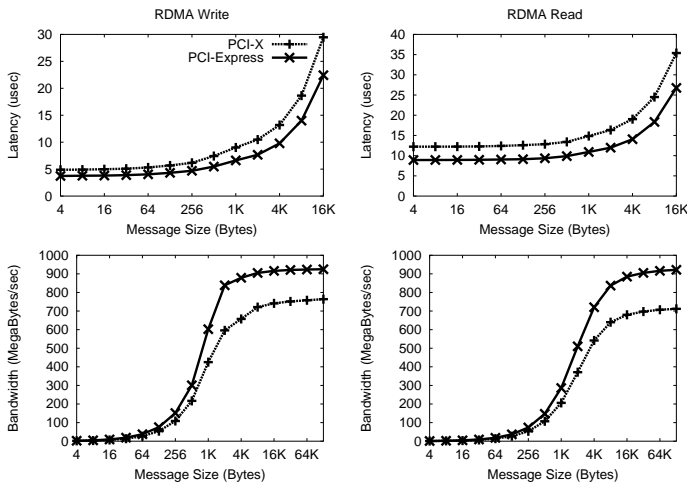


Figure 2. Performance comparison between PCI-X and PCI-Express using InfiniBand in terms of latency and bandwidth micro-benchmarks

Table 1. Capabilities of 64-bit PCI buses [16]

Interface	Peak Bandwidth (MegaBytes/s)	Cards/Bus
PCI (66 MHz)	532	1-2
PCI-X (66 MHz)	532	4
PCI-X (133 MHz)	1066	1-2
PCI-X (266 MHz)	2132	1

in Figure 3. It also allows aggregate bandwidth up to 4 GigaBytes/sec for PCI-Express x8 (8 lanes). This allows us to fully utilize the bandwidth of current generation InfiniBand hardware, which can transfer data at a rate of 1 GigaBytes/sec in each direction.

A comparison between PCI-X and PCI-Express in terms of the latency and bandwidth micro-benchmarks on InfiniBand is shown in Figure 2. Message latency for a four byte message sent through RDMA Write is approximately $3.8 \mu\text{s}$ using PCI-Express, while for PCI-X it is approximately $4.9 \mu\text{s}$. Small message latency is important in an SDSM system like HLRC, since request messages as shown in Figure 1 are small and may propagate through RDMA Write. Similarly, for RDMA Read the small message latency of RDMA Read on PCI-Express is approximately $4 \mu\text{s}$ lower than that on PCI-X. The difference increases with increasing message size. For both, the bandwidth of RDMA Read is close to the peak at message sizes greater than 4K, while for RDMA Write, the peak is reached at a lower message size. In the next section, we will discuss the overall design of the NEWGENDSM protocol.

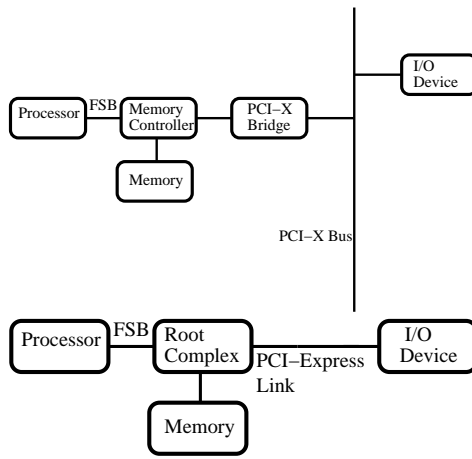


Figure 3. Architecture of PCI-X and PCI-Express

3 Overview of the NEWGENDSM protocol

In this section, we discuss the protocol NEWGENDSM proposed in [13]. We start out by examining the existing protocol termed ASYNC. Following that is a description of the design of NEWGENDSM and then finally we examine how NEWGENDSM might benefit from improved bus technology.

3.1 Base ASYNC protocol

ASYNC employs the home based lazy release consistency protocol. In this protocol every page and lock is assigned a home by the protocol. All requests for accesses to a page or a lock go to the home node. Similarly all updates for a page and a lock go to the home node. In ASYNC, updates or diffs for a page propagate to the home node at synchronization points such as a lock release or a barrier. We now examine page fetching and diffing in ASYNC.

3.1.1 Page fetching in ASYNC

First consider a page fetch operation. Figure 4 shows the protocol activity required by ASYNC for an example scenario. Initially

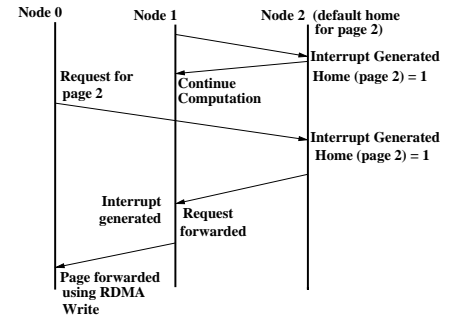


Figure 4. The original ASYNC protocol on a page fetch (for an example scenario)

all pages are assigned default home nodes. Let us assume that the default home for page two is node two. Now node one first requests page two from node two by sending it a message (RDMA Write with immediate data) which is processed by the asynchronous protocol handler. The handler on node two appoints node one as the home node and updates its data structures. It then finishes servicing the request by sending a reply to node one (RDMA Write). Node one on receiving this reply updates its data structures and continues computing. Now when node zero requests this page for the first time by sending a request to the default home node one, node two forwards this request to node one, which in turn processes the request and replies to node zero with the correct version of the page. We define *page fetch time* or *page time* as the elapsed time between sending a page request and actually receiving the page.

3.1.2 Diffing in ASYNC

Now we go through the protocol steps when computing and applying diffs. As shown in Figure 5, node zero arrives at a synchronization point such as a barrier

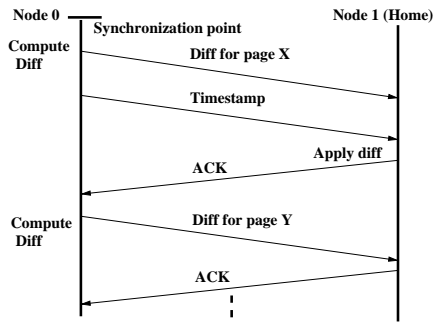


Figure 5. The original ASYNC protocol on a diff (for an example scenario)

or a lock. At this point, node zero must propagate all updates it has made to all pages to the home node. Assume node zero has modified pages X and Y. It computes the diff which is a run-length encoded string of the differences between the original page and the modified page. Following that it sends these differences to the home node one through RDMA Write followed by a message containing the time-stamp. The home node then applies these diffs. It then sends an ACK back to node zero, which acts as a signal to node zero that the buffer on node one, which were used to receive the diffs to page X have been freed up and may be reused. Node zero now computes the diffs for page Y and sends them to node one and so on. Multiple buffers at the receiver may be used to speedup the process. We will now see how parts of this protocol can be enhanced with the features available in InfiniBand.

3.2 NEWGENDSM protocol

In this section we discuss the protocol NEWGENDSM proposed in [13]. NEWGENDSM consists of two compo-

nents; ARDMAR (Atomic and RDMA Read) and DRAW (Diff with RDMA Write). ARDMAR uses the InfiniBand atomic operation *Compare and Swap* and *RDMA Read* for page fetching and synchronization. DRAW uses *RDMA Write* for diffing. First we describe ARDMAR followed by DRAW.

3.2.1 ARDMAR

In this section we describe the design of ARDMAR which is shown in Figure 6 for an example scenario. The Atomic operation compare and swap have been combined with the RDMA Read operation to completely eliminate the asynchronous protocol processing. Let us assume the same pattern of requests for a page as shown in Figure 4. Here assume that node one wants to access page two for the first time. Let $home_n(x)$ denote the last known value for the home of page x at node n. Initial values for $home_n(x)$ are -1 at the default home node (indicating that a home has not been assigned) and $x \bmod$ (number of nodes) at a non-home node. Initially $home_2(2) = -1$. Let us denote an issued atomic compare and swap operation as $CMP_SWAP(node, address, compare\ with\ value, swap\ with\ value)$ where address points to some location in node. Node 1 issues an atomic compare and swap $CMP_SWAP(2, home_2(2), -1, 1)$. The compare succeeds and now $home_2(2) = 1$. Now on completion of the atomic operation, node 1 knows that it is the home node ($home_2(2) = 1$) and can continue computation after appropriately setting the appropriate memory protections on the page.

Now assume that node zero wants to access page two for the first time. It also issues an atomic compare and swap operation $CMP_SWAP(2, home_2(2), -1, 0)$ which fails since $home_2(2) = 1$. Simultaneous with the atomic compare and swap, node zero also issues an RDMA Read to read in $home_2(2)$. The atomic compare and swap having failed node zero looks at the location read in by the RDMA Read. This location tells node zero that the actual home is now node one. Node zero now issues two simultaneous RDMA Reads. The first RDMA Read brings in the version of the page, while

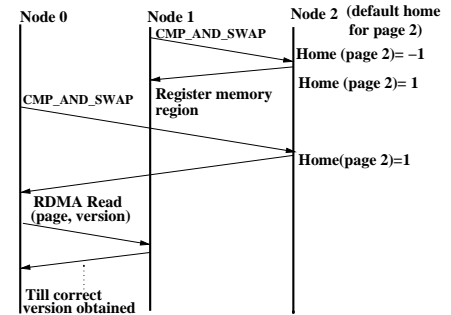


Figure 6. The proposed protocol ARDMAR (for the example scenario)

the second RDMA Read brings in the actual page. If the version does not match, both RDMA's are reissued until the correct version is obtained.

3.2.2 DRAW

Let us look at the design of DRAW. Figure 7 shows the protocol activity. DRAW uses RDMA Write to directly write the diffs to the page on the destination node. Again consider the diff creation and application activity shown in Figure 5.

DRAW

moves most of the diffing activity into node zero as shown in Figure 7. Assume that node zero has now initiated

computing the diff at

the synchronization point. Let $modified(X,n)$ refer to the n 'th position in page X. Let $clean(X,n)$ refer to the n 'th position in the twin of X where twin refers to a clean copy of page X. Let $buffer(t,n)$ refer to the n 'th position in communication buffer t . Let $RWRITE(source,dest,t,s,len)$ denote an RDMA Write descriptor initiated at node source bound for node dest, using buffer t , starting at location s and of length len . Let us assume further that $modified(X,i..j)$ differ from $clean(X,i..j)$. In this case, DRAW copies $modified(X,i..j)$ into $buffer(t,i..j)$ and creates $RWRITE(0,1,t,i,j-i)$. Similarly, assume that $modified(Y,b..f)$ differ from $clean(y,b..f)$. DRAW copies $modified(Y,b..f)$ into $buffer(t+1,b..f)$ and creates $RWRITE(0,1,t+1,b,f-b)$. Now if pages X and Y are the only modified pages, at node zero DRAW issues $RWRITE(0,1,t,i,j-i)$ and $RWRITE(0,1,t+1,b,f-b)$. In addition, a message containing the timestamps of pages X and Y is also sent. At node one, on receiving the messages containing the timestamps, DRAW updates the timestamps for pages X and Y.

3.3 Potential Benefits With PCI-Express

As discussed in Section 2.3, NEWGENDSM is a more synchronous protocol. Its two components are ARDMAR and DRAW. ARDMAR reads pages using RDMA Read. DRAW propagates DIFF's through RDMA Write. As a result, NEWGENDSM is potentially sensitive to network characteristics. These characteristics include latency and bandwidth of primitives like RDMA Read and Write. Ad-

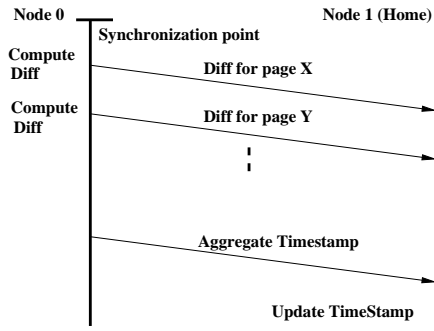


Figure 7. The proposed protocol DRAW (for the example scenario)

ditionally, since every node in the application might be executing the protocol, it is important that the network be able to support multiple probes from NEWGENDSM. This might result in a greater load on the network. As discussed in Section 3.2.1; ARDMAR might need to access the network multiple times to obtain a page. As a result, quick access to the network is an important requirement.

The requirements of NEWGENDSM match well with the architecture of PCI-Express, shown in Figure 3 and discussed in Section 2.3. PCI-Express is a serial point-to-point link interface, unlike the shared bus medium of PCI-X. As a result, there is less contention to access the network. This also translates into improved latency. Additionally, PCI-Express x8 allows an aggregate bandwidth of up to 4 GigaBytes/s, allowing NEWGENDSM to fully exploit the full aggregate bandwidth of InfiniBand 4X. Thus, multiple nodes executing NEWGENDSM might be able to tap into the network with lower latency and higher bandwidth, for improved performance.

4 Performance Evaluation

This section evaluates the performance of NEWGENDSM and ASYNC on PCI-X and PCI-Express architectures. Evaluation is in terms of micro-benchmarks and applications. First we describe the hardware setup in section 4.1. Following that NEWGENDSM is evaluated using the Page micro-benchmark in Section 4.2. The application level evaluation is presented in Section 4.3.

4.1 Experimental Test Bed

The experiments were run on a four node cluster. Each node has a dual 3.4 GHz Xeon processor (64-bit) and 512 MB main memory. The nodes also have both 64-bit, 133 MHz PCI-X and x8 PCI-Express interfaces. The InfiniBand MT23108 (PCI-X) and MT25208 (PCI-Express) adapters were installed in each node. The adapters were connected through an InfiniScale MT43132 switch. Each node has Linux installed and the kernel version is 2.4.21..

4.2 Micro-benchmark level evaluation

The performance of ARDMAR and ASYNC on PCI-X and PCI-Express were evaluated using the page fetch micro-benchmark. This micro-benchmark is modified from the original version implemented for the TreadMarks SDSM package [9]. *Page fetch time* is the elapsed interval between sending a request for a page and actually getting the page. It is measured as follows. The first node (master node) initially touches each of 1024 pages so that the home node is assigned to it. Following that each of the remaining nodes reads one word from each of the 1024 pages. This results in all the 1024 pages being read from the first node. As the number of nodes increases, the contention for a page at

the master node increases. The time of the second phase is measured.

Table 2 shows the results of running the page micro-benchmark on PCI-X and PCI-Express. We see that *ASYNC* performs slightly worse than *NEWGENDSM* at two and four nodes, when comparing the same architecture, i.e. PCI-X or PCI-Express. This matches with the observations in [13]. *NEWGENDSM* is expected to show improved performance at large system size. But we can see that PCI-Express performs better than PCI-X for both *ASYNC* and *NEWGENDSM*. There is approximately a 6-15% improvement when replacing PCI-X with PCI-Express at two nodes. This increases to approximately 18-21% at four nodes. We would expect the improvement to increase, with increasing system size. *ASYNC* and *NEWGENDSM* were not evaluated using the diff micro-benchmark discussed in [13], since the diff micro-benchmark does not measure improvement due to communication, the metric we are interested in.

4.3 Application level evaluation

In this section we evaluate our implementation using four different applications; Barnes-HUT (Barnes), three dimensional FFT (3Dfft), LU Decomposition (LU) and Integer Sort (IS). Out of these applications Barnes and LU have been taken from the SPLASH-2 benchmark suite [19] while 3Dfft and IS have been taken from the TreadMarks [9] SDSM package. The application sizes used are shown in Figure 3. All other parameters were kept the same as originally described in [19, 9]. We will now discuss the performance numbers for the different applications.

4.3.1 Application Characteristics

In this section, we discuss some of the application characteristics relevant to our design. The first application Barnes, is an N-Body simulation using the hierarchical Barnes-Hut method. It contains two main arrays, one containing the bodies in the simulation and the other the cells. Sharing patterns are irregular and true. A fairly large amount of diff traffic is exchanged at barriers, which are the synchronization points. 3DFFT performs a three dimensional Fast Fourier Transform. It exchanges a large volume of messages per unit time. It also uses a large number of locks and barriers for synchronization. The LU program factors a dense matrix into the product of a lower triangular and an upper triangular matrix. The factorization uses blocking to exploit temporal locality on individual submatrix elements. LU sends a large numbers of diffs. It also exchanges the maximum amount of data traffic. IS implements Bucket Sort. There is a global array containing the buckets, and a local array which the local node uses to sort its data. After each iteration, each node places its data in the global array and copies the data relevant to it into its local array. A large numbers of diffs are exchanged at intervals.

Table 3. Application sizes

Application	Parameter	Size
Barnes	Bodies	4096
3Dfft	Grid size	4
LU	Matrix dimension	16
IS	num of keys	2 ⁸

4.3.2 Effect on execution time

The execution time of the different applications at four nodes while using PCI-X and PCI-Express is shown in Figure 8. *ASYNC-X* refers to an application run with the PCI-X interface, with *ASYNC*. *ASYNC-E* refers to an application run over the PCI-Express interface. Similarly, *NEWGENDSM-X* and *NEWGENDSM-E* refer to run of applications with the *NEWGENDSM* protocol over PCI-X and PCI-Express respectively. As can be seen from Figure 8, PCI-Express can reduce the execution time of Barnes by up to 6%. For 3Dfft, when run over PCI-Express, there is a reduction in execution time by up to 13%. Similarly, for LU, there is an improvement up to 3%. For IS there is an improvement of approximately 10%.

4.3.3 Effect on wait time

We will now look at the breakdown of execution time of the applications, while running *NEWGENDSM* on PCI-X and PCI-Express. Barnes-X refers to the time taken by Barnes using *NEWGENDSM* on PCI-X, while Barnes-E refers to the time on PCI-Express. Similar notation is used for other applications. The breakdown of timing for Barnes, 3Dfft, LU and IS is shown in Figure 9. *Compute* time refers to the per-node time spent in application level processing. *Wait time* refers to time spent waiting at synchronization points such as barriers. *Page* time refers to the time needed to fetch an updated copy of a page from a remote node. From Figure 9, we can see that there is a small reduction in the wait time when PCI-X is replaced with PCI-Express. There is a more significant reduction in the page time discussed next.

4.3.4 Effect on page time

The breakdown of execution time of the applications while using *NEWGENDSM* is shown in Figure 9. Barnes-X refers to the time taken by Barnes using *NEWGENDSM* on PCI-X, while Barnes-E refers to the time on PCI-Express. Similar notation is used for other applications. The *Page* time refers to the per-node time spent in waiting for a particular page from a remote node. This includes the time needed to execute the protocol ARDMAR described in section 3.2.1. From Figure 9, we see that PCI-Express helps to

Table 2. Time for the page microbenchmark in μ s

Interface	ASYNC (two nodes)	ASYNC (four nodes)	NEWGENDSM (two nodes)	NEWGENDSM (four nodes)
PCI-X	36	51	33	50
PCI-Express	33	40	27	41

reduce the *Page* time. This is particularly true in the case of Barnes, 3Dfft and IS.

5 Related Work

Ever since the proposal for the first SDSM system IVY [10] there has been considerable research conducted into the SDSM systems. However SDSM was not found to be scalable. The benefits of implementing HLRC over low level protocol like VIA was examined in [15]. Also implementing a SDSM package like Treadmarks directly over a low level protocol like VIA or GM over Myrinet, was shown to substantially reduce wait times and improve scalability [5, 12]. A study of the effect of removing the interrupt handler in HLRC (GeNIMA) through the use of NIC support on Myrinet is discussed in [4]. A comparison of benefits of using network based support as opposed to a migratory home protocol in the Cashmere SDSM system was studied in [17]. The interconnect used in this study was memory channel. Integrating network based get operations into the sequentially consistent DSZOOM SDSM over the SCI interface is discussed in [14]. Cache entries are locked using atomic fetch and set operations before being modified by remote put operations. NEWGENDSM uses a lazy release consistency (LRC) model rather than a sequentially consistent model. Network interface support was used to perform virtual memory mapped communication in addition to DMA based communication along with protected, low-latency user-level message passing in the SHRIMP project [11]. A proposal for using active-memory support for SDSM systems to achieve software DSM with hardware DSM performance is discussed in [8]. The research in [18] explores the effect of kernel level access to InfiniBand primitives on SDSM performance.

6 Conclusions and Future Work

In this work we have examined the impact of the bus technology PCI-Express on the performance of the software Distributed Shared Memory System NEWGENDSM. NEWGENDSM employs RDMA and atomic operations in InfiniBand to implement the coherency protocol. We have evaluated the performance of NEWGENDSM over PCI-Express and PCI-X using micro-benchmarks and applications. The page micro-benchmark shows that PCI-Express improves performance of NEWGENDSM up to

21% compared to PCI-X. Application level evaluation using the SPLASH-2 suite shows a reduction of up to 13% in execution time when PCI-Express is replaced with PCI-X.

We would like to evaluate how using multiple threads on the same node, would impact the performance when using PCI-Express. The higher bandwidth of PCI-Express x8 coupled with InfiniBand 4X might show improved performance from multi-threaded support. We would also like to see how implementing true zero-copy support for diffing might impact performance in NEWGENDSM. We would also like to evaluate the scalability of NEWGENDSM on a cluster of up to 256 nodes using PCI-Express.

References

- [1] Infi niband Trade Association. www.infi-nibandta.org.
- [2] PCI and PCI-X. <http://www.pcisig.com>.
- [3] PCI Express architecture. <http://www.pcisig.com>.
- [4] C. L. A. Bilas and J. Singh. Using Network Interface Support to Avoid Asynchronous Protocol Processing in Shared Virtual Memory Systems. *International Symposium on Computer Architecture*, May 1999.
- [5] M. Banikazemi, J. Liu, . K. Panda, and P. Sadayappan. Implementing TreadMarks over VIA on Myrinet and Gigabit Ethernet: Challenges Design Experience and Performance Evaluation. *Int'l Conference on Parallel Processing*, sep 2001.
- [6] N. J. Boden, D. Cohen, et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, pages 29–35, Feb 1995.
- [7] Compaq, Intel, and Microsoft Corporation. *Virtual Interface Architecture Specification*, 1997.
- [8] M. Heinrich and E. Speight. Providing Hardware DSM Performance at Software DSM Cost. *Technical Report No. CSL-TR-2000-1008*, Cornell University, Ithaca, NY, November 2000.
- [9] P. Keleher, A. L. Cox, S. Dwarkadas, and W. Zwaenepoel. TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems. In *Proceedings of the 1994 Winter Usenix Conference*, Jan. 1994.
- [10] K. Li. IVY: A Shared Virtual Memory System for Parallel Computing. In *Proceedings of the International Conference on Parallel Processing*, pages 94–101, Los Alamitos, CA, 1988.
- [11] M.A. Blumrich, C. Dubnicki, E.W. Felten, Kai Li, M.R. Mesarina. Two Virutal Memory Mapped Network Interface Designs. *Proc. of the Hot Interconnects Symp.*, 1994.
- [12] R. Noronha and D. K. Panda. Implementing TreadMarks over GM on Myrinet: Challenges, Design Experience and

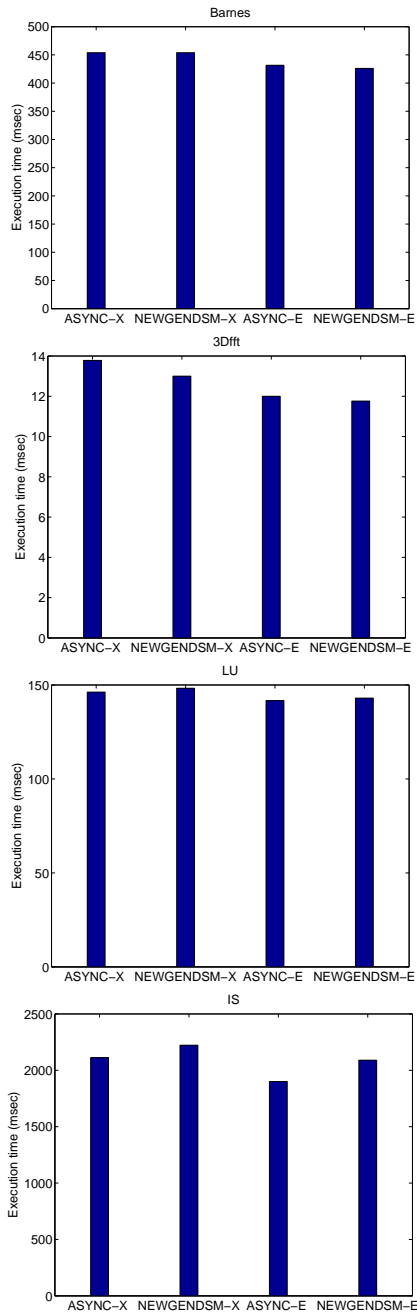


Figure 8. Execution time of different applications with PCI-X (ASYNC-X, NEWGENDSM-X) and PCI-Express (ASYNC-E, NEWGENDSM-E)

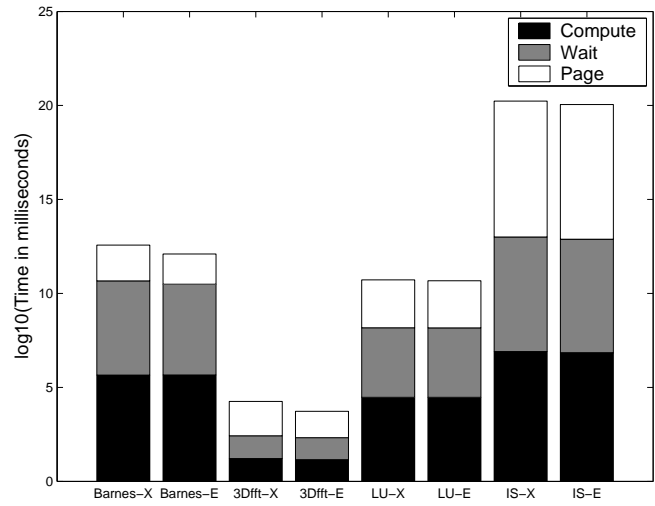


Figure 9. Breakdown time of different applications with PCI-X and PCI-Express using NEWGENDSM. App-X refers to runtime with PCI-X, while App-E refers to the time taken with PCI-Express.

Performance Evaluation. *Workshop on Communication Architecture for Clusters (CAC'03)*, held in conjunction with *IPDPS '03*, April 2003.

- [13] R. Noronha and D.K. Panda. Designing High Performance DSM Systems using Infi niBand Features. *Workshop on Distributed Shared Memory on Clusters (DSM' 04)*, held in conjunction with *CCGRID '04*, April 2004.
- [14] Z. Radovic and E. Hagerstern. Implementing Low Latency Distributed Software-Based Shared Memory. *Workshop on Memory Performance Issues*, held in conjunction with *ISCA '01*, feb 1996.
- [15] M. Rangarajan and L. Iftode. Software Distributed Shared Memory over Virtual Interface Architectur: Implementation and Performance. *Proc. of the Annual Linux Showcase, Extreme Linux Workshop, Atlanta*, October 2000.
- [16] Ravi Budruk, Don Anderson, Tom Shanley. *PCI Express System Architecture*. Addison Wesley, 2003.
- [17] R. Stets, S. Dwarkadas, L. Kontothanassis, U. Rencuzogullari, and M. L. Scott. The Effect of Network Total Order, Broadcast, and Remote-Write Capability on Network-Based Shared Memory Computing. In *International Symposium on High-Performance Computer Architecture*, 2000.
- [18] T. Birk, L. Liss and A. Schuster. Efficient Exploitation of Kernel Access to Infi niBand: a Software DSM Example. *Hot Interconnects*, August 2003.
- [19] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *International Symposium on Computer Architecture*, pages 24–36, 1995.
- [20] Y. Zhou, L. Iftode, and K. Li. Performance Evaluation of Two Home-Based Lazy Release Consistency Protocols for Shared Virtual Memory Systems. In *Proceedings of Operating Systems Design and Implementation Symposium*, October 1996.