

Supporting MPI-2 One Sided Communication on Multi-Rail InfiniBand Clusters: Design Challenges and Performance Benefits ^{*}

Abhinav Vishnu, Gopal Santhanaraman, Wei Huang,
Hyun-Wook Jin, and Dhabaleswar K. Panda

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

{vishnu, santhana, huanwei, jinhy, panda}@cse.ohio-state.edu

Abstract. In cluster computing, InfiniBand has emerged as a popular high performance interconnect with MPI as the *de facto* programming model. However, even with InfiniBand, bandwidth can become a bottleneck for clusters executing communication intensive applications. Multi-rail cluster configurations with MPI-1 are being proposed to alleviate this problem. Recently, MPI-2 with support for one-sided communication is gaining significance. In this paper, we take the challenge of designing high performance MPI-2 *one-sided communication* on multi-rail InfiniBand clusters. We propose a unified MPI-2 design for different configurations of multi-rail networks (*multiple ports, multiple HCAs and combinations*). We present various issues associated with one-sided communication such as *multiple synchronization messages, scheduling of RDMA (Read, Write) operations, ordering relaxation* and discuss their implications on our design. Our performance results show that multi-rail networks can significantly improve MPI-2 one-sided communication performance. Using PCI-Express with two-ports, we can achieve a peak *MPI_Put* bidirectional bandwidth of 2620 Million Bytes/s, compared to 1910 MB/s for single-rail implementation. For PCI-X with two HCAs, we can almost double the throughput and reduce the latency to half for large messages.

1 Introduction

High computational power of commodity PCs combined with the emergence of low latency and high bandwidth interconnects has led to the trend of *cluster computing*. In this area, Message Passing Interface (MPI) [6] has become the *de facto* standard for writing parallel applications. MPI-2 has been introduced as a successor of MPI-1 with *one-sided communication* as one of its main additional features. Recently, InfiniBand Architecture [8] has been proposed as the next generation interconnect for inter-process communication and I/O. Due to its open standard and high performance, InfiniBand is becoming increasingly popular for cluster computing. However, even with InfiniBand, network bandwidth can become the performance bottleneck for communication intensive applications. This is especially the case for clusters built with SMP

^{*} This research is supported in part by Department of Energy's grant #DE-FC02-01ER25506; National Science Foundation's grants #CNS-0403342 and #CCR-0311542; grants from Intel and Mellanox; and equipment donations from Intel, Mellanox, AMD, Apple and Sun Microsystems.

(2-16 way symmetric multiprocessor systems) machines, in which multiple processes may run on a single node and must share the node bandwidth. Multi-rail [11] (*multiple ports, multiple HCAs and combinations*) cluster configurations with MPI-1 are being proposed to alleviate this problem. Compared to MPI-1, MPI-2 is the next generation MPI standard with one-sided operations (such as `MPI_Put` and `MPI_Get`). This leads to the following challenges:

1. *How to design support for one-sided operations on multi-rail InfiniBand clusters?*
2. *How much benefits can be achieved compared to the single-rail implementation?*

In this paper, we take on these challenges. We propose a unified MPI-2 design with different configurations of multi-rail networks (*multiple ports, multiple HCAs and combinations*) for one-sided communication. We present various issues associated with one-sided communication (*multiple synchronization messages, scheduling of RDMA (Read, Write) operations, scheduling policies, ordering relaxation*) and discuss their implications on our design.

We implement our design on MVAPICH2¹ and evaluate it with micro-benchmarks on different multi-rail configurations. Our performance results show that multi-rail networks can significantly improve MPI-2 one-sided communication performance. Using two-ports on EM64T cluster with PCI-Express, we can achieve an *MPI_Put* bandwidth of 1500 Million Bytes/s (MB/s), and a bidirectional bandwidth of 2620 MB/s. Using two-HCAs on IA32 cluster with independent PCI-X buses, we can achieve a *MPI_Put* bandwidth of 1750 MB/s, and a bidirectional bandwidth of 1810 MB/s.

The rest of the paper is organized as follows: In section 2, we provide background information for InfiniBand, MVAPICH2 and multi-rail configurations. In section 3, we describe the multi-rail MPI-2 design for one-sided communication and discuss the design issues. In section 4, we present performance results of our multi-rail MPI-2 implementation. In section 5, we present the related work. In section 6, we conclude and discuss our future directions.

2 Background

In this section, we provide background information for our work. First, we provide a brief introduction of InfiniBand. Then, we discuss some of the internals of MPI-2 one-sided communication and their implementations over InfiniBand. We also present a brief overview of multi-rail InfiniBand clusters.

2.1 Overview of InfiniBand

The InfiniBand Architecture (IBA) [8] defines a switched network fabric for inter-connecting processing nodes and I/O nodes. It provides a communication and management infrastructure for inter-processor communication and I/O. In an InfiniBand network, processing nodes and I/O nodes are connected to the fabric by *Host Channel Adapters* (HCA). HCAs sit on processing nodes. InfiniBand Architecture supports both channel and memory semantics for Reliable Connection service. In channel semantics,

¹ MVAPICH/MVAPICH2 [13] are high performance MPI-1 and MPI-2 implementations from The Ohio State University, currently being used by more than 250 organizations across 28 countries.

send/receive operations are used for communication. In memory semantics, InfiniBand supports *Remote Direct Memory Access (RDMA)* operations, including RDMA write and RDMA read. RDMA operations are one-sided and do not incur software overhead at the remote side. In these operations, the sender can directly access remote memory by posting RDMA descriptors.

2.2 MPI-2 one-sided communication

In MPI-2 one-sided communication, the sender can access the remote address space directly. Such one-sided communication is also referred to as *Remote Memory Access* or *RMA* communication. In this model, the *origin process* (the process that issues the RMA operation) provides necessary parameters needed for communication. The area of memory on the *target process* accessible by the origin process is called a *Window*. MPI-2 specification defines various communication operations:

1. *MPI_Put* operation transfers the data to a window in the target process
2. *MPI_Get* operation transfers the data from a window in the target process
3. *MPI_Accumulate* operation combines the data movement to target with a reduce operation

As per the semantics of one-sided communication, the return of the one-sided operation call does not guarantee the completion of the operation. In order to guarantee the completion of one-sided operation, explicit synchronization operations must be used. We mainly focus on active synchronization in this paper.

2.3 Multi-Rail InfiniBand Configurations

Multi-rail networks can be built by using *multiple HCAs* on a single node, or by using *multiple ports* in a single HCA. In an MPI application, any pair of processes can communicate with each other. This is implemented in MPICH2 designs by an abstraction called *virtual channel*. A virtual channel can be regarded as an abstract communication channel between two processes. In [11], we have proposed enhanced virtual abstraction to provide a unified solution to support multiple HCAs, multiple ports, and multiple paths in a single port. In our proposed design, a virtual channel can consist of multiple *virtual subchannels* (referred as subchannels from here on-wards). Each subchannel refers to a path of communication between end nodes.

2.4 MVAPICH2

MVAPICH2[13] is our high performance implementation of MPI-2 over InfiniBand. The implementation is based on MPICH2. As a successor of MPICH[6], MPICH2[1] supports MPI-1 as well as MPI-2 extensions including one-sided communication. One sided communication can be implemented using a variety of approaches. One approach is to use the point to point implementation provided by MPICH2 for one-sided communication. This approach involves the remote host for communication and synchronization operations. In the second approach, the one-sided operations are implemented at the CH3 level by extending the CH3 interface [10, 9]. This approach shows benefits with respect to latency and bandwidth for regular communication patterns. It also provides

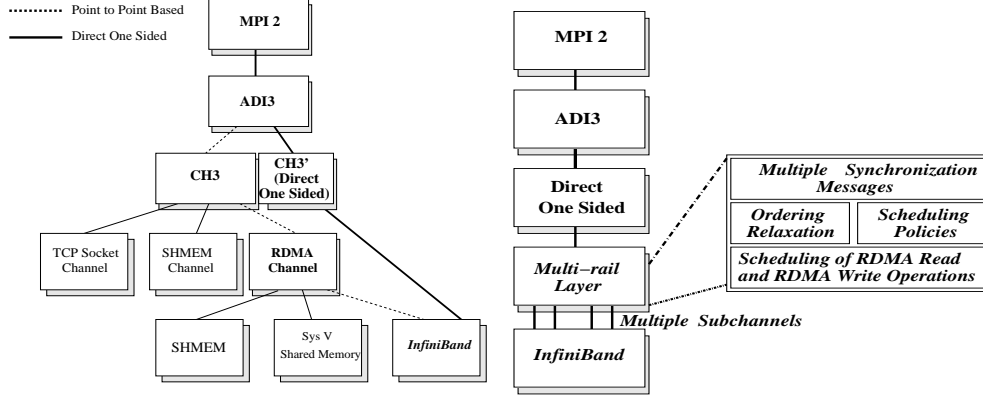


Fig. 1. Implementations of one-sided communication in MVAPICH2

Fig. 2. Basic Architecture

better overlap between computation and communication along with scalability. We refer to the first approach as *Point to Point Based* and second approach as *Direct One Sided*. Fig. 1 shows the path taken by these approaches. In this paper, we design the *Direct One Sided* over multi-rail InfiniBand clusters implementation along with *active* mode of synchronization.

3 Multi-rail Layer Design for MPI-2 One Sided Communication

In this section, we present the design issues involved with MPI-2 one-sided communication on multi-rail InfiniBand clusters.

3.1 Basic Architecture

The basic architecture of our design to support multi-rail networks for MPI-2 one-sided communication is shown in Figure 2. In the figure, we can see that besides the MPI-2, Direct One Sided layer and InfiniBand layer, our design consists of an intermediate layer, *Multi-rail Layer*.

This layer takes the responsibility of scheduling messages on the available subchannels. Besides this, it takes care of the correctness issues like *Multiple Synchronization Messages* and efficiency issues like *Scheduling Policies*, *Ordering Relaxation* and *Scheduling of RDMA Read and RDMA Write Operations*.

In this section, we discuss the design challenges involved for multi-rail MPI-2 design associated at the Multi-rail Layer.

Multiple Synchronization Messages: In order to initiate the one-sided communication, the origin process calls *win_start* to open a window. The target process *posts* the buffers for the window. Once the one-sided communication is done, a synchronization message needs to be sent to the target process. The receipt of synchronization message guarantees the data transfer of previously issued RMA operations. However, when multiple subchannels are used, data transfer on one subchannel might not have finished even though other subchannels would have received the synchronization message. Hence, we need to issue synchronization messages on each subchannel. It is to

be noted, that when the load on subchannels is balanced, the transfer of synchronization messages along multiple subchannels takes place in parallel, incurring very small overhead.

Scheduling of RDMA Read and RDMA Write Operations: In MPI-1, usually the two sided communication uses either RDMA Write or RDMA Read for data transfer in InfiniBand. For many MPI-2 applications, in one-sided communication, the *MPI_Put* and *MPI_Get* operations are implemented using RDMA Write and RDMA Read, respectively. Since RDMA Read and RDMA Write utilize bandwidth in different directions, it is important to schedule them independently with respect to each other’s load on different subchannels.

In order to achieve this, we propose a load based fragmentation policy discussed in the next section, which maintains independent queues of *MPI_Put* and *MPI_Get* operations issued in an *epoch*. Trivially, this policy would fragment the messages equally on all subchannels in the presence of only one kind of one-sided operation. In presence of a combination of one-sided operations, each having the same size, this policy would fall back to equal fragmentation.

Scheduling Policies Classification based on Message Size: In this paper, we classify the policies used for scheduling based at different layers. As proposed in [7], we use *reordering* and *no reordering* policies at the CH3’ (Direct One Sided) Layer. At the multi-rail layer, we do a classification of the policies based on the message size. We employ the following policies:

- *Round Robin*
- *Load Balanced Fragmentation*

For small messages, we employ round robin policy. In this policy, the complete message is sent using one of the available subchannels in a round robin fashion. Fragmentation incurs overhead of posting descriptors on multiple subchannels, which is significant for small messages. Hence, we employ a *switchover threshold*, messages of size less than this threshold are scheduled in a round robin fashion. For large messages, we primarily use Load Balanced Fragmentation policy. In this policy, we divide the message in chunks and schedule them, so that the load on all subchannels is balanced. This policy leads to optimal utilization of all subchannels for medium to large messages.

Ordering Relaxation: Two-sided communication requires messages to be processed in order at the receiver side. One sided communication imposes no ordering requirements for messages within an *epoch*, by the definition from the semantics. As a result, the one-sided approach does not need to maintain ordering at the receiver side. We simplify our design by incorporating this fact, reducing the overhead of bookkeeping at the receiver side.

4 Performance Evaluation

In this section, we evaluate the performance of our multi-rail MPI-2 design over InfiniBand. We show the performance benefit which can be achieved with multi-rail design compared to the single-rail implementation.

4.1 Experimental Testbed

We evaluated our implementation with multiple HCAs on IA32 systems comprising of independent PCI-X buses, and on EM64T systems comprising of PCI-Express bus and multiple ports per adapter. Our experimental testbed comprises of two clusters.

IA32 Cluster with Multiple HCAs: This cluster consists of two SuperMicro SUPER X5DL8-GG nodes with ServerWorks GC LE chipsets. Each node has dual Intel Xeon 3.0 GHz processors, 512 KB L2 cache, and PCI-X 64-bit 133 MHz bus. We have used InfiniHost MT23108 Dual-Port 4x HCAs from Mellanox. The ServerWorks GC LE chipsets have two separate I/O bridges and three PCI-X 64-bit 133 MHz bus slots. To reduce the impact of I/O bus contention, the two HCAs are connected to separate PCI-X buses connected to different I/O bridges.

EM64T Cluster with Multiple ports: This cluster consists of two EM64T nodes having 8X PCI Express slots. Each node has two Intel Xeon CPUs running at 3.4 GHz processors, 512 KB L2 cache and 1 GB of main memory. This cluster uses III Generation MT25208 4X Dual Port HCAs from Mellanox. A combined unidirectional bandwidth of 8X can be used, when both ports are used for communication.

4.2 One Sided Communication Micro-Benchmarks

In this section, we introduce the micro-benchmarks used to evaluate the MPI-2 one-sided operation performance. We use such as uni- and bi-directional bandwidth, as well as micro-benchmarks with other communication patterns.

Two processes are involved in uni-directional bandwidth test. The origin process starts a window access epoch, issues a *window* of RMA operations (MPI_Put for MPI_Put bandwidth test, MPI_Get for MPI_Get bandwidth test), and ends the access epoch. The target process just starts and ends a window exposure epoch. This step is repeated for multiple iterations. For bidirectional bandwidth test, both processes starts and end a window exposure epoch.

In the Interleaving test, the origin process issues a *window* of *MPI_Put* operations followed by a *window* of *MPI_Get* operations. Due to the impact of reordering at CH3' layer, interleaving of these operations provides almost bidirectional bandwidth throughput in comparison to unidirectional throughput.

4.3 Performance Benefits of Multi-Rail Design

To evaluate the performance benefits of our multi-rail MPI-2 design, we compare it with our original MVAPICH2 design, which can only use only one-port of a nic. In the multi-rail design, we use *load balanced fragmentation* for large messages and *round robin* scheme for small messages. We present performance comparisons using latency for *MPI_Get* operation and bandwidth and bidirectional bandwidth for *MPI_Put* operations.

Microbenchmark Evaluation for Basic One Sided Operations In Figures 3 and 5 we present the results for *MPI_Put* bandwidth and bidirectional bandwidth respectively for the IA32 cluster with multiple HCAs. We show the results for EM64T with two-ports on PCI-Express in Figures 4 and 6.

In Figure 3, we observe that for small messages (less than or equal to 1KBytes), both multi-rail design and the original implementation perform comparably. For large messages, multi-rail design outperforms the original implementation considerably. With multi-rail design, we can achieve a maximum peak unidirectional *MPI_Put* bandwidth of 1750 MB/s in comparison to 880 MB/s for our original implementation. We also notice, that due to the absence of rendezvous protocol, medium size messages (2KB - 16KB) can take advantage of load balanced fragmentation policy for multi-rail design.

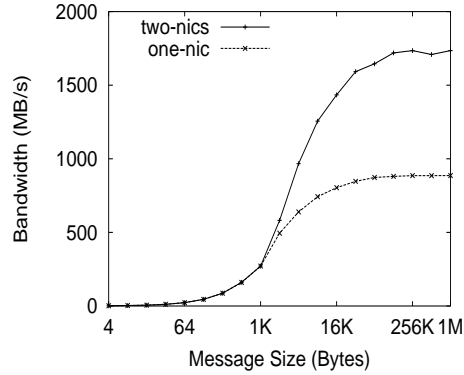


Fig. 3. *MPI_Put* Bandwidth on the IA32 Cluster

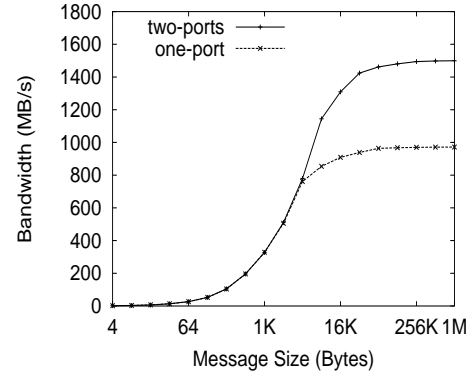


Fig. 4. *MPI_Put* Bandwidth on the EM64T Cluster

We observe a similar trend for dual-port on EM64T in Figure 4. For messages of size greater than 8KBytes, we use fragmentation policy. We can achieve a peak bandwidth of 1500 MB/s using multi-rail design, in comparison to 971 MB/s for the original implementation capable of using only one-port of a nic.

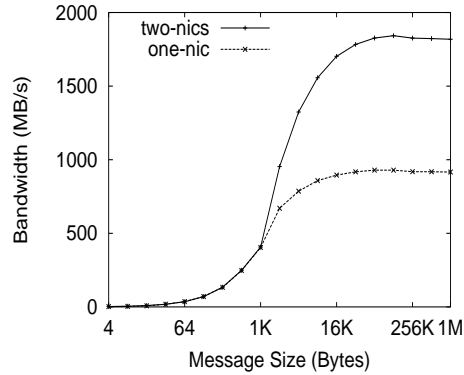


Fig. 5. *MPI_Put* Bidirectional Bandwidth on the IA32 Cluster

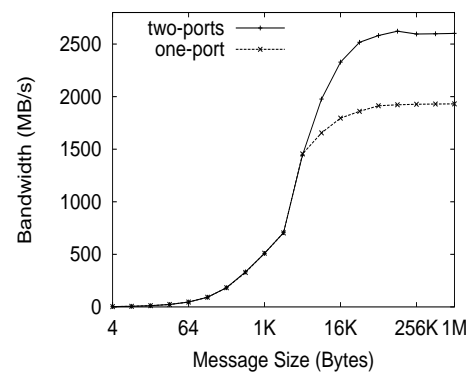


Fig. 6. *MPI_Put* Bidirectional Bandwidth on the EM64T Cluster

In figures 5 and 6, we compare the performance of *MPI_Put* bidirectional bandwidth for IA32 cluster and EM64T cluster, respectively. For IA32 cluster, due to the bottleneck of PCI-X, we can achieve only 941 MB/s for original implementation. However, using multi-rail design we can achieve a peak bidirectional bandwidth of 1810 MB/s. For EM64T cluster, we can achieve a peak bidirectional bandwidth of 2620 MB/s with two-ports in comparison to 1910 MB/s using the original implementation.

In figures 7 and 8, we present the results for *MPI_Get* latency for IA32 and EM64T cluster, respectively. We observe that we perform almost similar with the original implementation for small messages. For large messages, we can improve the latency by 45% for IA32 cluster and 33% for EM64T cluster by using multi-rail design.

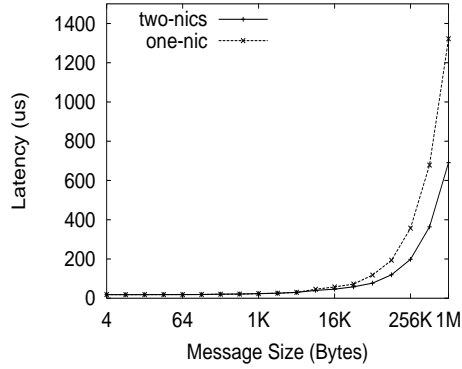


Fig. 7. *MPI_Get* Latency on the IA32 Cluster

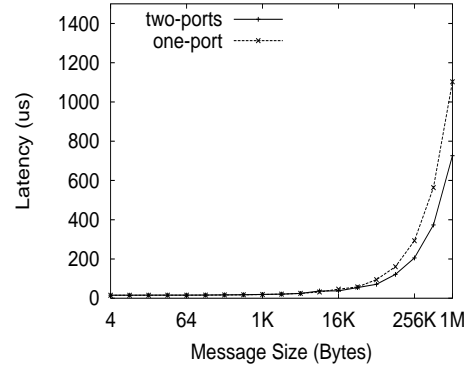


Fig. 8. *MPI_Get* Latency on the EM64T Cluster

Impact of Reordering on One Sided Communication Figure 9 shows the performance achieved by a combination of policies at the CH3' layer and Multi-rail layer. At the multi-rail layer we use load balanced fragmentation policy. At the CH3' layer, we compare impact of reordering with no reordering, when combined with the multi-rail policy specified above.

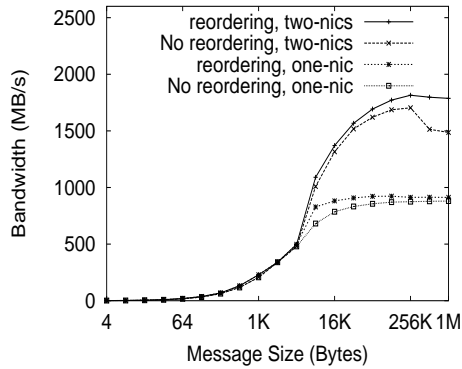


Fig. 9. Interleaved throughput on the IA32 Cluster

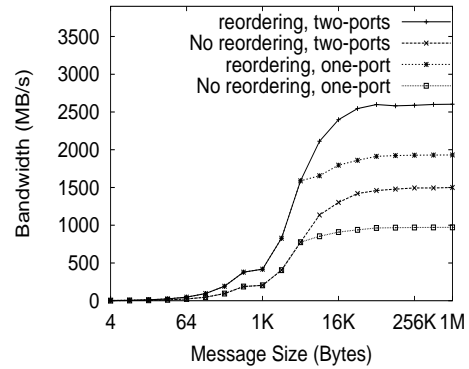


Fig. 10. Interleaved throughput on the EM64T Cluster

For IA32 cluster using two-nics, we can achieve almost 1703 MB/s without reordering, which is close to the multi-rail peak unidirectional bandwidth. With single-rail implementation, we can achieve a peak bandwidth of 880 MB/s without reordering. We notice that with reordering for two-nics, we can almost achieve 1800 MB/s, almost the peak bidirectional bandwidth with two-nics. With single-rail implementation, due to the limitation of PCI-X, we can achieve only 907 MB/s.

In figure 10, we evaluate the performance of CH3' layer reordering, compared to the no reordering policy for the EM64T cluster. We use the load balanced fragmentation at the multi-rail layer. Using two-ports and reordering, we can achieve 2604 MB/s, which is almost the peak bidirectional bandwidth available with two-ports. It is interesting to notice, that reordering with single-rail implementation outperforms the combination of no reordering with multi-rail implementation. We attribute it to the fact that, PCI-Express can achieve 8X bidirectional bandwidth with one-port. However, due to the contention at the NIC, we cannot achieve a combined 8X unidirectional bandwidth using two-ports. Using reordering with single-rail implementation, we can achieve 1900 MB/s. However we can only achieve a peak bandwidth of 1474 MB/s using multi-rail implementation with no reordering. With no reordering for single-rail implementation, we can achieve 962 MB/s, which is close to the unidirectional bandwidth available with the single-rail implementation.

5 Related Work

In this section we discuss related work on one-sided communication model as well as multi rail networks. In [15], reordering of one sided operations is proposed to reduce the cost of lock synchronization operation. Besides MPI, some other programming models which provide one-sided communication are ARMCI [14], GASNET [2] and BSP [5]. Using interconnection networks for different topologies has been studied in [4]. Using multirail networks to build high performance clusters is proposed in [3].

However, none of the above works have focussed on design of MPI-2 one-sided communication operations with multirail InfiniBand clusters.

6 Conclusions and Future Work

In this paper, we have presented the challenges (*Multiple synchronization messages, handling multiple HCAs, scheduling policies, ordering relaxation*) associated with designing MPI-2 one-sided communication over multirail Infiniband networks. We have implemented our design and presented the performance evaluation for microbenchmarks. We have observed that multirail InfiniBand clusters can significantly improve the performance for one-sided communication. Using a two rail cluster, we have achieved almost doubled the throughput and reduced the latency to half with *MPI_Put* and *MPI_Get* operations for large messages. We have also observed that reordering policy can significantly improve the performance for communication patterns with a mix of one-sided operations.

In future, we plan to evaluate our implementation on large scale clusters for applications with one-sided communication. We also plan to evaluate the scheduling policies in depth, to take care of different communication patterns for one-sided communication.

7 Software Distribution

As indicated earlier, the open-source MVAPICH2 [12] software is currently being used by more than 250 organizations world-wide. The latest release is 0.6.5. The proposed MPI-2 multirail one-sided communication solution will be available in the 0.7.0 release.

References

1. Argonne National Laboratory. MPICH2. <http://www-unix.mcs.anl.gov/mpi/mpich2/>.
2. D. Bonachea. GASNet Specification, v1.1. Technical Report UCB/CSD-02-1207, Computer Science Division, University of California at Berkeley, October 2002.
3. S. Coll, E. Frachtenberg, F. Petrini, A. Hoisie, and L. Gurvits. Using multirail networks in high-performance clusters. In *CLUSTER '01: Proceedings of the 3rd IEEE International Conference on Cluster Computing*, page 15, Washington, DC, USA, 2001. IEEE Computer Society.
4. J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*. The IEEE Computer Society Press, 1997.
5. M. Goudreau, K. Lang, S. B. Rao, T. Suel, and T. Tsantilas. Portable and Efficient Parallel Computing Using the BSP Model. *IEEE Transactions on Computers*, pages 670–689, 1999.
6. W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. *Parallel Computing*, 22(6):789–828, 1996.
7. W. Huang, G. Santhanaraman, H.-W. Jin, and D. K. Panda. Scheduling of MPI-2 One Sided Operations On InfiniBand. In *Int'l Parallel and Distributed Processing Symposium (IPDPS '05)*.
8. InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.0, October 24 2000.
9. W. Jiang, J. Liu, H. W. Jin, D. K. Panda, D. Buntinas, R. Thakur, and W. Gropp. Efficient Implementation of MPI-2 Passive One-Sided Communication on InfiniBand Clusters. *EuroPVM/MPI*, September 2004.
10. W. Jiang, J. Liu, H.-W. Jin, D. K. Panda, W. Gropp, and R. Thakur. High Performance MPI-2 One-Sided Communication over InfiniBand. *International Symposium on Cluster Computing and the Grid (CCGrid 04)*, April 2004.
11. J. Liu, A. Vishnu, and D. K. Panda. Building multirail infiniband clusters: Mpi-level design and performance evaluation. In *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing*, page 33, Washington, DC, USA, 2004. IEEE Computer Society.
12. J. Liu, J. Wu, S. P. Kini, D. Buntinas, W. Yu, B. Chandrasekaran, R. Noronha, P. Wyckoff, and D. K. Panda. MPI over InfiniBand: Early Experiences. Technical Report, OSU-CISRC-10/02-TR25, Computer and Information Science, the Ohio State University, January 2003.
13. Network-Based Computing Laboratory. MVAPICH: MPI for InfiniBand on VAPI Layer. <http://nowlab.cse.ohio-state.edu/projects/mpi-iba/index.html>, January 2003.
14. J. Nieplocha and B. Carpenter. ARMCI: A Portable Remote Memory Copy Library for Distributed Array Libraries and Compiler Run-Time Systems. *Lecture Notes in Computer Science*, 1586, 1999.
15. R. Thakur, W. Gropp, and B. Toonen. Minimizing Synchronization Overhead in the Implementation of MPI One-Sided Communication. In *EuroPVM/MPI*, September 2004.