High Performance MPI on IBM 12x InfiniBand Architecture

ABHINAV VISHNU, BRAD BENTON AND DHABALESWAR K. PANDA

Technical Report
OSU-CISRC-5/06-TR57

High Performance MPI on IBM 12x InfiniBand Architecture *

Abhinav Vishnu, Brad Benton†, and Dhabaleswar K. Panda

Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{vishnu, panda}@cse.ohio-state.edu

† IBM Austin
11501 Burnet Road
Austin, TX 78758
{brad.benton}@us.ibm.com

Abstract. InfiniBand is becoming increasingly popular in the area of cluster computing due to its open standard and high performance. I/O interfaces like PCI-Express and GX+ are being introduced as next generation technologies to drive InfiniBand with very high throughput. InfiniBand HCAs which support these I/O interfaces are being introduced. HCAs with throughput of 8x on PCI-Express have become available. Recently, support for HCAs with 12x throughput on GX+ has been announced. In this paper, we design an MPI (Message Passing Interface) on IBM 12x Dual-Port HCAs consisting of multiple send/recv engines per port. We propose and study the impact of various communication scheduling policies (binding, striping, round robin). Based on this study, we present a policy, EPC (Enhanced point-to-point and collective), which takes into account the kind of communication pattern; point-to-point blocking, non-blocking, and collective communication for data transfer. We implement our design and evaluate it with micro-benchmarks, collective communication and NAS parallel benchmarks. Our performance results show that 12x HCAs can significantly improve MPI communication performance. Using EPC on a 12x InfiniBand cluster with one HCA and one port, we can improve the performance by 41% for blocking communication with latency test and 63-65% for non-blocking communication, with the unidirectional and bi-directional tests, when compared with the default single-rail MPI implementation. We can achieve a peak unidirectional bandwidth of 2745 MB/s and bidirectional bandwidth of 5362 MB/s. Our evaluation on NAS Parallel Benchmarks shows an improvement of 7-13% in execution time along with a signification improvement in collective communication using Pallas benchmark suite. To the best of our knowledge, this is the first such design and evaluation of high performance MPI for IBM 12x InfiniBand HCA.

1 Introduction

InfiniBand Architecture [3] is an industry standard which offers low latency and high bandwidth, as well as many advanced features such as Remote Direct Memory Access (RDMA), multicast and QoS (Quality of Service). I/O interfaces like PCI-Express and GX+ are being introduced as next generation technologies to drive InfiniBand with very high throughput. InfiniBand HCAs which support these I/O interfaces are being introduced. InfiniBand HCAs with throughput of 8x on PCI-Express have become available.

* This collaborative research was initiated during Abhinav Vishnu's internship at IBM. The OSU part of the research is supported in part by Department of Energy's grant #DE-FC02-01ER25506 and National Science Foundation's grants #CNS-0403342 and #CNS-0509452; and equipment donations from IBM and Cisco Systems

Recently, support for HCAs with 12x link speed on GX+ has been announced. In this paper, we focus on IBM 12x HCAs with GX+ interface. Each IBM 12x HCA port consists of multiple send and receive DMA engines providing a aggregate link bandwidth of 12x in each direction. This leads to the following challenges:

1. *How to design efficient support at the MPI level for taking advantage of multiple send and receive engines at the HCA?*
2. *What are the trade offs in such designs?*
3. *How much performance benefits can be achieved at the MPI level for point-to-point, collective communication and applications with the proposed design?*

In this paper, we take on these challenges. We propose a unified MPI design for taking advantage of multiple send and receive engines on a port, multiple ports and HCAs. We study the impact of various communication scheduling policies *(binding, striping and round robin)* and discuss the limitations of these individual policies for different communication patterns, in context of IBM 12x InfiniBand HCA. To overcome this limitation, we present a policy, EPC (*Enhanced point-to-point and collective*), which takes into account *point-to-point blocking, non-blocking, collective communication* for data transfer. To enable this differentiation, we design a *communication marker* and discuss the need to integrate it with the ADI layer to obtain optimal performance.

We implement our design and evaluate it with micro-benchmarks, collective communication and NAS parallel benchmarks. Using EPC on a 12x InfiniBand cluster with one HCA and one port, we can improve the performance by 41% for blocking communication with ping-pong latency test and 63-65% for non-blocking communication, with the unidirectional and bi-directional throughput tests, when compared with the default single-rail MPI implementation [5]. We can achieve a peak unidirectional bandwidth of 2745 MB/s and bidirectional bandwidth of 5362 MB/s. We conclude that none of the previously proposed policies alone provide optimal performance in these communication patterns. Using NAS Parallel Benchmarks, we see an improvement of 7-13% in execution time along with a signification improvement in collective communication using pallas benchmark suite. To the best of our knowledge, this is the first such design and evaluation of high performance MPI for IBM 12x InfiniBand HCA.

The remaining part of the paper is organized as follows: In Section 2, we provide a brief overview of InfiniBand, MPI and IBM 12x InfiniBand HCA Architecture. In Section 3, we present the MPI design for IBM 12x architecture. Performance evaluations and discussion are presented in Section 4. In section 5, we present the related work. We present conclusions and discuss future directions in Section 6.

2 Background

In this section, we provide background information for our work. We provide a brief introduction of InfiniBand and IBM 12x InfiniBand HCAs.

2.1 InfiniBand

The InfiniBand Architecture (IBA) [3] defines a switched network fabric for interconnecting 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 Channel Adapters (CA).

HCA sits on processing node. Communication operations are described in Work Queue Requests (WQR), or descriptors, and submitted to the work queue. The completion of WQRs is reported through Completion Queues (CQs). InfiniBand supports different classes of transport services. In this paper, we focus on the Reliable Connection (RC) service. InfiniBand Architecture supports both channel and memory semantics. In channel semantics, send/receive operations are used for communication.

2.2 Overview of IBM 12x Dual-Port InfiniBand HCA

Each IBM 12x HCA comprises of two ports. The local I/O interconnect used is GX+, which can run over different clock rates of 633 MHz-950 MHz. Each 12x HCA port has multiple send and receive DMA engines. The aggregate link bandwidth of the send DMA engines and receive DMA engines is 12x in each direction, respectively. To schedule data on a DMA engine, the hardware send scheduler looks at the send queues of queue pairs, which are not being serviced currently. Given equal priority, the queue pairs are serviced in a round robin fashion. Hence, multiple queue pairs are needed to extract maximum parallelism from the 12x InfiniBand HCA. In this paper, we focus on the design to exploit maximum parallelism in this architecture, by using multiple queue pairs and proposing optimal scheduling policies for different communication patterns.

3 MPI Design for IBM 12x InfiniBand Architecture

In this section, we focus on designing an MPI substrate for IBM 12x InfiniBand Architecture. We begin with the introduction of overall design, which is followed by discussion on scheduling policies. We also present communication marker module, which resides in the ADI layer and differentiates between communication patterns.

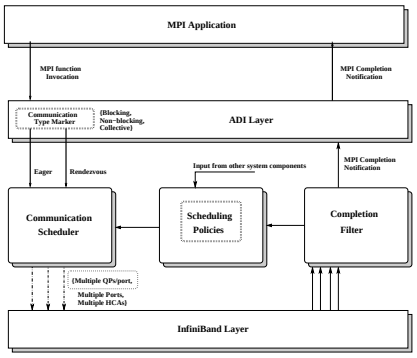

Fig. 1. Overall MPI Design for IBM 12x InfiniBand Architecture

3.1 Overall Design

The overall design is shown in Figure 1. Our previous design presented in [4, 8] supports using multiple ports and multiple HCAs. In our new design presented here, we enhance it by adding support for *multiple QPs per port*. In addition, in our enhanced design, we present a *communication marker* schedule, which differentiates between communication patterns, to obtain optimal performance for point-to-point and collective communication. These enhancements are shown with dotted boxes in Figure 1. In future, we plan to extend this design for other communication patterns like stencil communication.

3.2 Discussion of Scheduling Policies for different Communication Patterns

Point to Point Communication Point to point communication can be classified as *blocking* and *non-blocking* type of communication. In blocking communication, only one outstanding message is present. Round Robin policy uses the available QPs one-by-one in a round robin fashion. Using round robin policy may lead to under-utilization of the available send and receive DMA engines. *Striping* divides the messages amongst available queue pairs providing a much better utilization of available DMA engines. Similarly, for *non-blocking* communication, *striping* can provide benefits by exploiting parallelism in send and receive DMA engines. However, a large percentage of MPI applications perform mainly employ medium size messages for data transfer. In our previous work [4, 8], our performance evaluation comprised of mostly for two queue pairs, hence the impact of assembly-disassembly on the performance of medium size messages is not clearly reflected. However, as the number of queue pairs increase, the cost of assembly and disassembly due to striping may become significant. This cost is mainly due to posting descriptors for each stripe, and acknowledgment overhead of the reliable connection transport service of InfiniBand. It seems that *reverse multiplexing* or round robin is the only solution to solve this problem. However, blocking communication allows only one outstanding message during the communication. Hence, we need to employ different scheduling policies for *blocking* and *non-blocking* kind of communication.

Collective Communication Collective communication primitives based on point-to-point typically employ MPI_Sendrecv primitive for various steps in the algorithm. Each of the MPI_Sendrecv calls can further be dissected in one function call of MPI_Isend and MPI_Irecv each. As described in the previous section, this is a non-blocking form of communication, and round robin policy would be used. However, only one outstanding non-blocking call is available for each send/receive engine , which may lead to insufficient usage of available send DMA engines. Thus, we clearly need to differentiate amongst the non-blocking calls received from point-to-point communication and collective communication.

From the above discussion, we can conclude that a single scheduling policy is not sufficient for data transfer with different patterns. Hence, we present a policy, EPC (Enhanced point-to-point and collective), which falls back to optimal policies for respective communication patterns. In essence, for non-blocking communication, it uses *round robin*, for blocking communication, it uses *striping*. For collective communication, even though we have non-blocking calls, it falls back to *striping*. The efficiency of this policy is dependent upon the ADI layer to be able to differentiate between such communication patterns. Next, we present such a module, called *communication marker* module, which resides in the ADI layer and takes advantage of ADI layer data structures and parameters for differentiating amongst communication patterns.

3.3 Communication Marker

The communication marker module resides in the ADI layer of our design. The main purpose of this module is to be able to differentiate amongst different communication patterns invoked by the MPI Application. In essence, it differentiates between:

- Point-to-point
  - Blocking

  - Non-blocking
- Collective

Since our design is based on MPICH, this differentiation at ADI layer is possible. In particular for collective communication, a separate tag is used, which can be used to differentiate an ADI function call from point-to-point communication. In addition, the ADI layer decides the communication protocol eager/rendezvous depending upon the message size. We have used a rendezvous threshold of 16KBytes in performance evaluation.

4 Performance Evaluation

In this section, we present performance evaluation of IBM 12x HCAs using MPI Benchmarks. We have used MVAPICH [5], a high performance MPI implementation as the framework for implementation of our new design. We compare the performance of our enhanced design with MVAPICH-0.9.7 release version. The 1QP/port case refers to single-rail version of MVAPICH. Since MVAPICH-0.9.7 multi-rail version also supports only 1 QP/port, the results obtained on our experimental testbed are similar for MVAPICH-0.9.7, single-rail and multi-rail devices. The solution will be available in an integrated manner in MVAPICH-0.9.8 We show the performance results for simple micro-benchmarks, latency, bandwidth and bi-directional bandwidth followed by collective communication. This is followed by performance evaluation on NAS Parallel Benchmarks.

4.1 Experimental Testbed

Our experimental testbed consists of IBM Power5 InfiniBand cluster. The cluster is connected using IBM 12x Dual-Port HCAs. Each node in the cluster comprises 16 processors, shared L2 and L3 caches along-with 32 GB DDR2 533MHz main memory. Each node has multiple GX+ slots, which run at a speed of 950 MHz and CPU speed of 2.4 GHz. We have used 2.6.16 linux kernel and InfiniBand drivers from OpenIB-Gen2, revision 6713.

4.2 Performance Evaluation with Micro-Benchmarks and Collective Communication

In Figure 2, we present the results of pingpong latency, a form of blocking communication, for different configurations of our design. We can conclude that our design exhibits insignificant overhead with increasing number of queue pairs. Figure 3 shows the results for large message latency and compares the performance of EPC with existing policies. We notice that EPC performs equally well as even striping policy, proposed in our previous work [4, 8]. Binding and round robin policy do not improve the latency compared to 1 QP/port case, since only one outstanding message is possible in the latency test. Figure 4 and 5 compare the performance of EPC with the round robin policy with 1QP/port and 2QPs/port. Using round robin and EPC, we are able to almost double the throughput for uni-directional bandwidth test. An improvement of 40% is also noticed for bi-directional test. We notice that increasing number of queue pairs from 2-4 does not help bi-directional bandwidth test, since the bandwidth has been saturated. Figures 6 and 7, show the performance of uni-directional and bi-directional bandwidth tests for large messages. We compare the performance of EPC with the originally proposed even striping [4, 8]. Using both policies we are able to achieve, 2745 MB/s and 5263 MB/s results respectively for the above tests in comparison to 1661 MB/s and

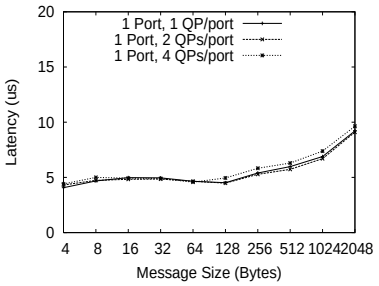

Fig. 2. MPI Latency For Small Messages

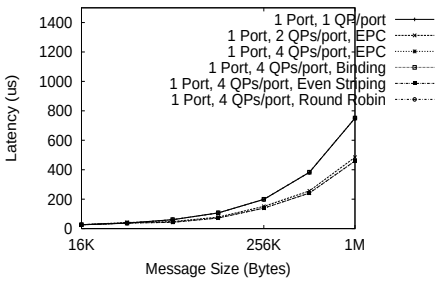

Fig. 3. MPI Latency For Large Messages

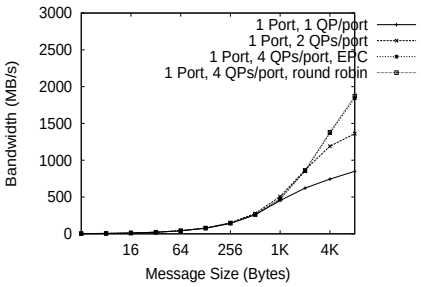

Fig. 4. Impact of Scheduling Policies on Small Message Uni-directional Bandwidth

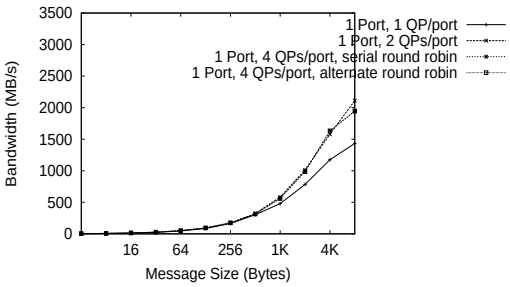

Fig. 5. Impact of Scheduling Policies on Small Message Bi-directional Bandwidth

3079 MB/s using single-rail implementation. However, even striping performs much worse than EPC for medium size messages. This can be attributed to the fact that dividing the data into multiple chunks leads to inefficient use of send engines, as they do not have enough data to pipeline, posting of descriptors for each send engine and receipt of multiple acknowledgments. For very large messages, the data transfer time is reasonably high, and as a result, the performance graphs converge.

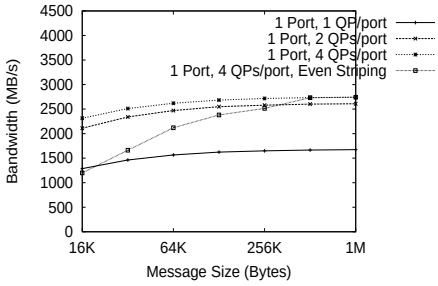

Fig. 6. Large Message Uni-directional Bandwidth

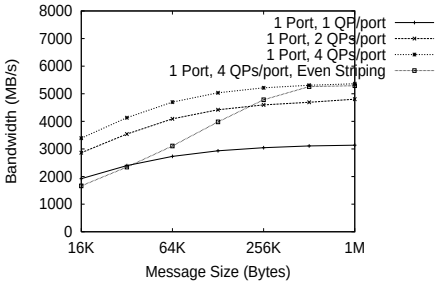

Fig. 7. Large Message Bi-directional Bandwidth

Figures 9 and 8 show the performance of MPI_Bcast and MPI_Alltoall using our enhanced design. We use 2x4 configuration for performance evaluation, where two nodes and four processes per node are used for communication. For MPI_Bcast, the performance does not improve up to very large messages. However, for MPI_Alltoall, even

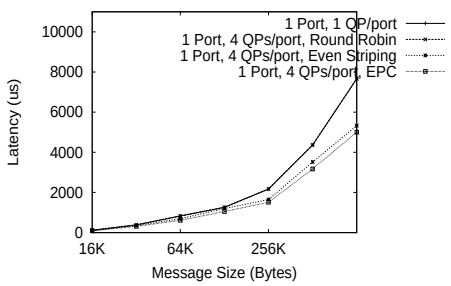

Fig. 8. Alltoall, Pallas Benchmark Suite, 2x4 Configuration

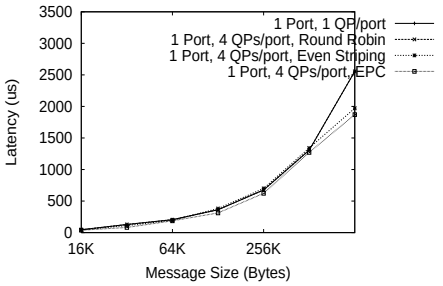

Fig. 9. Bcast, Pallas Benchmark Suite, 2x4 Configuration

for medium range of messages, we can see an improvement, due to efficient utilization of available send and receive DMA engines in comparison to single-rail implementation. Hence, differentiation at the ADI layer between non-blocking communication and collective communication significantly helps the performance of collective operations.

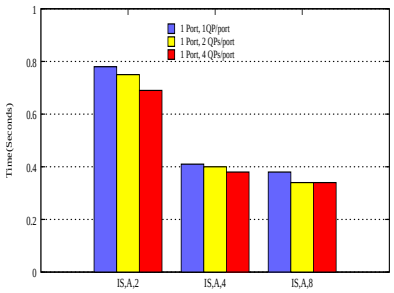

Fig. 10. Integer Sort, NAS Parallel Benchmarks, Class A

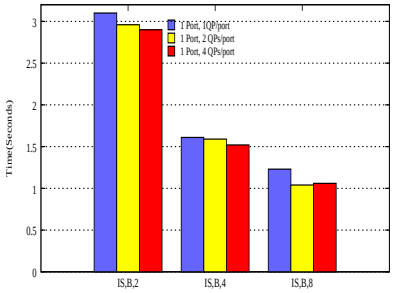

Fig. 11. Integer Sort, NAS Parallel Benchmarks, Class B

4.3 Performance Evaluation with NAS Parallel Benchmarks

Figures 10 and 11 show the results for Integer Sort, Class A and Class B respectively. We compare the performance for 2 (2x1), 4 (2x2) and 8 (2x4) processes respectively. Using two processes on Class A and B, the execution time improves by 13% and 9% respectively with 4 QPs/port. We use only EPC policy for comparison, since it performs equal or better than previously proposed policies, as shown by results from micro-benchmarks. For 4 processes, the execution time improves by 8% and 7% respectively. Since we use shared-memory communication for processes on same node, the percentage of network communication decreases with increasing number of processes and the performance benefits follow a similar trend. Figures 12 and 13 show the results for Fourier Transform, Class A and Class B, respectively. We see around 5%-7% improvement with increasing number of processes.

5 Related Work

Studies on the performance of high performance interconnects including InfiniBand, Myrinet, Quadrics, and 10 gigabit ethernet have been carried out in the literature [1, **?**].

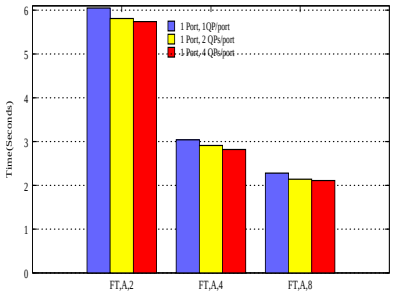

Fig. 12. Fourier Transform, NAS Parallel Benchmarks, Class A

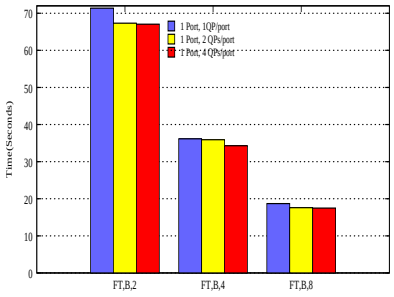

Fig. 13. Fourier Transform, NAS Parallel Benchmarks, Class B

We have also conducted performance evaluation of multirail configurations at the MPI level for InfiniBand [4, 8]. In this paper, we focus on the interaction between InfiniBand Architecture ,local I/O bus technologies and the number of send and receive engines in an HCA. OpenMPI [2] is a high performance MPI implementation capable of supporting InfiniBand, myrinet and TCP based devices. It allows striping across different interconnects. VMI2 [7] is a messaging layer developed by researchers at NCSA. An MPI implementation over VMI2, which runs over multiple interconnects like InfiniBand, Myrinet and Ethernet. LA-MPI [6] is an MPI implementation developed at Los Alamos National Labs. LA-MPI was designed with the ability to stripe message across several network paths. However, none of the above works have focussed on designing high performance MPI substrate over IBM 12x InfiniBand HCAs and exploiting the capability of multiple send/recv engine architecture.

6 Conclusions and Future Work

In this paper we have designed an MPI for IBM 12x InfiniBand architecture comprising of multiple send/recv DMA engines. We have studied the impact of various communication scheduling policies *(binding, striping, and round robin)*, and presented a policy, EPC (*Enhanced point-to-point and collective*), which takes into account *point-to-point blocking, non-blocking, and collective communication* for data transfer. We have discussed the need to strongly integrate our design with the ADI layer to obtain optimal performance. We have implemented our design and evaluated it with micro-benchmarks, collective communication and NAS parallel benchmarks. Our performance results show that 12x HCAs can significantly improve MPI communication performance. Using EPC on a 12x InfiniBand cluster with one HCA and one port, we can improve the performance by 41% for blocking communication with latency test and 63-65% for non-blocking communication, with the unidirectional and bi-directional throughput tests, when compared with the default single-rail MPI implementation. We can achieve a peak unidirectional bandwidth of 2745 MB/s and bidirectional bandwidth of 5362 MB/s. We have concluded that none of the previously proposed policies alone provide optimal performance in these communication patterns. Using NAS Parallel Benchmarks, we see an improvement of 7-13% in execution time along with a signification improvement in collective communication using Pallas benchmark suite. To the best of our knowledge, this is the first such design and evaluation of high performance MPI for IBM 12x InfiniBand HCA. In future, we plan to study the impact of these policies on other communication types like stencil communication, along with scalability issues for large scale clusters for different MPI Applications.

6.1 Acknowledgment
We would like to thank Allison White (IBM Poughkeepsie) for providing us cluster access with IBM 12x InfiniBand HCAs. We would also like to thank David Craddoc (IBM Poughkeepsie) and Chet Mehta (IBM Austin) for technical suggestions and constant support in this work.

References

1. L. Chai, R. Noronha, P. Gupta, G. Brown, and D. K. Panda. Designing a Portable MPI-2 over Modern Interconnects Using uDAPL Interface. In *EuroPVM/MPI*, 2005.
2. Edgar Gabriel and Graham E. Fagg and George Bosilca and Thara Angskun and Jack Dongarra and Jeffrey M. Squyres and Vishal Sahay and Prabhanjan Kambadur and Brian Barrett and Andrew Lumsdaine and Ralph H. Castain and David J. Daniel and Richard L. Graham and Timothy S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *EuroPVM/MPI*, pages 97–104, 2004.
3. InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.2, October 2004.
4. J. Liu, A. Vishnu, and D. K. Panda. Building Multirail InfiniBand Clusters: MPI-Level Design and Performance Evaluation. In *SuperComputing Conference*, 2004.
5. Network-Based Computing Laboratory. MVAPICH/MVAPICH2: MPI-1/MPI-2 for InfiniBand on VAPI/Gen2 Layer. http://nowlab.cse.ohio-state.edu/projects/mpi-iba/index.html.
6. Richard L. Graham and Sung-Eun Choi and David J. Daniel and Nehal N. Desai and Ronald G. Minnich and Craig E. Rasmussen and L. Dean Risinger and Mitchel W. Sukalski. A Network-Failure-Tolerant Message-Passing System for Terascale Clusters. volume 31, pages 285–303, Norwell, MA, USA, 2003. Kluwer Academic Publishers.
7. Scott Pakin and Avneesh Pant. VMI 2.0: A Dynamically Reconfigurable Messaging Layer for Availability, Usability, and Management. In *The 8th International Symposium on High Performance Computer Architecture (HPCA-8), Workshop on Novel Uses of System Area Networks (SAN-1)*, Cambridge, Massachusetts, February 2, 2002. Available from http://www.csl.cornell.edu/SAN-1/san1.pdf.
8. A. Vishnu, G. Santhanaraman, W. Huang, H.-W. Jin, and D. K. Panda. Supporting MPI-2 One Sided Communication on Multi-Rail InfiniBand Clusters: Design Challenges and Performance Benefits. In *International Conference on High Performance Computing, HiPC*, 2005.