High Performance MPI on IBM 12x InfiniBand Architecture *

Abhinav Vishnu Brad Benton† Dhabaleswar K. Panda

Network Based Computing Lab
Department of Computer Science and Engineering
The Ohio State University
{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. 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 a Message Passing Interface (MPI) on IBM 12x Dual-Port HCAs, which consist of multiple send/recv engines per port. We propose and study the impact of various communication scheduling policies (binding, striping and round robin). Based on this study, we present a new policy, EPC (Enhanced point-to-point and collective), which incorporates different kinds of communication patterns; 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. Using EPC on a 12x InfiniBand cluster with one HCA and one port, we can improve the performance by 41% with ping-pong latency test and 63-65% with the unidirectional and bi-directional bandwidth tests, compared with the default single-rail MPI implementation. Our evaluation on NAS Parallel Benchmarks shows an improvement of 7-13% in execution time for Integer Sort and Fourier Transform.

*This collaborative research was initiated during Abhinav Vishnu's internship at IBM. The OSU part of the research is supported in part by DOE grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; NSF grants #CNS-0403342 and #CNS-0509452; and equipment donations from IBM

1-4244-0910-1/07/$20.00 ©2007 IEEE.

1 Introduction

InfiniBand Architecture [4] 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 Quality of Service (QoS). I/O interfaces like PCI-Express and GX+ are being introduced as next generation technologies to drive InfiniBand with very high throughput. InfiniBand Adapters (also known as Host Channel Adapters) with throughput of 8x on PCI-Express have become available. 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 an 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 address 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 new policy, EPC (*Enhanced point-to-point and collective*), which incorporates different kinds of

communication patterns *point-to-point (blocking, non-blocking) and 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 for obtaining the 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% with ping-pong latency test and 63-65% with the unidirectional and bi-directional throughput tests, when compared with the default single-rail MPI implementation [7]. 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 provides optimal performance for 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.

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

2 Background

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

2.1 InfiniBand

The InfiniBand Architecture (IBA) [4] defines a switched network fabric for interconnecting processing nodes and I/O nodes. An InfiniBand network consists of switches, adapters (called Host Channel Adapters or HCAs) and links for communication. For communication, InfiniBand supports different classes of transport services (Reliable Connection, Unreliable Connection, Reliable Datagram and Unreliable Datagram). In this paper, we focus on the reliable connection model. In this model, each process-pair creates a unique entity for communication, called *queue pair*. Each queue pair consists of two queues; *send queue* and *receive queue*. The requests to send the data to the peer are placed on the send queue, by using a mechanism called *descriptor*. A descriptor describes the information necessary for a executing an operation. For RDMA (Remote Direct Memory Access) operation, it specifies the local buffer, address of the peer buffer and access rights for manipulation of remote buffer. InfiniBand also provides a mechanism, where different queue pairs can share their receive queues, called *shared receive queue* mechanism. The completions of descriptors are posted on a queue called *completion queue*. This mechanism allows a sender to know the status of the data transfer operation. Different mechanisms for notification are also supported (polling and asynchronous).

2.2 Overview of IBM 12x Dual-Port InfiniBand HCA

Each IBM 12x HCA consists of two ports. The local I/O interconnect used is GX+, which can run over different clock rates of 633 MHz-950 MHz. Figure 1 shows the block diagram of the IBM 12x InfiniBand HCA. In this paper, we use GX+ bus with 950MHz frequency. As a result, a theoritical bandwidth of 7.6GB/s can be provided. However, each port can provide an aggregate theoritical bandwidth of 12x (3GB/s). Each port has multiple send and receive DMA engines. However, the peak bandwidth of each send/recv engine varies with different implementations. The objective is to provide a uniform MPI for all these implementations.

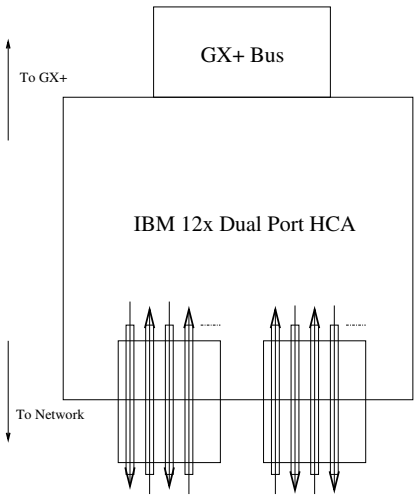

Figure 1. IBM 12x InfiniBand HCA Block Diagram

To schedule the data on a send engine, the hardware send scheduler looks at the send queues of different

queue pairs with send descriptors, which are not being serviced currently. Given equal priority, the queue pairs are serviced in a round robin fashion. As a result, multiple queue pairs should be used to utilize the send engines efficiently. In addition, efficient scheduling of data on these engines is also imperative to efficient utilization.

In the next section, we present an MPI functionality, which is able to take advantage of the presence of multiple send/recv engines in an efficient manner.

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 a communication marker module, which resides in the ADI layer and differentiates between communication patterns.

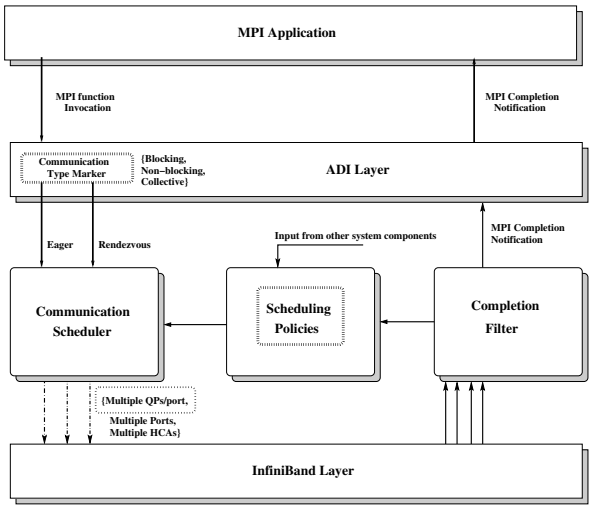

Figure 2. Overall MPI Design for IBM 12x InfiniBand Architecture

3.1 Overall Design

Figure 2 represents our overall design. Our previous design presented in [6, 9] supports using multiple ports and multiple HCAs. In our new design presented here, we enhance it by adding support for utilizing multiple send/recv engines per port. As mentioned in the section 2, multiple queue pairs per port are needed for such a support. 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 2.

3.2 Discussion of Scheduling Policies for different Communication Patterns

In this section, we present the discussion on scheduling policies. Even though, in our previous work, we have presented an initial discussion on scheduling policies, we discuss the limitations of the previously proposed scheduling policies for utilizing multiple send/recv engines in an efficient manner. We begin with a discussion on point-to-point communication.

3.2.1 Point-to-Point Communication

Point-to-point communication can be classified as *blocking* and *non-blocking* type of communication. In the blocking communication, only one message is outstanding in communication between a pair of processes. The round robin policy uses the available QPs one-by-one in a circular fashion [6, 9]. *Binding* policy allows a user to bind to a port of an HCA [6]. Using round robin policy may lead to under-utilization of the available send and receive DMA engines for such kind of communication pattern. *Striping* divides the messages among 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 mainly use medium size messages for data transfer. In our previous work [6, 9], our design and evaluation mostly used two queue pairs (one queue pair per port or adapter), hence the impact of striping on the performance of medium size messages was negligible. However, using multiple send/recv engines per port requires usage of multiple queue pairs per port. As the number of queue pairs increase, the cost of assembly and disassembly of data due to striping becomes significant. This cost is mainly due to posting a descriptor for each stripe, and acknowledgment overhead of the reliable connection transport service of InfiniBand. Hence, using the round robin policy for communication may outperform the striping policy.

From the above discussion, the need to differentiate between point-to-point communication patterns is clear. We incorporate this using a *communication marker* presented in the later part of the section.

3.2.2 Collective Communication

Collective communication primitives based on point-to-point use MPI_Sendrecv primitive for various steps in the algorithm. Each MPI_Sendrecv call can further be divided in one function call of MPI_Isend and MPI_Irecv each to the left and the right neighbor. Since MPI Collectives are blocking, each step in the algorithm is completed before executing the next step. 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 between 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 communication patterns. Some policies benefit blocking communication, while other benefit the non-blocking communication. In addition, for the non-blocking communication, the usage by collective communication can further complicate the scheduling policy decision. To resolve the above conflicts of policy selection, we present a new policy, EPC (Enhanced point-to-point and collective), which falls back to optimal policies for respective communication patterns. 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. In the next section, called *communication marker* module, which resides in the ADI layer and takes advantage of ADI layer data structures and parameters for differentiating between 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 among 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 the ADI layer is possible. 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. This value is also used as the *striping threshold*, the messages of size equal and above are striped on all available queue pairs equally.

4 Performance Evaluation

In this section, we present performance evaluation of IBM 12x HCAs using MPI Benchmarks. We have used MVAPICH [7], a high performance MPI implementation as the framework for implementing our new design. We compare the performance of our enhanced design with MVAPICH release version (referred to as original from here on). The 1QP/port case is referred to as the original version of MVAPICH. 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 [1].

4.1 Experimental Testbed

Our experimental testbed consists of an InfiniBand cluster with IBM nodes built with Power6 processor. The cluster is connected using IBM 12x Dual-Port HCAs. Each node in the cluster comprises 4 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. For our experimentation, we have used only one GX+ bus, one HCA and one port of an HCA. The objective is to evaluate the performance of multiple send/recv engines on one HCA. However, the experimentation can definitely be extended to usage of multiple ports, HCAs and combinations. In future, we plan to evaluate such combinations.

4.2 Overview of Experiments

In this section, we present a brief overview of the experiments. For micro-benchmarks, we have used latency, uni-directional and bi-directional bandwidth tests. The tests are written using two processes.

- **Latency Test:** This test performs communication between the processes in a ping-pong fashion. The

objective is to calculate the performance during the steady state. Hence, we ignore the first couple of iterations. The test is run for 1000 iterations, and the first 10 are not measured.

- **Uni-directional Bandwidth Test:** This test performs ping-ping type of communication. The sender issues a *window* of messages using MPI_Isend and waits for the acknowledgement from the receiver. The receiver waits for *window* receives to finish and sends the acknowledgement. This step is repeated for 20 iterations, the first couple of iterations are ignored as a warm-up phase. Since the idea is to observe the potential capacity of the network, we use a large window size, which is 64 in this case.

- **Bi-directional Bandwidth Test:** This test performs exchange form of communication. Both processes issue 64 MPI_Isend messages to each other after posting 64 MPI_Irecvs. The messages from the peer are used as acknowledgements, and hence no explicit acknowledgement is used. We use the warmup phase and the same number of iterations as the unidirectional bandwidth test.

  The source code which we have used for these tests is available publicly [7].

4.3 Performance Evaluation with Micro-Benchmarks and Collective Communication

In Figure 3, we present the results for the latency test. For small messages, it is not beneficial to stripe the message across multiple queue pairs as the startup time is dominant. Hence, even with increasing number of queue pairs, we use only one of the QPs for communication. The objective is to see the performance degradation from our design compared to the original case. From the figure, it is clear that our design adds negligible overhead compared to the original case.

Figure 4 shows the results for large message latency, comparing a set of parameters; scheduling policy and number of queue pairs used per port. The objective is to understand the efficiency of the communication marker for differentiating between communication types. Hence we compare the performance of EPC and policies proposed in the previous work. We notice that using 4QPs/port, EPC and striping perform comparably. Both binding and round robin are not able to take advantage of multiple queue pairs, since they use only one queue pair for an MPI message. An improvement of 33% is observed using EPC and striping.

Figure 5 compares the performance of EPC and round robin policy with the original case. Since mes-

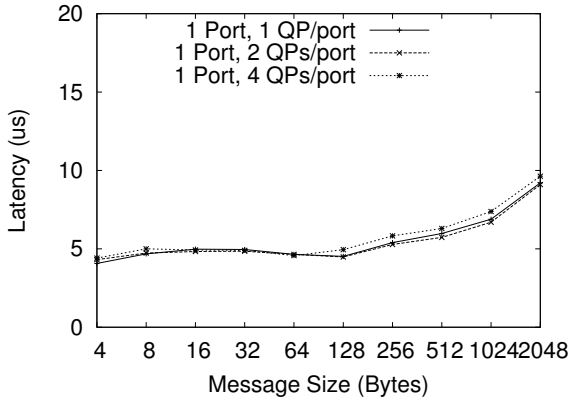

Figure 3. MPI Latency For Small Messages

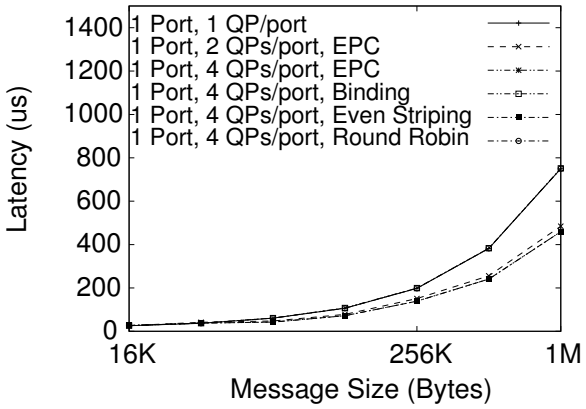

Figure 4. MPI Latency For Large Messages

sages are still on a smaller range, we do not use striping. Comparing 2QPs with 4QPs, we observe that performance gains are observed after 1KBytes. For very small messages (less thank 1KBytes), the startup time limits the usage of multiple queue pairs efficiently. However, from 1KBytes-8KBytes message range, as the transfer time increases, 4QPs show improvement in performance. The performance is similar to the round robin policy.

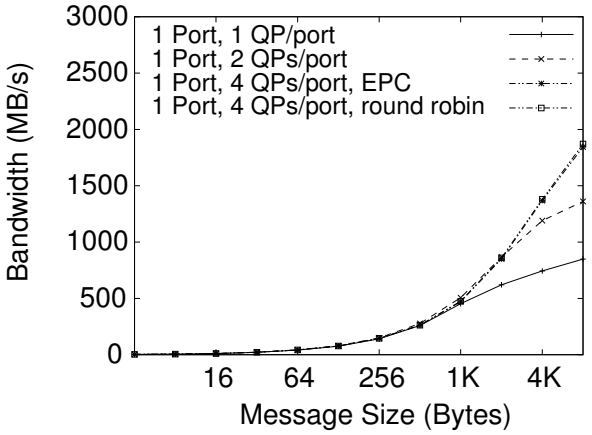

Figure 5. Impact of Scheduling Policies on Small Message Uni-directional Bandwidth

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 [6, 9]. 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 3079 MB/s using the original implementation. However, even striping performs much worse than EPC for medium size messages (16K - 64K). 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.

Figures8 show the performance of 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. However, for MPI_Alltoall, even 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, differentia-

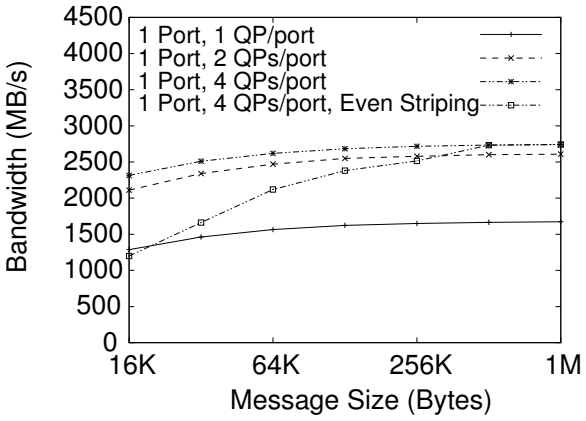

Figure 6. Large Message Uni-directional Bandwidth

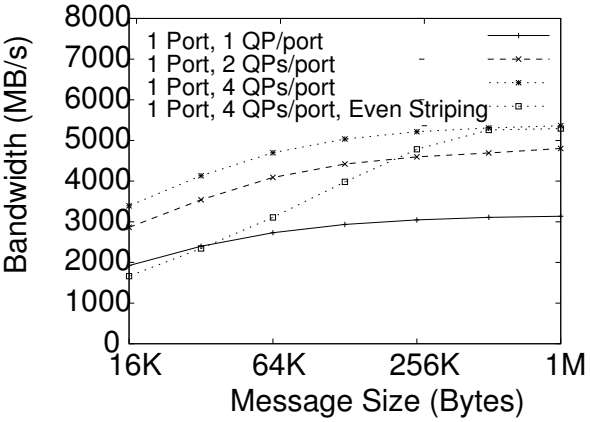

Figure 7. Large Message Bi-directional Bandwidth

tion at the ADI layer between non-blocking communication and collective communication significantly helps the performance of collective operations.

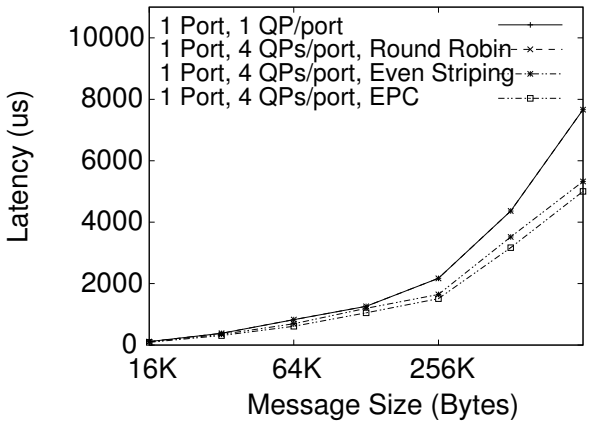

Figure 8. Alltoall, Pallas Benchmark Suite, 2x4 Configuration

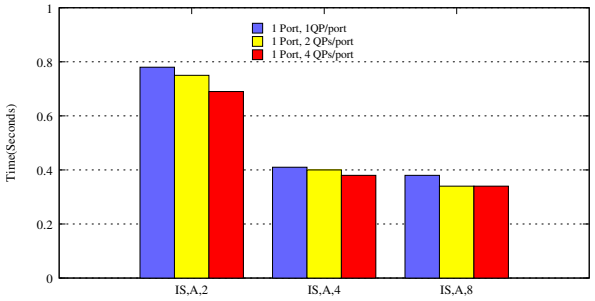

Figure 9. Integer Sort, NAS Parallel Benchmarks, Class A

4.4 Performance Evaluation with NAS Parallel Benchmarks

Figures 9 and 10 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 the same node, the percentage of network communication decreases with increasing number of processes and the performance benefits follow a similar trend. However, we do not see any performance degradation using our

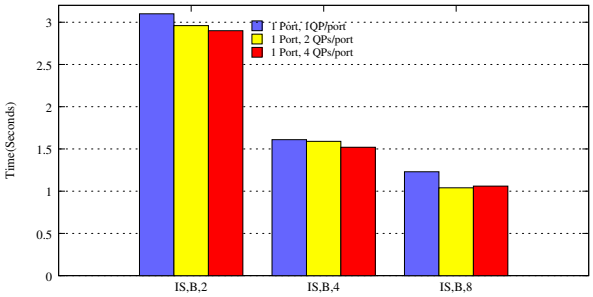

Figure 10. Integer Sort, NAS Parallel Benchmarks, Class B

enhanced design. Figures 11 and 12 show the results for Fourier Transform, Class A and Class B, respectively. We see around 5-7% improvement with increasing number of processes. Although, not included in the paper, we have not seen performance degaradation using other NAS Parallel Benchmarks.

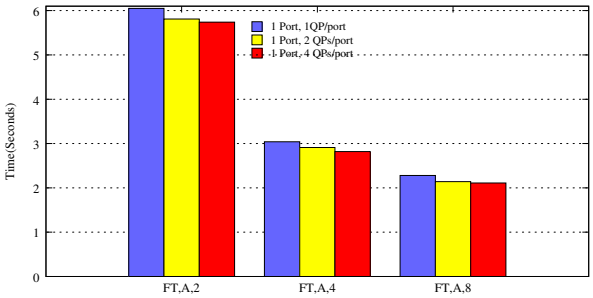

Figure 11. Fourier Transform, NAS Parallel Benchmarks, Class A

5 Related Work

Studies on the performance of high performance interconnects including InfiniBand, Myrinet and Quadrics have been carried out in the literature [5]. We have also conducted performance evaluation of multirail configurations at the MPI level for InfiniBand [6, 9]. 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 [8] is a messaging layer developed by the researchers at NCSA. An MPI implementation over VMI2, which runs over multiple interconnects like InfiniBand, Myrinet and Ethernet. LA-MPI [3] is an MPI implementation developed at Los Alamos Na-

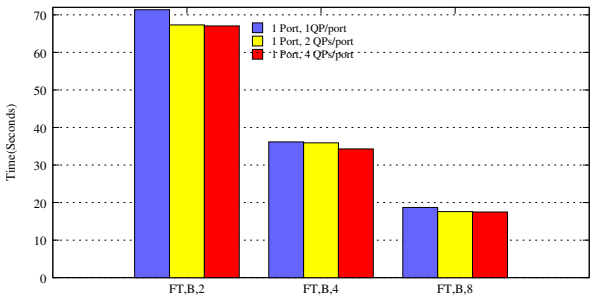

Figure 12. Fourier Transform, NAS Parallel Benchmarks, Class B

tional 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 new policy, EPC (*Enhanced point-to-point and collective*), which incorporates different kinds of communication patterns; *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% with ping-pong latency test and 63-65% with the unidirectional and bi-directional throughput tests compared with the default single-rail MPI implementation. 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. 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.

References

[1] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. Number 3, pages 63–73, Fall 1991.

[2] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *EuroPVM/MPI*, pages 97–104, 2004.

[3] R. L. Graham, S.-E. Choi, D. J. Daniel, N. N. Desai, R. G. Minnich, C. E. Rasmussen, L. D. Risinger, and M. W. Sukalski. A Network-Failure-Tolerant Message-Passing System for Terascale Clusters. volume 31, pages 285–303, Norwell, MA, USA, 2003. Kluwer Academic Publishers.

[4] InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.2, October 2004.

[5] J. Liu and B. Chandrasekaran and W. Yu and J. Wu and D. Buntinas, S. P. Kinis, P. Wyckoff, and D. K. Pand. Micro-Benchmark Level Performance Comparison of High-Speed Cl uster Interconnects. In *Hot Interconnect 11*, August 2003.

[6] J. Liu, A. Vishnu, and D. K. Panda. Building Multirail InfiniBand Clusters: MPI-Level Design and Performance Evaluation. In *SuperComputing Conference*, 2004.

[7] 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.

[8] S. Pakin and A. 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, Feb. 2, 2002.

[9] 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.