High Performance MPI over iWARP: Early Experiences *

S. Narravula A. Mamidala A. Vishnu G. Santhanaraman D. K. Panda

Department of Computer Science and Engineering
The Ohio State University
{narravul, mamidala, vishnu, santhana, panda}@cse.ohio-state.edu

Abstract

Modern interconnects and corresponding high performance MPIs have been feeding the surge in the popularity of compute clusters and computing applications. Recently with the introduction of the iWARP (Internet Wide Area RDMA Protocol) standard, RDMA and zero-copy data transfer capabilities have been introduced and standardized for Ethernet networks. While traditional Ethernet networks had largely been limited to the traditional kernel based TCP/IP stacks and hence their limitations, iWARP capabilities of the newer GigE and 10 GigE adapters have broken this barrier and thereby exposing the available potential performance.

In order to enable applications to harness the performance benefits of iWARP and to study the quantitative extent of such improvements, we present MPI-iWARP, a high performance MPI implementation over the Open Fabrics verbs. Our preliminary results with Chelsio T3B adapters show an improvement of up to 37% in bandwidth, 75% in latency and 80% in MPI allreduce as compared to MPICH2 over TCP/IP. To the best of our knowledge, this is the first design, implementation and evaluation of a high performance MPI over the iWARP standard.

1. Introduction

Compute clusters have become increasingly popular due to the high performance to cost ratios they offer. Rapid technology advances at largely affordable costs have led to the wide spread deployment of these clusters, with several organizations having multiple cluster deployments. TCP/IP has been the most popular protocol for all inter-node communication requirement. While TCP/IP based communication has its advantages in being the most popular protocol and supporting both across LAN and WAN communication, CPU and memory related costs for driving traditional TCP/IP stacks often impact communication performance and hence limit the scalability and efficiency of the clusters [22, 7].

To improve communication performance within clusters, modern interconnects like 10 Gigabit Ethernet, Quadrics [6], Myrinet [21] and InfiniBand [2] offer higher network bandwidths and lower communication latencies. In addition, interconnects like InfiniBand include features such as Remote Direct Memory Access (RDMA) that have enabled communication mechanisms providing a significantly higher performance. To extend the benefits of RDMA to the traditional Ethernet-based networks, a new Internet Wide Area RDMA Protocol (iWARP) standard has been recently introduced [9]. The iWARP standard basically allows for zero-copy transfer of data over the legacy TCP/IP communication stacks. Hence iWARP provides for a significantly higher communication performance. Applications need to leverage this available communication performance into better overall application performance. Additionally, the iWARP protocol being based on TCP/IP also allows high performance data transfers across WANs enabling users to run high performance applications in cluster-of-clusters scenarios. Currently several vendors including Chelsio [8], NetEffect [13] and Ammasso provide iWARP capable RNICs.

On the other hand, Message Passing Interface (MPI) has become the *de facto* standard in writing parallel computing applications. The benefits of the MPI standard lie in its ability to abstract the differences in the underlying networks from the applications and hence making the job of application writers easier. As expected, the communication performance observed by the applications is directly dependant on the performance of the underlying MPI library implementation on a high performance interconnect.

*This research is supported in part by DOE grants #DE-FC02-06ER25749 and #DE-FC02-06ER25755; NSF grants #CNS-0403342, #CNS-0509452 and #CCF-0702675; grants from Intel, Mellanox, Cisco systems, Linux Networx and Sun Microsystems; and equipment donations from Intel, Mellanox, AMD, Apple, Appro, Chelsio, Dell, Fujitsu, Fulcrum, Microway, PathScale, IBM, SilverStorm and Sun Microsystems.

Application writers can utilize available highly optimized MPI libraries like MVAPICH [4], MPICH-GM over Myrinet [21] and MPI/ELAN4 over Quadrics [6]. Typically current deployments of Ethernet networks utilize MPI over TCP/IP for their needs. Clearly, availability of a high performance MPI library utilizing iWARP is of critical importance to enable applications programmers to exploit the benefits offered by the modern 10 Gigabit Ethernet adapters supporting the iWARP protocol.

In order to enable easy and efficient development of higher level libraries and applications across the various modern RDMA enabled interconnects, a new initiative of having a single unified lower level software stack has been started under the Open Fabrics Alliance (OFA) [5]. Recently the OFA's software stack has incorporated the support for RDMA capable Ethernet adapters (iWARP enabled adapters or RNICs).

In our paper, we study the different MPI choices available for Ethernet networks. In particular we present MPI-iWARP, a high performance MPI over iWARP as a means to overcome the performance limitations observed in the traditional TCP/IP stack based software. Our design and implementation is based on the high performance MPI-2 implementation MVAPICH2 [4]. The current MVAPICH2 implementation supports OFA verbs. While OFA's software verbs are identical from the MPI library's perspective for both InfiniBand and iWARP-RNICS, certain issues need addressing for enabling MVAPICH2 over iWARP. In particular, the connection management requirements for iWARP are different. We have studied such requirements in this paper.

Our preliminary performance numbers show a 75% improvement in MPI level latency and a 37% improvement in MPI level bandwidth over the traditional MPICH2 over TCP/IP [20]. Several of Intel MPI benchmarks also show performance benefits of above 80% for the MPI-iWARP design.

Rest of this paper is organized as follows: Section 2 briefly describes the required background on iWARP and MVAPICH2. Section 3 details the design and implementation of MPI-iWARP. The experimental evaluations are presented in Section 4. The related work is discussed in Section 5 and lastly, the conclusions in Section 6.

2. Background

In this section, we briefly describe the required background information in iWARP and MVAPICH2.

2.1. iWARP: Internet Wide Area RDMA Protocol

The iWARP protocol defines RDMA operations over Ethernet networks [9]. As specified in [9], for iWARP, the basic message transport is undertaken by the TCP layer. Since TCP itself is a stream protocol and does not respect message boundaries, an additional MPA layer is introduced to enforce this. The actual zero-copy capability is enabled by the Direct Data Placement (DDP) Layer. The RDMA features provided by DDP are exported to the upper level protocol by the RDMAP layer. It is to be noted that the ordering of data within a single data-transfer is not guaranteed by the specifications of these layers. However, adapters often do guarantee this as an option. The iWARP protocol comprising of RDMAP, DDP, MPA and TCP layers are intended to be implemented in hardware for RNICs resulting in significant performance improvements over the traditional TCP/IP stacks.

The iWARP supports two types of communication semantics: Channel Semantics (Send-Receive communication model) and Memory Semantics (RDMA communication model). Remote Direct Memory Access (RDMA) [11] operations (RDMA Read and RDMA Write) allow processes to access the memory of a remote node process without the remote node CPU intervention. These operations are transparent at the remote end since they do not require the remote end to be involved in the communication.

The basic communication is achieved over connected end points known as the Queue Pairs (QPs). Each QP consists of a send queue and a receive queue. To enable data transfers, each QP needs to be setup and its state needs to be transitioned into the *connected* state. The communicating process initiates a data transfer by posting a descriptor. The descriptor typically holds all the required information for the data transfer like source/destination buffer pointers, type of transfer, etc. In case of RDMA operations, the descriptor also contains the remote buffer information.

2.1.1 RDMA-CM

RDMA-CM is an abstraction layer for connection management defined by OFA. It is designed to establish connections between the QPs of a pair of processes identified based on IP addresses and port numbers. The *RDMA-CM* library itself can setup connections over the multiple networks supported by OFA. The main responsibilities of the *RDMA-CM* library include exchanging necessary information and transitioning the QPs through their states into the connected state. It is to be noted that the *RDMA-CM* sets up the connections in a traditional client-server mechanism. The basic steps for a successful connection setup for each pair of QPs is shown

in Figure 1.

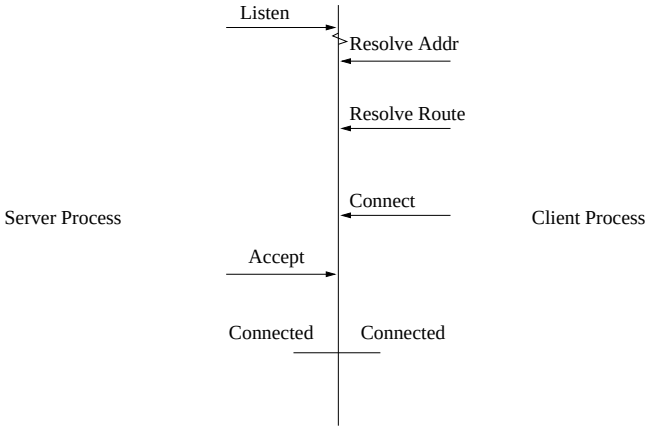

Figure 1. Basic Steps for Connection Setup Using RDMA CM

2.2. MVAPICH2: A High Performance MPI-2 Implementation

MVAPICH2 is a high performance implementation of MPI-2 over InfiniBand. The implementation is based on MPICH2. As a successor of MPICH[10], MPICH2[20] supports MPI-1 as well as MPI-2 extensions including one sided communication.

MVAPICH2 includes several features like: (i) RDMA fast path [18]: utilizing RDMA operations for small message transfers, (ii) RDMA-based one-sided operations [15]: mapping MPI one-sided operations to RDMA Read and RDMA Write for better performance, etc.

Software Distribution: MVAPICH2 is available for use as an open source package. It is currently being used in more than 525 organizations worldwide. The work presented in this paper is available in the latest MVAPICH2 release distribution starting with MVAPICH2 0.9.8. Further details are available in [12, 4].

3. Design and Implementation

In this section we describe the various design and implementation aspects of *MPI-iWARP*. In particular, we describe the required connection management and other iWARP compliance related design details.

We design and implement our MPI over iWARP based on the high performance implementation MVAPICH2, using the iWARP verbs implementation from Open Fabrics Enterprise Distribution (OFED).

3.1. RDMA-CM based Connection Management

Since iWARP protocol is based on TCP/IP, making an iWARP connection needs the iWARP stack to internally make a TCP/IP connection. Hence iWARP connection semantics essentially follow the 3-way handshake semantics of TCP to enable the underlying TCP connection. As detailed earlier in Section 2.1.1, the OFA software distribution provides for an abstraction layer called *RDMA-CM* that performs this connection setup for iWARP (and IB).

The main issue here that needs addressing is the mismatch in the connection semantics of *RDMA-CM* and the MPI processes. The MPI library typically assume a fully connected model (i.e. they should be able to send messages to any of the peers) and hence connections usually have to be setup between all the other MPI peer processes within the process group. On the other hand, in the *RDMA-CM*, the connection is setup in the traditional TCP/IP style, client-server mechanism. Due to this mismatch, all the process pairs need to be separated into *client-server* pairs before any setting up of connections using *RDMA-CM*.

The second issue is that the current connection setup in MVAPICH2 over OFA verbs follows a mechanism that is quite different from the mechanism used by *RDMA-CM*. Figure 2 shows the existing and the proposed *RDMA-CM*-based connection setup mechanism.

We address the first of these issues by using the process rank order. Between every pair of processes, the process with the lower rank takes the role of the server and the process with the higher rank takes the role of the client. The main steps taken to complete the connection setup using *RDMA-CM* are as follows. Firstly, each process identifies the port and IP address to use for its communication needs. After identifying and binding to the local port and address, the processes exchange this information. At this point all client processes initiate connections to the server processes which accept the same and the connections are established by the *RDMA-CM* library. As the final step, the processes synchronize with a barrier to make sure that all the peer processes are ready for communication.

These steps are repeated for the setup of each of the QPs between a pair of processes. It is to be noted that the actual setting of all the QPs in the process group is basically concurrent.

As mentioned earlier, MVAPICH2 supports multi-rail communication and RDMA based direct one-sided communication mechanisms. These mechanisms require additional QPs to be setup between each pair of processes. While all these connections need to be setup, we alleviate some of the possible contention issues by initiating the connect requests in the reverse order of rank. i.e. process *p[i]* issues a connection request to *p[i-1]* in step 1, *p[i-2]* in

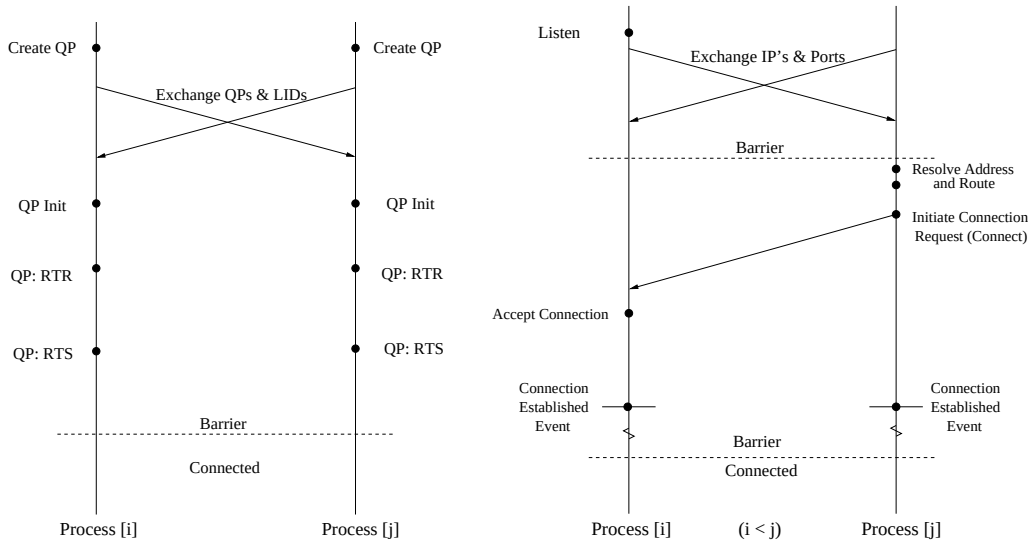

Figure 2. Connection Setup Mechanisms: (a) Existing OFA Gen2 Verbs and (b) RDMA-CM

step 2, and so on.

3.2. Connection Initiation for iWARP

The iWARP MPA specification requires the active side (client processes) to initiate the first data send or the first RDMA write. Due to this restriction, in the case of MPI processes with lower ranks should not initiate the first data transfer. Dealing with this issue during the actual MPI data transfers is non trivial. This is due to the fact that the MPI data transfer request can be initiated from any process depending on the MPI application requirements.

In our design, we make sure that this aspect is handled properly by having an extra dummy message sent from each of the client processes to all its server processes before marking the connection as being "connected". Figure 3 shows the complete *RDMA-CM* based connection setup for MVAPICH2.

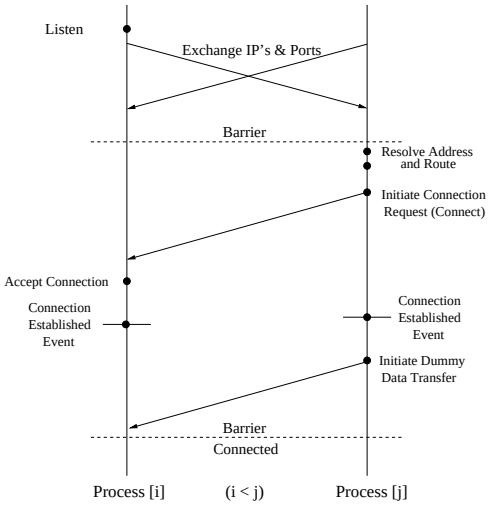

Figure 3. RDMA CM based Connection Setup for MVAPICH2

3.3. RDMA-based Data Transfers

MVAPICH2 supports an accelerated data transfer mechanism for small messages by utilizing RDMA writes. In this mechanism, instead of utilizing the zero copy capabilities of the networks, the messages are copied into the buffers from which they are transfered to remote buffers using RDMA Write operations. The remote side polls on the last bytes of the preassigned receive buffers to check for messages arriving using this path. Since the cost of data copies is low for small messages and RDMA Write provides better latency, this mechanism provides significant latency benefits over the normal Send/Receive based mechanisms.

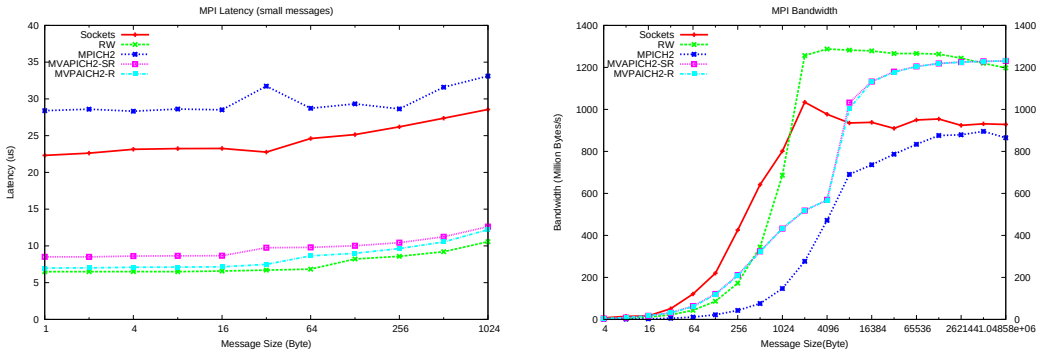

Figure 4. Basic Performance: (a) Latency and (b) Bandwidth

To utilize this technique it is essential that the underlying network places the data into the receive buffer in order. i.e. arrival of the last byte guarantees the arrival of the entire message. While most current generation adapters support this mechanism, this is not guaranteed by the iWARP specification as mentioned earlier. Hence, in our implementation we provide a switch for enabling this mechanism for adapters that do guarantee the in order placement of data.

4. Experimental Results

In this section we present a detailed performance evaluation of MPI-iWARP. We compare our performance numbers with the performance of MPICH2 over TCP/IP sockets as a reference.

For all our experiments we have used a four node cluster equipped with nodes having two quad core Intel Xeon 2.33GHz Nodes and a memory of 4GB each. These systems are equipped with a Chelsio T3B 10 GigE PCI-Express adapters (Firmware Version 4.2) plugged into an x8 PCI-Express slot. The Chelsio adapters are connected through a 24 port Fulcrum 10 GigE evaluation switch [1]. The MTU used for the path is 9000 bytes. The software stack we have used is OFED 1.2 rc4 and the operating systems is RH4 U4. MPICH2 1.0.5p3 is used for comparisons.

In the following subsections, we present the performance numbers for the following: (i) MPICH2: the basic MPI-2 implementation over TCP/IP [14], (ii) MVAPICH2-R: the MPI-iWARP implementation using MVAPICH2's RDMA fast path, (iii) MVAPICH2-SR: the MPI-iWARP implementation without RDMA fast path and (iv) MVAPICH2-1SC: the MPI-iWARP implementation with RDMA based direct one-sided operations enabled. We evaluate MPI-iWARP by measuring the performance of basic MPI latency and bandwidth, MPI one-sided operations - MPI Put and MPI Get, MPI collectives - MPI-Allgather, MPI-Allreduce and MPI-Barrier using IMB [3], followed by NAS parallel benchmarks - IS and CG.

4.1. MPI Send/Recv: Latency and Bandwidth

Figure 4(a) shows the basic latencies that we observe. The latency for the verbs level RDMA write operation over the T3 adapter is about 6.49 microseconds which is quite lower than the basic sockets over TCP/IP number which is about 22.3 microseconds. The corresponding latency for MPICH2 is about 27.84 microseconds and the latencies for MVAPICH2-R and MVAPICH2-SR are 6.89 us and 8.43 us, respectively. As we clearly observe, MVAPICH2-R adds a minimal overhead to the basic RDMA write latency. The difference in the performance of MVAPICH2-R and MVAPICH2-SR is the absence of RDMA fast path in the latter. Further we also note that the latency observed by the MVAPICH2-R is about 75% better than the latency observed by MPICH2. It is to be noted that large messages are bandwidth bound and hence for clarity of presentation we show the latencies of only the small messages.

The peak bandwidth that we observe for our test bed is about 1287 Million Bytes per second (MB/s) using the verbs level RDMA write operations. MPICH2 shows a peak bandwidth of about 895 MB/s out of a maximum bandwidth of 1034 MB/s that the sockets interface offers. The MVAPICH2-R and MVAPICH2-SR implementations both offer a peak bandwidth of about 1231 MB/s. The performance gain that we observe for MPI-iWARP variations over MPICH2 is about 37%. Figure 4(b) shows the basic bandwidth numbers.

4.2. MPI Put: Latency and Bandwidth

In this section, we evaluate the performance of MPI-2's one-sided Put operation. As shown in Figure 5(a), the basic latency that we observe for MPICH2's MPI-Put is about 36.5 microseconds. The performance of the MVAPICH2-R, MVAPICH2-SR and MVAPICH-1SC are 9.32, 10.09 and 9.41 microseconds, respectively. The performance gain observed for the iWARP based approaches is about

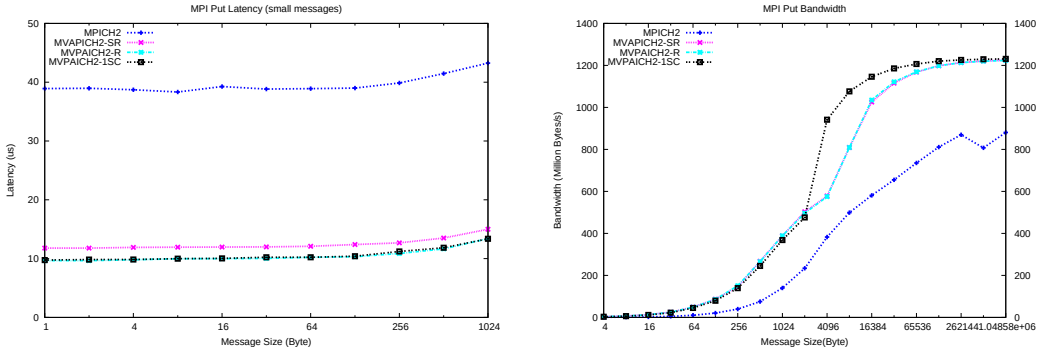

Figure 5. MPI Put Performance: (a) Latency and (b) Bandwidth

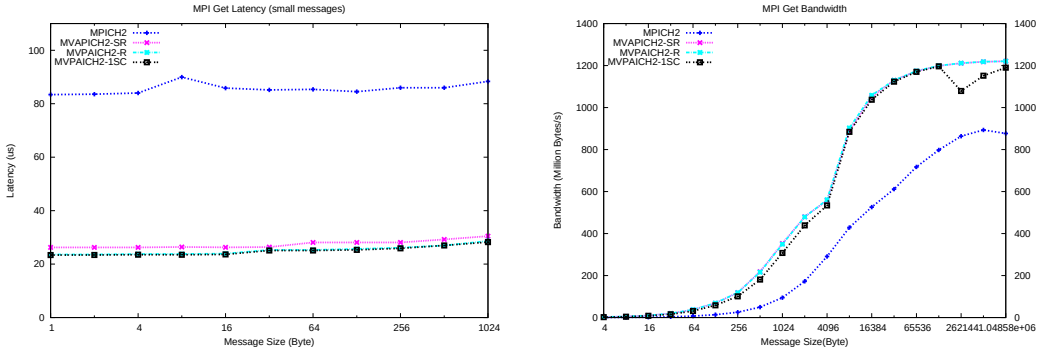

Figure 6. MPI Get Performance: (a) Latency and (b) Bandwidth

74% better as compared to MPICH2 over TCP/IP. The difference in MVAPICH2-R and MVAPICH2-SR in this case is about the same as the difference we observed in the previous subsection. In case of MVAPICH2-1SC, the small messages are internally sent over the basic MPI Send/Recv calls. Hence the latency of MVAPICH2-1SC is about the same as the MVAPICH2-R.

Figure 5(b) shows the performance of MPI Put bandwidth. The peak bandwidth seen using MPICH2 is 880 MB/s where as the MPI-iWARP implementations show a peak bandwidth of about 1231 MB/s, an improvement of about 40%. In this experiment, the MVAPICH2-1SC one-sided implementation which is directly based on RDMA operations, performs the best in all cases. In particular, for message size ranging from 4 KB to 64 KB the MVAPICH2-1SC implementation does significantly better than the MVAPICH2-R and MVAPICH2-SR.

4.3. MPI Get: Latency and Bandwidth

Figure 6(a) shows the latencies of MPI Get operations. MPICH2 shows a latency of 64.21 microseconds, MVAPICH2-R and MVAPICH2-1SC show a latency of 30.03 microseconds each and MVAPICH2-SR shows a latency of 33.95 microseconds.

For MPI Get bandwidth, we observe that the peak observed for MPI-iWARP implementations (1142 MB/s) is about 3.6 times better than the peak observed for MPICH2 (319 MB/s). Figure 6(b) shows the results for this experiment.

4.4. Intel MPI Benchmarks

In this section we compare the performance of MPI-iWARP implementation with that of basic MPICH2 using the Intel MPI benchmark suite [3]. This suite contains benchmarks for evaluating the performance of the various MPI collectives. For these experiments, we have up to 32 processes running on four nodes of our test bed.

Figure 7(a) shows the latency of MPI Allreduce operations. The allreduce latency for MPICH2 is about 264.0 microseconds as compared to 48.47 microseconds for the MVAPICH2-R implementation. Clearly, the MVPAICH2-R performance is over 80% better than that of MPICH2 over TCP/IP. In Figure 7(b) we show the performance of the MPI Allgather operation. The latencies for MPI allgather are 302.48 microseconds and 47.92 microseconds for MPICH2 and MVAPICH2-R, respectively. MVAPICH2-R is about 84% better for MPI Allgather.

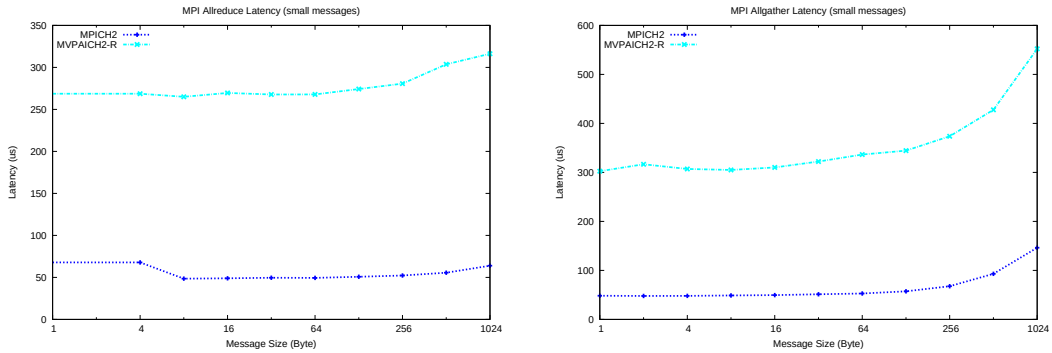

Figure 7. Intel MPI Benchmarks: (a) Allreduce Latency and (b) Allgather Latency

MPI Barrier performance is shown in Figure 8(a). As clearly seen, MPI-iWARP implementation significantly outperforms MPICH2 over TCP/IP. MVAPICH-R performs about 80% better than MPICH2 for a 32 process execution of IMB barrier.

4.5. NAS Parallel Benchmarks

In this section we present the application-level performance benefits of MPI-iWARP for the NAS parallel benchmarks. We run 16 processes on four nodes for these experiments.

Figure 8(b) shows the relative performances of IS and CG running over MVAPICH2-R with MPICH2 taken as the base case. We observe that IS performs about 15% better while running over MVAPICH2-R. Similarly, we show that CG performs about 4% better in case of MVAPICH2-R.

As clearly observed in our experiments, iWARP based MPI implementations provide significant performance gains over the traditional TCP/IP stack based MPI implementations.

5. Related Work

Several researchers have looked at improving application performance over modern networks. In particular, various MPI designs including [4, 16, 6, 21] have all provided high performance MPI implementations over various modern interconnects. While these implementations provide high performance MPI designs on networks other than Ethernet, our current work enables a high performance MPI over Ethernet networks.

Several optimizations like RDMA based data transfers [18], multiple data streams [17], novel collective algorithms [24, 19, 23], etc. all have further improved the performance of applications through the MPI implementations. Our current work provides an MPI implementation over Ethernet networks and can leverage the work done in many of these directions.

6. Conclusions

The growing popularity of compute clusters and the wide spread use of compute applications have gone hand in hand with the technology advances of modern interconnects and corresponding high performance MPIs. With the recent introduction of iWARP standard, RDMA and zero-copy data transfer capabilities have been introduced and standardized for Ethernet networks. While traditional Ethernet networks have largely been limited to the traditional kernel based TCP/IP stacks and their inherent limitations, the iWARP capabilities of the newer GigE and 10 GigE adapters have broken this barrier. These potential benefits that are now provided by the Ethernet networks can be exploited for increased performance gains and need to be leveraged by middleware and applications.

In order to enable applications to harness the performance benefits of iWARP and to study the quantitative extent of such improvements, in this paper we have presented a high performance MPI implementation over the Open Fabrics verbs, MPI-iWARP. Our preliminary results with Chelsio T3B adapters have shown an improvement of up to 37% in bandwidth, 75% in latency and about 80% for a MPI barrier (32 processes) as compared to MPICH2 over TCP/IP.

It is to be noted that the current costs of the iWARP adapters are expected to be higher than the corresponding simple Ethernet adapters. These costs however, are expected to decrease with wider adaptation of iWARP.

MPI-iWARP has been developed based on the MVAPICH2 code base and is now a part of the MVAPICH2 0.9.8 release distribution. As future work we plan to perform further in-depth application level evaluation and scalability studies of the MPI-iWARP design.

References

[1] Fulcrum Microsystems. `http://www.fulcrummicro.com/`.

[2] InfiniBand Trade Association. http://www.infinibandta.com.

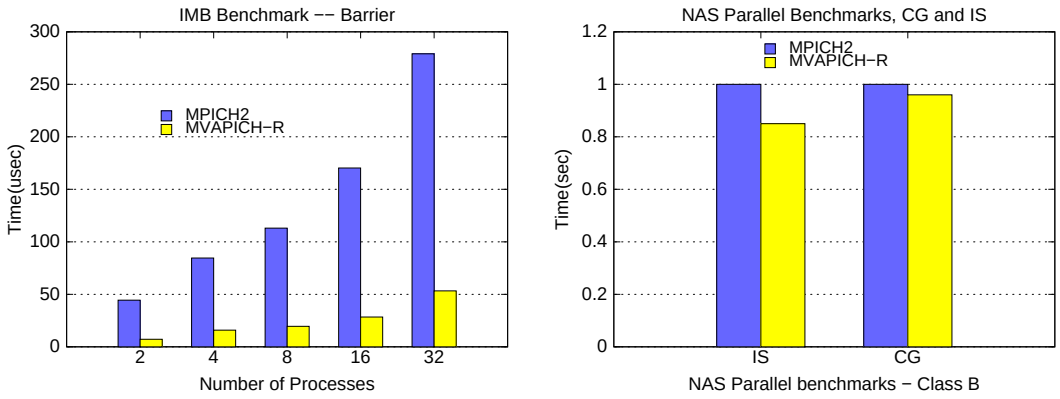

Figure 8. (a) Intel MPI Benchmark: Barrier Latency and (b) NAS Parallel Benchmarks: IS and CG

[3] Intel MPI Benchmarks. `http://www.intel.com/` .

[4] MVAPICH: MPI over InfiniBand and iWARP. http://mvapich.cse.ohio-state.edu/.

[5] Open Fabrics Alliance. `http://www.openib.org/`.

[6] Quadrics Ltd. `http://www.quadrics.com`.

[7] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. Sockets Direct Protocol over InfiniBand in Clusters: Is it Beneficial? Technical Report OSU-CISRC-10/03-TR54, 2003.

[8] Chelsio Communications. http://www.chelsio.com/.

[9] RDMA Consortium. Architectural Specifications for RDMA over TCP/IP. `http://www.rdmaconsortium.org/`.

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI, Message Passing Interface Standard. Technical report, Argonne National Laboratory and Mississippi State University.

[11] J. Hilland, P. Culley, J. Pinkerton, and R. Recio. RDMA Protocol Verbs Specification (Version 1.0). Technical report, RDMA Consortium, April 2003.

[12] Wei Huang, Gopal Santhanaraman, H.-W. Jin, Qi Gao, and Dhabaleswar K. Panda. Design and Implementation of High Performance MVAPICH2:MPI2 over InfiniBand. In *International Symposium on Cluster Computing and the Grid*, 2006.

[13] NetEffect Inc. http://www.neteffect.com/.

[14] J. Liu and W. Jiang and P. Wyckoff and D. K. Panda and D. Ashton and D. Buntinas and B. Gropp and B. Tooney. High Performance Implementation of MPICH2 over InfiniBand with RDMA Support. In *IPDPS*, 2004.

[15] W. Jiang, J. Liu, H. Jin, D. K. Panda, W. Gropp, and R. Thakur. High Performance MPI-2 One-Sided Communication over InfiniBand. In *IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 04)*, Chicago, IL, April 2004.

[16] J. Liu, W. Jiang, P. Wyckoff, D. K. Panda, D. Ashton, D. Buntinas, W. Gropp, and B. Toonen. Design and Implementation of MPICH2 over InfiniBand with RDMA Support. In *Proceedings of Int'l Parallel and Distributed Processing Symposium (IPDPS '04)*, April 2004.

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

[18] J. Liu, J. Wu, S. P. Kini, P. Wyckoff, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. In *17th Annual ACM International Conference on Supercomputing*, June 2003.

[19] A. R. Mamidala, A. Vishnu, and D. K. Panda. Efficient shared memory and rdma based design for mpi-allgather over infiniband. In *EuroPVM/MPI*, 2006.

[20] Mpich2, argonne. http://www-unix.mcs.anl.gov/mpi/mpich2/.

[21] Myricom. Myrinet Software and Customer Support. http://www.myri.com/scs/GM/doc/, 2003.

[22] S. Narravula, P. Balaji, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. Supporting strong cache coherency for active caches in multi-tier data-centers over infiniband. In *SAN-3 held in conjunction with HPCA 2004*, 2004.

[23] S. Sur, H.-W. Jin, and D. K. Panda. Efficient and Scalable All-to-All Exchange for InfiniBand-based Clusters. In *International Conference on Parallel Processing (ICPP)*, 2004.

[24] V. Tipparaju, J. Nieplocha, D.K. Panda. Fast Collective Operations Using Shared and Remote Memory Access Protocols on Clusters. In *International Parallel and Distributed Processing Symposium '03*, April 2003.