Design and Implementation of High Performance MVAPICH2 (MPI2 over InfiniBand)

WEI HUANG, GOPALAKRISHNAN SANTHANARAMAN, HYUN-WOOK JIN
QI GAO AND DHABALESWAR K. PANDA

Technical Report
OSU-CISRC-12/05-TR76

Design and Implementation of High Performance MVAPICH2 (MPI2 over InfiniBand) *

W. Huang G. Santhanaraman H.-W. Jin Q. Gao
D. K. Panda
Department of Computer Science and Engineering
The Ohio State University
Columbus, OH 43210
{*huanwei, santhana, jinhy, gaoq, panda*}*@cse.ohio-state.edu*

Abstract

MPICH2 provides a layered architecture for implementing MPI-2. In this paper, we provide a new design for implementing MPI-2 over InfiniBand by extending the MPICH2 ADI3 layer. Our new design aims to achieve high performance by providing a multi-communication method framework that can utilize appropriate communication channel/device to attain optimal performance without compromising on scalability and portability. We also present the performance comparison of the new design with our previous design MVAPICH2-0.6.5 based on MPICH2 RDMA channel. We show significant performance improvements in micro-benchmarks and up to 24% improvement in case of NAS CG benchmark for 16 processes.

1 Introduction

In the last decade, Message passing interface (MPI) has become the *de facto* standard for writing parallel applications. Next generation of supercomputers built with thousands of nodes are becoming commonplace, which poses challenges for researchers to provide high performance and scalable MPI designs. Further the need for a high performance MPI is growing with the increasing proliferation of clusters.

MPI-2 standard [**?**] has been proposed to extend the functionality of the earlier proposed MPI-1 standard [**?**] in the area of one sided communication, I/O and dynamic process management. MPICH-2 from Argonne National Laboratory is a popular implementation of MPI-2, which aims to combine performance with portability across different interconnects.

Over the past decade there is a strong trend towards RDMA capable networks since they can provide improved performance and scalability. With its RDMA capability and several advanced hardware features, InfiniBand has emerged as a strong player in the area of high performance computing.

In our group we have been working on providing a high performance implementation of MPI-2 over InfiniBand based on MPICH2 design. To scale well for large scale next generation clusters, the design should be able to fully exploit the high performance and the hardware features provided by InfiniBand. It should also be flexible to incorporate multiple communication channels provided by underlying hardware. Our goal is to achieve the optimal performance while not compromising on scalability and portability.

Previously, we implemented MPI2 over the MPICH2 RDMA channel [**?**] due to the ease of porting at this layer. Implementing at such a lower layer can have its disadvantages due to lack of flexibility in implementing novel solutions. In [**?**] we have studied the design alternatives and performance trade-offs associated in implementing at the different MPICH2 layers. We concluded that extending the ADI3 layer and implementing at this layer would provide the best opportunity to achieve better performance.

In this paper we describe our new extended ADI3 level design for MPI-2 over InfiniBand. We provide a framework that attempts to address the issues of performance and scalability mainly in the context of InfiniBand network, yet portable to other programming APIs provided by RDMA capable interconnects. We propose a layered design including an ADI3 extended layer, a multi communication method (MCM) layer which chooses the appropriate communication channel/device that can give the optimal performance, and a device layer which encapsulates all device specific information. We also provide schemes that can improve the scalability. More over, by keeping the device specific information to the lowest device layer, we ensure easy portability across other programming interfaces like OpenIB-Gen2 [**?**] and uDAPL [**?**].

The rest of the paper is organized as follows. In Section 2 we describe the background of our work and also discuss our previous designs and its limitations. We propose our new design in Section 3. In Section 4 we describe the potential benefits of

*This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506; National Science Foundation grants #CNS-0403342 and #CCR-0509452; grants from Intel, Mellanox, Sun, Cisco, and Linux Networx; and equipment donations from Apple, AMD, IBM, Intel, Microway, Pathscale, Silverstorm and Sun.

our new design with respect to performance, portability and scalability. We evaluate the performance in Section 5. In Section 6 we discuss the related work and finally conclusions and future work are discussed in Section 7.

2 Background

2.1 InfiniBand

InfiniBand defines a System Area Network(SAN) to interconnect processing nodes and I/O nodes. In addition to send/receive semantics it also provides RDMA semantics which can be used to directly access/modify the contents of remote memory. RDMA operations are one sided and do not incur software overhead on the remote side. It provides features like scatter/gather, shared receive queues, hardware multicast etc. Because of its high performance, InfiniBand has becoming more and more popular in the area of high performance computing [**?**].

There are two popular user level programming libraries for InfiniBand. VAPI [**?**] is the Mellanox implementation and OpenIB Gen2 [**?**] has recently come out as part of the new generation of IB stack provided by the OpenIB community. InfiniBand also supports User Direct Access Protocol Library (uDAPL), which has emerged as a standard that defines device independent interface for accessing the transport mechanisms of RDMA capable networks, including other interconnects like Myrinet and Quadrics.

2.2 MPICH2

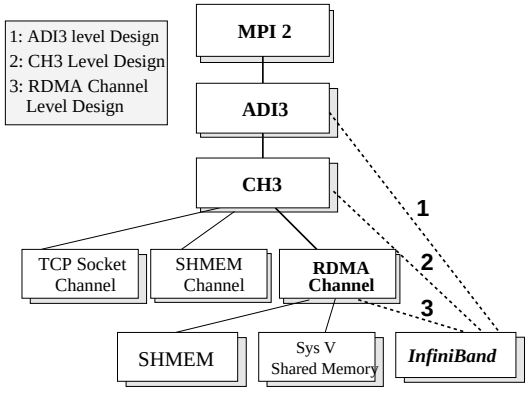

Figure 1. Layered Design of MPICH2

MPICH2 from Argonne National Laboratory is a popular implementation of MPI-2. It aims to combine performance with portability across different interconnects.

MPICH2 follows a layered design which is described in Figure 1. Lines 1, 2, and 3 illustrate the alternatives for implementing MPI2 over InfiniBand.

The ADI3 is a full featured, abstract device interface used in MPICH2. It is the highest layer in the MPICH2 hierarchy. It is responsible for all the point to point and one sided communications. ADI3 completes its tasks through function interfaces provided by the CH3 layer to increase portability.

The CH3 layer provides a *communication channel* that consists of a dozen interfaces. Those interfaces provide basic functionalities for eager, rendezvous communication protocols, as well as auxiliary functions such as process startup, finalize, etc. MPICH2 provides several implementations for CH3, including channels which communicate through shared memory and sockets, etc.

To further increase the portability over RDMA capable networks, MPICH2 also has proposed CH3 implementation based on RDMA channel interfaces. All communication operations that MPICH2 supports are mapped to just five functions at the RDMA channel, thus the RDMA channel provides the least porting overhead.

2.3 MVAPICH2

MVAPICH2 is a high performance implementation of MPI-2 over InfiniBand[**?**] from the Network Based Computing Laboratory at the Ohio State University. MVAPICH2 and MVAPICH (MPI-1 version) are currently being used by more than 280 organizations across the world.

We have earlier designed the MVAPICH2 based on the RDMA channel in MPICH2 [**?**], which is reflected in our earlier public released version MVAPICH2-0.6.5. In this design, we use eager and rendezvous schemes to support the communication interfaces of the RDMA channel. For small messages, we use the eager protocol. It copies messages to pre-registered buffers and sends them through RDMA write operations, which achieves good latency. And for large messages, a zero-copy rendezvous protocol is used, because using pre-registered buffers introduces high copy overhead. The user buffer is registered on the fly and sent directly through RDMA. However, as we have analyzed in our previous work [**?**], a design at RDMA channel has several disadvantages:

- First, there is a known limitation of MPICH2's CH3 implementation based on the RDMA channel. It makes only one outstanding request to the RDMA channel and will not issue the next one until the previous one has completed, which results in serialization of communication requests to the RDMA channel. This adversely affects the throughput. At RDMA channel we can hardly do anything to overcome this problem since we do not have control on the progress engine. The MPICH2 team also has observed this problem and the RDMA channel has been removed from their newest release.

- Second, at this layer many important data structures are hidden and this prevents us from performing many high level optimizations. For example, our earlier work has proposed to use the InfiniBand scatter/gather feature to benefit the non-contiguous datatype communication [**?**]. With this scheme the receiver understands the exact layout of the datatype and posts scatter descriptors, while the sender posts the exactly matching gather descrip-

tors. This Send Gather Receive Scatter (SGRS) avoids the copy overhead and shows great improvement over the packing/unpacking schemes, which is generally used to deal with non-contiguous datatype. However, this scheme requires both the sender and the receiver to understand the total layouts of the datatype, which is actually hidden from the RDMA channel. Thus such functionality cannot be implemented at the RDMA channel only.

To overcome these limitations, we propose ADI3 level design, which is reflected in the latest MVAPICH2 release version 0.9.0[**?**].

3 Design and Implementation

In this section we present the new design of MVAPICH2. We will elaborate the potential advantages of our design in Section 4. The main objectives we are trying to achieve through this new design are:

- Modern clusters provide multiple low level methods for communication, such as intra-node communication through file system shared memory, and different programming libraries for various interconnects, etc. Our new design should be able to take advantage of the available communication methods on a cluster to achieve the best performance.
- For portability reasons, it is desirable to have a concise device abstraction for each of these communication methods. And this abstraction should be well designed so that we can have enough information to perform most of the hardware-specific optimizations and enable the MPI library to achieve maximum performance and scalability.

As illustrated in Figure 2, we follow the basic idea of the layered structure of MPICH2. We start from an extension of the ADI3 layer (ADI3-Ex) in MPICH2. And below that we have our own design of the multi-communication method (MCM) layer and the device layer. The device layer provides abstractions of the communication methods on the cluster, which can be either an intra-node communication device or an inter-node communication device. And the MCM layer aims to exploit the performance benefits provided by these abstract devices.

The following subsections describe the design details for all these three layers.

3.1 ADI3-Ex Layer

The ADI3-Ex layer, which extends from the ADI3 layer in MPICH2, is the highest level in our design. We inherit the other MPI functionality from MPICH2 implementation.

Similar to the ADI3 layer, the main responsibilities of our layer include selecting appropriate internal communication protocol, eager or rendezvous, for each point to point or one sided operation from the user application. Our extensions for

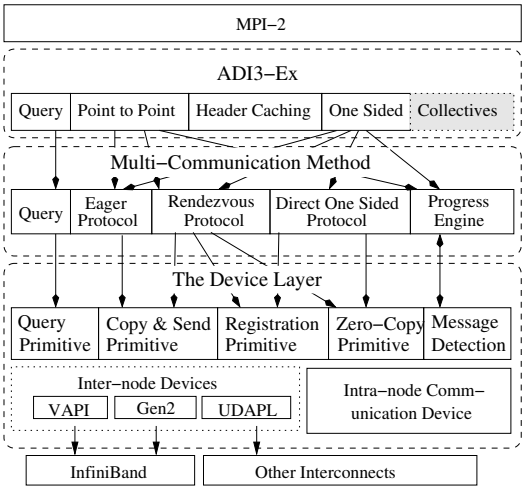

Figure 2. Overall new design of MVAPICH2

the ADI3-Ex layer are mainly for high level optimizations along the following fronts:

- Query: Instead of choosing the rendezvous or eager protocols solely based on the message size, as most of the current MPI implementations do, the ADI3-Ex layer makes decisions based on the preference of the MCM layer. Because the MCM layer dynamically chooses from the available communication devices to send out the message and the optimal point to switch from the eager to rendezvous protocol might be different for each device, this query process helps the ADI3-Ex layer to select the most efficient communication protocol for a specific message.
- Point to Point operations: Point to point communication can take advantage of header caching. The Header caching feature caches the content of internal MPI headers at the receiver side. As a result, if the next message to the receiver contains the same cached fields in header, we reduce the message size, thus reducing the small message communication latency. As discussed in [**?**], header caching can only be put at this layer since only ADI3-Ex layer is able to understand the content of the message.
- One Sided Operations: In our previous work we have extended ADI3 layer to directly implement one sided operation based on InfiniBand RDMA operations to achieve higher performance and less CPU utilization [**?**]. This piece of work is also incorporated into the added functionality of the ADI3-Ex layer based on the direct-one-sided interface provided by the MCM layer.
- Collectives: Our current implementation of this new design supports point to point and one sided communication. Collective operations in MPICH2 are implemented based on point to point communication, thus can take advantage of our design. The framework we propose can

also be extended to incorporate optimized algorithms for collectives which directly utilize the hardware capabilities of InfiniBand, such as hardware multicast [**?**], etc.

3.2 The Multi-communication Method Layer

The multi-communication method (MCM) layer implements the communication protocols selected by the ADI3-Ex layer using the communication primitives supported by the device layer. It understands the performance features of each communication device provided at the device layer and chooses the most suitable one to complete the communication at runtime. Right now our implementation supports one inter-node and one intra-node communication devices simultaneously, but in future we plan to extend this framework to support multiple communication devices. Figure 3 illustrates the interfaces and the basic functional modules of this layer.

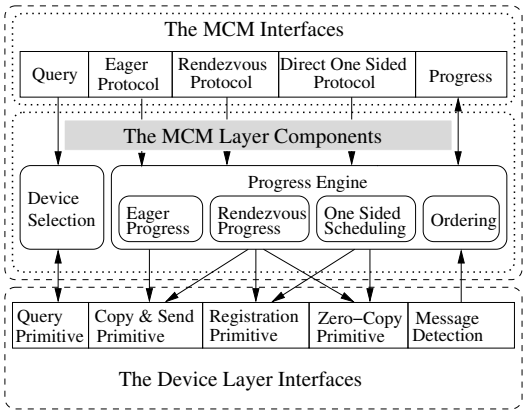

Figure 3. Design details of the MCM layer

The device selection component collects the performance characteristics from the device layer through the device query interfaces. It knows the message size and the destination from the ADI3-Ex, thus it is able to decide the most suitable device to complete this message and the preferred communication protocol. The information is passed back to the ADI3-Ex layer through the query interface.

Once the ADI3-Ex layer decides the communication protocol, the actual communication is taken care of by the progress engine at the MCM layer. There are several important components of the progress engine.

The eager and rendezvous progress components implement the eager and rendezvous communication protocols. The communication requests will be processed through the communication device chosen by the device selection components. Typically for eager protocols, the messages will be sent through the copy-and-send primitives provided by the device layer. And for rendezvous protocol, the user buffers involved in the communication are first sent to the device through the registration primitives for registration and then the data is sent through the zero-copy primitives. In our design the progress engines are supposed to mask the recoverable failures of the device level. For example, the ADI3-Ex layer may send an eager message too large for the device to send out at one time, in that case the message needs to be broken down to smaller sizes and sent out in multiple packets. Also the device may fail to register the user buffer for zero-copy rendezvous protocols, then large messages will also need to be broken down into pieces to be sent through copy-and-send primitives. As we will discuss in Section 4, this behavior benefits both the portability and scalability.

The direct one sided progress component implements the direct one sided communication support for one sided operations. The detailed design and implementation details are described in [**?**, **?**].

The ordering component helps to keep the correct ordering of the messages sent from different devices at the receiver side. The incoming message from a specific device is detected through the message detection interface provided by the device layer.

3.3 The Device Layer

The lowest device layer implements the basic network transfer primitives. The paradigm of this layer allows us to hide the difference between various communication schemes provided by system while exposing the maximum number of features (such as zero-copy communication) to the MCM layer. The interfaces provided by the device layer include copy-and-send primitives, zero-copy primitives, and the registration primitives which prepares for zero-copy communication.

Since there can be multiple devices existing at the same time, each device also implements the query primitive to provide the device selection component of the MCM layer with the basic performance features. This enables the MCM layer to select at runtime the most suitable device for a particular message.

The message detection primitive is queried by the progress engine at the MCM layer to detect the next incoming packet. The communication primitives at this layer are directly implemented based on the programming libraries provided by communication systems.

4 Potential Benefi ts of the New Design

In this section we take a closer look at how the proposed design achieves our design objectives with respect to performance, portability and scalability.

4.1 Performance

We start from the bottom most device layer to analyze the performance benefits of our design. In our design, the lowest devices will have enough information to optimize the performance based on hardware specific features. And these optimizations can be conducted without the involvement of higher layers.

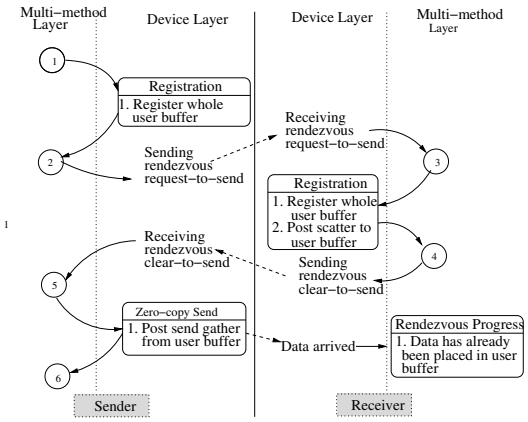

Figure 4. Incorporating SGRS

Let us take non-contiguous datatype communication as an example. In our design, the lowest device layer will get the whole datatype layout of the message from the MCM layer. With this information, we can incorporate the zero-copy SGRS scheme mentioned in Section 2. Fig 4 illustrates how the SGRS scheme works in our design. The layout exchange takes place as part of the rendezvous handshake. In the registration primitive, the device registers the user buffers involved in the communication, the receiver side also posts scatter receive descriptors. Upon receiving the rendezvous clear-to-send message, the sender will post the exactly matching gather send descriptor. Note that from view of the MCM layer, the whole process is exactly the same as a normal rendezvous transfer, where the user buffer is registered through the registration primitive and the RDMA is posted through the zero-copy primitives.

In addition, many other performance optimization schemes can be naturally implemented without the MCM layer being aware of it. For instance, the device layer can stripe large messages and send them through multiple rails [**?**] within the zero-copy primitive. Further any advance registration cache [**?**] can be implemented within the registration primitive. Hiding the details of the hardware-specific optimizations keeps the progress engines simple at the MCM layer. So we can focus more on high level optimizations, such as device selection components, or striping message across multiple devices. It also helps to keep the interface of the device layer concise thus helping the portability.

The MCM layer in our design is able to automatically choose the best available communication device among the available intra- or inter-node communication devices to communicate. Since the MCM layer understands the performance features of the devices through the query primitives provided by each device, under most circumstances the choice will be optimal. Further, the device selection component is an independent module in our design. In future it can be easily extended to consider more factors, such as the current communication load on each device.

At the ADI3-Ex layer, header caching will help to reduce the small message communication latency. And the one sided operations are also optimized through the direct one sided implementation.

4.2 Portability

Compared with the RDMA channel of old MVAPICH2 design, the device layer of our new design has increased the number of interfaces, to facilitate more features. However, we believe that the additional complexity for implementing a new device is limited, which allows us to maintain the good portability as delivered with the RDMA channel.

First, the added device layer communication interfaces in our new design are mainly the zero-copy communication interface. Those interfaces are very close to most of the programming interfaces provided by the RDMA capable networks and can be implemented without much overhead.

Further, since the MCM layer provides enough schemes to mask the recoverable failures as described in Section 3.2, a device may choose to only implement a part of the interfaces. For example, we implemented the intra-node communication based on file system shared memory. And in this case it is difficult to direct access memories of the remote processes without complex schemes involving the kernel [**?**]. Thus we only implement the copy-and-send primitives and return failure for every call to the registration primitive. In this case, the MCM layer will automatically switch to send large messages through copy-and-send primitives instead of attempting a zero-copy scheme.

In addition to the devices based on VAPI and shared memory, we have already implemented devices based on Gen2 verbs over InfiniBand and uDAPL programming interfaces.

4.3 Scalability

The requirement on the scalability of MPI usually contains two aspects:

- Memory scalability: The memory usage of MPI needs to be kept under a reasonable amount even when the process number of the parallel job is very large.
- Performance scalability: The communication performance, such as point-to-point latency, etc., should not be largely affected as the number of processes increases.

Usually these two aspects are tightly coupled. For example, to have better memory scalability, we typically need to reduce the size of pre-registered communication buffer for copy-and-send schemes as the number of processes increases. However in our earlier design, if the message cannot fit into one communication buffer, it has to be sent through rendezvous protocol. Since the rendezvous protocol requires memory registration and also involves a handshake process, the performance for small message sizes will be sub-optimal.

In our new design, the MCM layer keeps track of each unfinished request. Thus even if a message may not fit into the communication buffer at the device layer, the device layer can just send as much as it can. And the rest of the data will be sent out through further calls to the copy-and-send primitives. As a result, a message can be sent in a 'packetized' fashion. As we can see in Section 5, the packetization greatly helps the medium range message latency and throughput if the communication buffer is small.

Further, the MCM layer polls the incoming message through the message detection primitive of the device layer. The actual implementation is left to each communication device. Our previous work targeted towards scalability, such as RDMA polling set [**?**] and using Shared Receive Queue [**?**] can also be implemented at the device layer. These schemes help both the memory and performance scalability. In the old RDMA channel based design the lowest level only had control on the message detection with respect to each connection, hence it was not possible to incorporate such novel schemes.

5 Performance Evaluation

In this section we evaluate our new design with a set of micro-benchmarks as well as NAS Parallel Benchmarks (NPB 3.2 version). We evaluate VAPI and intra-node communication devices for our new design. We compare this new design with MVAPICH2-0.6.5, which is our old design based on the RDMA channel over VAPI on InfiniBand.

Our experimental testbed is a cluster of eight nodes. Each node is equipped with dual Intel Xeon 3.0GHz CPU, 2 Gigabytes memory, and a Mellanox MT23108 PCI-X InfiniBand HCA. All nodes are connected through a MTS2400 InfiniBand switch.

5.1 Inter-Node and Intra-Node Communication

We first look at the improvement on the basic point to point communication performance. We evaluate the latency and throughput between two processes for the two cases.

1. Inter-node - the two processes are on two different compute nodes
2. Intra-node - the two processes are on the same compute node

Figures 5 and 6 show the latency and throughput between two processes on different computing nodes. We observe that the 1 byte message latency is reduced from 5.8μs to 4.9μs, a 16% of improvement for the new design. This can be attributed to the header caching scheme we have implemented in the ADI3-Ex layer. Since our new design also overcomes the shortcoming of only one outstanding communication request, the message throughput increases greatly, especially for medium size messages. For message size of 64KB, we observe almost 29% improvement on throughput.

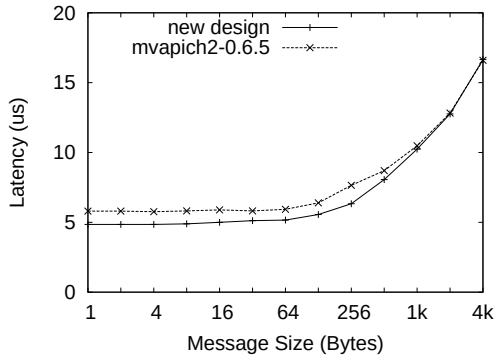

Figure 5. Inter-node communication latency

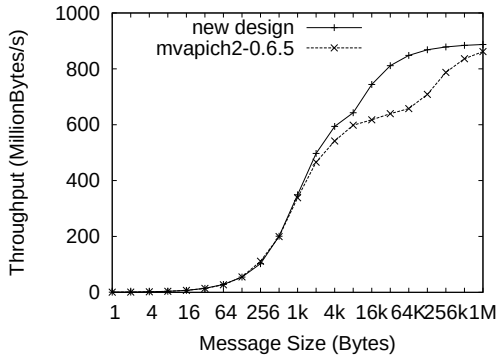

Figure 6. Inter-node communication throughput

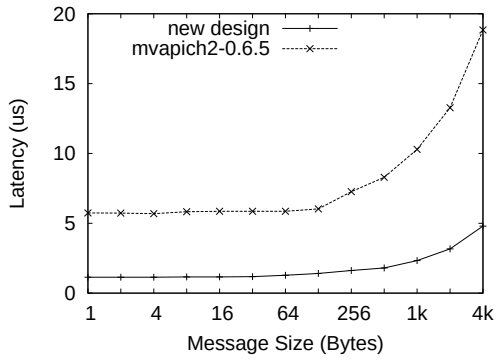

Figure 7. Intra-node communication latency

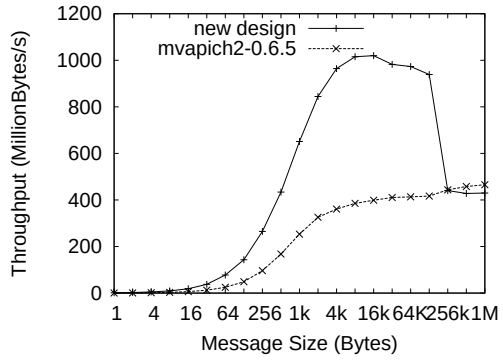

Figure 8. Intra-node communication throughput

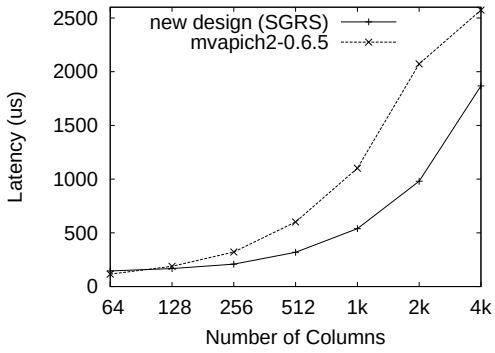

Figure 9. Non-contiguous communication latency

Figures 7 and 8 show the latency and throughput when the two processes are on the same node. Our previous design, mvapich2-0.6.5, did not have multiple communication devices, so the messages reach the HCA and loopback. However, the MCM layer in our new design automatically chooses to communicate through the intra-node communication device, which reduces the latency from 5.7μs to 1.1μs for small messages. And it also increases the throughput drastically especially for medium size messages. For example, for 16KB message, the throughput is improved by 150%, from 398 MB/s to 1019 MB/s. The drop for messages after 256KB is due to the cache effect. Even then it is comparable with the mvapich2-0.6.5 numbers. Note that the MCM layer has the ability to switch to using loopback for messages larger than 256KB, then the difference will no longer be there.

Our new design also incorporates the optimization for one sided communication based on direct one sided operations which utilize InfiniBand's RDMA capabilities. The benefits of this approach have already been demonstrated in [**?**].

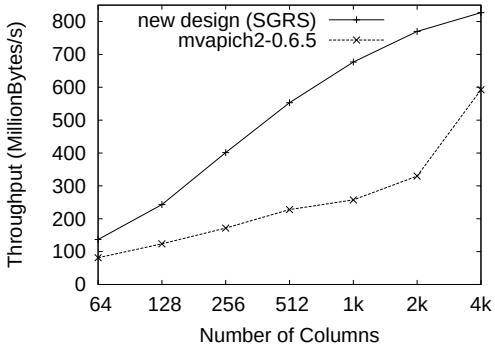

Figure 10. Non-contiguous communication throughput

5.2 Non-Contiguous Datatype Communication

Performance can largely benefit from the hardware-specific optimizations at the device level. Here we take the SGRS scheme (described in Section 4) as an example to see how it can enhance the performance for non-contiguous datatype communication.

The micro-benchmarks we use transfers increasing number of columns in a two dimensional $M * 8192$ integer matrix between 2 processes on 2 cluster nodes. These columns can be represented as vector datatype, which are widely used in scientific applications. Figures 9 and 10 show the comparison of latency and throughput between our old design, which uses pack and unpack approach involving copies on the sender and receiver side, and the new design with SGRS scheme incorporated. As we increase the number of columns from 64 to 4096, we clearly observe that the zero-copy SGRS scheme outperforms the copy based scheme.

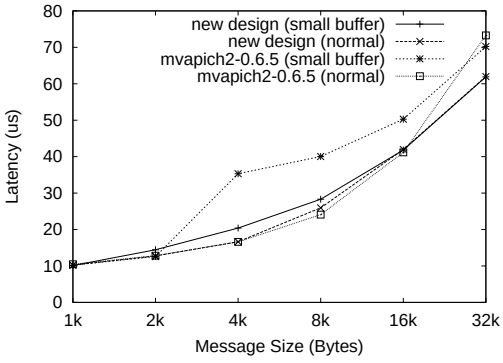

Figure 11. Effect of packetization on latency

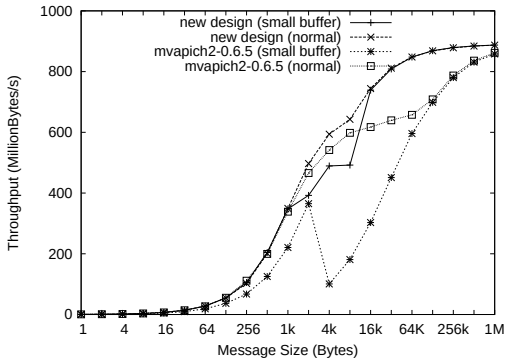

Figure 12. Effect of packetization on throughput

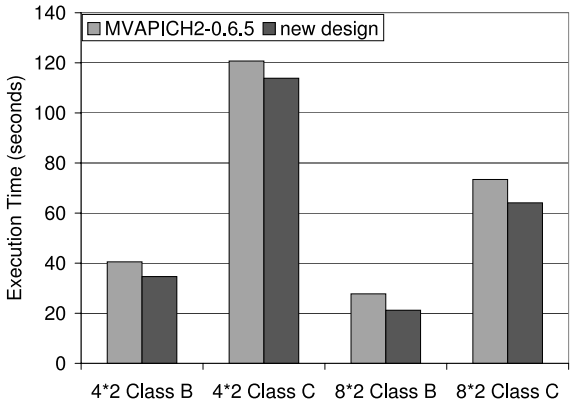

Figure 13. NAS-CG execution time comparison

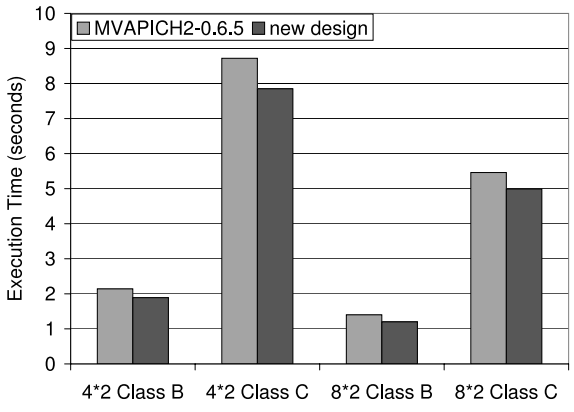

Figure 14. NAS-IS execution time comparison

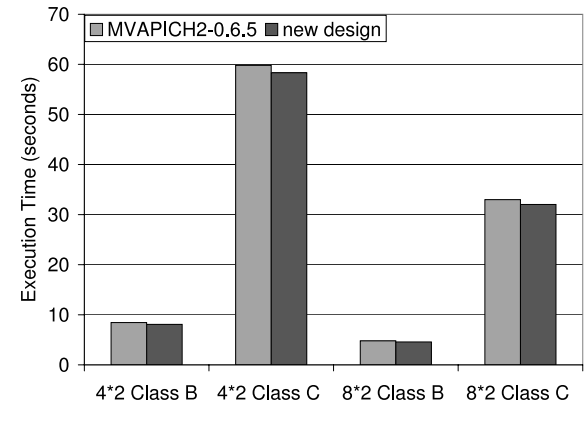

Figure 15. NAS-MG execution time comparison

5.3 Effects of Packetization

In Section 4.3 we mentioned that with the increase in number of nodes in a MPI job, we need to reduce the size of the communication buffers to keep the memory usage under a reasonable amount. With smaller size communication buffer, the role of packetization scheme is important. Figures 11 and 12 show the performance comparison with communication buffer of normal and reduced size for both our new design and mvapich2-0.6.5. We can clearly see that for the new design where the packetization scheme is incorporated, the performance only drops a little when we reduce communication buffer to 2KB. However for mvapich2-0.6.5 where the message is forced to go through rendezvous, the performance drops sharply.

5.4 NAS Parallel Benchmarks

Finally we evaluate the application level performance with NAS Parallel Benchmarks. We show the comparison between MVAPICH2-0.6.5 with our new design for NAS IS, CG and MG with class B and class C in Figures 13, 14 and 15, respectively. These experiments are done on two configurations: 4 dual processors and 8 dual processors. Due to the overall effect of intra-node communication device and the optimized progress engine, the new design performs considerably better. Especially for CG benchmark, we get up to 24% improvement with respect to the execution time. For MG and IS benchmarks, we get up to 5% and 14% improvement, respectively.

6 Related Work

There are several research studies on implementing MPI-2 on modern interconnects. MPICH2 [**?**] is a popular implementation from Argonne National Laboratory. Grabner et al. at University Chemnitz have implemented MPI2 for InfiniBand based on the MPICH2 CH3 layer[**?**]. Open MPI [**?**]

is another MPI-2 implementation based on component architecture. Our paper focuses on a new framework to design high performance and scalable MPI-2 over InfiniBand based on MPICH2 ADI3 layer.

There are several studies to improve different aspects of a communication system and middleware like MPI with respect to non-contiguous communication, intra-node communication, scalability, etc. Wang et al. [**?**] and Jin et al. [**?**] have proposed novel approaches for extracting communication performance on SMP based cluster. We believe that our design allows to adopt these schemes as well.

Byna et al. have proposed techniques for improving the performance of derived datatypes by automatically using packing algorithms that are optimized for memory-access cost [**?**]. Our SGRS design takes advantage of InfiniBand hardware features and can be considered complementary to this scheme in extracting better performance. Scalability issues related to VIA-based technologies in supporting MPI has been analyzed in detail by Brightwell et al. [**?**].

7 Conclusions and Future Work

In this paper we have presented a new MVAPICH2 design based on extending the MPICH2's ADI3 layer. We illustrate the overall design and explain the advantages of our new design. Within the design we propose, most of the hardware-related performance optimizations can be smoothly incorporated at the device layer, which brings high performance and scalability, while keeping almost the same portability as delivered by the RDMA channel interface provided by MPICH2.

In future, we plan to incorporate additional techniques and features, such as multi-rail, shared receive queue and RDMA polling set, into our design. We will also explore more intelligent device selection schemes, for instance, taking into account of the load balancing among the communication devices and simultaneously supporting multiple inter-node/intra-node communication devices. We also plan to incorporate collective operations into this layered design, allowing them to take advantage of hardware-specific features, such as hardware multicast.

References

[1] R. Brightwell and A. B. Maccabe. Scalability limitations of via-based technologies in supporting mpi. In *Fourth MPI Developer's and User's Conference*, 2000.

[2] S. Byna, W. Gropp, X. H Sun, and R. Thakur. Improving the performance of mpi derived datatypes by optimizing memory-access cost. In *Cluster'03*, 2003.

[3] DAT Collaborative. http://www.datcollaborative.org/udapl.html.

[4] E. Gabriel et al. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation. In *Proceedings of EuroPVM/MPI*, 2004.

[5] R. Grabner, F. Mietke, and W. Rehm. An MPICH2 Channel Device Implementation over VAPI on InfiniBand. In *Proceedings of the International Parallel and Distributed Processing Symposium*, 2004.

[6] H. -W. Jin, S. Sur, L. Chai, and D. K. Panda. LiMIC: Support for High-Performance MPI Intra-Node Communication on Linux Cluster. In *ICPP 05*, June 2005.

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

[8] 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 the International Parallel and Distributed Processing Symposium*, 2004.

[9] J. Liu, A. Mamidala, and D. K. Panda. Fast and scalable mpi-level broadcast using infiniband's hardware multicast support. In *Proceedings of IPDPS'04*, 2004.

[10] J. Liu, J. Wu, S. P. Kini, P. Wyckoff, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. In *Proceedings of 17th Annual ACM International Conference on Supercomputing (ICS '03)*, June 2003.

[11] Mellanox Technologies. Mellanox IB-Verbs API (VAPI), Rev. 1.00.

[12] MPICH2 Homepage. http://www-unix.mcs.anl.gov/mpi/mpich2/.

[13] Open InfiniBand Alliance. http://www.openib.org.

[14] G. Santhanaraman, J. Wu, W. Huang, and D. K. Panda. Designing Zero-copy MPI Derived Datatype Communication over InfiniBand: Alternative Approaches and Performance Evaluation. In *Special Issue of International Journal of High Performance Computing Applications (IJHPCA)* , 2005.

[15] Top 500 Supercomputer Site. http://www.top500.com.

[16] Marc Snir, Steve Otto, Steve Huss-Lederman, David Walker, and Jack Dongarra. *MPI–The Complete Reference. Volume 1 - The MPI-1 Core, 2nd edition.* The MIT Press, 1998.

[17] S. Sur, L. Chai, H.-W. Jin, and D. K. Panda. Shared Receive Queue based scalable MPI Design for InfiniBand Clusters. Tech Report OSU-CISRC-10/05-TR66.

[18] H. Tezuka, F. O'Carroll, A. Hori, and Y. Ishikawa. Pin-down cache: A virtual memory management technique for zero-copy communication. In *Proceedings of the 12th International Parallel Processing Symposium*, 1998.

[19] W. Huang, G. Santhanaraman, H.-W. Jin, and D. K. Panda. Design Alternatives and Performance Trade-offs for Implementing MPI-2 over InfiniBand. In *EuroPVM/MPI 05*, Sept. 2005.

[20] W. Huang, G. Santhanaraman, H.-W. Jin, and D. K. Panda. Scheduling of MPI-2 One Sided Operations over InfiniBand. In *CAC 05 (in conjunction with IPDPS 05)*, Apr. 2005.

[21] W. Jiang, J. Liu, H. -W. Jin, D. K. Panda, W. Gropp, and R. Thakur. High Performance MPI-2 One-Sided Communication over InfiniBand. In *CCGrid 04*, Apr. 2004.

[22] MVAPICH Project Website. http://nowlab.cse.ohio-state.edu/projects/mpi-iba/index.html.

[23] Kwan-Po Wong and Cho-Li Wang. Push-pull messaging: A high-performance communication mechanism for commodity smp clusters. In *ICPP*, 1999.