Design Alternatives and Performance Trade-offs for Implementing MPI-2 over InfiniBand *

Wei Huang, Gopalakrishnan Santhanaraman,
Hyun-Wook Jin, and Dhabaleswar K. Panda

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

Abstract. MPICH2 provides a layered architecture to achieve both portability and performance. For implementations of MPI-2 over InfiniBand, it provides the flexibility for researchers at the RDMA channel, CH3 or ADI3 layer. In this paper we analyze the performance and complexity trade-offs associated with implementations at these layers. We describe our designs and implementations, as well as optimizations at each layer. To show the performance impacts of these design schemes and optimizations, we evaluate our implementations with different micro-benchmarks, HPCC and NAS test suite. Our experiments show that although the ADI3 layers adds complexity in implementation, the benefits achieved through optimizations justify moving to the ADI layer to extract the best performance.
keywords: MPI-2, InfiniBand, RDMA channel, CH3, ADI3

1 Introduction

In the last decade, MPI (message passing interface) has become the *de facto* standard for programming parallel applications. MPI-1 standard [12] was proposed by the MPI forum to provide a uniform standard for MPI developers. As a follow-up, MPI-2 [9] standard aims to extend MPI-1 in the areas of one sided communication, I/O and dynamic process management.

MPICH2 [10] from Argonne National Laboratory is one popular implementation of the MPI-2 standard. It aims to combine performance with portability over different interconnects. It tries to achieve this by maximal sharing of platform independent code like MPI datatypes, groups, and communicators, etc., and calls the Abstract Device Interface (ADI3) for platform dependent code. The porting to different interconnects is achieved by having a separate ADI implementation for each interconnect. To further ease the porting, the ADI itself is also layered and can be implemented in terms of lower level interfaces.

In the field of High Performance Computing, InfiniBand [5] is emerging as a strong player. InfiniBand supports several advanced hardware features including RDMA capability. To implement MPI-2 on InfiniBand, MPICH2 provides

* This research is supported in part by Department of Energy's Grant #DE-FC02-01ER25506, National Science Foundation's grants #CNS-0204429, and #CUR-0311542, and a grant from Intel.

the flexibility to implement the ADI3, or its lower level interfaces like CH3 and RDMA channel layers. Understanding the benefits and limitations of implementing at each layer (ADI3, CH3, and RDMA channel) is very important to come up with an efficient design. The lower layer interfaces are easier to port at the cost of some performance penalties. This is rather expected, but it would be of more interest to quantitatively understand the performance impact and try to come up with different levels of optimizations at each layer. To the best of our knowledge, there is no literature that does an in-depth study on this topic.

In this paper, we attempt to do an detailed analysis of the performance and complexity trade-offs for implementing MPI-2 over InfiniBand at the RDMA channel, CH3, and ADI3 layer. We focus on the point to point and one sided operations in the current work. For fair comparison, we provide our design and implementation at each of these layers. In the rest of the paper, Section 2 introduces the background of our work. Section 3 describes and analyzes our design choices and optimizations. In Section 4 we conduct performance evaluations. Conclusions and future work are presented in Section 5.

2 Background

2.1 InfiniBand Architecture

The InfiniBand Architecture (IBA) [5] 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 the remote memory. RDMA operations are one sided and do not incur software overhead on remote side. Further, InfiniBand verbs provide scatter/gather features to handle non contiguity. InfiniBand verbs specification also provides useful features like atomic operations and multi-cast, etc.

2.2 Layered Design of MPICH2 and MVAPICH2

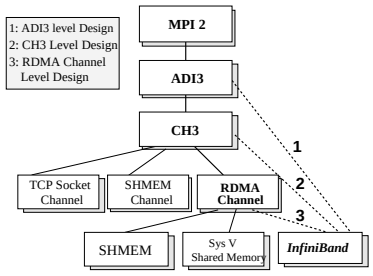

Fig. 1. Layered Design of MPICH2

MPICH2 supports both point to point and one sided operations. Figure 1 describes the layered approach provided by MPICH2 for designing MPI-2 over RDMA capable networks like InfiniBand. Implementation of MPI-2 on InfiniBand can be done at one of the three layers in the current MPICH2 stack: RDMA channel, CH3 or ADI3. One of the objectives of such kind of design is to get a better balance between performance and complexity.

RDMA channel is at the bottom most position in the hierarchical structure. All communication operations that MPICH2 supports are mapped to just five functions at this layer. Among them only two (*put* and *read*) are communication functions, thus the porting overhead is minimized. The interface needs to conform to stream semantics. It is especially

designed for the architectures with RDMA capabilities, which directly fits with the InfiniBand's RDMA semantics.

The CH3 provides a *channel* device that consists of a dozen functions. It accepts the communication requests from the upper layer and informs the upper layer once the communication has completed. It is responsible to make communication progress, which is the added complexity associated with the implementations at this layer. From a performance perspective, it has more flexibility to improve the performance since it can access more performance oriented features than the RDMA channel layer. Argonne National Lab[10] and the University of Chemnitz[3] have both developed their CH3 devices for Infiniband.

The ADI3 is a full featured, abstract device interface used in MPICH2. It is the highest portable layer in MPICH2 hierarchy. A large number of functions must be implemented to bring out an ADI3 design, but meanwhile it provides flexibility for many optimizations, which are hidden from lower layers.

MVAPICH2 is a high performance implementation of MPI-2 over InfiniBand [6] from the Ohio State University. The latest release version implements the point to point communication at the RDMA channel layer [8] and also optimizes the one sided communication at the ADI3 level [7, 4]. The continuous research progress of this project has further motivated us to carry out the proposed in-depth study of design alternatives and performance trade-offs.

3 Designs and Implementations at Different Layers

For an implementation over InfiniBand, MPICH2 design provides choices at three different layers. In this sense, understanding the exact trade-offs of the performance constraints and implementation complexity at each layer will be critical to have an efficient design. To study these issues and to carry out a fair comparison, we have designed and implemented MPI2 over InfiniBand at each of the MPICH2 hierarchy: RDMA channel, CH3 and ADI3, respectively. We present our strategies in the following subsections.

3.1 RDMA Channel Level Design and Implementation

At RDMA channel, as mentioned in Section 2.2, all architecture dependent communication functionalities are encapsulated into a small set of interfaces. The interface needs to provide only stream semantics and the communication progress of MPI messages are left to the upper layers.

Our design is purely based on the RDMA capability of InfiniBand [8]. Fig. 2 illustrates the design issues at this layer. For short messages, the eager protocol is used to achieve good latency. It copies messages to pre-registered buffers and sends them through RDMA write. For large messages, using the eager protocol will introduce high copy overhead, so a zero-copy rendezvous protocol is used. User buffer is registered on the fly and sent through RDMA. Registration cache [13] is implemented to reduce the registration overhead.

The RDMA channel receives the communication requests from the CH3 layer above it. In the current stack, the CH3 layer makes only one outstanding request to the RDMA channel and will not issue the next request until the previous one has completed. This results in the serialization of the communication requests,

which causes inefficient utilization of the network. For small messages, an optimization would be to copy the message to the pre-registered buffer and immediately report completion to the CH3 layer. By this early completion method, the CH3 layer can issue the next communication request so that multiple requests can be issued to the RDMA layer. But for large messages which are sent through the rendezvous protocol, we can only report completion after the whole rendezvous process finishes since we need to hold the user buffer for zero-copy send, making it difficult to obtain higher bandwidth for medium-large messages at the RDMA layer. We show this performance impact in Section 4.

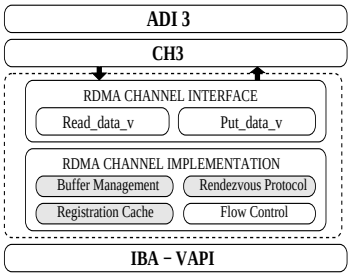

Fig. 2. RDMA Channel level design and implementation

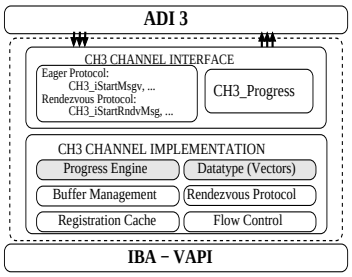

Fig. 3. CH3 level design and implementation

3.2 CH3 Level Design and Implementation

Fig. 3 shows the basic function blocks in our CH3 level design. Messages can be sent through eager or rendezvous protocols similar to the approach taken by the RDMA channel implementation. Hence functionalities such as buffer management, registration cache, etc., need to be also implemented at the CH3 layer.

The need to implement progress engine is a significant difference between the CH3 level and the RDMA channel design. The CH3 layer must keep track of all the communication requests coming from the ADI3 layer, finish them and report completions. ADI3 can keep sending requests but the underlying network resources are limited. So we implement a queue to buffer the requests which cannot be finished immediately due to the lack of network resources. Requests in the queue will be retried when resources become available again. The benefit of implementing the progress engine is obvious. We can get access to all the requests at the sender side. Now for large messages we can start multiple rendezvous progresses at the same time so that the throughput is expected to be greatly improved as compared to the RDMA channel level design.

Datatype communication can also be optimized at this layer. ADI3 flattens the datatype and provides the datatype information to the CH3 layer as a vector list. With this information, a CH3 level design can have a global picture of all the buffer vectors that need to be sent in a particular MPI message. So optimizations such as zero-copy datatype [11, 14] can be applied at this level.

3.3 ADI3 level Design and Implementation

Compared to the CH3 and RDMA channel, the ADI3 interface is full-featured. This allows the implementations to take opportunities for more efficient commu-

nication. To re-implement the whole ADI3 is very complicated, so we decided to reuse most of the CH3 level implementation described in Section 3.2 and perform several optimizations at the ADI3 layer, shown in Fig 4. These optimizations are possible at the ADI3 layer since it is allowed direct access to several global data structures which are abstracted out for the lower layers.

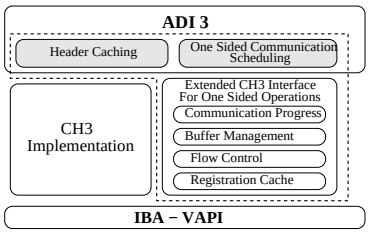

Fig. 4. Optimizations at ADI3 layer

Header caching is an optimization that can be implemented at the ADI3 to reduce the small message latency. The basic idea is to cache some fields of the MPI header for each connection at the receiver side. So that if the next message between this connection has the same header information in those cached fields, we can reduce the size of MPI header being sent. If these fields differ, there is a copy overhead at the receiver for the header caching, which is quite negligible according to our experience. It is be noted that MPI header caching cannot be performed at lower layers since only ADI3 is supposed to know the contents in a MPI header.

Another significant optimization is in the area of one-sided communication. Originally in MPICH2, one sided operations are implemented by the point to point interfaces provided by CH3. Our previous work [7] has shown that by directly using the RDMA features provided by InfiniBand instead of going through the point to point path, the performance for the one-sided operations can be greatly enhanced. We can also schedule one sided operations to achieve much better latency and throughput. The scheduling schemes are described in detail in [4]. These optimizations are done by extending the CH3 interface [7]. At the CH3 layer we cannot distinguish between data for two sided and one sided operations and hence cannot perform such optimizations. The latest MPICH2 also has extended the CH3 one sided interface for shared memory architectures, which reflects the potential to optimize one sided operations at the ADI3 layer.

4 Performance Evaluation

In this section we evaluate our implementations at RDMA channel, CH3 and ADI3 layers by a set of micro-benchmarks, HPC Challenge Benchmark [2] and NAS test suite [1]. Tests are conducted on on two different clusters. The first cluster (Cluster A) consists of computing nodes with dual Intel Xeon 3.0 GHz processors, 2GB memory, and MT23108 PCI-X HCAs. They are connected by an InfiniScale MTS2400 switch. The second cluster (Cluster B) is equipped with dual Intel Xeon 2.66 GHz processors, 2GB memory, and MT23108 PCI-X HCAs, connected through an InfiniScale MTS14400 switch.

4.1 Point to Point Communication

Point to point communication test are conducted on Cluster A. Fig. 5 shows the uni-directional bandwidth test results for our implementation at RDMA channel, CH3 and ADI3 layers. For medium-large messages, the bandwidth is significantly improved by up to 28% by moving from RDMA channel to CH3 layer, because

CH3 handles multiple send requests simultaneously. ADI3 layer shows similar numbers as CH3 layer. It is to be noted that all bandwidth numbers in this paper are reported in MillionBytes/Sec (MB/s).

Fig. 6 shows the ping-pong latency. By getting rid of the stack overhead, the one byte message latency drops from 5.6us at RDMA layer to 5.3us at CH3 layer. By performing header caching at ADI3 layer, the number drops further to 4.9us. Header caching technique is applied to messages smaller than 256 bytes. So for this range the ADI3 level numbers consistently outperform CH3 numbers.

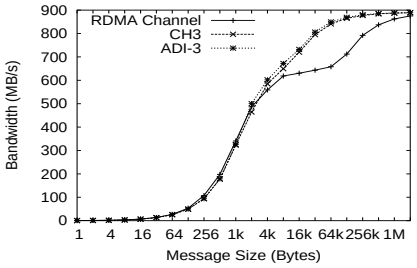

Fig. 5. Unidirectional bandwidth

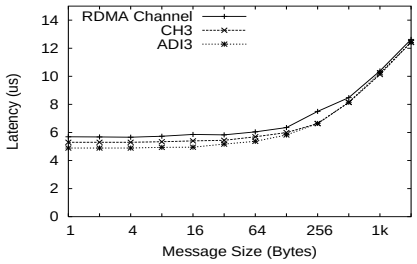

Fig. 6. Point to point latency

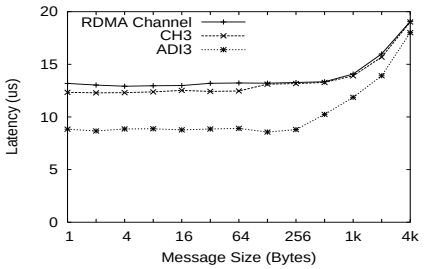

Fig. 7. MPI_Put latency

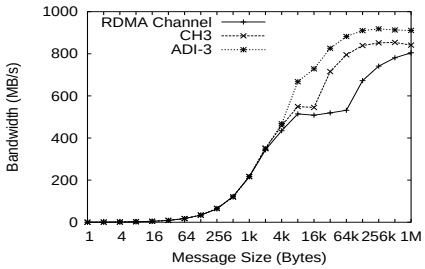

Fig. 8. One sided throughput test

4.2 One Sided Operations

Evaluation with one sided operations are also conducted on Cluster A. The results for MPI_Put test are shown in Fig. 7. The test times the ping-pong latency for performing the put operation followed by synchronization. For the CH3 and the RDMA Channel level design, one sided operations are implemented based on point to point communication. Their numbers are similar, with the CH3 level design performing slightly better because point to point communication is optimized. By optimizing one sided operation at ADI3, we observe 30% reduction in MPI_Put latency as compared to the CH3 level design.

We also measure the throughput of one sided communication. Here the origin process issues 16 MPI_Put and 16 MPI_Get operations of the same size. The target process just starts an exposure epoch. We measure the maximum throughput we can achieve (MillionBytes/sec) for multiple iterations of the above sequence. Fig. 8 shows the results. The improvement on point to point bandwidth makes the CH3 level design outperform the RDMA channel level design by up to 49%. And with one sided scheduling at the ADI3 layer, the peak throughput can reach around 920MB/s, which is another 8.1% higher than the CH3 level design numbers.

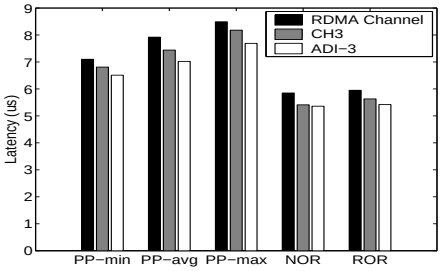

Fig. 9. HPCC 8 bytes latency. PP-min: minimum ping-pong latency; PP-avg: average ping-pong latency; PP-max: maximum ping-pong latency; NOR: Natural ordered ring access latency; ROR: random ordered ring access latency

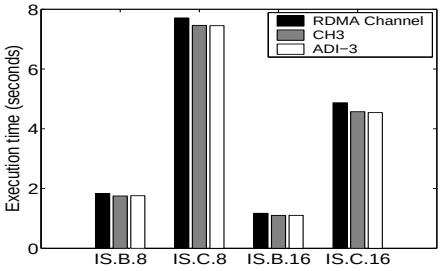

Fig. 10. NAS (IS) results on Cluster B

4.3 HPC Challenge (HPCC) Suite

The HPCC suite [2] contains tests which evaluate the latency for different types of communication patterns. It performs the ping-pong tests between all possible different pairs of processors. It also uses two different ring types of communication patterns to evaluate latency: Naturally Ordered Ring and Randomly Ordered Ring. HPCC tests were performed on 16 nodes of Cluster B and we report 8 bytes latency numbers. As shown in Fig. 9, we clearly observe an performance improvement up to 7% for the CH3 level design over the RDMA Channel level design; and the benefit from header caching at the ADI3 layer enhances the performance by up to another 6%.

4.4 NAS Integer Sort

In this section we show the performance evaluation for the NAS-IS benchmark [1]. IS is an integer sort benchmark kernel that stresses the communication aspect of the network. The experiments were conducted for classes B and C on 8 nodes and 16 nodes of Cluster B. The results are shown in Fig. 10. The ADI3 level implementation here shows up to 7% improvement comparing with the RDMA channel level design. The ADI3 level optimizations does not directly help the communication pattern that is observed in NAS-IS. Hence the performance seen at the CH3 level is the same as that of the ADI3 level.

5 Conclusions and Future Work

In this paper we have analyzed the trade-offs associated with implementing MPI-2 over InfiniBand at the RDMA channel, CH3 and ADI3 layer of MPICH2. We have also described the various optimizations that are possible at each level.

With respect to design complexity, a CH3 level design needs to implement the progress engine, which is the main cause of added complexity. A fully featured ADI3 level design is very complicated but optimizations like header caching, one sided communication scheduling can be done at this level.

With respect to performance, the CH3 and ADI3 level design can increase the bandwidth significantly, up to 28% for bandwidth test comparing with the

RDMA channel level design. Header caching at the ADI3 can lower the small message latency to 4.9 us, a 12.5% improvement comparing with 5.6 us achieved by the RDMA channel level design. One sided scheduling at the ADI3 level also greatly improves the performance. We see an enhancement up to 30% in MPI_Put latency and 8.1% in throughput test compared with the CH3 level design, which in turn shows 49% improvement on throughput over the RDMA channel level design. Effects of these optimizations also show benefits at the application level evaluation of HPCC and NAS suite. As a conclusion, although the ADI3 layers adds complexity in implementation, the benefits achieved through optimizations justify moving to the ADI layer to extract the best performance.

As a part of future work we would like to come up with a full fledged MPI-2 design over InfiniBand at the ADI3 layer to deliver good performance. We are planning to support communication through shared memory, and optimize collective operations using InfiniBand's RDMA and multi-cast features.

References

1. D. H. Bailey, E. Barszcz, L. Dagum, and H.D. Simon. NAS Parallel Benchmark Results. Technical Report 94-006, RNR, 1994.
2. HPC Challenge Benchmark. http://icl.cs.utk.edu/hpcc/.
3. 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.
4. W. Huang, G. Santhanaraman, H. W. Jin, and D. K. Panda. Scheduling of MPI-2 One Sided Operations over InfiniBand. Workshop On Communication Architecture on Clusters (CAC), in conjunction with IPDPS'05, April 2005.
5. InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.2.
6. Network Based Computing Laboratory. http://nowlab.cis.ohio-state.edu/.
7. J. Liu, W. Jiang, H. W. Jin, D. K. Panda, W. Gropp, and R. Thakur. High Performance MPI-2 One-Sided Communication over InfiniBand. International Symposium on Cluster Computing and the Grid (CCGrid 04), April 2004.
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. Message Passing Interface Forum. MPI-2: A Message Passing Interface Standard. *High Performance Computing Applications*, 12(1–2):1–299, 1998.
10. MPICH2. http://www-unix.mcs.anl.gov/mpi/mpich2/.
11. G. Santhanaraman, J. Wu, and D. K. Panda. Zero-Copy MPI Derived Datatype Communication over InfiniBand. EuroPVM-MPI 2004, September 2004.
12. M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI–The Complete Reference. Volume 1 - The MPI-1 Core, 2nd edition.* The MIT Press, 1998.
13. 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.
14. J. Wu, P. Wyckoff, and D. K. Panda. High Performance Implementation of MPI Datatype Communication over InfiniBand. In *Proceedings of the International Parallel and Distributed Processing Symposium*, 2004.