MIBA: A Micro-benchmark Suite for Evaluating InfiniBand Architecture Implementations *

B. Chandrasekaran[1], Pete Wyckoff[2], and Dhabaleswar K. Panda[1]

[1] Department of Computer and Information Sciences
The Ohio State University, Columbus, OH 43201
{chandrab,panda}@cis.ohio-state.edu
[2] Ohio Supercomputer Center
1224 Kinnear Road, Columbus, OH 43212
pw@osc.edu

Abstract. Recently, InfiniBand Architecture (IBA) has been proposed as the next generation interconnect for I/O and inter-process communication. The main idea behind this industry standard is to use a scalable switched fabric to design the next generation clusters and servers with high performance and scalability. This architecture provides various types of new mechanisms and services (such as multiple transport services, RDMA and atomic operations, multicast support, service levels, and virtual channels). These services are provided by components (such as queue pairs, completion queue, and virtual-to-physical address translations) and their attributes. Different implementation choices of IBA may lead to different design strategies for efficient implementation of higher level communication layer/libraries (such as Message Passing Interface (MPI), sockets, and distributed shared memory). It also has an impact on the performance of applications.
Currently there is no framework for evaluating different design choices and for obtaining insights about the design choices made in a particular implementation of IBA. In this paper we address these issues by proposing a new micro-benchmark suite (MIBA) to evaluate the InfiniBand architecture implementations. MIBA consists of several micro-benchmarks which are divided into two major categories: non-data transfer related micro-benchmarks and data transfer related micro-benchmarks. By using the new micro-benchmark suite, the performance of IBA implementations can be evaluated under different communication scenarios, and also with respect to the implementation of different components and attributes of IBA. We demonstrate the use of MIBA to evaluate the second generation IBA adapters from Mellanox Technologies.

1 Introduction

Emerging distributed and high performance applications require large computational power as well as low latency, high bandwidth and scalable communication

* This research is supported in part by Sandia National Laboratory's contract #30505, Department of Energy's Grant #DE-FC02-01ER25506, and National Science Foundation's grants #EIA-9986052 and #CCR-0204429.

subsystems for data exchange and synchronous operations. In the past few years, the computational power of desktop and server computers has been doubling every eighteen months. The raw bandwidth of network hardware has also increased to the order of Gigabits per second. During the past few years, the research and industry communities have been proposing and implementing many user-level communication systems such as AM [20], VMMC [7], FM [14], EMP [17,18], U-Net [19,21], and LAPI [16] to address some of the problems associated with the traditional networking protocols. In these systems, the involvement of operating system kernel is minimized and the number of data copies is reduced. As a result, they can provide much higher communication performance to the application layer.

More recently, InfiniBand Architecture [10] has been proposed as the next generation interconnect for I/O and inter-process communication. In InfiniBand, computing nodes and I/O nodes are connected to the switched fabric through Channel Adapters. InfiniBand provides a Verbs interface which is a superset of VIA [9,8]. This interface is used by the host systems to communicate with Host Channel Adapters. InfiniBand provides many novel features: three different kinds of communication operations (send/receive, RDMA, and atomic), multiple transport services (such as reliable connection (RC), unreliable datagram (UD), and reliable datagram (RD), different mechanisms for QoS (such as service levels and virtual lanes). In addition to providing scalability and high performance, InfiniBand also aims to meet applications' need for Reliability, Availability and Serviceability (RAS).

Recently several companies have started shipping out InfiniBand hardware. It is now a challenging task to report the performance of InfiniBand architectures accurately and comprehensively. The standard tests such as ping-pong latency and bandwidth give very little insight into the implementation of various components of the architecture. It does not evaluate the system for various communication scenarios. Therefore it does not depict all the characteristics of a real life application. Hence there is a need to study the aspects of various components involved in the communication. For example, the design choices in the implementation of virtual to physical address translation may lead to different performance results.

InfiniBand architecture specification offers a wide range of features and services. This is a motivating factor for computer architects to develop highly efficient implementations of higher-level programming model layers such as MPI [12,11], sockets [4] and distributed shared memory [13]. Also the architecture provides a promising efficient communication subsystem for various applications such as web servers and data centers. The various features and services offered by the InfiniBand architecture increases the number of design choices for implementing such programming models and applications. Hence there is a need for a framework to evaluate various such design choices.

The hardware products for the InfiniBand Architecture are still in their early stages but are rapidly developing. More features and still better performance of the hardware are expected in the near future. A systematic and in depth study of various components by a framework would provide valuable guidelines to hardware vendors to identify their strengths and weaknesses in their implementations and bring out better releases of the InfiniBand products.

The requirements of such a framework are:

1. To evaluate various implementations of InfiniBand architecture and compare their strengths and weakness in a standardized manner.
2. To evaluate the system for various communication scenarios.
3. To provide insights to developers of programming model layers and applications and to guide them in adopting appropriate and efficient strategies in their implementations.
4. To give valuable guidelines to InfiniBand hardware vendors about their implementations so that it can be optimized.

Traditional models of computation and communication are not sufficient to address the requirements listed above. We take on the challenge of designing a micro-benchmark suite to comprehensively evaluate the InfiniBand Architecture. This suite is divided into two major categories: Non-Data transfer related and Data Transfer related. Under the first category, we include micro-benchmarks for measuring the cost of several basic non-data transfer related operations: creating and destroying Queue Pairs, creating and destroying Completion Queues, and memory registration and deregistrations. The cost of such operations are evaluated by varying various parameters associated with them. The second category consists of several data-transfer related micro-benchmarks. The main objective here is to the isolate different components (such as virtual-to-physical address translation, multiple data segments, and event handling) and study them by varying their attribute values. This would clearly bring out the importance of that component in the critical path of communication. It would also help us to evaluate such components for various implementations of InfiniBand. The micro-benchmark suite would provide valuable insights to the developers of high performance parallel applications and data center enterprise applications

The micro-benchmarks are evaluated on a Linux based InfiniBand cluster. The benchmark suite evaluates the Verbs Application Programmers Interface (VAPI) over InfiniHost(TM) MT23108 Dual Port 4X Host Channel Adapter (HCA) cards provided by Mellanox Technologies [1].

The rest of the paper is organized in the following manner. Section 2 gives an overview of the IBA architecture. Sections 3 and 4 describe the Mellanox HCAs and their Verbs API interface. Section 5 describes the benchmark tests in detail. In Section 6 we present the results. Related work, conclusions and future work are presented in Sections 7 and 8.

2 InfiniBand Architecture Overview

The InfiniBand Architecture defines a System Area Network (SAN) for interconnecting processing nodes and I/O nodes. Figure 1 provides an overview of the InfiniBand architecture. It provides the communication and management infrastructure for inter-processor communication and I/O. The main idea is to use a switched, channel-based interconnection fabric. The switched fabric of InfiniBand Architecture provides much more aggregate bandwidth. Also, a switched fabric can avoid single point of failure and provide more reliability. InfiniBand Architecture also has built-in QoS mechanisms which provide virtual lanes on each link and define service levels for each packets.

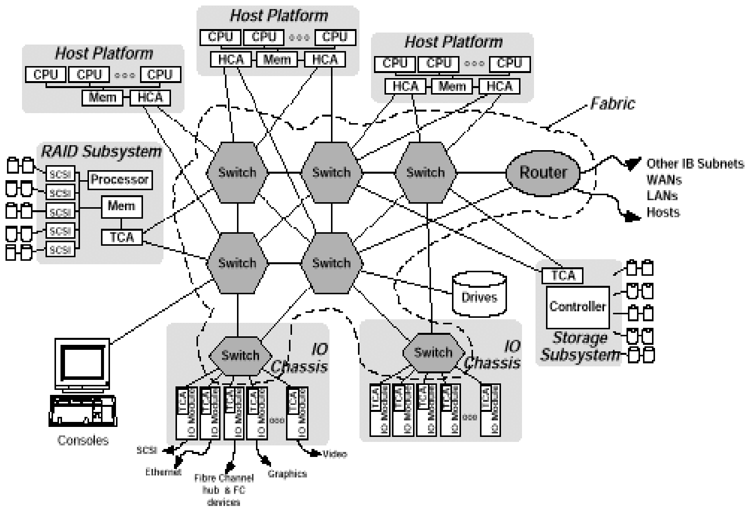


Fig. 1. Illustrating a typical system configuration with the InfiniBand Architecture (Courtesy InfiniBand Trade Association)

In an InfiniBand network, processing nodes and I/O nodes are connected to the fabric by Channel Adapters (CA). Channel Adapters usually have programmable DMA engines with protection features. They generate and consume IBA packets. There are two kinds of channel Adapters: Host Channel Adapter (HCA) and Target Channel Adapter (TCA). HCAs sit on processing nodes. Their semantic interface to consumers is specified in the form of InfiniBand Verbs. Unlike traditional network interface cards, Host Channel Adapters are connected directly to the system controller. TCAs connect I/O nodes to the fabric. Their interface to consumers are usually implementation specific and thus not defined in the InfiniBand specification.

The InfiniBand communication stack consists of different layers. The interface presented by Channel Adapters to consumers belongs to the transport layer. A queue-based model is used in this interface. A Queue Pair in InfiniBand Architecture consists of two queues: a send queue and a receive queue. The send queue holds instructions to transmit data and the receive queue holds instructions that describe where received data is to be placed. Communication operations are described in Work Queue Requests (WQR) and submitted to the work queue. Once submitted, a Work Queue Request becomes a Work Queue Element (WQE). WQEs are executed by Channel Adapters. The completion of work queue elements is reported through Completion Queues (CQ). Once a work queue element is finished, a completion queue entry is placed in the associated completion queue. Applications can check the completion queue to see if any work queue request has been finished.

3 Mellanox Hardware Architecture

Our InfiniBand platform consists of several InfiniHost HCAs and an InfiniScale switch from Mellanox [1]. In this section we will give a brief introduction to both the HCA and the switch.

InfiniScale is a full wire-speed switch with eight 4X ports or 1X InfiniBand Ports. These ports have an integrated 2.5 Gb/s physical layer serializer/deserializer and feature auto-negotiation between 1X and 4X links. There is also support for eight Virtual Data Lanes (VLs) in addition to a Dedicated Management Lane (VL15). Additionally, there is also support for link packet buffering, inbound and outbound partition checking and auto-negotiation of link speed. Finally, the switch has an embedded RISC processor for exception handling, out of band data management support and performance monitoring counter support.

The InfiniHost MT23108 dual 4X ported HCA/TCA allows for a bandwidth of up to 10 Gbit/s over its ports. It can potentially support up to 2^{24} QPs, End to End Contexts and CQs. Memory protection along with address translation is implemented in hardware itself. PCI-X support along with DDR memory allows portions of host memory to be configured as a part of system memory using a transparent PCI bridge allowing the host to directly place HCA related data without going over the PCI-X bus. The DDR memory allows the mapping of different queue entries namely work queues entries (WQE's) and execution queue entries to different portions of the system space transparently. At its heart, the HCA picks WQE's in a round robin fashion (the scheduler is flexible and supports more complex scheduling including weighted round robin with priority levels) and posts them to execution queues allowing for the implementation of QoS at a process level. Different WQE's specify how the completion notification should be generated. In the following section, we discuss the software interface to InfiniBand.

4 InfiniBand Software Interface

Unlike other specifications such as VIA, InfiniBand Architecture doesn't specify an API. Instead, it defines the functionality provided by HCAs to operating systems in terms of Verbs[10]. The Verbs interface specifies such functionality as transport resource management, multicast, work request processing and event handling.

Although in theory APIs for InfiniBand can be quite different from the Verbs interface, in reality many existing APIs have followed the Verbs semantics. One such example is the VAPI interface [1] from Mellanox Technologies. Many VAPI functions are directly mapped from corresponding Verbs functionality. This approach has several advantages: First, since the interface is very similar to the Verbs, the efforts needed to implement it on top of HCA is reduced. Second, because the Verbs interface is specified as a standard in the InfiniBand Architecture, it makes the job much easier to port applications from one InfiniBand API to another if they are both derived from Verbs.

As we have mentioned earlier, the communication in Verbs is based on queue-pairs. InfiniBand communication supports both channel (send/receive) and memory (RDMA) semantics. These operations are specified in work queue requests and posted to send or receive queues for execution. The completion of work queue requests is reported through completion queues (CQs). Note that all communication memory must be registered first. This step is necessary because the HCA uses DMA operation to send from or receive into host communication buffers. These buffers must be pinned in memory and the HCA must have the necessary address information in order to carry out the DMA operation.

5 Micro-benchmark Suite for InfiniBand

In this section we discuss the MIBA micro-benchmark suite. Besides quantifying the performance seen by the user under different circumstances, MIBA is also useful to identify the time spent in each of the components during communication. The micro-benchmark tests can be categorized into two major groups: non-data transfer related micro-benchmarks and data transfer related micro-benchmarks. These categories are discussed in detail in the rest of the section. Note that not all features supported by the IBA specification are available in the current implementations. We have evaluated most of the components that are available. We plan to extend the micro-benchmark suite as more features become available.

5.1 Non-Data Transfer Operations

In this category we measure the costs of the following operations:

Create, Modify and Destroy Work Queues: A Work Queue (or Queue Pair) is the virtual interface that the hardware provides to an IBA consumer, and communication takes place between a source QP and a destination QP. IBA supports various transport services through these QPs. To establish a reliable connection the QP must transit several states. This is established by appropriate *modify* operations on these QPs. To establish a Reliable connection, the *modify* operation is performed as per the IBA specification [10]. Here we measure the cost of setting up and tearing down the connection. The *modify* operation would represent the setting up of connections and the *destroy* operation represents the tearing down of the connection. QP connection does not correlate directly with TCP connection because of protection and other requirements. Note that the cost of such an operation would depend on parameters like the maximum number of WQEs supported by that QP.

Create and Destroy Completion Queues: Completion Queues (CQ) serve as the notification mechanism for the Work Request completions. It can be used to multiplex work completions from multiple work requests across queue pairs on the same HCA. We measure the cost to create and destroy CQs. Again, such a cost will depend on the attributes of the CQ.

Memory Registration and Deregistration: The IBA architecture provides sophisticated high performance operations like RDMA and user mode IO. To manage this, appropriate memory management mechanisms are specified. Memory Registration operation allows consumers to describe a set of virtually contiguous memory locations that can be accessed by the HCA for communication. We measure the cost for registering and deregistering the memory.

Work Request Processing Operations: Work Requests are used to submit units of work to the Channel Interface. Some types of work requests are Send/Receive, RDMA read/write, and Atomic operations. A work request usually triggers communication between the participating nodes. The results from a Work Request operation are placed in a completion Queue Entry. This result can be retrieved by polling the completion queue. We measure the cost of work request operations, polling on completed work request operations, and polling on pending work request operations (empty CQs). The cost indicates the host overhead involved in communication. If the cost is less, then more CPU cycles can be allocated for other computation.

5.2 Data Transfer Operations

In this category, the basic operations which are used for transfer of data are evaluated under different scenarios. The rest of the section describes them in detail.

5.2.1 Basic Tests: These micro-benchmarks are used to find the latency, uni-directional bandwidth, bi-directional bandwidth, and CPU utilization for our base configuration. The base configuration has the following properties: 100% buffer reuse, one data segment, polling on Completion Queue, one connection, no notify mechanism. These properties are described in more detail in later in this section.

Latency Test: Latency measures the time taken for a message of a given size to reach a designated node from the source or the sender node. For measuring the latency, the standard ping-pong test is used. We calculate the latency for both synchronous (Send/Receive on RC) and asynchronous operations (RDMA on RC). The *ping* side posts two work requests: one for send and another for receive. It then polls for the completion of the receive request. The *pong* side posts a receive request, waits for it to complete and then posts a send work request. This entire process is repeated for sufficient number of times (so that the timing error is negligible) from which an average round trip time is produced, then it is divided by two to estimate the one way latency. This test is repeated for different message sizes.

Bandwidth Test: The objective of the bandwidth test is to determine the maximum sustained date rate that can be achieved at the network level. To measure the bandwidth, messages are sent out repeatedly from the sender node to the receiver node for a number of times and then the sender waits for the last message to be acknowledged. The time for sending these back to back messages is measured and the timer is stopped when the acknowledgment for the last message is received. The number of messages being send is kept large enough to make the time for transmission of the acknowledgment of the last message negligible in comparison with the total time.

In order to avoid overloading of the HCA, we use the concept of a *window size* w. Initially w messages are posted. Following which the sender waits for the send completion of $w/2$ messages. Upon completion, another $w/2$ messages are posted. This pattern for waiting for $w/2$ messages and posting $w/2$ messages are repeated sufficient number of times. Since there is always $w/2$ outstanding messages we make sure that the there is sustained data movement on the network. However, if the HCA is faster in dispatching the incoming work requests than the host posting a work request, then there might not be any change in the results for various window sizes.

Bi-directional Bandwidth Test: Networking layer in IBA like any other modern interconnects supports bidirectional traffic in both the directions. The aim of this test is to determine the maximum sustained date rate that can be achieved at the network level both ways. To measure the bidirectional bandwidth, messages are sent out from both sender and receiver repeatedly, both wait on the completion of the last receive. The time for sending these back to back messages is measured. Similar to the bandwidth test, we incorporate *window size* here.

CPU Utilization Test: Higher level applications usually involve a computation cycle followed by communication cycle. If the time spent on communication is small, the valuable CPU cycles can be allocated for useful computation. This raises an important question: *how many CPU cycles are available for computation when communication is performed in tandem?* CPU utilization test is similar to the bi-directional bandwidth test where computation is gradually inserted. Each iteration of a measurement loop includes four steps: post receive work request for expected incoming messages, initiate sends, perform computational work, and finally wait for message transmission to complete. As the amount of work increases, the host CPU fraction available to message passing decreases.

5.2.2 Address Translation A very important component of any user-level communication system is the virtual-to-physical address translation. In InfiniBand, the HCA provides the address translation[21]. In the basic setup, messages are sent from only one buffer. Usually hardware implementations *cache* the physical address of this buffer and hence the cost of virtual-to-physical address translation is not reflected in the latency or bandwidth tests. However by varying the percentage of buffer reused one can see significant difference in the basic test results. Studying the impact of virtual-to-physical address translation can help higher level developer optimize buffer pool and memory management implementations.

To capture the cost of address translation and effectiveness of physical address cache, we have devised two schemes. In Scheme 1, if P is the fraction (or percentage) of buffer reuse then there are $1/P$ buffers used by the test. Access to such buffers are evenly distributed across the basic tests (latency and bandwidth). Here we try to evaluate the effectiveness of the caching scheme. If the cache is effective enough to hold the address of all the $1/P$ buffers then there should be no variation in the results. In Scheme 2, if P is the fraction (or percentage) of buffer reuse and n is the total number of messages communicated between the two sides then n/P messages use the same buffer while $(1 - n/P)$ messages use different buffers. Again, different buffer access are evenly distributed across the test. Here we try to evaluate the cost of virtual-to-physical address translation. As the percentage of buffer reuse decreases, more and more new buffers are accessed.

Illustration: Assume that we have ten buffers numbered 0 to 9 and buffer reuse percentage is 25%. In Scheme 1, the buffer access sequence would be 0, 1, 2, 3, 0, 1, 2, 3,..., and so on. If the cache is big enough to fit all the buffers then there will be no change in the latency and bandwidth numbers. In Scheme 2, the access sequence would be 0, 1, 2, 3, 0, 4, 5, 6, 0, 7, 8, 9,..., and so on. The buffer '0' is reused 25% of the time and the rest of the time different buffers which are not in the cache are used.

5.2.3 Multiple Queue Pairs IBA architecture specification supports 2^{24} QPs. For connection oriented transport services like RC, a QP is bound exclusively for one connection. Hence as the number of connections increases, the number of active QPs increases. Therefore, it is important to see whether the number of active QPs has any effect on the basic performance. This information is important for applications which run on many nodes and there is a need to establish reliable connection between the nodes. This benchmark thus provides valuable information regarding the scalability of the InfiniBand architecture for large scale systems.

5.2.4 Multiple Data Segments IBA supports scatter and gather operations. Many high level communication libraries such as MPI which support gather and scatter operations can use this feature directly. Therefore it is necessary to study the impact of the number of gather and scatter data segments on the basic performance.

5.2.5 Maximum Transfer Unit Size The maximum payload size supported by a particular connection may take any of the following values: 256, 512, 1024, 2048, or 4096 bytes. A smaller memory transfer unit (MTU) may improve the latency for small messages while a larger MTU may increase the bandwidth for larger messages due to smaller overload per payload. Hence depending on the MTU the results of the base tests may vary. Therefore the higher level communication library and applications developers must be aware of such variations. We measure the performance through the basic tests by varying the MTU.

5.2.6 Maximum Scatter and Gather Entries The maximum number of scatter gather entries (SGE) supported by a QP can be specified during creation of that QP. A larger SGE may potentially increase the size of Work Request posted to the HCA. On the other hand, a QP with smaller SGE may not be flexible if the application uses scatter and gather operations of large data segments frequently. Hence it is important that the application developer be aware of this trade-off. We measure the performance by varying the SGE values.

5.2.7 Event Handling IBA also supports event notification. On completion of the work request, a consumer defined event handler is invoked which does the required functions. In our micro-benchmark suite the main thread waits on a semaphore while the event handler signals the semaphore, upon completion of the Work Request operations. Event handling is preferred to polling in scenarios where the application is better off performing other computation rather than waiting on polling. We evaluate the performance when event handling instead of polling is used for the basic tests.

5.2.8 Impact of Load at HCA In all the basic tests only two nodes communicate between themselves and the HCA is exclusively used by the corresponding nodes. An interesting challenge would be to evaluate the performance of the system when a HCA is involved in more that one communication, hence causing contention for HCA resources. The objective here is similar to that of the CPU utilization test. The test is carefully designed so as to avoid contention at the host processors or at the PCI-bus and to create contention only at the HCA. Two nodes (Sender and Receiver) are involved in bandwidth test described previously. Other nodes try to load the HCA of the sender by sending RDMA messages with negligible size. RDMA messages are used because there is no contention at the sender host processor. The message size is chosen to be small (4 bytes in this case) so that the contention at PCI bus at the sender side (also at the switch and wire) are minimal. We measure the results for the basic test by varying the number of other nodes involved in sending RDMA messages to the sender.

6 Performance Evaluation and Discussion

In this section we evaluate VAPI over Mellanox HCA, the currently available implementation of IBA.

6.1 Experimental Testbed

Our experimental testbed consists of a cluster system of 8 SuperMicro SUPER P4DL6 nodes. Each node has dual Intel Xeon 2.40 GHz processors with a 512KB

L2 cache and a 400 MHz front side bus. The machines are connected by Mellanox InfiniHost MT23108 DualPort 4X HCA adapter through an InfiniScale MT43132 Eight 4x Port InfiniBand Switch. The HCA adapters work under the PCI-X 64-bit 133MHz interfaces. The Mellanox InfiniHost HCA SDK build id is thca-x86-0.2.0-build-001. The adapter firmware build id is fw-23108-rel-1_18_0000.

6.2 Non-Data Transfer Operations

The results obtained for non-data transfer benchmarks are presented in Table 1, Figure 2, Figure 3(a), and Figure 3(b). Table 1 summarizes the cost of connection management and work request operations. Connection is established by *modify* QP operation and is destroyed by *destroy* QP operation as described in section 5.1. It is observed that the creation and tearing of connection is costly. When a reliable connection is created or destroyed, the resources for that connection must be allocated or freed. Hence the cost. This provides valuable information to the developers of application which requires dynamic creation of connections. In that case the developer may choose to implement Reliable Datagram (RD) instead of Reliable Connection. Note that RD is not currently supported in available IBA implementation but is expected soon. The cost for posting a work request is less implying that the CPU overhead for communication is less for Mellanox HCAs. Figure 2 shows the cost of memory registration and deregistration. The memory registration cost increases exponentially after 1MB and is around 100 milliseconds for 64MB. Figures 3(a) and 3(b) show the cost of CQ and QP operations with respect to the maximum number of outstanding requests expected on that queue. Note that the QP operations here do not involve setting up of connections hence the cost for QP destroy operation shown in the Figure 3(b) is not as high the cost of the QP destroy operation indicated in Table 1.

Table 1. Non-Data Transfer Micro-Benchmarks

Operation	**Time in μs**
Creating a Connection (*modify QP*)	195.5
Tearing Down Connection (*destroy QP*)	218.2
Posting a Receive Work Request	0.6
Posting a Send Work Request	0.7
Polling on Complete Queue	1.0
Polling on Empty Queue	0.3

6.3 Data Transfer Operations

In this section we present the data-transfer related benchmark results. All the tests use Send-Receive primitives unless explicitly specified as RDMA.

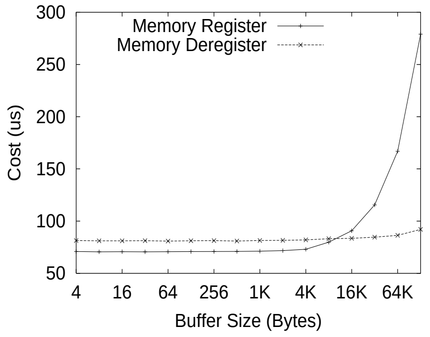
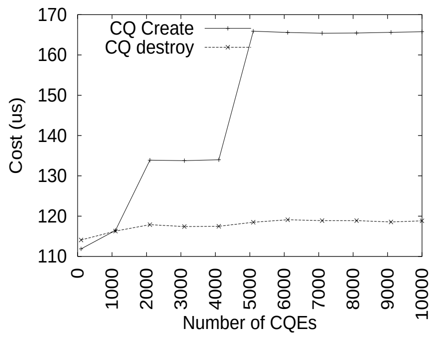
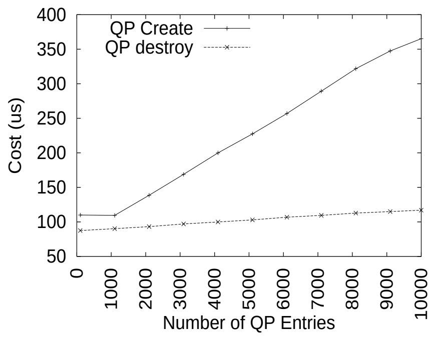


Fig. 2. Cost of Memory Operations

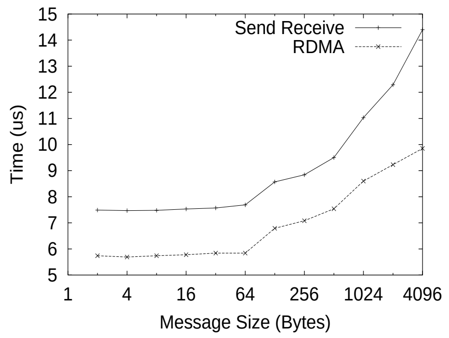


(a) Completion Queue operations

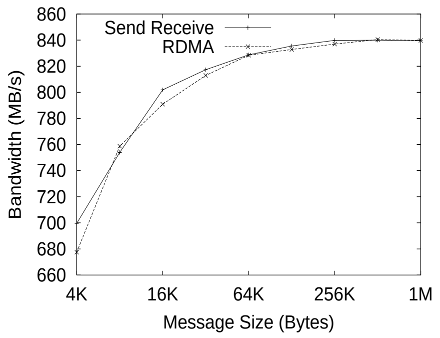


(b) Queue Pair operations

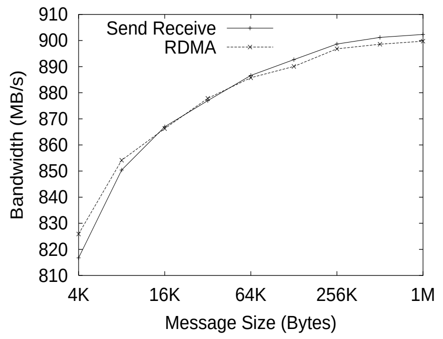
Fig. 3. Cost of CQ and QP operations



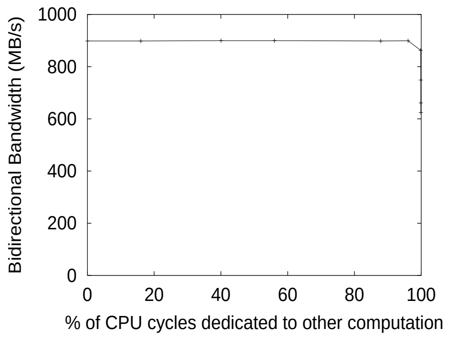
(a) Latency



(b) Bandwidth

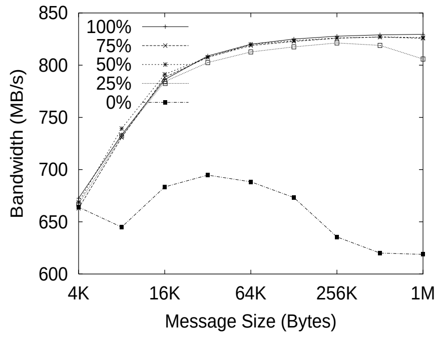


(c) Bi-directional Bandwidth

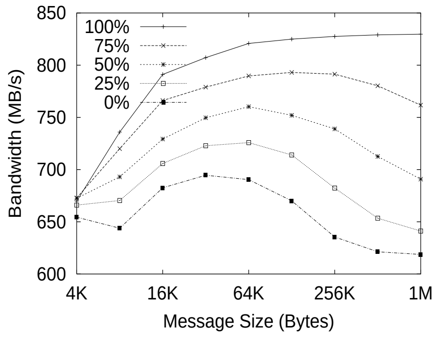


(d) CPU Utilization

Fig. 4. Basic Tests



(a) Bandwidth for Scheme 1



(b) Bandwidth for Scheme 2

Fig. 5. Impact of Virtual-to-Physical Address Translation

6.3.1 Basic Tests Here we present the results for the base settings described in section 5.2.1. The latency and bandwidth results are as shown in Figure 4(a) and Figure 4(b). The one-way RDMA latency is 5.7μs and peak unidirectional bandwidth is around 840MBps. Currently available PCI-X bus supports a bandwidth of around 1GBps. This and the chipset limitations are the reason why the bi-directional bandwidth (Figure 4(c)) is not twice that of the unidirectional bandwidth. There is no variation for different window sizes for both bandwidth and bi-directional bandwidth. Figure 4(d) shows the CPU utilization. The peak bi-directional bandwidth when there is no computation involved is around 900MBps. We increase the computation gradually to see how the communication is affected. From the graph we can see that there is fall in the bandwidth after 96% of CPU cycles are allocated for computation. *We can achieve the peak bandwidth performance even when 96% of the CPU cycles are used for computation.*This shows low CPU Utilization.

6.3.2 Address Translation Figure 5 shows the impact of virtual-to-physical address translation for the two schemes described in section 5.2.2. Scheme 1 shows no decrease in performance for up to 25% of buffer reuse (Figure 5(a)). This is because of the effective caching mechanism by the Mellanox HCAs. Figure 5(b) shows the cost of address translation. As the percentage of buffer reuse is decreased, more and more address translations have to be performed. For large messages, we can notice that there is a drop in the bandwidth values. This is because as the message size increases, it occupies more and more pages and hence requires more entries in the cache increasing the probability of cache misses.

6.3.3 Multiple Queue Pairs This benchmark test shows that there is no difference in the latency and bandwidth numbers as we vary the number of connections established by a node. We varied the number of QP connections up to 64 and the latency and bandwidth numbers remained the same. This shows excellent scalability of the Mellanox HCAs.

6.3.4 Multiple Data Segments This benchmark evaluates the performance of data transfer when multiple data segments are used, as described in section 5.2.4. It is observed that as the number of segments increases, the latency increases. Figure 6 shows the latency for different number of segments. Here each segment is of equal size. In the graph, the total message size (sum of size of all the segments) is plotted on the x-axis and time taken for the latency test is plotted on the y-axis. Note that each data segment has to be copied to the HCA through DMA. Hence as the number of segments increases, the number of DMAs increase. Therefore the performance of PCI and the corresponding chipsets are also major components in the impact of multiple data segments.

6.3.5 Impact of Maximum Transfer Unit size (MTU) This benchmark evaluates the performance of data transfer when MTU values are varied as described in section 5.2.5. Figure 7 shows that smaller MTU values have lower latency for small messages, but the bandwidth for smaller MTU values are significantly less. This is because larger MTU packets have lesser overhead per packet. MTU 1kB performs better than MTU 2048 in bandwidth test. This may be due to effective pipelining for MTU 1kB.

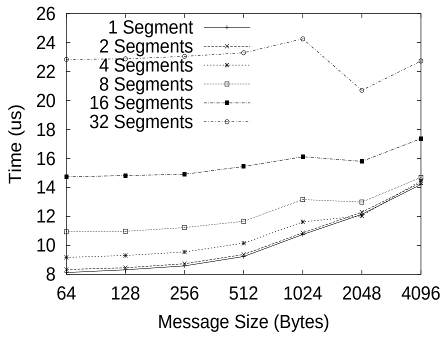
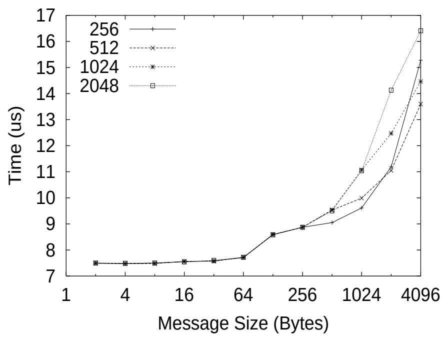
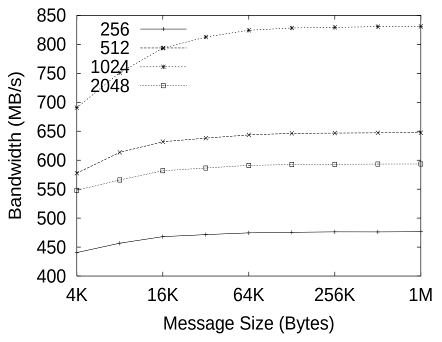


Fig. 6. Impact of Multiple Data Segments

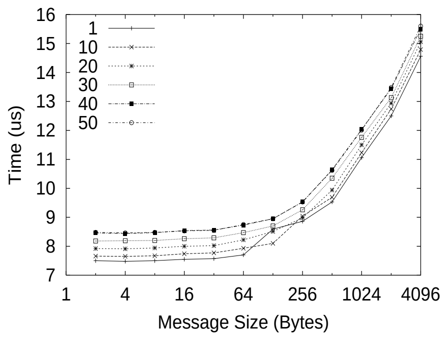


(a) Latency

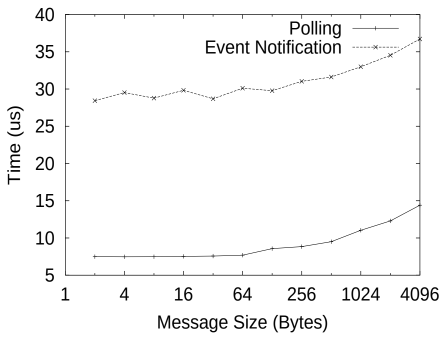


(b) Bandwidth

Fig. 7. Impact of MTU



(a) Maximum SGE



(b) Event Notification

Fig. 8. Impact of SGE and Event Notification on Latency

6.3.6 Maximum Scatter and Gather Entries This benchmark evaluates the performance of data transfer when the maximum SGE supported by a QP is varied. Figure 8(a) shows the impact on the latency as the maximum number of scatter gather entries are varied. There is no significant difference observed for the bandwidth test.

6.3.7 Event Handling Figure 8(b) shows the impact of event notification as compared to polling. We can see that the latency is significantly higher for event notification. This is due to the cost of invoking the event handler upon work completion and subsequent operation on the semaphore to notify the main thread about completion. However, event notification may help certain applications and hence it is important for the developers for such applications to be aware of the cost. No significant difference is noticed for the bandwidth test.

6.3.8 Impact of Load at HCA Figure 9 shows the impact of contention for HCA resources from other communication. The graph is plotted by varying the number of contending nodes. The contending nodes try to load the HCA of the sender node in the basic test as described in section 5.2.8. We can see that as the number of contending nodes increases, the bandwidth drops but not significantly. This shows that the scalability of the HCA with respect to the number of contending nodes.

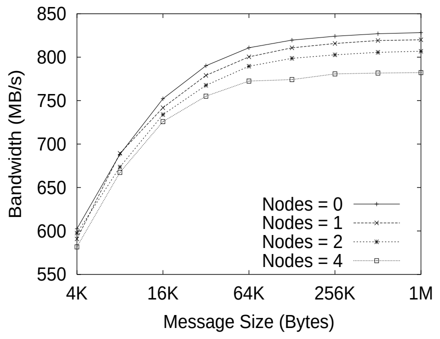


Fig. 9. Impact of contention from other communication on bandwidth

7 Related Work

To the best of our knowledge this is the first attempt to comprehensively evaluate InfiniBand Architecture using a micro-benchmark suite. Our benchmark is based on VIBe [5] Micro-Benchmark developed earlier in our group for VIA architecture. Bell et al [6] used a variant of LogGP [2] model to evaluate several current generation high performance networks like Cray T3E, the IBM SP, Quadrics, Myrinet 2000, and Gigbit Ethernet. They have also compared performance of MPI layer in these networks. NPB [3] benchmarks is an application level benchmark to evaluate the performance the system using MPI. Saavedra et al [15] developed a micro-benchmark to evaluate the memory subsystem of KSR1 architecture. Our micro-benchmark is a more in-depth evaluation at a lower layer API with the focus on IBA.

8 Conclusions and Future Work

In this paper we have proposed a new micro-benchmark suite for evaluating InfiniBand Architecture implementations. In addition to the standard latency and bandwidth test, we have presented several tests that help in obtaining a clear understanding of the implementation details of the components involved in the InfiniBand Architecture. It clearly provides valuable insights for the developers of higher layers and applications over IBA.

IBA products are rapidly maturing. This tool will help hardware vendors to identify the strengths and weaknesses in their releases. As the products are released, more and more features of InfiniBand Architecture will be available. Some of the feature include service levels, virtual level to service level mapping, reliable datagram, partitioning and atomic operations. These features are important for large systems such as cluster based data centers and also for higher level communication libraries such as Message Passing Interface (MPI) standard and distributed shared memory. This micro-benchmark would then provide guidelines to make design choices in the implementation of such systems and libraries. We are planning to extend the micro-benchmark suite in tandem with the development of IBA products.

MIBA software distribution

The code for the benchmark suite described in this paper is available. If you are interested, please contact Prof. D. K. Panda (panda@cis.ohio-state.edu).

Acknowledgments

We would like to thank Jiuxing Liu, Sushmitha Prabhakar Kini, and Jiesheng Wu for their help with the experiments. Our appreciation is also extended to Jeff Kirk and Kevin Deierling from Mellanox Technologies for their insight and technical support on their InfiniBand hardware and software.

References

1. Mellanox Technologies. http://www.mellanox.com.
2. Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. LogGP: Incorporating long messages into the LogP model for parallel computation. *Journal of Parallel and Distributed Computing*, 44(1):71–79, 1997.
3. D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, D. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, Fall 1991.

4. P. Balaji, P. Shivam, P. Wyckoff, and D.K. Panda. High Performance User Level Sockets over Gigabit Ethernet. In *Cluster Computing*, September 2002.
5. M. Banikazemi, J. Liu, S. Kutlug, A. Ramakrishna, P. Sadayappan, H. Sah, and D. K. Panda. Vibe: A micro-benchmark suite for evaluating virtual interface architecture implementations. In *Int'l Parallel and Distributed Processing Symposium (IPDPS)*, April 2001.
6. C. Bell, D. Bonachea, Yannick Cote, Jason Duell, Paul Hargrove, Parry Husbands, Costin Iancu, Michael Welcome, and Katherine Yelick. An evaluation of current high-performance networks. In *International Parallel and Distributed Processing Symposium (IPDPS'03)*, 2003.
7. M. Blumrich, C. Dubnicki, E. W. Felten, K. Li, and M. R. Mesarina. Virtual-Memory-Mapped Network Interfaces. In *IEEE Micro*, pages 21–28, Feb. 1995.
8. Compaq, Intel, and Microsoft. VI Architecture Specification V1.0, December 1997.
9. D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A.M. Merritt, E. Gronke, and C. Dodd. The Virtual Interface Architecture. *IEEE Micro*, pages 66–76, March/April 1998.
10. InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.0, October 24 2000.
11. J. Liu, J. Wu, S. P. Kinis, D. Buntinas, W. Yu, B. Chandrasekaran, R. Noronha, P. Wyckoff, and D. K. Panda. MPI over InfiniBand: Early Experiences. Technical Report, OSU-CISRC-10/02-TR25, Computer and Information Science department, the Ohio State University, January 2003.
12. J. Liu, J. Wu, S. P. Kinis, P. Wyckoff, and D. K. Panda. High Performance RDMA-Based MPI Implementation over InfiniBand. *International Conference on Supercomputing*, June 2003.
13. R. Noronha and D. K. Panda. Implementing TreadMarks over GM on Myrinet: Challenges Design Experience and Performance Evaluation. *Workshop on Communication Architecture for Clusters (CAC'03), to be held in conjuction with IPDPS '03*, April 2003.
14. S. Pakin, M. Lauria, and A. Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM). In *Proceedings of the Supercomputing*, 1995.
15. Rafael H. Saavedra, R. Stockton Gaines, and Michael J. Carlton. Micro benchmark analysis of the KSR1. In *Supercomputing*, pages 202–213, 1993.
16. G. Shah, J. Nieplocha, J. Mirza, C. Kim, R. Harrison, R. K. Govindaraju, K. Gildea, P. DiNicola, and C. Bender. Performance and experience with LAPI - a new high performance communication library for the ibm rs/6000 sp. *International Parallel Processing Symposium*, March 1998.
17. P. Shivam, P. Wyckoff, and D. K. Panda. EMP: zero-copy OS-bypass NIC-driven gigabit ethernet message passing. In *Proceedings of SC '01*, Denver, CO, November 2001.
18. P. Shivam, P. Wyckoff, and D. K. Panda. Can user level protocols take advantage of multi-CPU NICs? In *Proceedings of IPDPS '02*, Ft. Lauderdale, FL, April 2002.
19. T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A User-level Network Interface for Parallel and Distributed Computing. In *ACM Symposium on Operating Systems Principles*, 1995.
20. T. von Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser. Active Messages: A Mechanism for Integrated Communication and Computation. In *International Symposium on Computer Architecture*, pages 256–266, 1992.
21. M. Welsh, A. Basu, and T. von Eicken. Incorporating Memory Management into User-Level Network Interfaces. In *Proceedings of Hot Interconnects V*, Aug. 1997.