Micro-benchmark Level Performance Evaluation and Comparison of High Speed Cluster Interconnects

A Thesis

Presented in Partial Fulfillment of the Requirements for

the Degree Master of Science in the

Graduate School of The Ohio State University

By

Balasubramanian Chandrasekaran, B.E.

* * * * *

The Ohio State University

2003

Master's Examination Committee:

Prof. Dhabaleswar K. Panda, Advisor

Prof. P. Sadayappan

Approved by

Advisor

Department of Computer and Information Science

ABSTRACT

Two high speed interconnects: Myrinet and Quadrics, are widely used in high performance cluster computing. InfiniBand was recently introduced and is being increasingly used as an interconnect for cluster. In this thesis, we provide a set of micro-benchmarks to comprehensively study and characterize different performance aspects of these three interconnects. The micro-benchmark suite includes not only traditional tests and performance parameters, but also those specifically tailored to the interconnects' advanced features such as user-level access for performing communication and remote direct memory access. Our performance evaluation and comparison consists of two parts. First, we evaluate the two APIs for InfiniBand: VAPI and IBAL. We have designed the tests to evaluate and compare the advanced features offered by the InfiniBand Architecture for the two interfaces. Next, we perform a detailed performance comparison of the three interconnects using our micro-benchmarks. In order to explore the full communication capability of the interconnects, we have implemented the micro-benchmark suite at the low level messaging layer provided by each interconnect. Our performance results show that all three interconnects achieve low latency, high bandwidth and low host overhead. However, they show quite different performance behaviors when handling completion notification, unbalanced communication patterns and different communication buffer reuse patterns.

To Amma, Appa, Ammu and Devi

ACKNOWLEDGMENTS

I would like to express my sincere appreciation and gratitude to my advisor Dr. Dhabaleswar K. Panda for his constant guidance and motivation. His valuable support has gone a long way in the completion of this work.

I am grateful to Dr. P Sadayappan for consenting to serve on my Master's examination committee.

For generously sharing their thoughts, ideas and comments, I would like to acknowledge Darius, Gopal, Jiesheng, Jiuxing, Pavan, Savitha, Sushmitha, and Weikuan.

I would like to thank Kevin Deierling, Jeff Kirk and Ezra Silvera from Mellanox Technologies for their support with the InfiniBand hardware and software.

My heartfelt thanks to my family, roommates and friends for their support, love and affection.

VITA

February 13,1980	Born - Chennai, India
1997 - 2001	B.E. Computer Science and Engineering
September 2001 - December 2001 March 2003 - June 2003	Graduate Teaching Assistant, The Ohio State University.
January 2002 - March 2003, June 2003 - present	Graduate Research Associate, The Ohio State University.

PUBLICATIONS

Balasubramanian Chandrasekaran, Pete Wyckoff, and Dhabaleswar K. Panda, "MIBA: A Micro-benchmark Suite for Evaluating InfiniBand Architecture Implementations", *Performance TOOLS 2003* , September 2003.

Jiuxing Liu, Balasubramanian Chandrasekaran, Weikuan Yu, Jiesheng Wu, Darius Buntinas, Sushmitha P. Kini, Pete Wyckoff, and Dhabaleswar K. Panda, "Micro-Benchmark Level Performance Comparison of High-Speed Cluster Interconnects", *Hot Interconnects 11* , August 2003.

Jiuxing Liu, Balasubramanian Chandrasekaran, Jiesheng Wu, Weihang Jiang, Sushmitha Kini, Weikuan Yu, Darius Buntinas, Pete Wyckoff, and Dhabaleswar K. Panda, "Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics", *In SuperComputing 2003 Conference* , November 2003.

Sandhya Senapathi, Balasubramanian Chandrasekaran, Don Stredney, Han-Wei Shen, and Dhabaleswar K. Panda, "QoS-aware Middleware for Cluster-based Servers to Support Interactive and Resource-Adaptive Applications", *In High Performance Distributed Computing 12* , June 2003.

FIELDS OF STUDY

Major Field: Computer and Information Science

Studies in High Performance Computing: Prof. D.K. Panda

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Cluster computing systems are becoming increasingly popular for day-to-day computational needs. This is due to the availablity of cost-effective and affordable commodity components[17, 15]. The idea is to use Commodity of the self (COTS) components and build a cluster, which not only satisfies the computational requirements of various applications but also provides a cost effective solution. With the rapid advances in networking and processor technologies, High Performance Computing Clusters (HPCC) has become an attractive and alternate choice for High Peformance Computing (HPC).

A high performance computing cluster consists of several computing nodes connected by a high-speed interconnect. The computing node consists of one or more central processing unit (CPU), memory, and I/O devices including a Network Interface Card (NIC) for communication. Two high speed interconnects, Myrinet[3] and Quadrics[14], are widely used in designing such high performance clusters. Recently, InfiniBand[5] has been introduced and is being increasingly used as an interconnect for clusters. These interconnects provide a rich set of advanced features such as memory protected user-level access to the network interface[16] and remote direct memory access (RDMA). These high speed interconnects are designed to achieve low latency

(less than 10μs) and high bandwidth (of the order of Gbps). Most of the communication overhead can be offloaded to the sophisticated NICs present in the computing nodes.

Such high performance is made available directly to the applications by the efficient implementation of higher-level programming models such as Message Passing Interface (MPI)[11], high performance user-level sockets, and distributed shared memory. Also, upper layers such as file systems and data center servers can use the features offered by these high speed interconnects. Due to the advanced and rich set of features offered by these interconnects, the designer of such an upper layer now has various design choices to make. Therefore, an interesting question ariese: *How to choose an interconnect and how to come up with a good design choice based on the communication characteristics of the upper layer and the interconnect?* In order to answer these questions one has to have a detailed understanding of the communications characteristics of all the interconnects.

Some interconnects provide more than one interface for user-level access. For example, InfiniBand as two interfaces VAPI and IBAL. The two APIs follow Verbs as described InfiniBand Architecture specification[6]. Verbs API (VAPI) was developed by Mellanox[10]. It closely follows InfiniBand Architecture specification and offers high performance. InfiniBand Access Layer (IBAL) was developed by Sourceforge[7]. It is a software component that provides an interface to the InfiniBand fabric to multiple concurrent users. This interface exposes the capabilities of the InfiniBand Architecture and augments the verbs software transport interface with support for higher-level operations required by most users of InfiniBand. Other advanced components such as SDP, IPoIB, OpenSM (a subnet manager for IBA) are developed on top

of this Access Layer. The interesting question here is: *How can we conduct a meaningful performance comparison among the two interfaces of InfiniBand Architecture?*

Standard tests such as ping-pong latency and bandwidth, which are used to evaluate the interfaces and interconnects, 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. Similarly, the latency test results for a scenario where the buffers are reused might be significantly different from latency test results for a scenario where buffers are not reused at all.

Another way to evaluate different user-level interface and network interconnects is to use real world applications. However, real applications usually run on top of a middleware layer such as the Message Passing Interface (MPI). Therefore, the performance we see reflects not only the capability of the network interconnects, but also the quality of the MPI implementation and the design choices made by different MPI implementers. Thus, to provide more insights into the communication capability offered by each interconnect, it is desirable to conduct tests at a lower level.

In this thesis, we propose a set of micro-benchmarks to evaluate and compare the the user-level interfaces and high speed interconnects. The thesis can be broadly divided into two categories.

1.1 Comparison of InfiniBand Architecture over VAPI and IBAL

Using a set of micro-benchmarks we evalaute and compare the InfiniBand Architecture implementations for VAPI and IBAL. In addition to the standard tests, we have also included micro-benchmarks which evaluate the advanced features provided by the InfiniBand.

InfiniBand Architecture was initially designed for Data Centers. The specification offers a wide range of features and services. This coupled with high performance makes it a good high speed interconnect for cluster. Our micro-benchmark tests are tailored to evalaute the advanced features offered by InfiniBand Architecture such as connection management and scatter-gatter capabilites. The micro-benchmark suite is developed based on VIBe[1], which was developed earlier in our group.

1.2 Comparison of InfiniBand, Myrinet, and Quadrics

We extend the benchmarks to do a comprehesive performance evaluation of the three high speed interconnects: InfiniBand, Myrinet, and Quadrics. Our micro-benchmarks are relatively simple and well-defined. Therefore it is possible to implement them directly on the low level messaging layer provided by all the interconnects. The benchmarks also concentrate on the remote memory access capabilities provided by each interconnect. Our micro-benchmark suite also includes hot-spot tests that reveal performance characteristics of different interconnects that are difficult to get by a simple two-node test.

The rest of the thesis is organized as follows. In Chapter 2, we provide a detailed description of all the Micro-benchmarks. In Chapter 3,We provide a detailed peformance evaluation of the InfiniBand Architecture with respect to the two interfaces: VAPI and IBAL. In Chapter 4, we provide the performance comparison of the three high speed interconnets: InfiniBand, Myrinet and Quadrics. Conclusions and ongoing work are discussed in Chapter 5.

CHAPTER 2

MICRO-BENCHMARKS

In this chapter, we provide a comprehensive description of the micro-benchmarks that are used to compare and evaluate the three interconnects: InfiniBand, Myrinet, and Quadrics. The algorithms and the scenarios for each micro-benchmark is discussed in detail. We not only include the traditional tests such as latency and bandwidth tests but also include tests which evaluate the advanced features of these modern interconnects. The tests can be broadly classified as 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 connections, memory registration and deregistrations, and posting send and receive descriptors. 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 the three interconnects.

2.1 Non-Data Transfer Operations

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

2.1.1 Memory Registration and De-registration

All the three interconnects support user-level access to the network and advanced communication semantics such as RDMA. However, in order to take advantage of these features the memory used for communication must be registered or *pinned*. To do this, appropriate memory management mechanisms are specified by the corresponding APIs. Memory Registration operation allows the user to describe a set of virtually contiguous memory locations that can be accessed by the NIC for communication. We measure the cost for registering and deregistering the memory.

2.1.2 Descriptor Operations

Send and receive descriptors are submitted to the NIC to request send and receive of messages. Some types of send and receive descriptors are Send/Receive, RDMA read/write, and Atomic operations. Posting of send and receive requests usually triggers communication between the participating nodes. Completion of the communication request can be detected by various mechanisms. For InfiniBand, the notification is through Completion Queues. The user can poll on the completion queue to detect completion of communication. For Quadrics and Myrinet, notification is through events. The user can poll on the event to detect completion.

We measure the cost of posting of send and receive descriptors. We also measure the cost of polling on queues and events to detect completion. The cost indicates the

host overhead involved in communication. If the cost is less, then more CPU cycles can be allocated for other computation.

2.1.3 Connection Operations

InfiniBand supports Reliable connection for communication. Connections are managed by Queue Pairs (QP) and Completion Queues (CQ) in IBA. In this micro-benchmark we measure the cost of creating and destroying a connections. This cost is critical for applications which makes dynamic connections. Based on the result for this test the application can choose either static or dynamic connections.

2.2 Data Transfer Operations

In this category we measure the cost of data transfer related operations. The micro-benchmarks include the traditional tests such as latency, bandwidth, bi-directional latency, and bi-directional bandwidth. It also includes tests which evaluate the interconnects in relatively complex communication scenarios thereby evaluating different components in the communication subsystem. The base configuration has the following properties: 100% buffer reuse, one data segment, polling for completion detection, one connection, and no notify mechanism. These properties are described in more detail later in this section.

2.2.1 Latency

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) and asynchronous operations (RDMA). Note that Quadrics supports only RDMA

operation. The *ping* side posts two descriptors: one for send and another for receive. It then polls for the completion of the receive request. The *pong* side posts a receive descriptor, waits for it to complete and then posts a send descriptor. 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.

2.2.2 Bandwidth

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 NIC, we use the concept of a *Window size W*. A similar method has been used in [2]. 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 at least $W/2$ outstanding messages we make sure that the there is sustained data movement on the network. Also, at any point in time there can be at most W outstanding messages and hence this makes sure that the NIC is not overloaded. However, if the

NIC is faster in dispatching the incoming requests than the host posting the messages, then there might not be any change in the results for various window sizes.

2.2.3 Bi-directional Latency and Bandwidth Test

All the modern interconnects supports traffic in both the directions. Bi-directional latency and bandwidth put more stress on the communication subsystem as compared to the uni-directional tests. Therefore they may be more helpful in understanding the bottleneck in communication. In bi-directional latency, test both sides send simultaneously and wait on the receive. The time between sending the message and receiving the message is measured. The aim of the bi-directional bandwidth 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.

2.2.4 Host Overhead in Communication

Latency and Bandwidth tests measure the efficiency of the interconnects. Another important factor in communication is the host involvement in the communication. The less the host CPU is involved in communication, more the time it can spend on other useful computation. All the interconnects have advanced network interface cards and most of the communication overhead is transfered to the NIC. This raises an important question: *how many CPU cycles are available for computation when communication is performed in tandem?*

In this test, we measure the host overhead in communication in both latency and bi-directional bandwidth test. For latency test, we just directly measure the time for posting a descriptor and polling for completion. For bandwidth test, we gradually

insert computation between the communication. Each iteration of a measurement loop includes four steps: posting receive descriptors for expected incoming messages, initiate sends, perform computational work, and finally wait for message transmission to complete. As the amount of work increases, the fraction of the host CPU available for communication decreases. This is similar to a general loop in a higher level application, which usually involves a computation cycle followed by a communication cycle. If the time spent on communication is small, then valuable CPU cycles can be allocated for useful computation.

2.2.5 Overhead in Completion Notification

All the three interconnects support remote direct memory access (RDMA) operations. The arrival of RDMA messages can be detected by various ways. One way to detect the arrival of messages at the receiver side is to poll on the memory content in the destination buffer. This approach can be used to minimize the receiver overhead. However, this method is hardware dependent because it relies on the order in which the DMA controller writes to host memory.

In this test, we measure the cost of notification mechanism for RDMA messages in each of the interconnects. The network interconnects we have studied support different mechanisms to report the completion of remote memory operations. For example, InfiniBand uses Completion Queues, while Quadrics rely on event abstractions. Myrinet does not have a notification mechanism for RDMA messages. Therefore, we simulated the notification by using a separate send operation. The receiver side polls on this message to detect for the completion of the RDMA message.

2.2.6 Overhead in Blocking

As seen in the previous section, polling is a mechanism that is used for completion notification. However, polling burns CPU cycles and is not good for multitasking. Polling is good in scenarios where a set of nodes is allocated exclusively for a particular application. However, in scenarios like Data Center where there are multiple applications sharing the same CPU, polling is not a good idea. The application can then use interrupts instead of polling. However, there is additional overhead involved in interrupts. In this test we measure the cost of using interrupts.

2.2.7 Impact of Buffer Reuse

A very important component of any user-level communication system is the virtual-to-physical address translation[18]. In the latency and bandwidth tests, 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 the physical address cache, we have devised two schemes. In Scheme 1, we vary the working set for the buffers. The number of buffers, W, used for the test is varied. If there are N iterations in the test, then these W buffers ($W \leq N$) are used in a round robin fashion. Here we try to evaluate the effectiveness of the caching scheme. If the cache is effective

enough to hold the address of all the W buffers then there should be no variation in the results.

In Scheme 2, *we vary the buffer reuse rate.* If R is the fraction (or rate) of buffer reuse and N is the total number of messages communicated between the two sides (number of iterations) then N/R messages use the same buffer while $(1 - N/R)$ messages use different buffers. Again, different buffer accesses 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. In Scheme 1, if the number of buffers $W = 4$, then 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, if the buffer reuse rate $R = 25\%$, then 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.

2.2.8 Hot-Spot Test

In all the basic tests only two nodes communicate between themselves and the NIC is exclusively used by the corresponding nodes. An interesting challenge would be to evaluate the performance of the system when a NIC is involved in more that one communication, hence causing contention for NIC resources. We have designed hot spot tests to evaluate the interconnect under unbalanced communication scenarios. We have implemented two hot spot tests. Similar tests have been conducted in [13].

Hot Spot Send Test

In hot-spot send test, a master node sends a message to all the slave nodes and receives an acknowledgment from one of the slave nodes. The test is repeated by varying the number of slave nodes. Here the 'hot-spot' is at the sending side.

Hot Spot Send and Receive

In hot-spot send-receive test, a master node sends a note (small message) to all the slave nodes and receives one message each from all the slave nodes. The test is repeated by varying the number of slave nodes. Here the 'hot-spot' is at the receiving side.

2.2.9 Impact of Multiple Connections

For connection oriented communications like in InfiniBand, the latency and bandwidth may depend on the number of open connections. Connections are represented by Queue Pairs (QP) in InfiniBand. 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. Applications usually run on many nodes and there is a need to establish connections between the nodes. This benchmark thus provides valuable information regarding the scalability of the InfiniBand architecture for large scale systems.

2.2.10 Impact of Multiple Data Segments

InfiniBand supports scatter and gather operations. Gather scatter operations can be used to send non-contiguous memory segments as a single message. This avoids

posting of multiple descriptors or copying of data segments. Many high level communication libraries such as ARMCI[12] 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.

The maximum number of scatter gather entries (SGE) supported by a connection must be specified during the creation of that connection. A larger SGE may potentially increase the size of descriptor posted to the NIC for all messages in that connection. This may increase the latency for all messages in that connection. On the other hand, a connection 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 basic performance of latency and bandwidth on one data segment by varying the maximum number of scatter gather entries supported by that connection.

We also measure the latency when multiple data segments are used. Note that even though gather and scatter operations avoid posting of multiple descriptors, an additional DMA is always required for every new data segment. This cost is measured in the test by varying the number of data segments.

2.2.11 Impact of Maximum Transfer Unit Size

InfiniBand allows an user to specify the memory transfer unit (MTU) value when a connection is created. 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 may improve the pipelining of 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 latency and bandwidth tests may vary. Therefore the higher level communication library and applications developers must be aware of such variations. We measure the performance of basic latency and bandwidth test by varying the MTU.

CHAPTER 3

EVALUATION OF INFINIBAND ARCHITECTURE IMPLEMENTATIONS (VAPI AND IBAL)

In this chapter we evaluate the InfiniBand Architecture implementation for VAPI and IBAL[4]. In addition to the standard tests, our micro-benchmarks evaluate the advanced features offered by InfiniBand. The micro-benchmark tests can be categorized into two major groups: non-data transfer related micro-benchmarks and data transfer related micro-benchmarks, as described in chapter 2.

3.1 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-x86-1.0.0-build-001. The adapter firmware build id is fw-23108-rel-2_00_0000.

3.2 Non-Data Transfer Operations

The results obtained for non-data transfer benchmarks described in section 2.1 are presented in Table 3.1. It summarizes the cost of memory operations, posting of descriptors and connection management operations. Connection is established by *modify* QP operation and is destroyed by *destroy* QP operation provided by VAPI and IBAL.

InfiniBand, like most modern interconnects, require that the communication memory must be registered. We measure the cost of memory registration and de-registration in both VAPI and IBAL. The cost of memory operations is around 70-80 μs for both VAPI and IBAL. If this cost is too high for the application then the application can use pinned-down cache scheme and lazy de-registration scheme.

The cost for posting a descriptor and polling is less for both VAPI and IBAL implying that the CPU overhead for communication is less for InfiniBand Architecture. Because of this low overhead in communication, the host CPU cycles can be spent on useful computation. Note that the cost for posting a descriptor and polling is slightly higher for IBAL as compared to VAPI.

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 high 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.

Table 3.1: Non-Data Transfer Micro-Benchmarks (Time in μs)

Operation	**VAPI**	**IBAL**
Memory registration (4KB)	73.06	80.56
Memory de-registration (4KB)	82.05	70.84
Posting a Receive Descriptor	0.56	0.84
Posting a Send Descriptor	0.72	1.33
Polling on Complete Queue	1.08	1.11
Polling on Empty Queue	0.28	0.43
Creating a Connection (*modify QP*)	183.15	282.61
Tearing Down Connection (*destroy QP*)	198.63	256.89

3.3 Data Transfer Operations

In this section we present the data-transfer related benchmark results.

3.3.1 Latency

Standard *ping-pong* test described in section 2.2.1 is used to measure the latency. Both VAPI and IBAL support Send-Receive (S/R) and RDMA. The latency results are as shown in Figure 3.1. The one-way VAPI latency for RDMA and Send-Receive is $5.5\mu s$ and $7.5\mu s$ respectively. For IBAL, the RDMA latency is $6\mu s$ and Send-Receive latency is $8.3\mu s$. We can see that for both the interface RDMA performs better than Send-Receive. RDMA is asynchronous and is transparent to the receiver. Hence it avoids receive side overhead and consequently performs better as compared to Send-Receive.

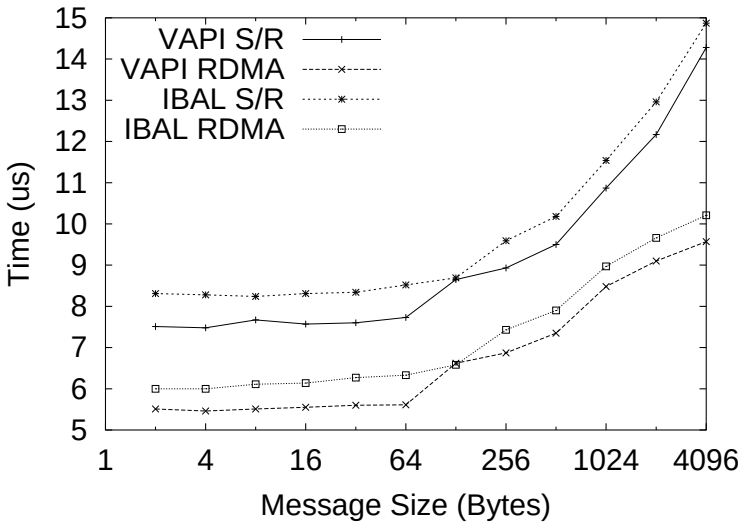


Figure 3.1: Latency

3.3.2 Bandwidth

We measure bandwidth for both Send-Receive and RDMA over VAPI and IBAL. Figure 3.2 gives the bandwidth results. Both Send-Receive and RDMA shows comparable performance. The peak bandwidth for VAPI is 838 MBps and for IBAL it is 833 MBps. In sections 3.2 and 3.3.1, we have seen that IBAL has additional overhead as compared to VAPI. Due to this additional overhead IBAL has lesser bandwidth as compared to VAPI for message sizes between 1 KB and 64 KB.

3.3.3 Bi-directional Latency and Bandwidth

Bi-directional tests put more stress on the interconnect as compared to uni-directional tests. From Figure 3.3(a), we can see that the bi-directional latency performance is worse as compared to the uni-directional tests. The bi-directional bandwidth performance is comparable for VAPI and IBAL for both Send-Receive

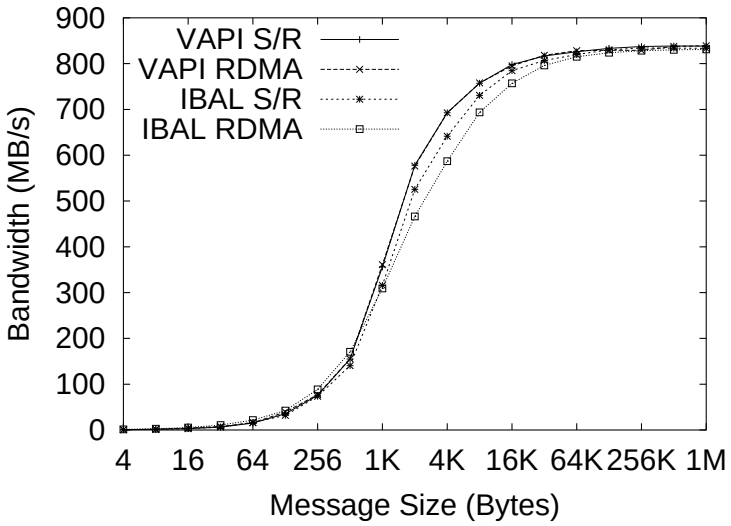


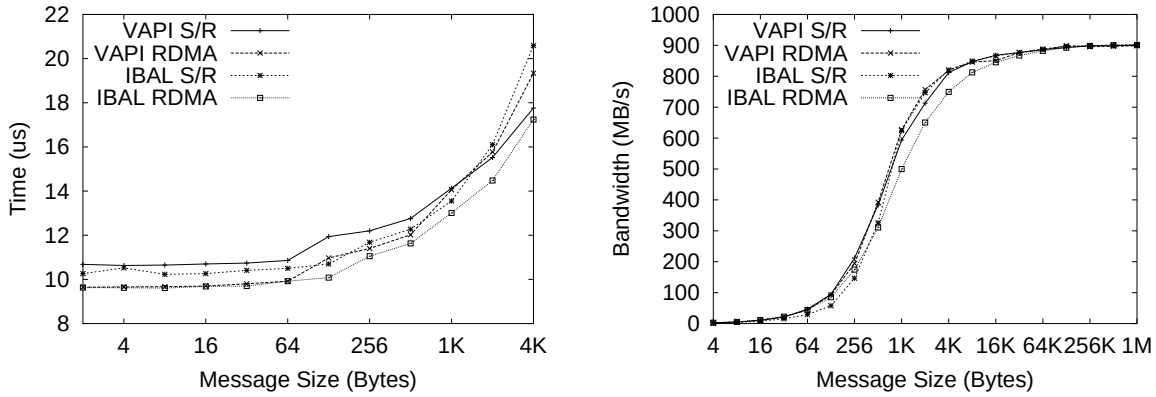
Figure 3.2: Bandwidth

and RDMA (Figure 3.3(b)). The peak bi-directional bandwidth is around 902 MBps. Currently available PCI-X bus supports a bandwidth of around 1GBps. This and the chipset limitations are the reason why the bi-directional bandwidth is not twice that of the unidirectional bandwidth.

3.3.4 Host Overhead in Communication

Figure 3.4(a) shows the host CPU overhead in the latency test. Clearly IBAL has a higher overhead as compared to VAPI. This difference in overhead is the cause of the difference in the basic latency of VAPI and IBAL.

Figure 3.4(b) shows the CPU utilization. The peak bi-directional bandwidth when there is no computation involved is around 902 MBps. We increase the computation gradually to see how the communication is affected. From the graph we can see that there is a drop in the bandwidth after 99.5% of CPU cycles are allocated for



(a) Bi-directional Latency

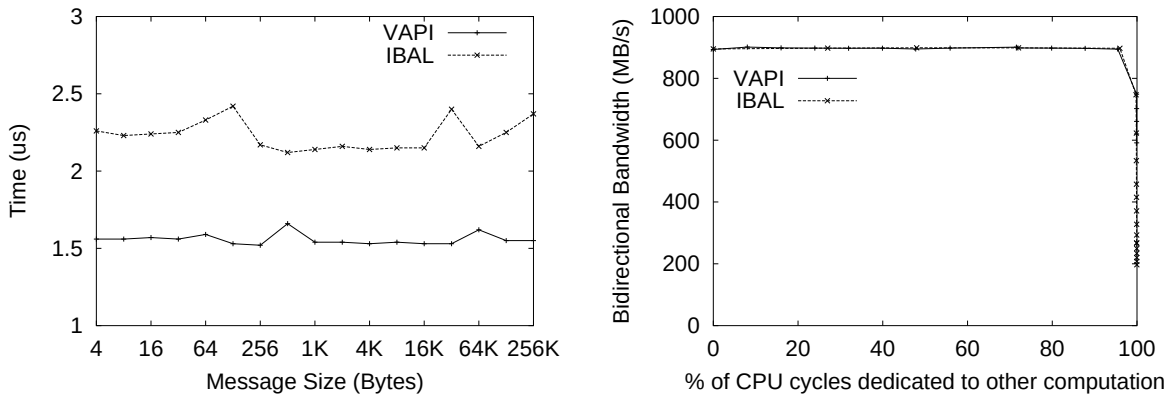
(b) Bi-directional Bandwidth

Figure 3.3: Bi-directional Tests

computation. *We can achieve the peak bandwidth performance even when 99.5% of the CPU cycles are used for computation.*This shows low CPU Utilization for both VAPI and IBAL.

3.3.5 Overhead in Completion Notification

InfiniBand supports special communication primitive called *RDMA Immediate* in addition to RDMA and Send-Receive. *RDMA Immediate* can be used to notify the receiver about the completion of a RDMA operation. This notification is useful in scenarios where the RDMA operation should not be transparent to the receiver as described in section 2.2.5. We measure the overhead due to notification for both VAPI and IBAL. Figure 3.5 shows that overhead is comparable for the two interfaces.



(a) Latency Overhead

(b) CPU Utilization

Figure 3.4: Host Overhead in Communication

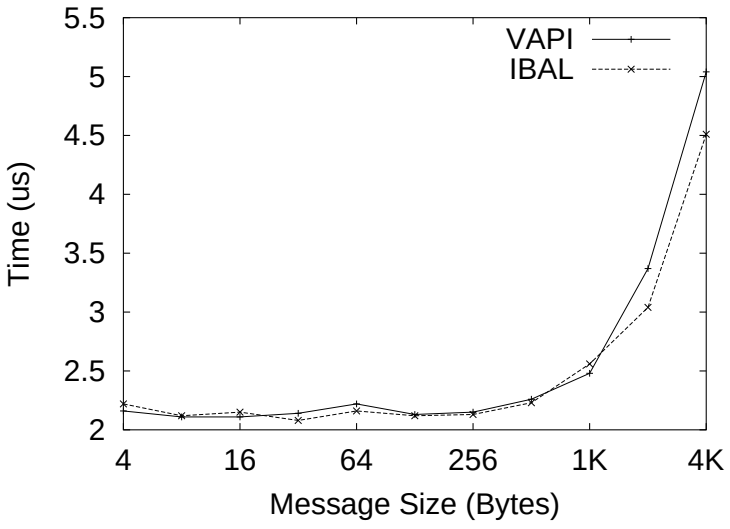


Figure 3.5: Overhead in Completion Notification

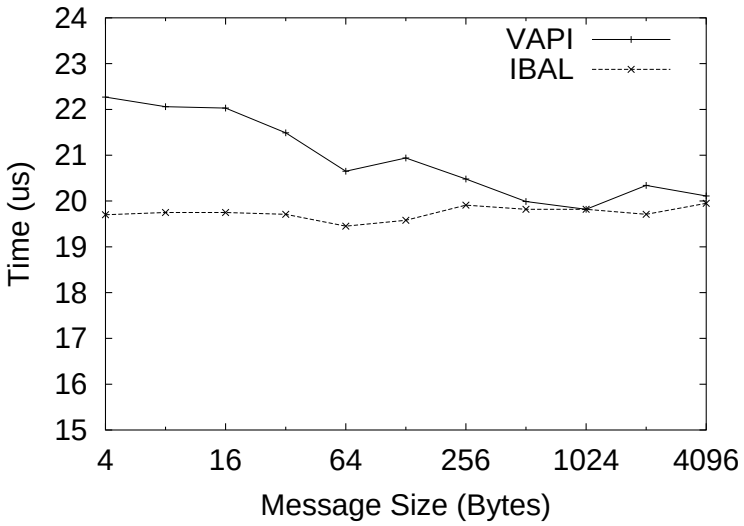


Figure 3.6: Overhead in Blocking

3.3.6 Overhead in Blocking

Figure 3.6 shows the impact of blocking as compared to polling for VAPI and IBAL. 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. IBAL has less overhead as compared to VAPI for interrupt handling.

3.3.7 Impact of Buffer reuse

Figure 3.7 shows the impact of virtual-to-physical address translation for the two schemes described in section 2.2.7. Both VAPI and IBAL shows the same trend. This

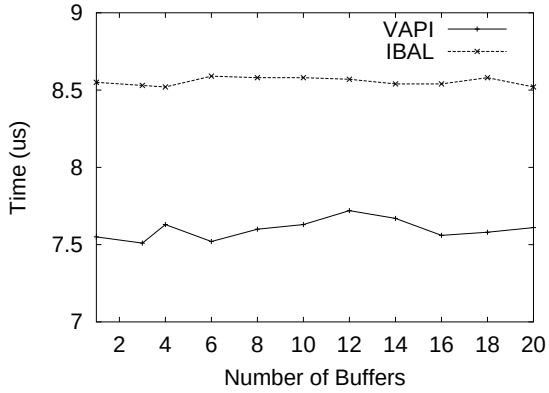
is because, virtual-to-physical address translation is done by the NIC independent of the API.

In both Schemes there is no decrease in latency for small messages. For small messages, the number of virtual to physical address translation is less and hence the number cache entries is less. We can see a drop in the bandwidth in both the schemes in Figure 3.7. This shows the efficiency of the address translation cache in the NIC. The bandwidth falls sharply after five buffers (each of size 512 KBytes). This gives the optimal working set for the buffers. Figure 3.7(b) shows the cost of address translation. As the percentage of buffer reuse is decreased, more and more address translations have to be performed.

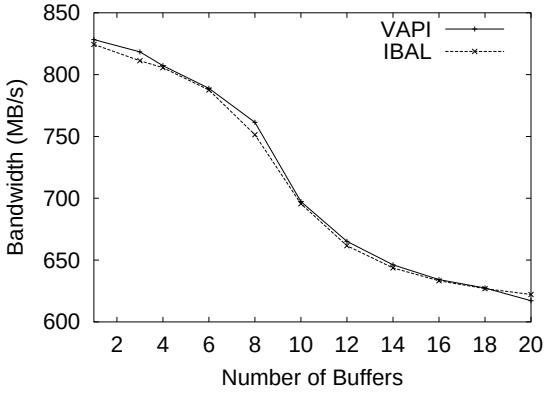
3.3.8 Impact of Multiple Connections

InfiniBand supports connection oriented communication. As the number of connections increases, the number of active Queue Pairs increases. All the active Queue Pairs are usually associated with only one Completion Queue. The polling time on the Completion Queue may increase with increase in the number of Queue Pairs associated with it. Hence, as the number of connections increases, there may be a drop in the performance with respect to latency and bandwidth.

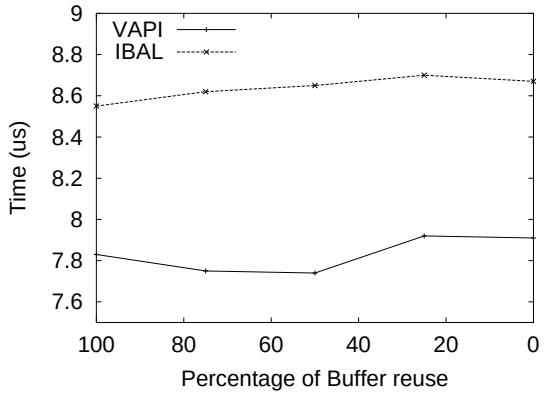
However, Figures 3.8 and 3.9 show that there is no difference in the latency and bandwidth numbers as we vary the number of connections established by a node. We varied the number of QP connections up to 64 and the latency and bandwidth numbers remained the same. This shows excellent scalability of the InfiniBand Architecture.



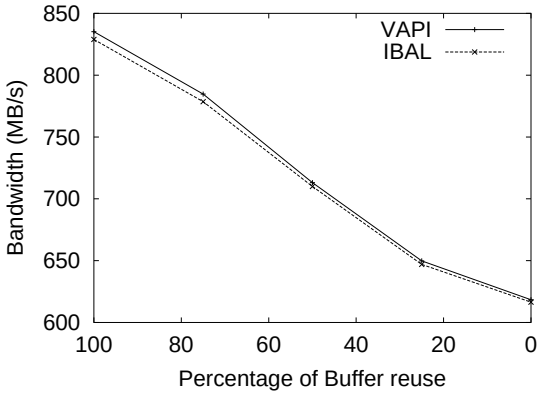
(a) Latency (Size=64 Bytes) for Scheme 1



(b) Bandwidth (Size=512 KB) for Scheme 1

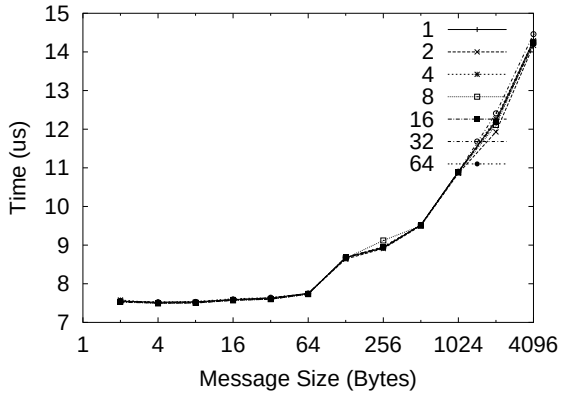


(c) Latency (Size=64 Bytes) for Scheme 2

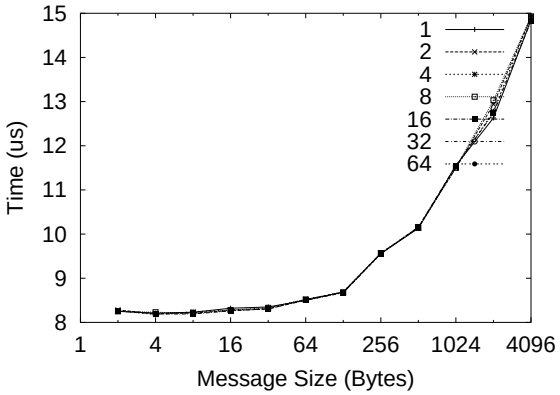


(d) Bandwidth (Size=512 KB) for Scheme 2

Figure 3.7: Impact of Buffer Reuse

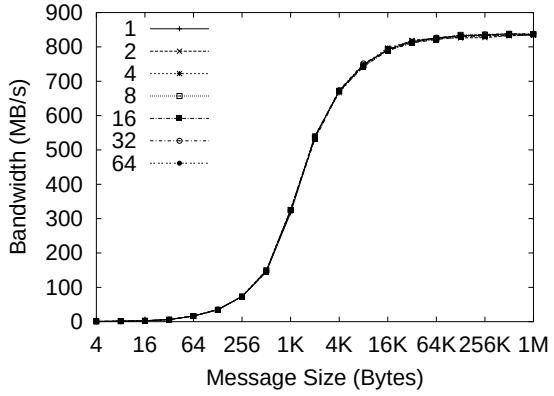


(a) VAPI

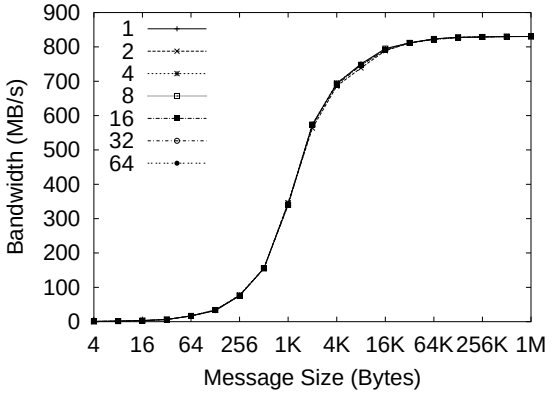


(b) IBAL

Figure 3.8: Impact of Multiple Connections on Latency



(a) VAPI



(b) IBAL

Figure 3.9: Impact of Multiple Connections on Bandwidth

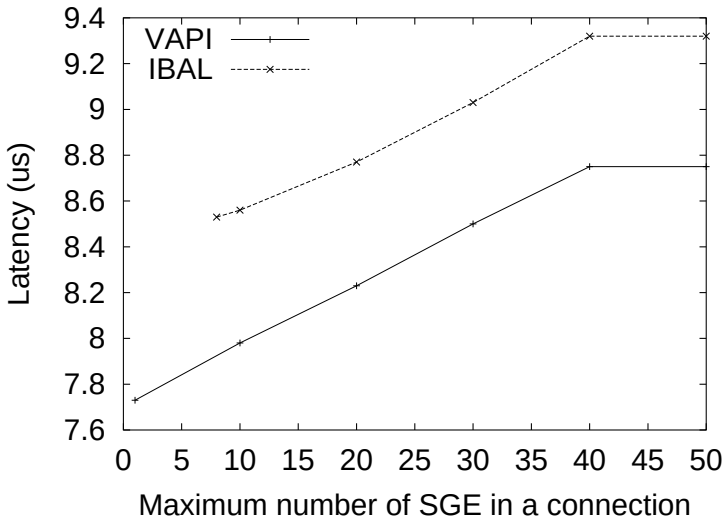
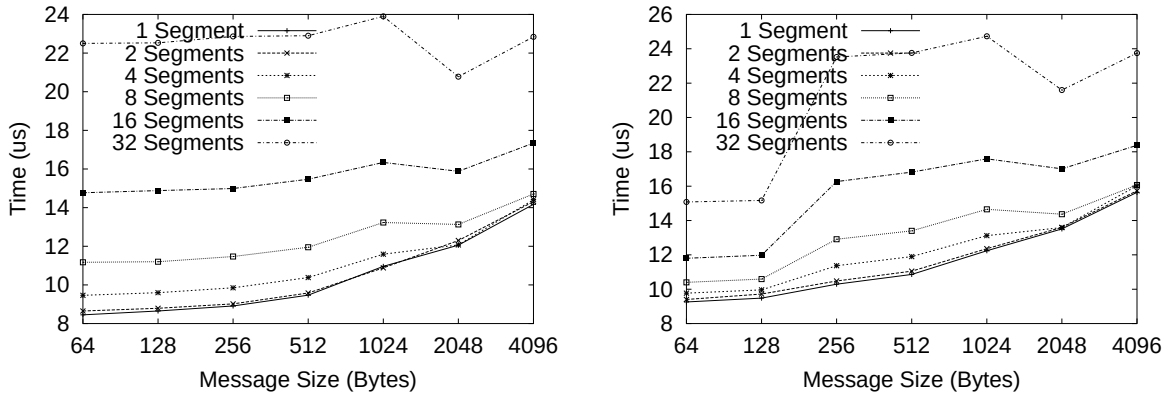


Figure 3.10: Impact of Maximum SG Entries in a QP on the Latency for a 64 Byte message

3.3.9 Impact of Multiple Data Segments

This benchmark evaluates the performance of data transfer when the number of Scatter Gather Entries (SGE) supported by a connection is varied and when multiple data segments are used, as described in section 2.2.10. Figure 3.10 shows the impact on the latency as the maximum number of scatter gather entries are varied for both VAPI and IBAL[1]. As the number of SGE supported by a connection increases, there is an increase in the descriptor size for every message posted in that connection. This increase in the descriptor size increases the latency for all messages posted in that connection. There is no significant difference observed for the bandwidth test because of pipelining.

[1]Due to a bug in the driver, IBAL only supports connections with SGE greater that or equal to eight.



(a) VAPI

(b) IBAL

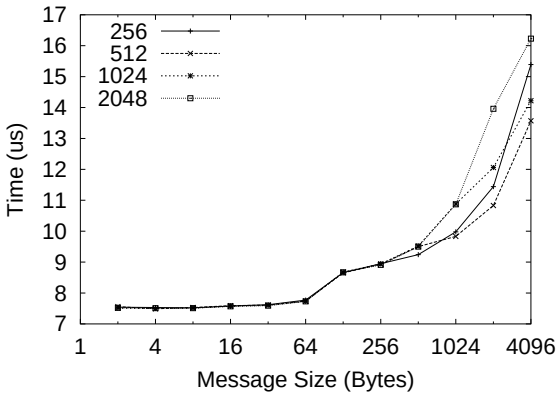
Figure 3.11: Impact of Multiple Data Segments

It is observed that as the number of segments increases, the latency increases for both VAPI and IBAL. Figure 3.11 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 NIC 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.

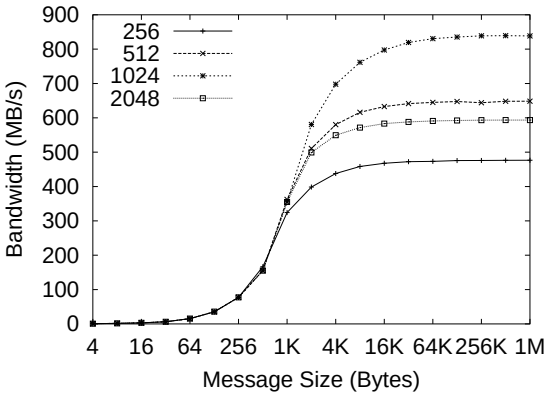
3.3.10 Impact of Maximum Transfer Unit size

This benchmark evaluates the performance of data transfer when MTU values are varied as described in section 2.2.11. Figure 3.12 shows that there is not much difference in latency for small messages, but the bandwidth for smaller MTU values are significantly less. This is because larger MTU packets have lesser overhead per

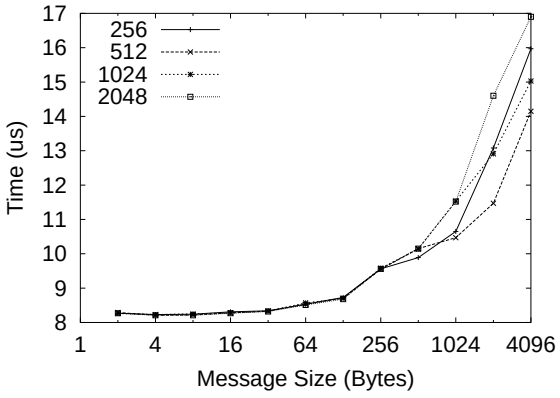
packet. MTU 1KB performs better than MTU 2048 in bandwidth test. This is due to effective pipelining for MTU 1KB.



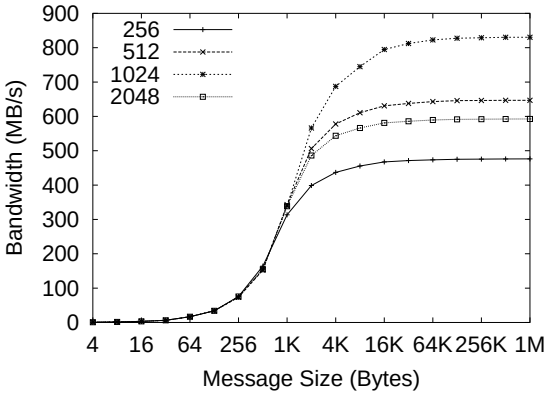
(a) Latency for VAPI



(b) Bandwidth for VAPI



(c) Latency for IBAL



(d) Bandwidth for IBAL

Figure 3.12: Impact of MTU

CHAPTER 4

PERFORMANCE COMPARISON OF INFINIBAND, MYRINET, AND QUADRICS

To provide more insights into the communication behavior of the three interconnects, we use our micro-benchmarks described in chapter 2 and study their performance characteristics. In this chapter, we evaluate InfiniBand, Myrinet, and Quadrics using the micro-benchmarks [9]. For InfiniBand, we have used VAPI interface for the micro-benchmarks.

4.1 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 were connected by InfiniBand, Myrinet, and Quadrics interconnects. For InfiniBand, we used Mellanox InfiniHost MT23108 DualPort 4X HCA adapter through an InfiniScale MT43132 Eight 4x Port InfiniBand Switch. The InfiniHost HCA adapters and Myrinet NICs work under the PCI-X 64-bit 133MHz interfaces. The Mellanox InfiniHost HCA SDK build id is thca-x86-1.0.0-build-001. The adapter firmware build id is fw-23108-rel-2_00_0000-rc12-build-002. For Myrinet, we used 225MHz Lanai-XP processors through a 8-port

Table 4.1: Non-Data Transfer Micro-Benchmarks (Time in μs)

Operation	**InfiniBand**	**Myrinet**	**Quadrics**
Memory registration (4KB)	73.06	2.42	-
Memory de-registration (4KB)	82.05	101.80	-
Posting a Receive Descriptor	0.56	0.17	0.12
Posting a Send Descriptor	0.72	0.18	0.25
Polling on Queue	1.08	0.88	0.30

Myrinet-2000 switch. The Quadrics Elan3 QM-400 cards were attached to the 8 nodes and connected through an Elite16 switch. The Quadrics cards use 64-bit 66MHz PCI slots. We used the Linux Red Hat 7.2 operating system.

4.2 Non-Data Transfer Operations

In this section, we measure the cost of non data transfer operations for the three interconnects. Non-data transfer operations include the cost of memory registration and de-registration, posting of send and receive descriptor and polling. Table 4.1 summarizes the results. We can see that InfiniBand memory registration and de-registration cost is higher as compared to that of Myrinet. In Quadrics, the user need not register the communication buffer because the NIC is capable of triggering a page-fault if that buffer is not in physical memory. For the descriptor and polling operations, we again see that InfiniBand as higher overhead as compared to Myrinet and Quadrics.

4.3 Data Transfer Operations

In this section, we measure the cost of the following data transfer operations:

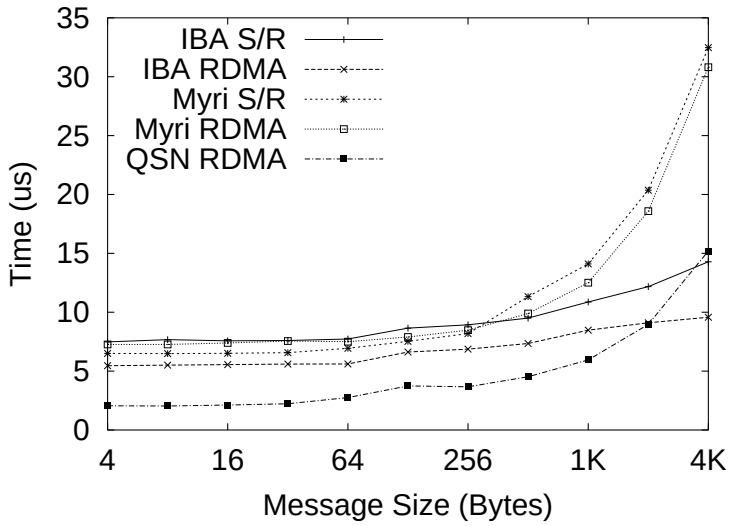


Figure 4.1: Latency

4.3.1 Latency

End-to-end latency, as described in section 2.2.1, has been frequently used to characterize the performance of interconnects. All the interconnects under study support access to remote nodes' memory space. Thus, we also measured the latency to finish a remote put operation. InfiniBand/VAPI and Myrinet/GM also support send/receive operations. Figure 4.1 shows the latency results. For small message, Quadrics/Elanlib achieves the best latency, which is 2.0μs. InfiniBand RDMA latency is around 5.5μs and send/receive latency is around 7.5μs. Myrinet has a small message latency of about 6.5μs for send/receive. Its RDMA (directed send) has a slightly higher latency of 7.3μs. For messages less than 64 bytes, Myrinet send/receive can combine data and descriptor at the sender side. Therefore it offers better performance than directed send.

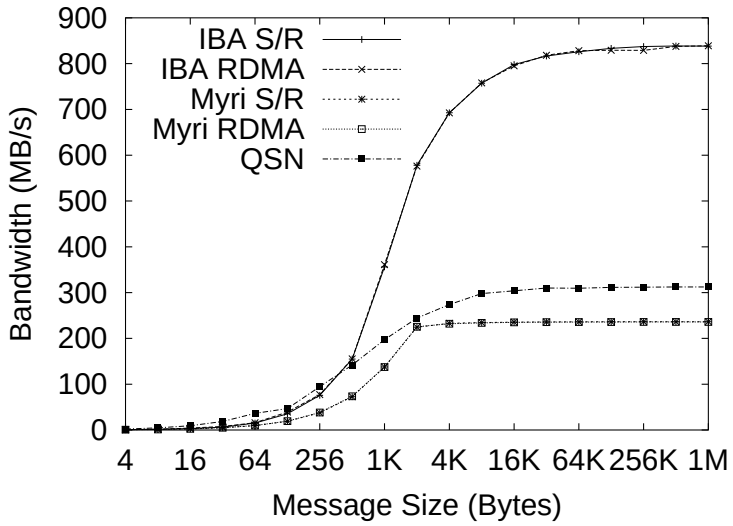


Figure 4.2: Bandwidth

4.3.2 Bandwidth

The bandwidth test is used to determine the maximum sustained data rate that can be achieved at the network level. In this test, a sender keeps sending back-to-back messages to the receiver until it has reached a pre-defined window size W, as described in section 2.2.2. Then it waits for W/2 messages to finish and sends out another W/2 messages. In this way, the sender ensures that there are at least W/2 and at most W outstanding messages. Figure 4.2 shows the bandwidth results with very large window size. Figure 4.3 shows the bandwidth with different window size.

The peak bandwidth for InfiniBand, Myrinet, and Quadrics is around 838MB/s, 236MB/s and 314MB/s, respectively. We can see that InfiniBand is more sensitive to the value of window size W and it performs much better for large messages.

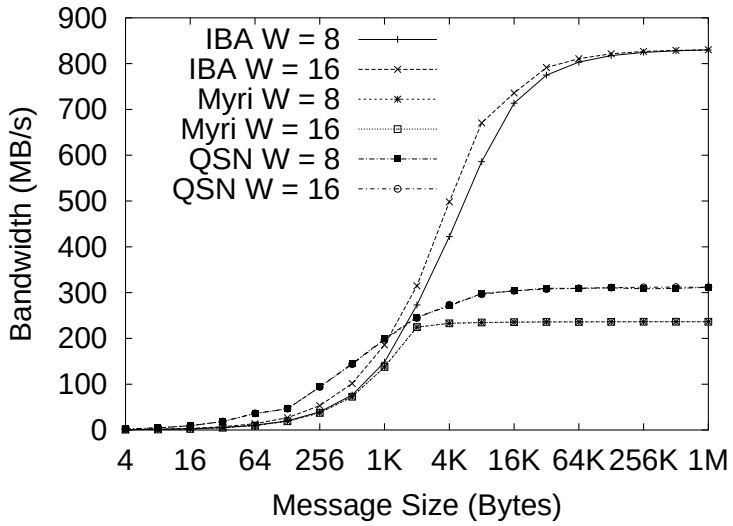


Figure 4.3: Bandwidth with Window Size

4.3.3 Bi-directional Latency and Bandwidth

Compared with uni-directional latency and bandwidth tests, bi-directional latency and bandwidth tests put more stress on the PCI bus, the network interface cards, and the switches. Therefore they may be more helpful to us to understand the bottleneck in communication. The tests are carried out in a way described in section 2.2.3, similar to the uni-directional tests. The difference is that both sides send data simultaneously. From Figure 4.4, we can see bi-directional latency performance for all interconnects is worse than their uni-directional latency except for Quadrics. Figure 4.5 shows results for bandwidth. We see that for InfiniBand, the PCI-X bus becomes the bottleneck and limits the bandwidth to around 902MB/s. Although Quadrics has better uni-directional bandwidth than Myrinet, its peak bi-directional bandwidth is only around 319MB/s, which is less than Myrinet's 471MB/s.

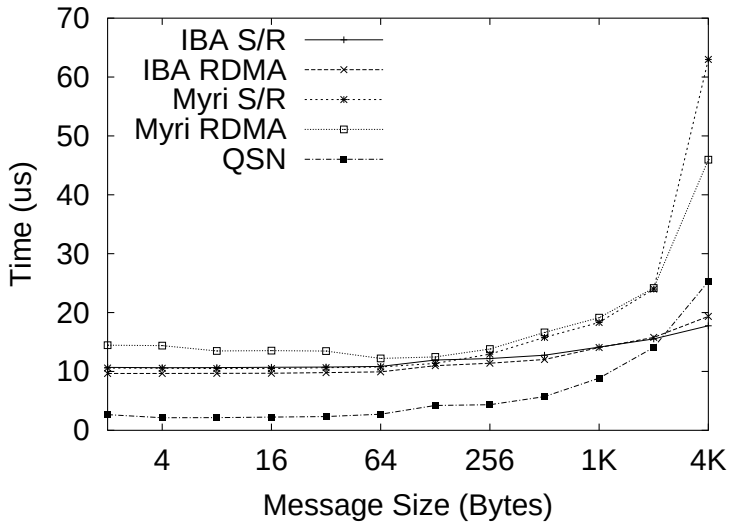


Figure 4.4: Bi-directional Latency

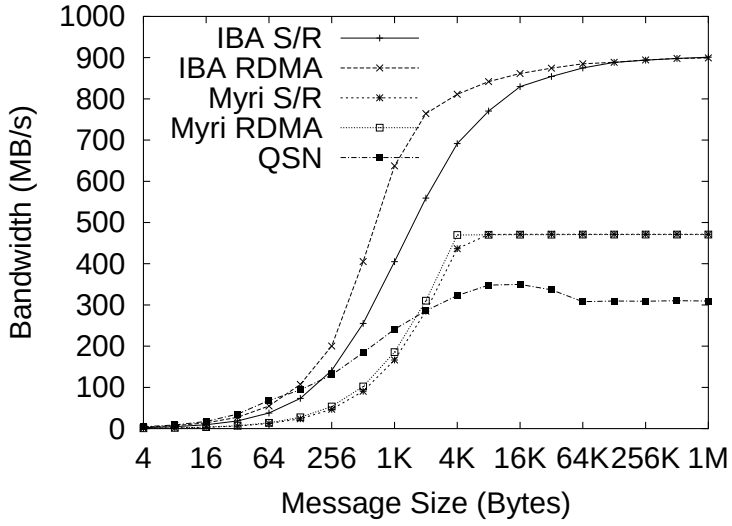


Figure 4.5: Bi-directional Bandwidth

4.3.4 Host Overhead in Communication

We define host communication overhead as the time CPU spends on communication tasks. The more time CPU spends in communication, the less time it can do computation. Therefore this can serve as a measurement of the ability of a messaging layer to overlap communication and computation. We characterize the host overhead for both latency and bandwidth tests, as described in section 2.2.4. In the latency test, we directly measure the CPU overhead for different message sizes. In the bandwidth test, we insert a computation loop in the program. By increasing the time of this computation loop, eventually we see a drop in the bandwidth.

Figure 4.6 presents the host overhead in the latency test. InfiniBand has the highest overhead, which is 1.5μs. Quadrics overhead is around 0.7μs. Myrinet has the least overhead, which is around 0.5μs. Myrinet reduces the overhead further for messages less than 64 bytes by combining data and send descriptors. Figure 4.7 shows the impact of computation time on bandwidth. All three interconnects can overlap communication and computation quite well. Their bandwidth drops only after over 99% of running time is used for computation.

4.3.5 Overhead of Completion Notification

The network interconnects we have studied support different mechanisms to report the completion of remote memory operations. For example, InfiniBand uses CQ, while Myrinet and Quadrics rely on event abstractions. We measure the cost of completion notification as described in section 2.2.5. Figure 4.8 shows the increase in latency when using these mechanisms for remote memory access at the receiver side. Quadrics has very efficient notification mechanism, which adds only 0.4μs overhead for large

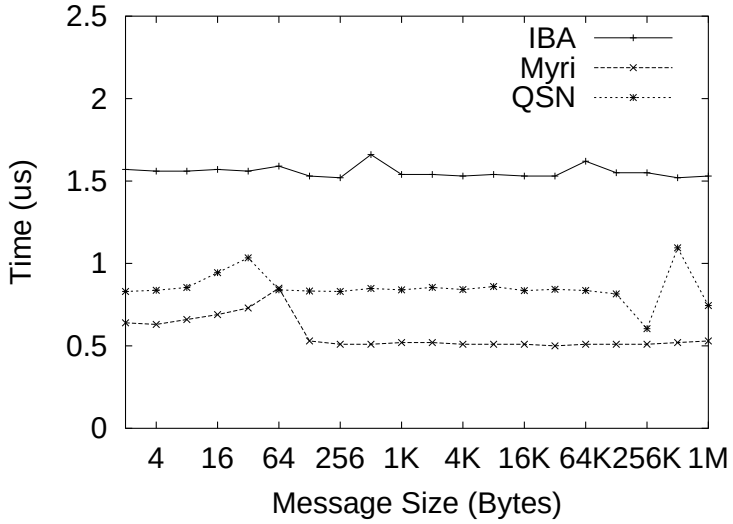


Figure 4.6: Host Overhead in Latency Test

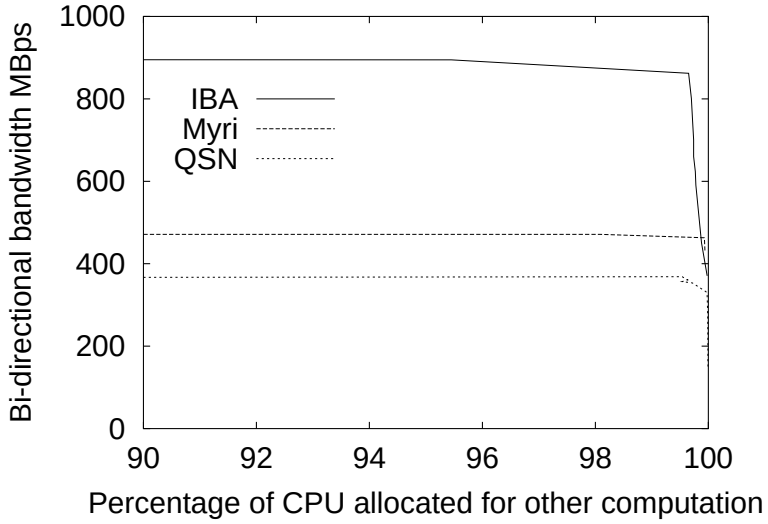


Figure 4.7: CPU Utilization in Bandwidth Test

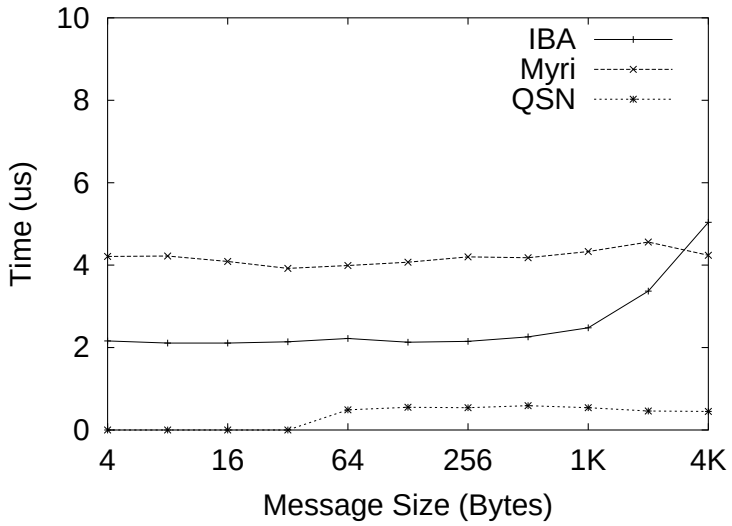


Figure 4.8: Overhead due to Completion

messages. For messages less than 64 bytes, there is no extra overhead. InfiniBand has an overhead of around 2.1μs. Myrinet GM's directed send does not have a mechanism to notify the receiver of message arrival. Therefore, we simulated the notification by using a separate send operation. This adds around 3-5μs overhead.

4.3.6 Overhead in Blocking

Instead of busy polling, the upper layer can also use blocking to wait for completions. We measure the cost of interrupt in this micro-benchmark as described in section 2.2.6. From Figure 4.9 we can observe that InfiniBand has the highest overhead, which is over 20μs. The overheads for Myrinet and Quadrics are about 11μs and 13μs, respectively.

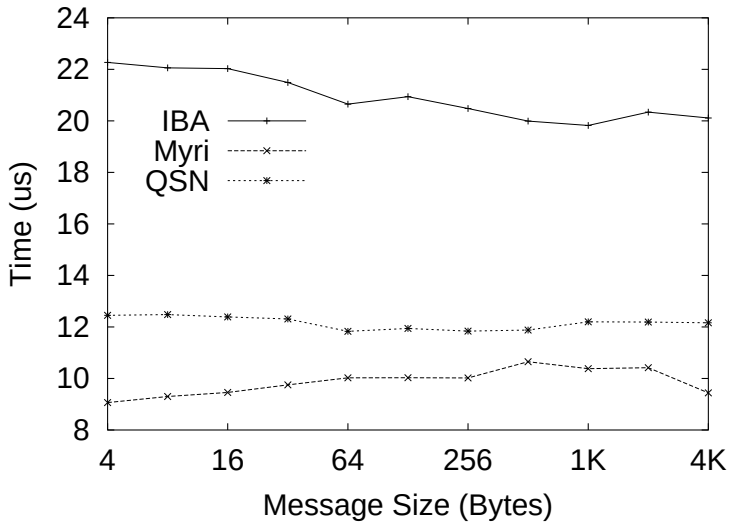


Figure 4.9: Overhead due to Blocking

4.3.7 Impact of Buffer Reuse

In most micro-benchmarks that are designed to test communication performance, only one buffer is used at the sender side and the receiver side, respectively. However, in real applications a large number of different buffers are usually used for communication. The buffer reuse pattern can have a significant impact on the performance of interconnects that support user-level access to network interfaces such as those studied in this paper.

To capture the cost of address translation at the network interface, we have designed two schemes of buffer reuse pattern as described in section 2.2.7. In scheme 1, N different buffers of the same size are used in FIFO order for multiple iterations. By increasing the number N, it may happen that eventually the performance drops. Basically, this test shows how many different communication buffers can be handled at the same time in the network interface without degrading performance.

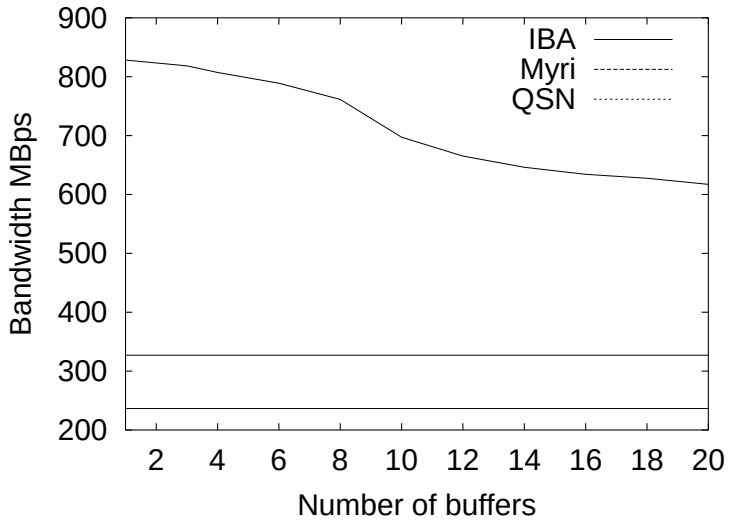


Figure 4.10: Bandwidth (size=512K) Buffer Scheme 1

Figure 4.10 shows the bandwidth results with 512KBytes messages. We can see that up to 25 buffers, Myrinet and Quadrics show no performance degradation. However, InfiniBand performance drops when more than 10 buffers are used.

Scheme 2 is slightly more complicated. In this scheme, the test consists of N iterations and we define a buffer reuse percentage R. For the N iterations of the test, N*R iterations will use the same buffer, while all other iterations will use completely different buffers. By changing buffer reuse percentage R, we can see how communication performance is affected by buffer reuse patterns. From Figures 4.11 and 4.12, we can see that Quadrics is very sensitive to buffer reuse patterns. Its performance drops significantly when the buffer reuse rate decreases. InfiniBand also shows similar behavior. Myrinet latency increases slightly when the buffer reuse rate decreases, but its bandwidth performance is not sensitive to the buffer reuse rate.

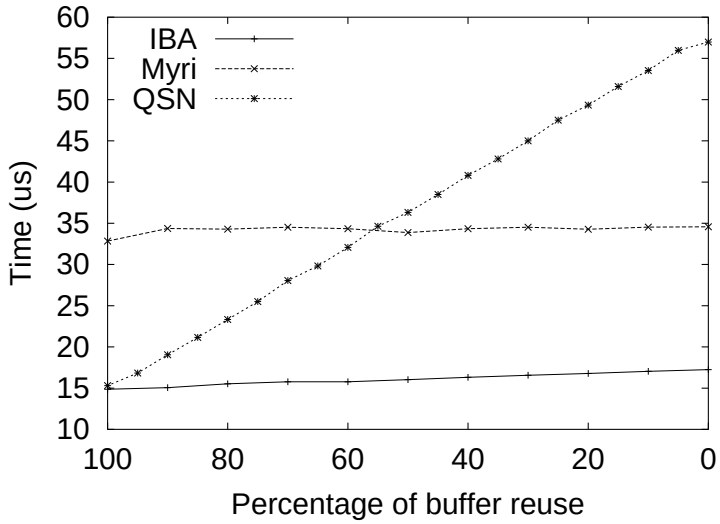


Figure 4.11: Latency (size=4K) Buffer Scheme 2

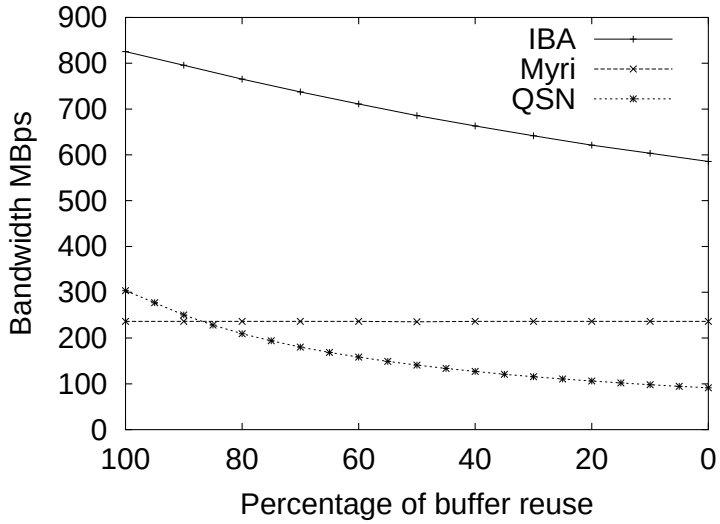


Figure 4.12: Bandwidth (size=512K) Buffer Scheme 2

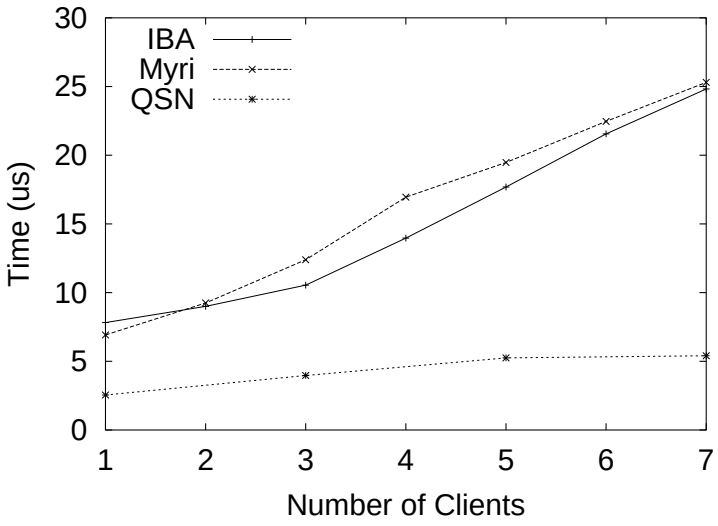


Figure 4.13: Hot Spot Send Test

4.3.8 Hot-Spot Tests

Hot-spot tests are designed to measure the ability of network interconnects to handle unbalanced communication patterns. We have used two sets of hot-spot tests as described in section 2.2.8. In hot-spot send tests, a master node keeps sending messages to a number of different slave nodes. In hot-spot receive tests, the master node receives messages from all the slave nodes. We vary the number of slave nodes. Figures 4.13 and 4.14 show the hot spot performance results. We can see that Quadrics scales very good when the number of slaves increases. On the other handle, InfiniBand and Myrinet do not scale very well.

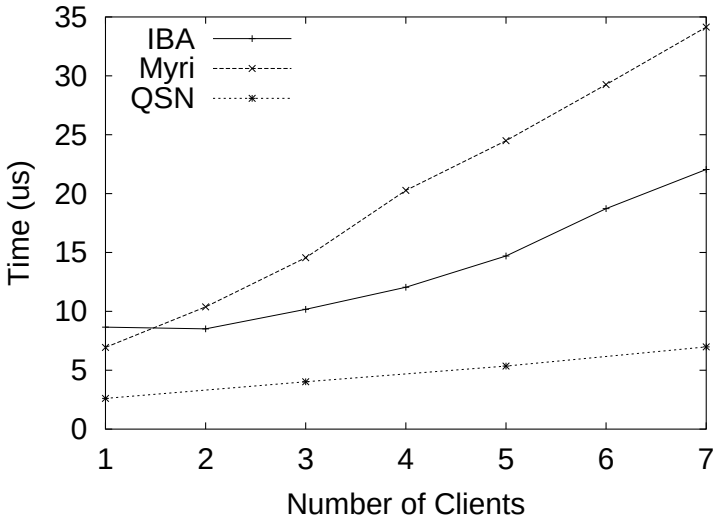


Figure 4.14: Hot Spot Send-Receive Test

CHAPTER 5

CONCLUSIONS

In this thesis, we have used a set of micro-benchmarks to evaluate three high performance cluster interconnects: InfiniBand, Myrinet and Quadrics. We provide a detailed performance evaluation for their communication performance by using a set of micro-benchmarks. We show that in order to get more insights into the performance characteristics of these interconnects, it is important to go beyond simple tests such as latency and bandwidth. Specifically, we need to consider the performance impact of certain features such as remote memory access, completion notification and address translation mechanisms in the network interface.

We further used the micro-benchmark suite to evaluate the two user-level interfaces of InfiniBand: VAPI and IBAL. We designed micro-benchmarks which evaluated the advanced features of InfiniBand Architecture. Several tests were presented that help in obtaining a clear understanding of the implementation details of the components involved in the InfiniBand Architecture. It clearly provides insights for the developers of higher layers and applications over IBA.

5.1 Ongoing Work

We have extended the micro-benchmarks to the MPI layer. We have presented a comprehensive performance evaluation of MPI Implementations over InfiniBand, Myrinet and Quadrics [8]. Our performance evaluation consists of two major parts. The first part consists of a set of MPI level micro-benchmarks that characterize different aspects of MPI implementations. The second part of the performance evaluation consists of application level benchmarks. We have used the NAS Parallel Benchmarks and the sweep3D benchmark. We not only present the overall performance results, but also relate application communication characteristics to the information we acquired from the micro-benchmarks. Our results show that the three MPI implementations all have their advantages and disadvantages.

5.2 Future Work

The micro-benchmark presented in this thesis gives valuable guidelines to the upper layer developers. The suite can further be extended by adding tests which are specifically tailored to a particular domain such as distributed shared memory, data centers, and socket direct protocol. Such a micro-benchmark suite will evaluate an interconnect relative to the communication characteristics of the domain and give more specific information about the interconnect for that particular domain.

InfiniBand products are rapidly maturing and new IBA adapters and interfaces are expected in the near future. The micro-benchmarks presented in this thesis can easily be extended to evaluate upcoming new implementations of InfiniBand Architecture.

BIBLIOGRAPHY

[1] M. Banikazemi, J. Liu, S. Kutlug, A. Ramakrishna, P. Sadayappan, H. Shah, and D. K. Panda. VIBe: A Micro-benchmark Suite for Evaluating Virtual Interface Architecture (VIA) Implementations. In *IPDPS*, April 2001.

[2] Christian Bell, Dan 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)*, April 2003.

[3] N. J. Boden, D. Cohen, et al. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, pages 29–35, Feb 1995.

[4] Balasubramanian Chandrasekaran, Pete Wyckoff, and Dhabaleswar K. Panda. MIBA: A Micro-benchmark Suite for Evaluating InfiniBand Architecture Implementations. In *Performance TOOLS 2003*, September 2003.

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

[6] InfiniBand Trade Association, InfiniBand Architecture Specification, Volume 1, Release 1.0. http://www.infinibandta.com.

[7] Linux InfiniBand Project. http://infiniband.sourceforge.net.

[8] Jiuxing Liu, Balasubramanian Chandrasekaran, Jiesheng Wu, Weihang Jiang, Sushmitha Kini, Weikuan Yu, Darius Buntinas, Pete Wyckoff, , and Dhabaleswar K. Panda. Performance Comparison of MPI Implementations over InfiniBand, Myrinet and Quadrics. In *SuperComputing 2003 Conference*, November 2003.

[9] Jiuxing Liu, Balasubramanian Chandrasekaran, Weikuan Yu, Jiesheng Wu, Darius Buntinas, Sushmitha P. Kini, Pete Wyckoff, and Dhabaleswar K. Panda. Micro-benchmark level performance comparison of high-speed cluster interconnects. In *Hot Interconnects 11*, August 2003.

[10] Mellanox Technologies. http://www.mellanox.com.

[11] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*, Mar 1994.

[12] J. Nieplocha and B. Carpenter. ARMCI: A portable remote memory copy library for distributed array libraries and compiler run-time systems. *Lecture Notes in Computer Science*, 1586, 1999.

[13] Fabrizio Petrini, Adolfy Hoisie, Wu chun Feng, and Richard Graham. Performance Evaluation of the Quadrics Interconnection Network. In *Workshop on Communication Architecture for Clusters 2001 (CAC '01)*, April 2001.

[14] Quadrics, Ltd. http://www.quadrics.com.

[15] T. Sterling, D. Savarese, D. J. Becker, J. E. Dorband, U. A. Ranawake, and C. V. Packer. BEOWULF: A parallel workstation for scientific computation. In *Proceedings of the 24th International Conference on Parallel Processing*, pages I:11–14, Oconomowoc, WI, 1995.

[16] 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.

[17] M. Warren, D. Becker, M. Goda, J. Salmon, and T. Sterling. Parallel supercomputing with commodity components, 1997.

[18] M. Welsh, A. Basu, and T. von Eicken. Incorporating Memory Management into User-Level Network Interfaces. In *Proceedings of Hot Interconnects V*, Aug. 1997.