An Evaluation of the Myrinet/GM2 Two-Port Networks

Reza Zamani Ying Qian Ahmad Afsahi
Department of Electrical and Computer Engineering
Queen's University
Kingston, Canada K7L 3N6
{zamanir, qiany, ahmad}@ee.queensu.ca

Abstract

It is important to systematically assess the features and performance of the new interconnects for high-performance clusters. This paper presents the performance of the two-port Myrinet networks at the GM2 and MPI layers using a complete set of microbenchmarks. We also present the communication characteristics and the performance of the NAS Multi-Zone benchmarks and SMG2000 application under the MPI and MPI-OpenMP programming paradigms.

We found that the host overhead is very small in our cluster, and the Myrinet is sensitive to the buffer reuse patterns. Our applications achieved a better performance for MPI than the mixed-mode. All the applications studied use only nonblocking communications, thus able to overlap their communications with the computations. Our experiments show that the two-port communication at the GM and MPI levels (except for the RDMA read, and overlap) outperforms the one-port communication for the bandwidth. However, this did not translate in a considerable improvement at least for our applications.

1. Introduction

With the introduction of several high-speed interconnects, and the availability of commodity computational engines, it has been very promising to build cost-effective cluster computing systems. In such systems, the interconnection network and the communication system software are the keys to the performance.

Several high-performance interconnects have been recently introduced including the Quadrics QsNet [18], and QsNet II [1], InfiniBand [16], Myrinet [5], SCI [8], and Sun Fire Link [19]. Each one of these interconnects provides different levels of performance, programmability, and integration with the operating systems. Recently, Myricom has introduced a dual-port Myrinet "E-Card" [5] for higher performance with a new messaging layer (GM2.1).

Applications usually run on top of a middleware such as the Message Passing Interface (MPI) [17], which itself runs on top of a user-level messaging layer such as the GM. To determine if applications can benefit from a particular interconnect, it is essential to assess the various features and the performance of the interconnect at both the user and middleware levels. Meanwhile, the communication characteristics of applications would also help to understand their performance on clusters.

An important decision for application programmers is the choice of programming paradigm for clusters of Symmetric Multiprocessors (SMP). MPI is the *de-facto* standard for parallel programming on network-based computing systems. OpenMP [9] has emerged as the standard for parallel programming on shared-memory systems. It is open to debate whether pure message-passing or mixed MPI-OpenMP is the programming of choice for higher performance on SMP clusters.

In this work, we do a comprehensive assessment of the features and performance of the two-port Myrinet networks using a complete set of microbenchmarks written at the user level (GM2.1), and MPI level. This paper has a number of contributions. Specifically, it first contributes by presenting the performance of the GM2.1 for one-sided and two-sided communications. The second part evaluates the quality of MPI implementation on top of GM. It includes measurements for the MPI latency, bandwidth, parameters of the LogP model, intra-node performance, computation/communication overlap, buffer reuse impact, polling/blocking impact, and the collective communication performance. Thirdly, it studies the MPI communication characteristics of the newly released NAS Multi-Zone benchmarks [20] and the SMG2000 application [21], as well as their performance under MPI and MPI-OpenMP on our Myrinet SMP cluster.

The rest of this paper is organized as follows. In Section 2, we provide an overview of the Myrinet interconnect, GM, and MPICH [10] over GM. We describe our experimental framework in Section 3. Section 4 presents the GM performance. MPI performance is presented in Section 5. Application benchmarks characteristics and performance are studied in Section 6. In Section 7, related work is presented. Finally, we conclude our paper in section 8.

2. Overview of Myrinet, GM, and MPICH-GM

Myrinet [5] is based on packet-switching technology, where the packets are wormhole-routed through a network consisting of switching elements and *network interface cards* (NIC). Each NIC, attached to the I/O bus, contains a programmable processor and on-board memory providing much flexibility in designing communication software.

The new M3F2-PCIXE-2 two-port Myrinet/PCI-X interface card (E-card) has a 64-bit, 133MHz PCI-X interface, and a programmable Lanai-2XP RISC processor operating at 333MHz with 2MB local memory. Each port has a 2.0+2.0 Gbps data rate. The standard firmware executing in the Lanai-2XP distributes packets adaptively across the two ports, such that the two ports act as a single 4.0+4.0 Gbps data-rate port. It also provides multi-path dispersive routing in the switch fabric that diffuses hot spots and increases the utilization of the network bisection for large network. The high-availability is instantaneous; by removing the fiber cable from either port, communication will continue using the remaining port.

GM is a commercial open source networking protocol from Myricom, which runs on top of the Myrinet network. It provides a protected user-level interface to the NIC so that multiple user processes can share it simultaneously. The Myrinet software support for the latest E-card requires GM 2.1.0 or later (or MX, yet to be released).

GM supports both *send/receive* and *Remote Direct Memory Access* (RDMA) operations. The send/receive mode is a two-sided operation, where both the sender and the receiver of a message are involved in the communication. However, RDMA is a one-sided operation, where a node has access to the remote memory of other nodes. Operations finish without the remote side being involved. The RDMA operations include RDMA write (*put*), where a process can directly write to the remote memory, and RDMA read (*get*) where it can read the contents of remote memory (RDMA read has been recently added to the GM API). In both the send/receive and RDMA communication models, communication buffers must be registered and deregistered in the physical memory at both ends to enable DMA transfer in and out of those regions.

MPICH [10] is a portable implementation of the MPI. MPICH-GM is implemented by targeting its Channel Interface to the GM messaging layer. MPICH-GM uses the *eager* protocol for sending small (less than 16KB), and control messages via GM send/receive operations. It uses the *rendez-vous* protocol for sending large messages via GM one-sided *put* operation.

3. Experimental Framework

In this paper, we first present the RDMA, and send/receive performance of the Myrinet network at the GM level using our own microbenchmarks. To gain a better understanding of the communication subsystem in the Myrinet SMP clusters, we have also devised a set of microbenchmarks at the MPI level. Our MPI microbenchmarks include detailed latency and bandwidth measurements at the intra-node and inter-node levels, buffer reuse impact, polling/blocking impact, communication/computation overlap, LogP parameters, and collective communication performance. We present and compare performance under both the two-port and the one-port. Note that by two-port we mean both ports of the E-card are connected to the switch fabric via their fiber cables. For the one-port experiments, we removed one of the fibers, and re-ran the GM mapper.

We evaluate the performance of our Myrinet SMP cluster with the newly released NAS Multi-Zone benchmarks (NAS-MZ) [20] and the SMG2000 from ASCII Purple benchmark [21]. We have profiled the MPI communication characteristics of these applications to better understand their performance on our system. The LU-MZ, BT-MZ, and SP-MZ benchmarks in the NAS Multi-Zone suite have been primarily designed to evaluate SMP clusters. They have six workload classes: small (S), workstation (W), large (A), larger (B and C), and largest (D). We experimented with the classes B and C. We also used a fixed workload of 128x64x64 for the SMG2000.

We ran our experiments on an 8-node dedicated SMP cluster interconnected with the Myrinet E-cards, and a Myrinet-2000 16-port switch. Each node is a Dell PowerEdge 2650 that has two Intel Xeon MP 2.0 GHz Processors and 1GB RAM running Linux RedHat 9 with SMP kernel version 2.4.24. Each node has a two-port Myrinet E-card on a 64-bit, 133 MHz PCI-X slot. Our PCI-X/DMA performance for a bus read, bus write, and bus read and write bandwidth is 775MB/s, 1040MB/s, and 873MB/s, respectively. We used the MPICH-GM version 1.2.5..10 as the message-passing library on top of the GM2.1.0 messaging layer. We used the version 7.1 of Intel C and FORTRAN compilers. We wrote our own profiling code using the wrapper facility of the MPI for gathering the communication traces of the NAS-MZ benchmarks. However, we used the Intel Vampir for the SMG2000.

4. GM Performance

Figure 1 shows the cost and bandwidth of the memory registration and deregistration in GM. Up to 64KB message sizes, registration is cheaper than deregistration. Registration/deregistration is a costly operation, and operating system dependent. This cost is avoided for short messages in MPICH-GM at the expense of a memory copy into pre-registered buffers.

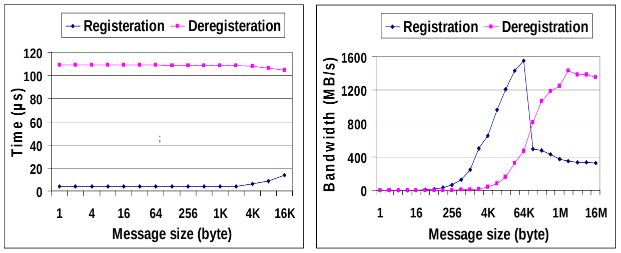

Figure 1. GM registration/deregistration.

4.1. RDMA Performance

The message latency and bandwidth are two important metrics for many parallel and distributed computations. Latency is defined as the time it takes for a message to travel from the sender process address space to the receiver process address space. Bandwidth is reported as the total number of bytes per unit time delivered during the time measured.

In the *unidirectional latency/bandwidth* test, the sender transmits a message repeatedly to the receiver, and then waits for the last message to be acknowledged. The *bidirectional* test is the *ping-pong* test where the sender sends a message and the receiver upon receiving the message, immediately replies with the same message size. This is repeated sufficient number of times to eliminate the transient conditions of the network. In the *both-way* test, both the sender and receiver send data simultaneously. This test puts more pressure on the communication subsystem, and the PCI-X bus.

Figure 2 presents the unidirectional and both-way latency and bandwidth for the RDMA write under the two-port and one-port communications. Note that because GM does not have a notification mechanism at the receiving side, we didn't do the bidirectional test (it actually needs a send/receive follow-up message). One can see that the number of ports does not affect the short message latencies. However, the two-port bandwidth sharply increases after 4KB messages for both unidirectional and both-way compared to the one-port. The two-port both-way bandwidth achieves a maximum of 788MB/s. The two-port unidirectional bandwidth is the same as the one-port both-way bandwidth, and equal to 479MB/s. The one-port unidirectional bandwidth is 246MB/s.

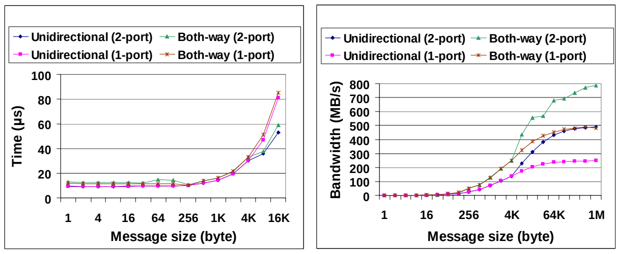

Figure 2. GM RDMA write performance.

Figure 3 shows the unidirectional and both-way latency and bandwidth for the RDMA read under the two-port and one-port communications. The latency for both RDMA write and read remains at roughly 10µs for up to 128 bytes (*RDMA* read is slightly more expensive) and then increases. The difference in bandwidth is more evident, where the RDMA write outperforms the RDMA read. This is consistent with the other high-performance interconnects, and implies applications and middleware should prefer RDMA write over read. Contrary to the RDMA write, the RDMA read has a higher bandwidth for the one-port (481MB/s for both-way, and 246MB/s for unidirectional) compared to the two-port (259MB/s for both-way, and 137MB/s for unidirectional). This is unclear to us. We are currently looking into how RDMA read has been implemented.

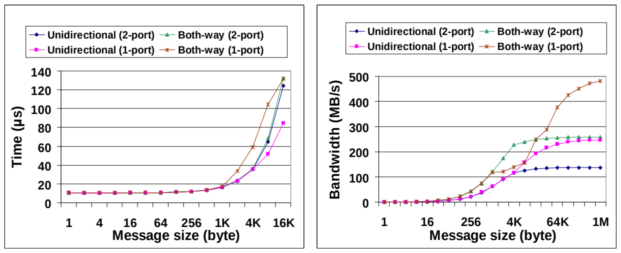

Figure 3. GM RDMA read performance.

4.2. Send/Receive Performance

Figure 4 shows the unidirectional, bidirectional, and both-way send/receive latency/bandwidth under the two-port and one-port communications. GM send/receive has a minimum short message unidirectional latency of 5.25 microseconds. The latency remains roughly the same for up to 64 bytes. Like the RDMA write, the two-port bandwidth outperforms the one-port. The peak unidirectional and bidirectional bandwidth is 489MB/s, while the sustained both-way bandwidth is 789MB/s. This is a major improvement over the Myrinet networks using the one-port D-card (471MB/s peak bandwidth [14]).

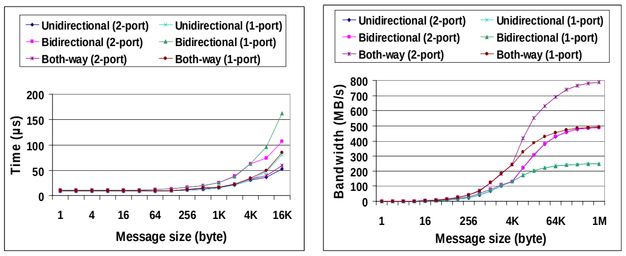

Figure 4. GM send/receive performance.

5. MPI Microbenchmarks and Performance

In this section, we present our MPI microbenchmark tests evaluating the quality of the MPI layer on top of the GM messaging layer.

5.1. Latency

In deriving the bidirectional latency (and bandwidth in Section 5.2), we used matching pairs of sends and receives under different MPI send modes; that is, the *standard* mode, the *synchronous* mode, the *buffered* mode, and the *ready* mode. In the synchronous mode, the send call returns when the receiver receives the message. In the buffered mode, the message is buffered and the send call returns without waiting for the receiver. In the standard mode, MPI implementation chooses to buffer or to wait for the receiver. In the ready mode, matching receive must have been issued before the send.

The latency for intra-node communication is shown in Figure 5. The intra-node latency for 1-byte message remains at 1.38µs for unidirectional ping, 1.46µs for standard and ready modes, 2.29µs for buffered, and 4.51µs for the synchronous mode. This shows that synchronous mode should be avoided for short message transfers. The buffered mode has the worst performance for large messages.

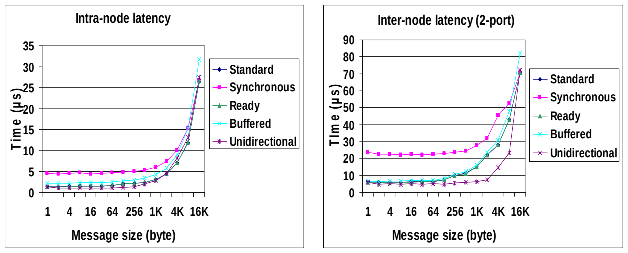

Figure 5. MPI unidirectional, and bidirectional latency.

The latency for two-port inter-node communication is also shown in Figure 5 (one-port has a similar performance for short messages). An overall observation satisfies the argument that intra-node latency should normally be better than the inter-node. However, in our system, intra-node messages larger than 128KB (not shown here) have larger latencies than their counterparts. This could be due to insufficient memory in our platform, or unoptimized intra-node communication in MPICH. The synchronous mode shows quite large inter-node latency, 23.6µs for up to 512-byte messages. The inter-node latency for 1-byte message remains at 5.7µs for unidirectional ping (compared to 5.25µs for GM), 6.1µs for standard (compared to 5.53µs for GM) and ready modes, and 6.84µs for the buffered mode. This shows MPICH-GM has very well been ported to GM. A quick comparison with other high-performance interconnect are as follows: Quadrics QsNet: 4.6µs [18], and QsNet II: ~3µs [1], Myrinet D-card: 6.7µs [13], InfiniBand: 6.8µs [13], and Sun Fire Link: 5µs [19].

We also measured the average standard ping-pong latency (two-port) when simultaneous messages are in transit between pairs of send and receive processes, as shown in Figure 6. There are up to 16 processes on 8 nodes, one process per processor. Note that the pair-wise send and receive processes are on different nodes. The results are interesting as the latency in each case does not change much when the number of pairs is increased. The latencies stay at 6µs from 2-byte to 64-byte messages, running with up to 8 processes.

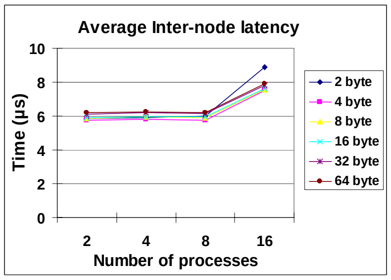

Figure 6. MPI inter-node latency under load.

5.2. Bandwidth

Figure 7 presents the unidirectional and bidirectional bandwidth for intra-node communication. We can see that the intra-node bandwidth arrives at a maximum of around 700MB/s (except for the buffered mode) at 64KB but suddenly drops. The reason could be the same as the one stated in section 5.1. Figure 7 also presents the unidirectional and bidirectional bandwidth for inter-node communication. The MPI bandwidth, except for the buffered mode, achieves approximately 440MB/s. This is almost 90% of the unidirectional bandwidth at the GM level. We also see a sudden drop at 16KB for unidirectional bandwidth where MPICH-GM switches from eager protocol to the rendezvous protocol.

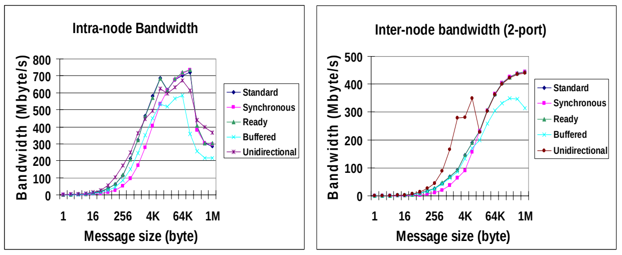

Figure 7. MPI unidirectional, and bidirectional bandwidth.

We have also evaluated the MPI both-way bandwidth. Figure 8 compares the unidirectional, bidirectional (standard mode), and the both-way bandwidth for different message sizes. MPICH-GM delivers a both-way bandwidth of 690MB/s equal to 87% of the GM performance. The results verify that the Myrinet with two-port outperforms the one-port. A quick comparison with other high-performance interconnect are as follows: Quadrics QsNet: 308MB/s [18], and QsNet II: ~900MB/s [1], Myrinet D-card: 471MB/s [13], InfiniBand: 841MB/s [13], and Sun Fire Link: 792MB/s [19].

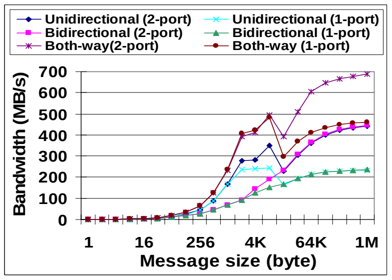

Figure 8. MPI inter-node bandwidth.

The Myrinet aggregate bandwidth when simultaneous messages are in transit between pairs of send and receive processes is shown in Figure 9. Note that the aggregate bandwidth is the sum of individual bandwidths. The result indicates that the network is capable of providing higher bandwidth with increasing number of communication pairs. For 16 processes, Myrinet achieves the maximum bandwidth of roughly 2500MB/s, and 1800MB/s for two-port and one-port, respectively.

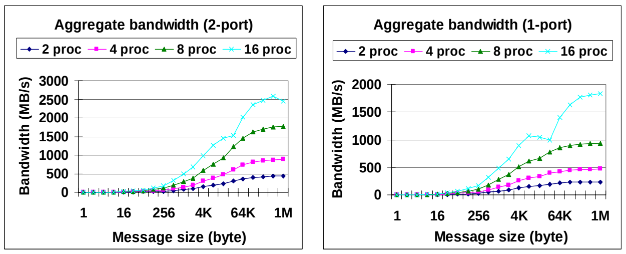

Figure 9. Aggregate MPI inter-node bandwidth.

5.3. Buffer Reuse Impact

To evaluate the impact of MPI buffer management on the communication performance, we measure the latency and bandwidth when different buffers are used for message transfers. Applications employing different message buffers may need to register/deregister their user buffers to achieve zero-copy communication. Pinning/unpinning are expensive, as seen in Section 4, thus, application buffer reuse pattern may have significant impact on the communication performance in high-performance interconnects. Note that our test in this section is similar to [13]. We ran our standard latency/bandwidth test for thousand times but with a buffer reuse pattern of 0%, 25%, 50%, 75%, and 100%. A 100% buffer reuse pattern means that only one user buffer is used for all thousand message transfers. A 0% buffer reuse pattern means that each time a new user buffer is used. We experimented with different message sizes.

Figure 10 shows the impact of buffer reuse pattern on the MPI latency and bandwidth using the two-ports. It is clear that with decreasing buffer reuse percentage the performance drops. Specifically, the latency starts increasing with decreasing buffer reuse percentage after 4KB. The higher buffer reuse percentages (50% and 75%) affect the bandwidth starting at 64KB messages, while this happens at smaller messages (4KB) for lower buffer reuse percentages (0% and 25%). Similar behavior, though with smaller bandwidths, is noted for the one-port case (not shown here).

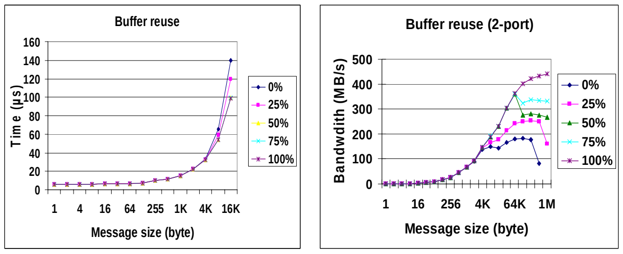

Figure 10. Buffer reuse impact on MPI.

5.4. Polling/Blocking Impact

MPICH-GM by default uses the *polling* method, however, it provides the ability to change the behavior. Three modes are supported: *polling*, *blocking*, and *hybrid*. In the blocking method, the MPI message reception call sleeps in the kernel, but will be interrupted when the matching message arrives. In the hybrid mode, the MPI message reception call polls for 1 second, and then goes to sleep. Normally, polling achieves low latency but has high CPU utilization. Blocking increases the latency due to interrupt and context switch, but achieves lower CPU utilization. It is interesting to see if blocking or hybrid mode can provide any advantage over polling in our system. The results are shown in Figure 11. Polling has a slightly better performance than the hybrid. However, they both outperform the blocking method. Again, the two-port performance is better than the one-port.

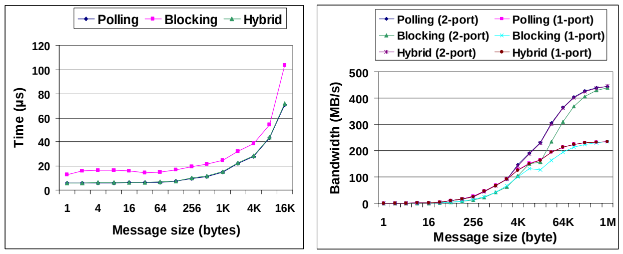

Figure 11. Polling/blocking impact on MPI.

5.5. LogP Parameters

The time spent for communication at the source and destination is a major contributing factor to the performance of applications in cluster computing systems. *LogP* model has been proposed to gain insights into different components of a communication step [7]. LogP models sequences of point-to-point communications of short messages. *L* is the network hardware latency for one-byte message transfer. *O* is the combined overhead in processing the message at the sender (o_s) and receiver (o_r). *P* is the number of processors. The gap, *g*, is the minimum time interval between two consecutive message transmission from a processor. *LogGP* model [3] extends LogP to also cover long messages.

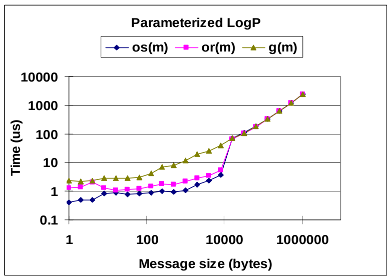

Figure 12. LogP parameters.

An efficient method for measurement of LogP parameters has been proposed in [12]. The method is called *parameterized LogP* and subsumes both LogP, and LogGP models. Our microbenchmark result depicted in Figure 12, in logarithmic scales, shows the o_s, o_r, and *g* for different message sizes in our Myrinet cluster for the two-port case. Both o_s and o_r have a sudden increase at 16KBytes (from 5 to 70 microseconds) due to protocol switch in MPICH-GM. The combined host overhead for a 1-byte message is 0.8µs; gap is 2.3µs, and L is 4.2µs. What is apparent is that the overhead increases with the message size, and the overhead at the receiving end is larger than the source side, most probably due to polling. It seems to us probably the network interface is not quite powerful as the CPU has to do more work with larger message sizes, both at the send and at the receiving sides.

5.6. Communication/Computation Overlap

Overlapping computation with communication can improve the application performance. A programmable NIC, such as the Myrinet E-card, offloads some of the host's functionality in a message transfer. Therefore, there will be opportunities for overlapping computation with the communication.

We use the bidirectional latency (standard mode) test but this time with non-blocking send and receive function calls along with their corresponding wait calls. However, we insert computation in between until the original latency increases. We do this test for different message sizes. Our test is similar to the work in [13]. Figure 13 shows the overlap potential for different message sizes. There is between 3µs to 7µs overlap for up to 512 bytes. This increases until a drop is observed due to the protocol switch in MPICH-GM. From then on, the overlap increases again. The two-port potential overlap is shown to be less than the one-port, mostly because the NIC is involved in extra operations such as adaptive distribution of packets across the two ports, among other things.

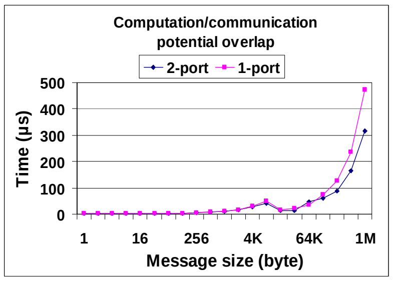

Figure 13. Computation/communication overlap.

5.7. Collective Communications

Efficient implementation of collective communication algorithms is one of the keys to the performance of network computing applications. Therefore, it is interesting to evaluate their performance. We have chosen the *broadcast*, *gather*, *scatter*, *reduce*, *allreduce*, *alltoall*, *alltoallv*, and *barrier* operations as representatives of the mostly used collective communication operations in parallel applications. Our experiments are done with processes located on the same node and/or on different nodes. In the inter-node case, we evenly divided the processes among the eight SMP nodes. We measured the performance in terms of their completion time, as shown in Figure 14. *Broadcast* has the best performance, where *allreduce* has the worst, followed by *alltoallv* and *reduce*.

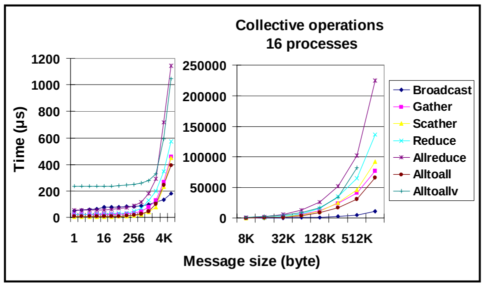

Figure 14. Collective communications.

6. Application Benchmarks

It is important to discover if the performance can be optimally used at the application layer. We experimented with the NAS Multi-Zone 3.0 benchmark suite [20] from NASA, as well as the SMG2000 benchmark from the ASCI Purple suite [21]. The parallel implementation uses hybrid parallelism: MPI for the coarse-grain parallelism and OpenMP for the loop-level parallelism.

6.1. Communication Characteristics

Knowing the communication characteristics of the applications may help to understand the reasons behind their performance. In this paper, we study the number of sends and receives per process, average message size per send, and the number of collective communication primitives used in the applications.

Table 1 presents the average number of send calls per process, and the average message size per process for all applications under different classes and number of processes. It should be mentioned that all four applications use only nonblocking send and receive calls in their point-to-point communications. The number of sends is equal to the number of receives for each process. Using nonblocking calls allow these applications to benefit from overlapping their communications with the computations.

The applications studied have diverse communication characteristics. For instance, the number of send calls for BT and LU decreases when the number of processes increases, while the average message size decreases. SMG2000 shows the opposite trend. SMG2000 uses short messages on average, while others use large message sizes. The number of collectives for these applications is very small.

Table 1. MPI Communication characteristics of the NAS-MZ and SMG2000 applications

Applications and classes	# Processes	Nonblocking pt-2-pt (per process)		Collectives (per process)					
		# Calls	Average size (KB)	#Broadcast	#Reduce	#All_reduce	#Barrier	#All-gather	#All gatherv
BT-B	**2**	**9648**	**123**	**3**	**3**	**-**	**2**	**-**	**-**
	4	**8040**	**128**	**3**	**3**	**-**	**2**	**-**	**-**
	8	**5276**	**137.8**	**3**	**3**	**-**	**2**	**-**	**-**
	16	**3065**	**138.3**	**3**	**3**	**-**	**2**	**-**	**-**
BT-C	**2**	**20502**	**156.6**	**3**	**3**	**-**	**2**	**-**	**-**
	4	**21809**	**167.4**	**3**	**3**	**-**	**2**	**-**	**-**
	8	**17135**	**177.8**	**3**	**3**	**-**	**2**	**-**	**-**
	16	**10804**	**186.2**	**3**	**3**	**-**	**2**	**-**	**-**
SP-B	**2**	**6416**	**112.5**	**3**	**3**	**-**	**2**	**-**	**-**
	4	**10827**	**120.8**	**3**	**3**	**-**	**2**	**-**	**-**
	8	**6416**	**112.5**	**3**	**3**	**-**	**2**	**-**	**-**
	16	**4010**	**123.8**	**3**	**3**	**-**	**2**	**-**	**-**
SP-C	**2**	**12382**	**146.3**	**3**	**3**	**-**	**2**	**-**	**-**
	4	**24261**	**163**	**3**	**3**	**-**	**2**	**-**	**-**
	8	**24261**	**153.6**	**3**	**3**	**-**	**2**	**-**	**-**
	16	**12832**	**146.3**	**3**	**3**	**-**	**2**	**-**	**-**
LU-B	**2**	**2008**	**234.4**	**7**	**4**	**-**	**2**	**-**	**-**
	4	**2008**	**234.4**	**7**	**4**	**-**	**2**	**-**	**-**
	8	**1506**	**271.9**	**7**	**4**	**-**	**2**	**-**	**-**
	16	**1004**	**290.6**	**7**	**4**	**-**	**2**	**-**	**-**
LU-C	**2**	**3012**	**633.8**	**7**	**4**	**-**	**2**	**-**	**-**
	4	**2008**	**633.8**	**7**	**4**	**-**	**2**	**-**	**-**
	8	**1506**	**742.1**	**7**	**4**	**-**	**2**	**-**	**-**
	16	**1004**	**796.3**	**7**	**4**	**-**	**2**	**-**	**-**
SMG	**2**	**44303**	**0.92**	**-**	**-**	**15**	**1**	**1**	**1**
	4	**51941**	**0.54**	**-**	**-**	**14**	**1**	**1**	**1**
	8	**52423**	**0.32**	**-**	**-**	**14**	**1**	**1**	**1**
	16	**50827**	**0.32**	**-**	**-**	**14**	**1**	**1**	**1**

6.2. Application Performance

Figure 15 and Figure 16 present the execution time of our applications on our two-port platform. The legend in Figure 15, and the X-axis in Figure 16 show the number of processes and threads for each case. For instance, “4P2T” means that there are four processes evenly divided among four nodes of our cluster, where each process has two threads running on its respective node.

This way, we are able to compare the performance of the applications under pure MPI, and the mixed MPI-OpenMP. From the results, one can easily claim that the MPI version of the applications perform better than their mixed-mode versions. For instance, for BT-MZ with class C (BT-C), 2P1T runs faster than 1P2T; 4P1T runs faster than 2P2T; and so on. The same is true for the SMG2000.

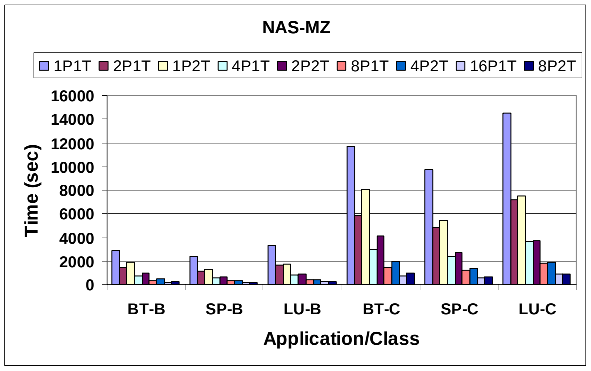

Figure 15. NAS-MZ performance (two-port).

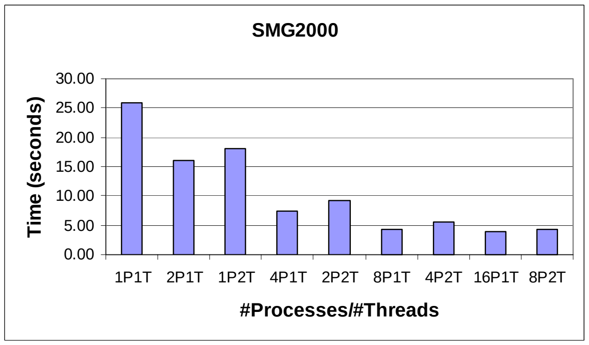

Figure 16. SMG-2000 performance (two-port).

Interestingly, the speedup, from 1 to 16 processes, for the MPI version of the NAS-MZ benchmarks is linear (not shown here). Applications reach almost perfect speedup. However, the speedup for SMG2000, MPI version, is not as good as the NAS-MZ. The speedup with 2, 4, 8, and 16 processes are 1.62, 3.50, 6.13, and 6.51, respectively. We are currently investigating the different reasons behind their performance.

We also ran the applications with only one-port. The difference in performance was minimal, with at most 3% improvements for the two-port cases. The message sizes for SMG2000 are short (less than 1KB) so the two-port Myrinet network cannot offer any improvement over the one-port. However, this is not the case for the other applications. Having a close look at the distribution of message sizes for the NAS-MZ applications (not shown here) reveal that the BT-MZ uses a large number of different message sizes. It uses up to 21 different message sizes in class C (16 in class B). The shortest and the longest messages are 42KB, and 446KB, respectively. The distribution of message sizes sent by the SP-MZ, and LU-MZ are bimodal (112KB, and 168KB for SP-B; 146KB, and 227KB for SP-C; 234KB, and 346KB for LU-B; and 633KB, and 958KB for LU-C). Message sizes and the number of messages sent for the NAS-MZ applications suggest these applications are bandwidth-bound. Therefore, the two-port Myrinet network should improve the performance over the one-port. A setback in performance improvement could be associated with the fact that the one-port NIC can offer a better computation/communication overlap than the two-port. Our study in this section is preliminary. We intend to get the execution traces of these applications to find the communication/computation ratio, and to see if these applications are sensitive to the gap, or overhead.

7. Related Work

Research groups in academia and industry have been studying the performance of the interconnection networks at the user-level, the MPI level, and the application level [13, 14, 1, 18, 4, 11, 19]. In [13], the authors assessed the performance of the MPI implementation on InfiniBand, Myrinet, and Quadrics. They studied microbenchmarks at the MPI level as well as NAS benchmarks. Our work is similar to their work. However, we have investigated the recently introduced Myrinet two-port networks. We have also expanded the microbenchmarks, and studied the performance at the GM level, as well as the characteristics of the NAS Multi-Zone benchmarks.

The performance of Myrinet and GigaNet were evaluated in [11] using microbenchmarks, and the NAS benchmark. The authors in [14] studied the user-level messaging layers of the InfiniBand, Myrinet, and Quadrics. Some interconnects including the SP switch, and Cray T3E were studied using user-level and MPI-level microbenchmarks [4]. Sun Fire Link was evaluated at the RSM, MPI, and application level in [19]. The authors in [1, 18] studied the QsNet, and QsNet II.

Researchers have characterized communication patterns of MPI applications [21, 2]. LogP and its variants were proposed to gain insights into the different components of a communication step [7, 3, 12]. Sensitivity of applications to latency, and overhead was inspected in [15]. Performance comparison of applications in MPI, and MPI-OpenMP were done in [6, 19].

8. Conclusions

The interconnection networks and the supporting communication system software are the deciding factors in the performance of clusters. It is desirable to understand their performance at the user-level messaging layer, middleware, and at the application level. In this paper, we have done a detailed evaluation of the two-port Myrinet network with its new GM layer. We have presented the performance at the GM and MPI levels using an extended set of microbenchmarks, as well as the application level.

We discovered that the two-port communication outperforms the one-port for the bandwidth, except for the RDMA read and computation/communication overlap. The performance of RDMA write was better than the read operation. We found that the host overhead is very small in our cluster. The Myrinet network showed sensitivity to the buffer reuse patterns. MPICH-GM delivered much of the performance offered at the user-level.

All the applications studied used only nonblocking communications. They had diverse communication characteristics, though. Our applications achieved a better performance under MPI than the mixed-mode. The NAS-MZ had an ideal speedup, but the SMG2000 was not scalable very much. The difference in performance between the two-port and one-port was minimal for the applications. We plan to complete our study to find the factors that may have caused this minimal improvement.

Acknowledgments

We would like to thank the anonymous referees for their insightful comments. This work was supported by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC), the Queen's University, Canada Foundation for Innovation (CFI), and Ontario Innovation Trust (OIT).

References

[1] D. Addison, J. Beecroft, D. Hewson, M. McLaren and F. Petrini, "Quadrics QsNet II: A Network for Supercomputing Applications". In *Hot Chips 15*, August 2003.

[2] A. Afsahi and N. J. Dimopoulos, "Efficient Communication Using Message Prediction for Clusters of Multiprocessors", *Concurrency and Computation: Practice and Experience*, Volume 14, Issue 10, 2002, pp. 859-883.

[3] A. Alexandrov, M. Ionescu, K. E. Schauser, and C. Scheiman, "LogGP: Incorporating Long Messages into the LogP Model - One Step Closer Towards a Realistic Model for Parallel Computation", *7th Annual ACM Symposium on Parallel Algorithms and Architecture (SPAA'95)*, July 1995, pp. 95-105.

[4] C. Bell, D. Bonachea, Y. Cote, J. Duell, P. Hargrove, P. Husbands, C. Iancu, M. Welcome, and K. Yelik, "An Evaluation of Current High-Performance Networks", *17th International Parallel and Distributed Processing Symposium (IPDPS 2003)*, April 2003.

[5] N. J. Boden, D. Cohen, R. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W-K Su, "Myrinet: A Gigabit-per-Second Local Area Network", *IEEE Micro*, Vol.15, No.1, February 1995, pp. 29-36.

[6] F. Cappello, D. Etiemble, "MPI versus MPI+OpenMP on IBM SP for the NAS benchmarks", *Proceedings of Supercomputing Conference (SC'2000)*, 2000.

[7] D. E. Culler, R. M. Karp, D. A. Patterson, A. Sahay, K. E. Schauser, E. Santos, R.Subramonian, and T. von Eiken, "LogP: Towards a Realistic Model of Parallel Computation", *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 1993, pp. 1-12.

[8] Dolphin Interconnect Solutions: http://www.dolphinics.com/

[9] N. R. Fredrickson, A. Afsahi, and Y. Qian, "Performance characteristics of OpenMP constructs, and application benchmarks on a large symmetric multiprocessor", *17th ACM International Conference on Supercomputing* (*ICS'03)*, 2003, pp. 140-149.

[10] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard, *Parallel Computing*, 22(6):789-828, 1996.

[11] J. Hsieh, T. Leng, V. Mashayekhi, and R. Rooholamini. "Architectural and Performance Evaluation of GigaNet and Myrinet Interconnects on Clusters of Small-Scale SMP Servers", *Supercomputing Conference (SC'00), 2000*.

[12] T. Kielmann, H. E. Bal, and K. Verstoep, " Fast Measurement of LogP Parameters for Message Passing Platforms", *4th Workshop on Runtime Systems for Parallel Programming (RTSPP), held in conjunction with IPDPS 2000*, 2000, pp. 1176-1183.

[13] J. Liu, B. Chandrasekaran, J. Wu, W. Jiang, S. Kini, W. Yu, D. Buntinas, P. Wyckoff, and D. K. Panda, "Performance comparison of MPI implementations over InfiniBand, Myrinet, and Quadrics", *Supercomputing Conference (SC'03)*, 2003.

[14] J. Liu, B. Chandrasekaran, Weikuan Yu, Jiesheng Wu, Darius Buntinas, Sushmitha Kini, Dhabaleswar K. Panda, and Pete Wyckoff, "Microbenchmark Performance Comparison of High-Speed Cluster Interconnects", *IEEE Micro*, 2004, Vol. 24, No.1, pp. 42-51.

[15] R. Martin, A. Vahdat, D. Culler, and T. Anderson, "Effects of Communication Latency, Overhead, and Bandwidth in a Cluster Architecture", *International Symposium on Computer Architecture*, 1997.

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

[17] Message Passing Interface Forum: MPI, A Message Passing Interface standard, Version 1.2, 1997.

[18] F. Petrini, S. Coll, E. Frachtenberg, and A. Hoisie, "Performance Evaluation of the Quadrics Interconnection Network", *Journal of Cluster Computing*, 2003, pp. 125-142.

[19] Y. Qian, A. Afsahi, N. R. Fredrickson, and R. Zamani, "Performance Evaluation of the Sun Fire Link SMP Clusters", *18th International Symposium on High Performance Computing Systems and Applications, HPCS 2004*, May 2004, pp. 145-156.

[20] R. F. Van der Wijngaart_, H. Jin. NAS Parallel Benchmarks, Multi-Zone Versions NAS Technical Report NAS-03-010, July 2003.

[21] J. S. Vetter and F. Mueller, "Communication Characteristics of large-scale scientific applications for contemporary cluster architectures", *Journal of Parallel and Distributed Computing*, 63 (2003) 853-865.