Network Performance in High Performance Linux Clusters

Ben Huang, Michael Bauer, Michael Katchabaw
Department of Computer Science
The University of Western Ontario
London, Ontario, Canada N6A 5B7
(huang|bauer|katchab}@csd.uwo.ca

Abstract*—Linux-based clusters have become more prevalent as a foundation for High Performance Computing (HPC) systems. With a better understanding of network performance in these environments, we can optimize configurations and develop better management and administration policies to improve operations. To assist in this process, we developed a network measurement tool to measure UDP, TCP and MPI communications over high performance networks, such as Gigabit Ethernet and Myrinet. In this paper, we report on the use of this tool to evaluate the network performance of three high performance interconnects in HPC clusters: Gigabit Ethernet, Myrinet, and Quadrics' QsNet and discuss the implications of those results for configurations in HPC Linux clusters.*

Keywords: Network performance analysis, benchmarking tools.

1. Introduction

It is now possible to build powerful platforms for high performance computation from off-the-shelf computers and network devices. In particular, the Linux-based commodity cluster constructed with general purpose computers is an increasingly popular model and seems to be a trend for future HPC [14,17]. Commodity cluster computing can be characterized as being cost effective, flexible, extensible, and easy to maintain. However, since these commodity clusters are built with essentially standalone computers and network devices, they do not necessarily guarantee high performance. The performance power of a standalone computer usually depends on its operating system, CPU and memory speed, and a variety of other factors. For HPC clusters, network communication is another key factor for cluster performance—a communication bottleneck in an HPC cluster may lead to a significant loss of overall performance in the cluster and the applications making use of it. Detailed network performance analyses that identify these bottlenecks and other performance issues are capable of yielding insight that developers can use to build better applications and administrators can use to better configure, manage, and administer their clusters.

Traditional network measurement tools, however, do not necessarily work well in evaluating the performance of HPC environments, as the network interconnecting a cluster plays a more critical role in supporting the applications distributed across the compute nodes in the environment. Furthermore, some functionality and parameters required for detailed performance evaluations cannot be found in currently available network benchmarking tools. To better understand network behavior in HPC environments, we developed a Linux-based network benchmark toolset, named Hpcbench [5]. Hpcbench is able to accurately and effectively measure UDP, TCP and MPI communication throughput and latency in high performance networks, recording the detailed communication parameters and kernel statistics under a variety of parameters.

In this paper, we first provide a survey of present network benchmarking tools and related work in Section 2, and justify the need to develop a new benchmark tool for HPC environments. In Section 3, we discuss the design and implementation of Hpcbench. In Section 4, we use Hpcbench to test and analyze HPC networks through a collection of experiments. Our experiments were based on three of the most commonly used interconnects in HPC systems:

Gigabit Ethernet, Myrinet [1,2] and Quadrics' QsNet [10]. Finally, in Section 5, we provide concluding remarks, and discuss future work.

2. Related Work

Measurement of network performance in high performance computing environments is recognized as an important element. Typically, however, the focus is on the performance of a particular interconnect or protocol [3,15]. For example, there has been work to measure different MPI libraries [6,9] or specific interconnects, such as Quadrics QsNet [10]. These measurements are always useful, but as computational hardware changes and libraries improve, it is important to be able to measure not just the interconnects but the protocols, particularly to understand their configurations within a particular environment. As stated [15] in a study of protocol-dependent message passing: "It is vital to take the time to measure and optimize the performance of the OS and message-passing system when dealing with gigabit speed hardware."

To study the high performance networks one would find in an HPC environment, it is preferable to use an active measurement model as it allows direct measurements of network performance and behaviour without the inaccuracies or assumptions introduced by other approaches. While this can cause disruptions to the HPC environment during experimentation, this inconvenience is well worth the better quality results that can be obtained in the process. There are many existing tools available that involve active measurements. Some of the most frequently used tools include Udpmon[4], Netperf[8], Iperf[6] and NetPIPE[16]. However, all of these tools were designed as general purpose network benchmarking tools, and have their own limitations and restrictions that make them unsuitable for HPC environments. For example, Udpmon measures UDP communication using hardware-dependent assembly language to access an Intel CPU cycle counter for high-precision timing, and therefore it can only be used on IA32/IA64 platforms. Iperf supports multi-threading and parallel TCP streams, but does not measure network latency. NetPIPE works well with connection-oriented protocols, such as TCP and MPI, but does not support UDP communication.

Although these tools work reasonably well for the specific tasks for which they were designed, they are limited in functionality and lack a feature set capable of supporting many of the interesting experimental scenarios for HPC environments. As examples, none of the above tools could test non-blocking communication, and none specialize in high performance interconnects, capable of testing all three of the most common communication protocols in commodity clusters: UDP, TCP and MPI. In determining appropriate configurations for specific cluster environments, the administrator may need to explore the impact of various parameters for each of these protocols for specific environments, such as block size, buffer size or even specific network card configuration settings. In theory, it is possible to modify these tools to support network performance analysis in HPC environments, since most of the tools are open source. However, in practice, their implementations are quite complex and difficult to extend for the additional required functionality.

3. Overview of Hpcbench

With this in mind, we felt that the best option was to implement our own network benchmarking tool, Hpcbench, focusing specifically on HPC environments. Hpcbench was designed to measure the high-speed, low-latency communication networks in Linux-based HPC systems. The objectives of Hpcbench include high accuracy and efficiency; support for UDP, TCP and MPI communications; tunable communication parameters of interest in an HPC environment; and detailed recording of test results, communication settings, and system information.

Hpcbench was written in C and uses BSD sockets and MPI APIs. It is comprised of three independent sets of benchmarks measuring UDP, TCP and MPI communications. As the benchmarks are based on a common infrastructure, the implementation and usage of each benchmark are quite similar, allowing us to easily compare the results for the different communication protocols.

Currently, Hpcbench can be freely downloaded from its website at: http://hpcbench.sf.net. For further details, the reader should refer to [5].

4. Network Performance Analysis and Experiences Using Hpcbench

In this section, we report on our experiences in using Hpcbench in the analysis of the network performance of three high performance interconnects: Gigabit Ethernet, Myrinet, and Quadrics' QsNet. (Note that the experimental results presented here are highlights of some of the more interesting results of our study; for complete results, the reader is urged to consult [5] for details.) To avoid unwelcome loads and interactions that could complicate results, all experiments were conducted using dedicated and completely idle machines in our experimental environment. All MPI communication is based on MPICH 1.2.5 [7] built with default settings.

4.1 Testbed Introduction

Our testbed includes two Linux-based clusters named "mako" and "hammerhead" in SHARCNET [12], a distributed HPC Network in Ontario, Canada.

The hammerhead cluster consists of 28 Compaq ES40 Alpha SMP servers. Each server is configured with 4 x 833MHz Alpha EV6 processors, 4GB RAM, an Alteon AceNIC Gigabit Ethernet Adaptor (PCI 64bits/33MHz), a Quadrics QSW Elan3 PCI Network Adaptor, and Redhat Linux 7.2 as its operating system, with kernel 2.4.21-3.7qsnet #9 SMP. For its interconnects, it uses a Nortel Passport 8600 switch (Gigabit Ethernet) and a Quadrics' QsNet switch.

The mako cluster consists of eight HP DL360 Intel Xeon SMP servers. Each server is configured with 4 x 3GHz Intel Xeon Hyperthreading processors, 2 GB RAM, a Broadcom Tigon 3 Gigabit Ethernet Adaptor (PCI-X 64bits/100MHz), a Myricom PCI-X Network Adaptor, and Redhat Linux 9.0 as its operating system, with kernel 2.4.20-8smp #1 SMP. For its interconnects, it uses an HP ProCurve 2800 (Gigabit Ethernet) switch and a Myrinet switch. We evaluate UDP, TCP and MPI over Gigabit Ethernet in this experimentation. TCP/IP communication was not configured for operation over the proprietary Myrinet and QsNet networks, so we only tested their MPI communication based on the same version of MPICH.

4.2 Communication Throughput on Gigabit Ethernet

We select two idle nodes in two clusters and test their UDP, TCP and MPI communication over Gigabit Ethernet.

Figure 1 shows the results of UDP unidirectional throughput versus datagram size, measured by Hpcbench in an exponential test mode with default socket buffering. The Intel cluster achieved 961Mbps peak UDP throughput, while the Alpha cluster gave 530Mbps peak UPD throughput. In high throughput situations, the UDP sender's CPU load was approximately 10% in the Intel system compared to 15% for the Alpha system; the receiver's CPU load was approximately 15% in the Intel system and 20% for the Alpha system.

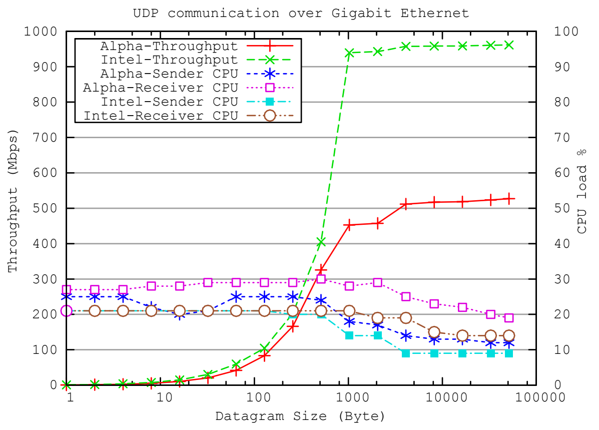

Figure 1: UDP Throughput on Gigabit Ethernet

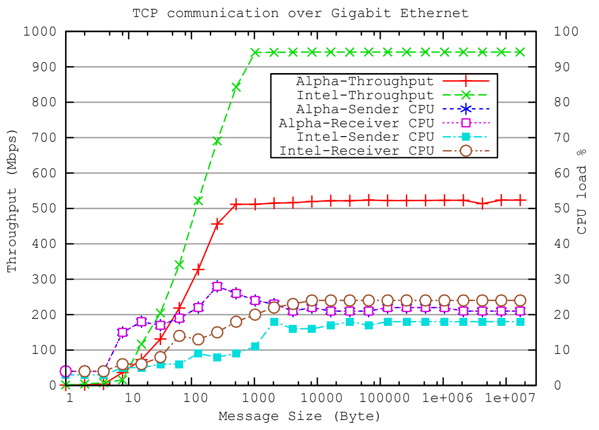

Figure 2: TCP Throughput on Gigabit Ethernet

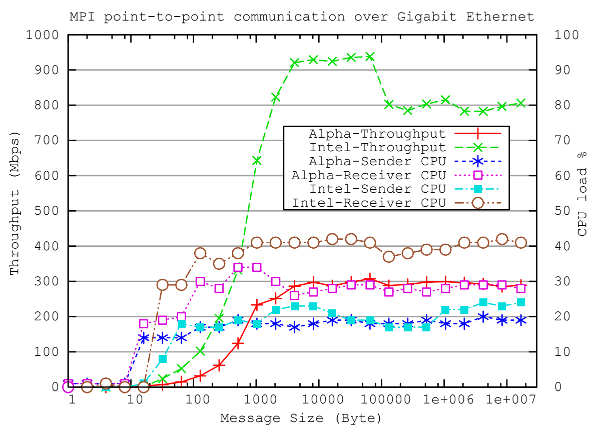

Figure 3: MPI Throughput on Gigabit Ethernet

Figure 2 shows the TCP unidirectional throughput on the Gigabit Ethernet environment as measured by Hpcbench. When message size is greater than 2KB, the Intel system delivered a stable throughput around 940Mbps and the Alpha system delivered about 520Mbps throughput. At the same time, the CPU usage varied from 18% to 29% in both systems (the receivers had slightly higher load than the senders).

Figure 3 shows the MPI point-to-point communication over Gigabit Ethernet measured by Hpcbench. The Intel system delivered a peak 938Mbps unidirectional throughput, which dropped to around 800Mbps when message size was larger than 64KB. The Alpha system only provided about 310Mbps peak MPI throughput. It is likely that the poor performance in this case came from inefficiencies in the implementation of TCP-based MPICH for that platform, because TCP communication in the same system has much higher throughput.

The above experiments demonstrate that the Intel Xeon system performed much better than the Alpha system, with higher throughput and lower CPU usage. To help understand this difference, we used Hpcbench to conduct further UDP tests. UDP was used for further testing rather than TCP because UDP communication has less protocol overhead than TCP and other connection-oriented protocols, and UDP does not utilize transmission control. Usually, UDP can better reflect maximum network throughput than TCP.

Using Hpcbench, UDP experiments with larger socket buffer sizes were conducted. This testing found that the Alpha cluster could provide a maximum 650Mpbs UDP unidirectional throughput with a better selection of buffer size. From detailed log files generated by Hpcbench, significant data loss was observed in the sender during the UDP processing when the socket buffer was quite large (1MB, for example). At the same time, however, no data loss was observed in the network according to the log data collected by Hpcbench. Consequently, it is reasonable to believe that the bottleneck in the Alpha cluster came from the relatively slow sender. For further details, please refer to [5].

4.3 Communication Throughput Using Myrinet and QsNet

In the Intel cluster, we conducted experiments to test MPI point-to-point communication over Myrinet using Hpcbench. The results are shown below in Figure 5.

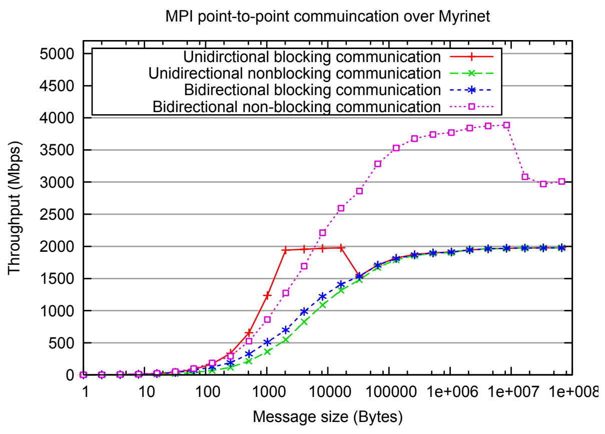

Figure 4: MPI Communication Throughput on Myrinet

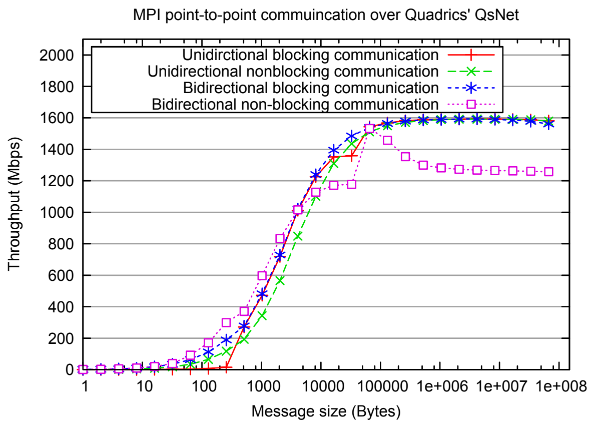

Figure 5: MPI Communication Throughput on QsNet

As shown in Figure 4, unidirectional MPI throughput over Myrinet could reach 1995Mbps, while non-blocking (synchronized MPI_Isend and MPI_Irecv) bidirectional throughput achieved 3885Mbps. In contrast, QsNet in the Alpha cluster delivered a maximum of 1600Mbps unidirectional MPI throughput, while the non-blocking bidirectional throughput dropped to around 1250Mbps when message exceeded 500KB, as shown in Figure 5. This may have occurred because the Alpha machines were not fast enough to handle the heavy system load.

Consulting the data logs recorded by Hpcbench, we observed that the communication on Myrinet and QsNet only consumed a single CPU's clock cycles in a 4-processor SMP system. In fact, communicating processes completely occupied CPU1's utilization, resulting in a sharp 25% overall CPU load in both Intel and Alpha clusters. This could come from the fact that both Myrinet and QsNet technologies use a zero-copy (OS-bypassing) technique for message passing between two nodes. Unlike traditional interrupt-driven TCP/IP communication, there is no interrupt interaction between the Linux kernel and the NIC in Myrinet and QsNet communications. Instead, the data goes directly from user space into the NIC without kernel processing. Consequently, different CPUs in an SMP system are unable to cooperate to handle one communication session. During benchmarking, the sender application attempted to send as much data as possible to the NIC, and used as many CPU resources it could get to do so. A similar situation also occurred on the receiver side.

We also conducted multi-link communication experiments over Myrinet using Hpcbench. When the number of communication links exceeded the number of CPUs, we found that the Intel Xeon system had reached a 100% CPU load in both the sender and the receiver, while the overall throughput in the network remained nearly the same. On the other hand, the kernel is involved in TCP/IP communications. Consequently, multi-link communication can introduce a high system load, but will not overwhelm the system for 100% usage. When the network becomes saturated or congested, the kernel will block application transmission to ensure that communication can be serviced properly.

4.4 Communication Latency on Gigabit Ethernet, Myrinet and QsNet

High throughput does not necessarily imply low network latency. The overall performance of some applications is very sensitive to network latency. Hpcbench measures network latency in terms of round trip time (RTT) from the application layer, using various underlying transports, giving us UDP RTT, TCP RTT and MPI RTT. The traditional ping utility instead evaluates the ICMP RTT with a relatively coarse resolution (milliseconds), and may not work properly or accurately enough in a low-latency network.

Figure 6 shows the Intel Xeon cluster's RTTs for different protocols on the Gigabit Ethernet with message (datagram) size less than the Maximum Transmission Unit (MTU, 1500 Bytes in our experimental testbed). The results show that the UDP and TCP's RTTs were around 56-60µsec for tiny messages (1~32 Byte), implying around a 28µsec one way network latency for these two protocols. MPI RTTs, on the other hand, started from nearly 72µsec, making it a higher latency protocol than TCP and UDP in our experimentation, according to Hpcbench. Figure 7 demonstrates that the network latency in the Alpha cluster was much higher than that of the Intel cluster according to experimentation with Hpcbench. The minimum RTTs in the Alpha cluster were about 240µsec for UDP and TCP, and about 350µsec for MPI communication.

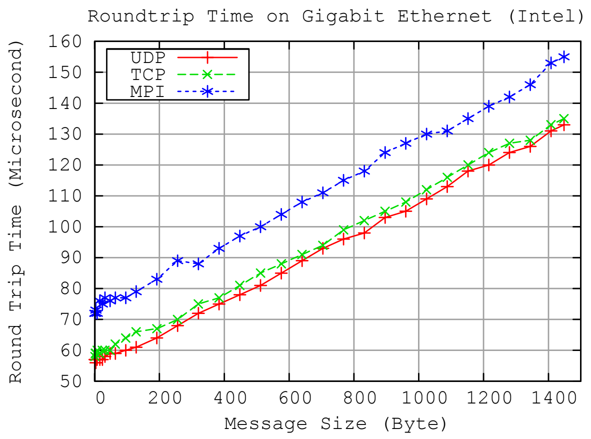

Figure 6: Intel Cluster RTT on Gigabit Ethernet

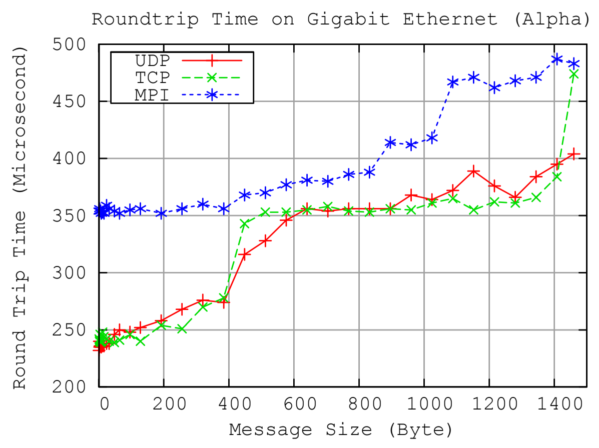

Figure 7: Alpha Cluster RTT on Gigabit Ethernet

From the statistics recorded by Hpcbench, we observed that the interrupt coalescence technique [11] was used in the Alpha cluster but not in the Intel cluster. Interrupt coalescence was used by Alpha machines' Gigabit Ethernet network cards to reduce the system load for network communication. Unfortunately, this technique can also introduce significantly larger network latency. We can also see that all curves in Figure 8 are not linearly smooth, in part due to the effect of interrupt coalescence.

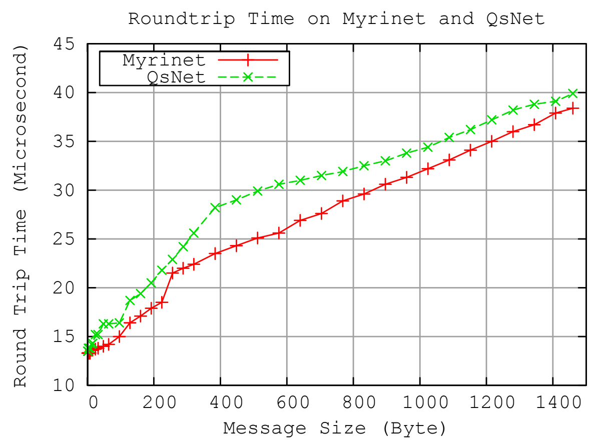

Figure 8: MPI RTT on Myrinet and QsNet

Figure 8 shows MPI communication latency in Myrinet and QsNet. The RTTs with tiny messages could be as low as 14μsec in both interconnects, which is significantly lower than Gigabit Ethernet.

In many HPC environments, including the current configuration of SHARCNET, Myrinet and QsNet are used only to provide a message passing environment for parallel computing. From our findings, these two interconnects are confirmed ready to support IP. In such an environment, other communication, such as shared storage access, could benefit from the advantage of these high-speed, low-latency technologies.

5. Conclusions and Future Work

In this paper, we analyzed the network performance (throughput and latency) of Gigabit Ethernet, Myrinet and Quadrics' QsNet using UDP, TCP, and MPI communication mechanisms as transports. We looked at a variety of parameters affecting overall performance, including packet size, buffering, and blocking versus non-blocking communication primitives. These experiments reconfirmed beliefs about network performance, providing empirical evidence to support these intuitions:

- Protocol overhead can have a significant impact on performance in an HPC network. For example, in Gigabit Ethernet experiments, we saw that UDP outperformed TCP, which outperformed MPI (based on TCP in this case). This was the case for both throughput and latency testing.
- Node configuration can have a significant impact on performance in an HPC network. Our experimentation showed that, for Gigabit Ethernet, our Intel Xeon cluster with faster processors and system bus outperformed our Alpha cluster. We also directly observed the effects of interrupt coalescence in our Alpha cluster on latency.
- Depending on cluster configuration, it is possible for the compute nodes themselves to be a performance bottleneck. Our experimentation demonstrated, for example, that the processors in our Alpha systems were unable to keep up with the load placed on them, limiting effective throughput.
- Technologies such as Myrinet and QsNet can outperform Gigabit Ethernet, both in terms of throughput and latency. If they can be afforded, they are definitely an asset to HPC cluster systems.

In addition to verifying such intuitions, our experimentation also yielded some interesting results about the specific configurations under test, and also illustrated the necessity of such experimentation in order to determine appropriate configurations to maximize throughput:

- For UDP and TCP communication over Gigabit Ethernet on both the Alpha and Intel clusters, message sizes greater than 1Kb achieved greater throughput but message sizes greater than 1Mb did not result in improved throughput.
- For MPI communication over Gigabit Ethernet on the Alpha cluster, the greatest throughput was achieved when message size was 4Kb or greater. Again, there was little improvement with message sizes beyond 4Kb. For the Intel cluster, the best throughput was achieved with message sizes between 4Kb and 100Kb. Message sizes beyond 100Kb actually resulted in degraded performance.
- For MPI uni- and bidirectional nonblocking communication on Myrinet on the Intel cluster, message sizes greater than 1Mb resulted in the best throughput. However, throughput for bidirectional nonblocking communication decreased for message sizes greater than 10Mb.
- For QsNet on the Alpha cluster, generally, message sizes of 10Kb resulted in the best throughput for both blocking and nonblocking communication. The exception occurs with bidrectional nonblocking communication where throughput decreases when message sizes larger than 10Kb are used. This seems to be a result of the slower performance of the Alpha processors, as they seem unable to handle the volume of packets.

There are a number of interesting directions for subsequent work. While Hpcbench already supports a large number of variables and protocol options for experimentation, there are other features that should be added, for example support for other MPI methods of communication besides point-to-point. Naturally, as the feature set of Hpcbench increases, more work will be required in terms of interfacing with the toolset and in managing the larger sets of possible experiments.

Another interesting topic for future experimental study is the relationship between network performance, computational performance and particular applications. For example, in a Gigabit Ethernet cluster, different applications place different demands on the network. Are there network configurations that are better for certain types of applications? For certain types of application mixes, are there preferred configurations for overall performance.

References

[1] Boden, N., et al, *Myrinet: A Gigabit-per-Second Local Area Network*, IEEE Micro, Vol. 15, No. 1, 1995, pp. 26-36.

[2] Chun, B., et al. *Virtual Network Transport Protocols for Myrinet*. Technical report. UC Berkeley, 1998.

[3] Hughes-Jones, R., et al. *Performance Measurements on Gigabit Ethernet NICs and Server Quality Motherboards*. First International Workshop on Protocols for Fast Long-Distance Networks, February 2003.

[4] Hughes-Jones, R., *Udpmon network measurement tool*, http://www.hep.man.ac.uk/u/rich/net/.

[5] Huang, B. Network Performance Studies in HPC environments. Master's Thesis, The University of Western Ontario, Canada. March 2005.

[6] Iperf Homepage, http://www.iperf.org.

[7] MPICH Implementation and Documentation, http://www-unix.mcs.anl.gov/mpi/.

[8] Netperf Homepage, http://www.netperf.org.

[9] Nguyen, K. and Le, T. *Evaluation and Comparison Performance of Various MPI Implementations on an OSCAR Linux Cluster*. Proceedings of the International Conference on Information Technology: Computers and Communications, 2003.

[10] Petrini, F., et al. *The Quadrics Network (QsNet): High-Performance Clustering Technology*. IEEE Micro, January-February 2002, pp. 46-57.

[11] Prasad, R., et al. *Effects of Interrupt Coalescence on Network Measurements*. Proceedings of Passive and Active Measurement (PAM) Workshop, April, 2004.

[12] SHARCNET Homepage, http://www.sharcnet.ca.

[13] Sumimoto, S., et al. *High Performance Communication using a Commodity Network for Cluster Systems*. Proceedings of the IEEE, 2000.

[14] The TOP500 Supercomputers list, http://www.top500.org.

[15] Turner, D., and Chen, X. *Protocol-Dependent Message-Passing Performance on Linux Clusters*. Proceedings of the IEEE International Conference on Cluster Computing, 2002.

[16] Turner, D., Oline, A., Chen, X., Benjegerdes, T., *Integrating New Capabilities into NetPIPE*. PVM/MPI 2003: 37-44.

[17] Sterling, T.. Beowulf Cluster Computing with Linux. The MIT Press, 2002.