Available online at www.sciencedirect.com

SCIENCE DIRECT®

J. Parallel Distrib. Comput. 65 (2005) 1253–1260

Journal of Parallel and Distributed Computing

www.elsevier.com/locate/jpdc

Research Note

Analyzing MPI performance over 10-Gigabit ethernet[☆]

Justin (Gus) Hurwitz*, Wu-chun Feng

Research and Development in Advanced Network Technology (RADIANT), Computer and Computational Sciences Division, Los Alamos National Laboratory, MS D-451, Los Alamos, NM 87545, USA

Received 1 April 2005; accepted 1 April 2005
Available online 18 July 2005

Abstract

Recent work with 10-Gigabit (10 GbE) network adapters has demonstrated good performance in TCP/IP-based local- and wide-area networks (LANs and WANs). In the present work we present an evaluation of host-based 10 GbE adapters in a system-area network (SAN) in support of a cluster. This evaluation focuses on the performance of the message-passing interface (MPI) when running over a 10 GbE interconnect. We find that MPI over 10 GbE provides communications performance comparable to that of TCP alone and fairly competitive with more exotic technologies such as MPI over Quadrics. The optimization of MPI and MPI-based applications to make use of this performance, however, is a non-trivial task. Consequently, it is difficult for MPI-based applications to realize this performance when running current-generation 10 GbE hardware.
Published by Elsevier Inc.

Keywords: 10 GbE; Ethernet; MPI; SAN; Interconnect; Cluster

1. Introduction

Recently, we have shown that first-generation host-based 10 GbE adapters can sustain multi-gigabit throughput (4.8–7.3 Gb/s) with low latency (10–15μs) on commodity hardware [3,1,4]. However, these works focus soley on optimizing TCP over 10 Gigabit Ethernet (10 GbE) for bulk data-transfer applications rather than on 10 GbE performance in traditional message-passing environments found in clusters and supercomputer.

Given the relatively low latencies achieved in previous tests, 10 GbE has potential to be an important and viable technology in the system-area network (SAN) environment in support of clusters and supercomputers. Ethernet has long been discounted as a competitive solution for cluster interconnects because it typically had best-case latencies on the order of hundreds of μs. In comparison, common cluster interconnects, such as Quadrics, Infiniband, and Myrinet, provide deterministic any-to-any latencies of sub-10-μs [5,9].

Over the past decade, the message-passing interface (MPI) has clearly become the standard protocol for application-layer communication in cluster environments [12]. As such, an understanding of MPI's performance over a given interconnect is essential to understanding which interconnect is best suited to the purpose of a given cluster. To provide a metric of 10 GbEs viability in cluster environments, we therefore evaluate the performance of MPI running over 10 GbE's. In conjunction with work such as [5], this evaluation provides a picture of where 10 GbE currently fits as a cluster interconnect.

2. Background and previous work

At the time of this writing, the only commercially available host-based 10 GbE adapter is Intel's PRO/10 GbE LR Server Adapter. This card, which is the same one used in

[☆] This work was supported by the US DOE office of science through LANL contract W-7405-ENG-36. This paper is also available as the following LANL technical report: LA-UR 03-5728.

* Corresponding author. Fax: +1 505 665 4934.
E-mail addresses: ghurwitz@lanl.gov (J. Hurwitz), feng@lanl.gov (W.-c. Feng).

0743-7315/$ - see front matter Published by Elsevier Inc.
doi:10.1016/j.jpdc.2005.04.011

our previous testing, is described in detail in [3,1,4]. It is built around Intel’s 82597EX single-chip 10 GbE controller, which interfaces with 512-KB of flash memory and 1310 nm single-mode serial optics. The host interface runs via a 133-MHz PCI-X bus interface. This interface, the fastest used by production 10 GbE hardware on PC architectures, limits theoretical throughput to 8.5 Gb/s (133-MHz*64-bits).

When running in low-end commodity hardware (e.g., 2.2-GHz Intel Xeon-based systems), these cards demonstrated performance of nearly 5 Gb/s in a local-area network (LAN) environment. Similar configurations were used to saturate an OC-48 (2.5 Gb/s) trans-Atlantic connection to set the Internet2 Land Speed Record in February 2003. On Opteron- and Itanium2-based server-class machines, these cards show sustained performance of better than 7 Gb/s in a LAN and 6.25 Gb/s in a wide-area network (WAN).

These numbers establish 10 GbE as a critical technology in achieving multi-gigabit throughputs in LANs and WANs. Indeed, Intel’s adapter is currently the only way to interface a single computer directly with a high-speed WAN. Little work, however, has previously been done to consider 10 GbE in other environments. In particular, 10 GbEs potential role in the SAN remains a hotly contested topic.

SANs present a networking environment that is fundamentally different from that of LANs and WANs. Throughput, which is the most visible and one of the most important performance metrics in LANs and WANs, is relatively unimportant in the SAN. This is not to discount its importance; but other metrics are equally, and often more, important in the SAN. Many cluster-based applications depend heavily upon calculations that are limited by inter-node latency [13]. Inter-node latency is the time it takes for a small amount of data to get from one node to another; as opposed to bandwidth, which is the amount of data that can get from one node to another in that time. Similarly, many calculations occur simultaneously with data transfers; again, this is different from LANs, where computation is generally secondary to the transfer of data. Such calculations can be significantly slowed if network-related overhead requires too much processing time. It is therefore important that SAN interconnects not place too heavy a burden on the CPU.

Our work with 10 GbE has demonstrated inter-node latencies on the order of 10–15 μs, when running in a back-to-back configuration. With new LAN/SAN-oriented 10 GbE layer-2 switches capable of port-to-port latencies of under 1 μs [11,2], we can achieve end-to-end latencies an order of magnitude less than past incarnations of Ethernet. While 10 GbE’s latency may not be as low as that of specialized SAN interconnects, it is low enough to warrant consideration in high-end computing clusters.

Perhaps of more concern than 10 GbEs inter-node latency in a SAN environment is the overhead required to process 10 GbE traffic. Other SAN technologies typically perform much, if not all, network-related processing on dedicated chipsets that are integrated with the adapters. This offloading of network processing frees the CPU to work almost entirely on application computation. Current 10 GbE adapters are only capable of offloading a small portion of this network overhead. Even so, our previous work indicates that the CPU is not entirely consumed by network processing at 10 GbE, and hence, that it may have usable cycles left for application use during heavy network communication.

Although previous work has shown that 10 GbE *might* have utility as a SAN interconnect, the work is not conclusive. To evaluate its potential beyond speculation, we run a series of synthetic benchmarks to determine 10 GbEs actual performance under common SAN-interconnect situations.

3. Testing environment

To run our benchmarks, we setup a simple two-node cluster environment with two Opteron-class machines connected in a back-to-back topology with 10 GbE. We opt for such a configuration, rather than one with more nodes, due to the current state of 10 GbE switching technology. All currently available 10 GbE-capable switches are designed to be used as layer-3 IP routers. These switches do not offer the latency characteristics that are required in a SAN interconnect. However, a new generation of layer-2 10 GbE switches that offer sub-μs port-to-port latency is now entering the market, e.g., the Fujitsu 10 GbE switch recently demonstrated at SC2003 [11,2]. If 10 GbE is to be used in a SAN environment, it will likely be used in conjunction with switches such as these.

Rather than evaluate a larger number of 10 GbE-enabled nodes in an environment that poorly represents actual implementations, we prefer to use a two-node system the more accurately models switching latencies. As a consequence, many common MPI benchmarks (e.g., all-to-all and allreduce) are of marginal value, and are therefore not reported.

In setting up our testing environment, we have taken care to apply a baseline level of optimizations to both our hardware and our software. These optimizations are of the general nature that we expect would be used as the default in a cluster configuration. We do not apply any benchmark or network-specific optimizations, because such run-time optimizations cannot be made in production environments. The results, as should be expected, are therefore not the best possible for each specific test.

3.1. Testing hardware

Each of our nodes is a dual AMD Opteron 246 running at 2.0 GHz. The CPUs run on Tyan S2880 motherboards with 2-GB of PC3200 memory. The AMD Opteron has a 6.4 GB/s (51.2 Gb/s) memory bus and a HyperTransport-based (6.4 GB/s) connection to two PCI-X buses. Each PCI-X bus can run at 133 MHz. The 10 GbE adapters are each placed in a PCI-X slot in each of the machines. This places a theoretical limit of 8.5 Gb/s on each adapter. The 10 GbE adapters are then connected to each other by a pair of single-mode, fiber-optic cables.

Previous testing has demonstrated that these systems, in this configuration, can sustain better than 7.3 Gb/s from end-host to end-host when running TCP/IP. However, a substantial amount of adapter- and application-specific tuning is required to achieve this throughput. We forego this level of custom optimization in our present evaluation. Regardless, we expect 7.3 Gb/s to be an empirical upper-bound for our performance evaluation.

We also note that the 10 GbE adapters support MTU sizes up to 16 KB. However, because typical Ethernet environments only provide support for 1500- or 9000-byte MTUs, we limit our tests to use these two common MTU sizes in order to better model expected real-world performance.

Finally, with respect to hardware, we configure the 10 GbE cards to use an interrupt delay of approximately 5 μs. Though this setting increases end-to-end latency by 5 μs, we generally achieve optimal TCP throughput performance with it. This is the default value used by the adapters as shipped.

Due to the latency-sensitive nature of SANs, we also ran our tests without interrupt delay. Unfortunately, this resulted in a great deal of connection instability (dropped packets and similar errors). We present a preview of these results below to demonstrate the problem. The remainder of our results exclude these lower-latency configurations due to their instability. Further exploration of this problem is beyond the scope of this presentation.

3.2. *Testing software*

Our benchmarks run MPI over TCP in a two-node cluster. Each node runs a Linux installation with a 2.4.25 kernel and the standard Linux TCP stack. We configure the kernel to allow a maximum TCP window of 16 MB.

There are several generally available MPI implementations that will work in this environment. We choose to run the LAM-MPI to provide MPI support. Previous experience with this MPI has given us great confidence in its ability to work well in high-performance networks. We discounted the use of other comparably high-performance implementations (such as LAMPI) out of consideration for the pending releases of new versions.

We use standard installation options with the sole exception of setting the *rpi_tcp_short* value to 2 MB (from the default value of 64 KB). This value controls the threshold at which LAM-MPI switches from its short-message to long-message protocols. The short-message protocol requires fewer transfers per message but does not perform as well as the long-message protocol when the message size is very large. We experience poor performance with settings less than about 512 KB, little difference between 512 KB and 2 MB, and virtually no difference when running with even larger settings.

There are certainly other options that could be tuned to improve performance. However, for the sake of comparison, we have kept our configuration as general as possible. We consider the modification of *rpi_tcp_short* necessary and appropriate in any high-bandwidth and low-latency network.

The first test the we run uses NTTCP [8] to measure TCP performance. Because MPI is running over TCP, these results provide an important baseline against which to compare our MPI results. Furthermore, NTTCP results in this environment can be directly compared to those reported in previous work. NTTCP operates by sending a set number of packets of a given size to the remote host and reports throughput as the ratio of the total number of bytes sent to the wall-clock time taken to send them. We run this test sending 4096 iterations of packets ranging in size from 8 bytes (8 B) to 3,145,728 bytes (3 MB).

The next test that we run is NetPipe [7]. NetPipe is a protocol-independent benchmarking tool that allows us to run identical throughput and latency tests over both TCP and MPI. These results give us a benchmark understanding of the communications overhead burden that LAM-MPI adds to TCP processing. NetPipe performs two types of throughput tests: ping-pong and stream. Ping-pong tests measure the time required for a pair of hosts to transmit data from one to the other and back. This is an important metric for latency-sensitive (and many bandwidth-intensive) applications. The stream test sends a set amount of data and reports the time required to transmit it, much in the same way as NTTCP. NetPipe's stream algorithm is more representative of application-level throughput, whereas NTTCP represents the transport-level bandwidth.

NetPipe is also used to measure host-to-host latency. This measurement is made by dividing the transfer time by the total number of transfers in each direction.

Finally, to measure MPI-application performance, we run MPBench (now part of llcbench [6]). MPBench is a suite of MPI applications that are used to measure common MPI performance metrics. The focus of these benchmarks is the application, as opposed to the network. As a result, these benchmarks are representative of the performance that can be expected by a typical MPI application, as opposed to a typical network-measurement tool.

MPBench evaluates several performance metrics: uni- and bi-directional throughput, round-trip time (connection latency), MPI_Send() latency, and broadcast, reduce, all-to-all and all-reduce throughputs. Of these tests, only the first set is of interest to our two-node environment; the rest are variants of the first two throughput tests and produce results that are roughly identical to the two-node case.

In selecting these benchmarks, we have tried to find a happy-medium between application-oriented and micro benchmarks. The wide range of applications, and their specific performance requirements, makes it an impossible task to present generally useful "application" benchmarks. Traditional micro-benchmark results, on the other hand, are difficult metrics from which to extrapolate application- and system-level performance. While both types of benchmarking should ultimately be performed, neither is particularly salient to our present inquiry, viz., whether 10 GbE is a workable interconnect for MPI-based applications generally.

3.3. Methodology

Each test is run multiple times to confirm that the results are consistent, both with respect to the given test and with respect to similar tests. Any observed abnormalities are examined and re-tested to determine whether they arise from factors unrelated to the tests themselves.

We monitor the system load throughout our testing. System load is represented as a unitless number and is generally calculated as the ratio of the number of processes added to a CPU's run queue to the number of processes removed in a given time. A load of 1.0 means that one CPU is constantly busy; less than 1.0 means that one CPU has free cycles; and greater than 1.0 means that more work is being requested of a CPU than it is capable of doing. The "optimal" load in a multiprocessor system is generally equal to the number of CPUs in that system.

To monitor the load, we access and timestamp the kernel's current load computation at 5-s intervals and record it to a file. We then timestamp the beginning and end of each test. The 5-s granularity is the highest resolution measurement of load supported by the kernel. This allows us to identify the load for the duration of each test that we run.

Because the load is measured at so coarse a granularity, we do not present graphs of the data. To determine what values to report, we compare the measurements from multiple runs to confirm that each represents a stable value, and report the highest value. Where aberations are noticed, we rerun the appropriate tests. Such aberations can result from system processes (e.g., crontab jobs) automatically running on the testing systems.

4. Experimental results

MPI over 10 GbE uses TCP as its transport; similarly, MPI-based applications use MPI as their transport. It is therefore necessary to understand our testbed's TCP performance before considering its MPI performance, and its MPI performance before considering its MPI-based application performance. Our results are therefore presented in this order.

To facilitate comparison of results, all of our throughput results are plotted in the same domain and range and at the same scale. The only exception are the results in Fig. 1, in which the interrupt delay was disabled.

4.1. NTTCP results

We first test 10 GbE TCP throughput with the NTTCP program (Fig. 1). The peak throughput observed with 1500- and 9000-byte MTUs are approximately 3.6 and 6.0 Gb/s, respectively. This peak falls off substantially for payloads larger than 400 KB when using a 9000-byte MTU. A similar, though less substantial, drop occurs at the same point when using a 1500-byte MTU.

This drop is related to the kernel's ability to allocate buffer memory. Simultaneous with this drop, we see a large increase in system load, from well below 0.5 to about 0.8 and with spikes as high as 1.2. Additionally, the operating system reports a large number of memory-allocation errors.

We have not observed this drop in previous testing. While it is important that its cause ultimately be understood, we consider this a question of optimization beyond what can be expected in a typical SAN. This behavior is taken as the baseline performance for the remainder of our tests.

Additionally, these graphs include throughput when running with a 0-μs interrupt delay. It is clear that running without this delay is detrimental to performance; such behavior is expected, due to the increased interrupt load placed on the host system. In addition to generally decreased throughput we also observe a great deal of "jitter" in the performance. This jitter is best characterized as a large number of very low performance data points (performance drops at these point by as much as 50%). These points are not consistent between test runs (whereas the variability in performance seen with the 5-μs delay does persist between runs). They appear to be caused by bursts of lost packets.

In light of this behavior, we exclude tests that disable the interrupt delay from our results. The performance is too unreliable either to be confidently reported or to be used in a production environment. Again, it is possible (and we believe likely) that this problem could be corrected with the proper, if somewhat extensive, optimizations. Alternatively, we hypothesize that this instability is caused by the same memory allocation errors seen above for large (400KB) payloads.

4.2. NetPipe results

We next run a series of tests with NetPipe (Fig. 2). Throughout these tests, we see the same drops in throughput and similar load characteristics that were observed in the NTTCP tests.

The first set of results are from NetPipe's stream test. As expected, this test, which is very similar to NTTCP, yields observed TCP throughput that is similar to that of NTTCP, albeit marginally lower: roughly 3.0 Gb/s for 1500-byte packets and 5.8 Gb/s for 9000-byte packets. Contrary to expectations, however, MPI throughput is almost equal to that of TCP. In some cases, particularly around peak through-

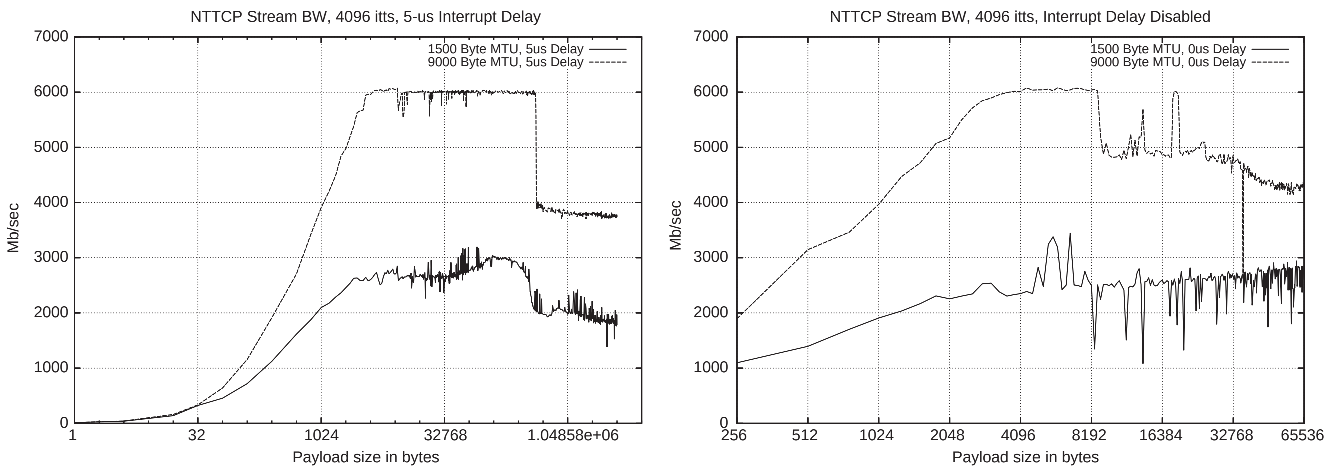

Fig. 1. NTTCP results, with 5- and 0-μs interrupt delay.

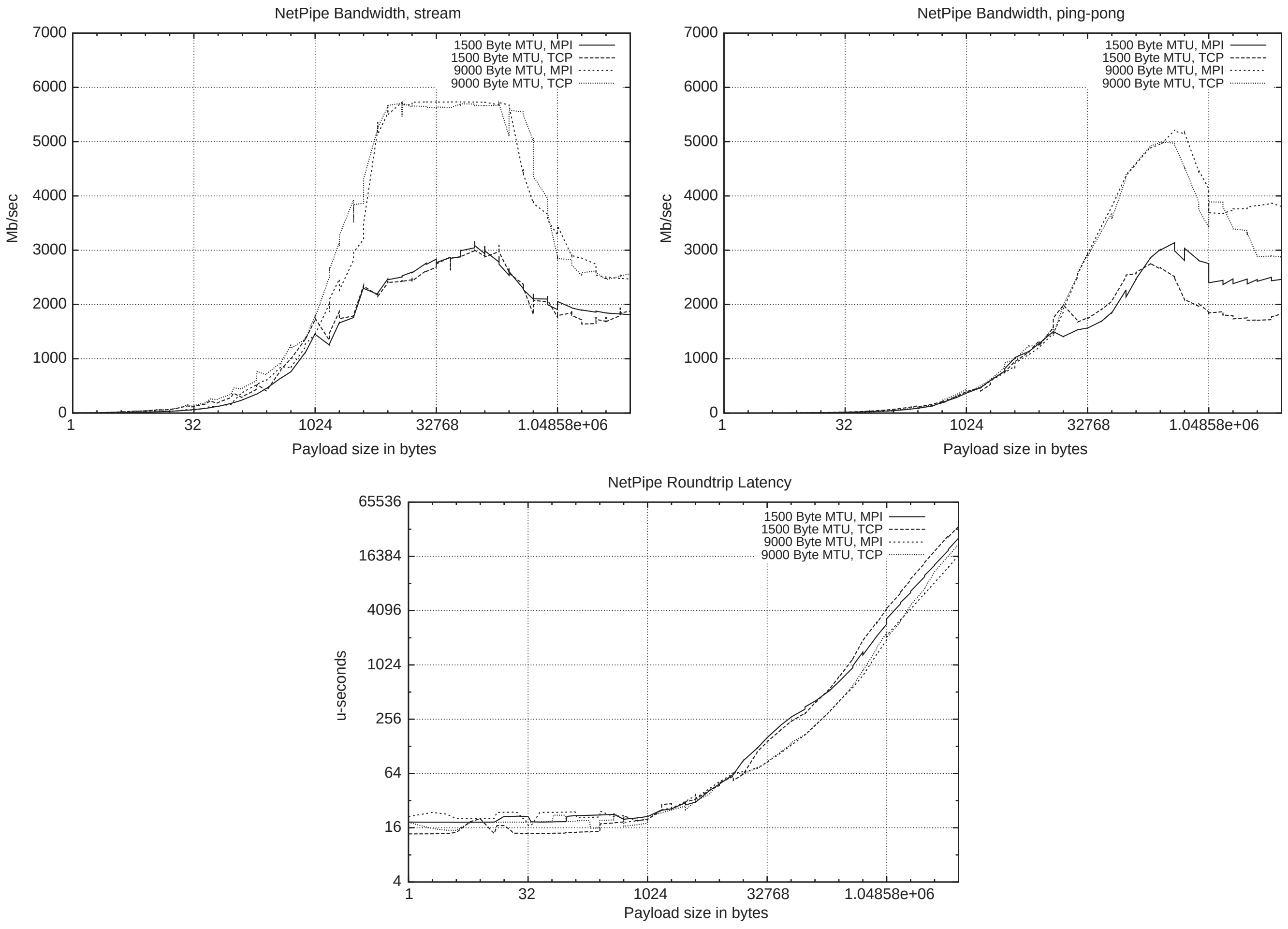

Fig. 2. NetPipe results.

put, MPI outperforms TCP! Additionally, MPI throughput is more stable, especially at its peak. These results can be attributed to the implicit tuning and TCP optimizations that LAM-MPI enables, transparent to the user and application.

These improvements are far more apparent when running NetPipe's ping-pong tests. In these tests, the same TCP falloff seen in Fig. 1 is observed. Given the nature of these tests, we expect peak throughput to be lower than in the

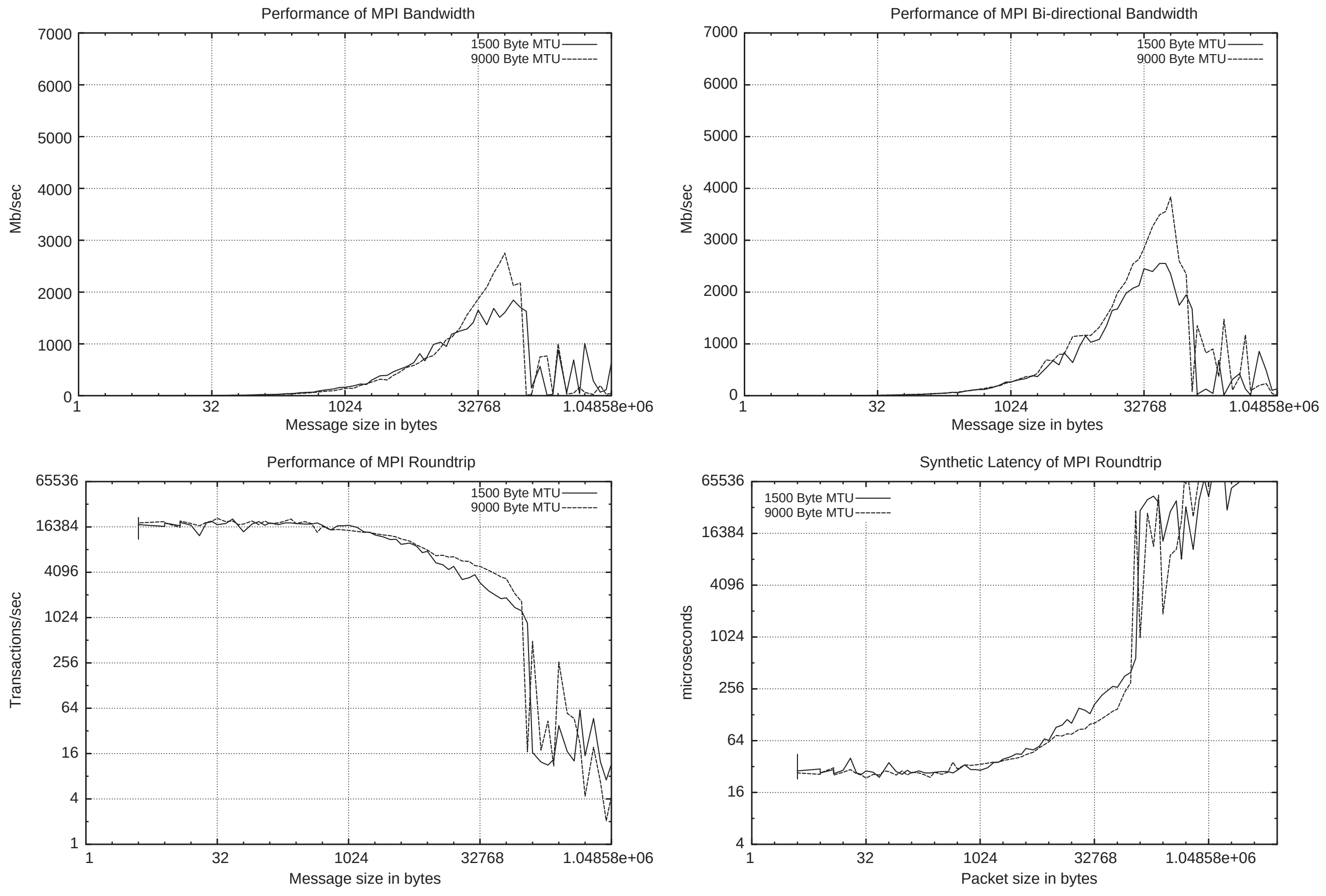

Fig. 3. MPBench results.

stream tests and the "ramp-up" to that throughput to take longer. Again, the tests meet these general expectations. MPI performance, however, is noticeably greater than that of TCP, and the throughput falloff is significantly smaller. Peak throughput for TCP is roughly 2.8 and 5.0 Gb/s for 1500- and 9000-byte MTUs, respectively; post-falloff throughput is roughly 1.7 and 2.7 Gb/s, respectively. For MPI, peak throughput is roughly 3.1 and 5.2 Gb/s for these respective MTUs; post-falloff throughput is roughly 2.5 and 3.9 Gb/s for these respective MTUs. The improved performance of MPI is, again, attributed to implicit TCP optimizations.

The final NetPipe results that we present are the latencies observed in the above tests. Here we observe latency on the order of 13–17 μs for small TCP packets, and 17–22 μs for small MPI packets. Latencies remain in these ranges for packets up to roughly 1500-bytes. Larger packets require multiple transfers when using a 1500-byte MTU. Additionally, latency increases due to time spent transferring memory across the memory bus on the host systems.

As packet size increases, we observe two trends. First, packets sent with a 9000-byte MTU have lower latency than those sent with a 1500-byte MTU. Generally, we see about a two-fold difference in latency between the different MTUs. Second, packets sent with MPI have lower latency than those sent with TCP. These results again indicate MPI optimizations to TCP. Furthermore, they help to explain the results from our ping-pong tests: decreased latency improves ping-pong performance.

4.3. MPBench results

MPBench is the final test suite that we run. The results of the tests that are pertinent to our experimental setup are shown in Fig. 3. The latency values shown in the figure are computed manually as the inverse of half the number of transaction per second. Other methods of measuring latency, such as those used by benchmark, can be artificially deflated due to the buffering behaviour of the MPI_send() call.

We observe very high CPU load throughout all of these tests. The average load ranges between 1.75 and 2.25, with frequent readings on the order of 2.5. On dual-processor systems, such as our test systems, this means that the CPUs were not able to keep up with system demands.

The first pair of tests we discuss are the uni- and bi-directional throughput tests. We measure peak uni-directional throughput of about 1.9 and 2.8 Gb/s for

1500- and 9000-byte MTUs, respectively. We observe bidirectional peaks of 2.5 and 3.7 Gb/s for the same respective MTUs.

These tests show a number of interesting features. Foremost, we observe a near-complete drop in performance for payloads greater than 128 KB. This drop, while similar to the one observed in our TCP tests, begins to occur with much smaller packets and is far more significant than the TCP drop. Previous experience with MPBench has shown that the LAM-MPI short-message to long-message crossover point can cause a similar drop at 64 KB. In these experiments, however, we are unable to eliminate this problem by altering the crossover point. Given the increased system load in these tests, this could be the same memory allocation problem seen with larger payloads in earlier tests.

The next thing to notice is the slope of MPBench's throughput curve relative to that of NetPipe. The slope of MPBench's curve grows far more gradually than either of NetPipe's curves. This indicates that, for a given packet size, MPBench's achieved throughput is less than that of NetPipe. Given the approximate differences in slope, it seems reasonable that MPBench is only capable of, at best, 60–75% of NetPipe's peak throughput.

In addition to the throughput tests, we present the results of MPBench's round-trip test. MPBench reports an average of about 20,000 round-trip transactions per second for small messages (up to about 1500 bytes). Larger messages show a gradual decrease in this value, up to 128 KB. As in the other MPBench results, we see a dramatic drop in performance after this point.

These results are inversely proportional to twice the latency. They therefore correspond to a latency of 25–30 μs for small packets and about 250 μs for 128-KB packets. These numbers correspond well to the NetPipe latency results, with an additional overhead of a few μs. As with the NetPipe latency tests, we see steady performance as the message sizes approach about 1500 bytes, after which performance degrades. We also see the results for the 9000-byte MTU behave in a similar manner: as the message size grows large, we see about a two-fold difference in performance relative to the smaller MTU.

5. Synopsis of results

One apparent trend throughout our tests is that poor performance at one layer adversely affects performance at every higher layer. Though intuitive, this idea is so important to these tests that it deserves explicit mention.

A poorly optimized TCP stack results in a poor MPI performance. Application-layer latencies are worse than protocol- and link-layer latencies. An MPI implementation may make use of application-layer TCP optimizations (such as we observe in our tests) that improve performance relative to TCP. Such "performance tweaks" cannot, however, overcome fundamental limitations in the optimization of TCP. The result is that MPI over TCP over 10 GbE requires optimizations at all three levels to perform well. The necessary optimizations might well be different for different applications. It is unreasonable to expect this level of custom configuration in a SAN environment.

Our NetPipe results indicate that some MPI-based applications can reach very high performance when running over 10 GbE. These tests also confirm the conclusions of our previous work that TCP's performance over 10 GbE is not CPU-limited.

The principal difference between the NetPipe and MPBench benchmarks is how they manage buffer memory. NetPipe is specifically tuned to measure peak throughput. As such, it uses an optimized algorithm to get higher performance out of MPI. For instance, it pre-posts receives to MPI, which pre-allocates buffer space for data reception. MPBench, on the other hand, is designed to model typical application performance. It therefore implements a far simpler transfer algorithm, and consequently, does not achieve nearly as high performance.

The dramatic differences between NetPipe and MPBench performance again demonstrate the importance of configuration and optimization when running MPI over 10 GbE.

6. Conclusion

It is difficult to compare MPI over 10 GbE to MPI over other SAN interconnects. This is largely because the MPI specification has many implementations. Each implementation, while compatible with MPI, has intrinsic peculiarities that affect network performance in positive or negative ways. Achieving maximum MPI performance requires the MPI implementation to be tuned to its operating environment and presumes that this environment itself is already tuned. Similarly, each application ought to incorporate implementation-specific optimizations if it is to reach peak performance.

Much of the promise of 10 GbE is that it allows the performance of TCP-based applications running over Fast Ethernet or Gigabit Ethernet to be increased without requiring modifications to the application code itself. In light of the above conditions, this advantage does not entirely carry over to MPI-based applications.

The proprietary MPIs that many other SAN interconnects support obviate much of the need for difficult configuration and optimization that is required when running MPI over 10 GbE. Their MPI implementations are inherently tuned to their particular network environments. As well, they frequently take advantage of comprehensive processing offload techniques to reduce the load on the host CPU, which makes performance less dependent upon how applications interface with the MPI API.

This raises a difficult question: since other MPIs are inherently tuned to their interconnects, do we consider a comparison to 10 GbE fair if it is tuned or untuned? There is little doubt that our benchmarks could be optimized to greatly improve application performance. However, such results would undermine many of the features that make 10 GbE an attractive interconnect. Furthermore, they would lead to unrealistic performance expectations.

While this is an important question at present, we expect its importance to wane with the next generation of host-based 10 GbE adapters. The first generation hardware used in our tests is now two years old; a next generation of hardware, from several manufacturers, is fast approaching the market. Our basic conclusion is that the first generation of host-based 10 GbE adapters is a suitable interconnect for many MPI-based applications, which may require burdensome optimizations for other applications. We are confident that the next generation of 10 GbE adapters can only increase performance, and decrease the burdens of optimization.

Many new 10 GbE adapters promise to include substantial protocol offload support (such as TOE and RDMA over IP [10]). On its face, these technologies will greatly increase TCP performance of 10 GbE. Indeed, our initial testing of a prototype TOE-enabled 10 GbE adapter has demonstrated throughput in excess of 7.1 Gb/s using a 1500-byte MTU with a latency of 9 µs, and barely measurable increase in system load. Such improvements to TCP performance should offer comparably great increases in performance for MPI running over TCP.

In addition, many of these adapters will include on-board programmable logic. While initial offload support is principally targeted at TCP, the programmability of new 10 GbE adapters might be leveraged to enable MPI offload support. With such developments on the horizon, 10 GbE's future role in the SAN environment looks bright.

Regardless of its future potential, the use of MPI over 10 GbE faces substantial burdens in the current SAN environment. The question today is not whether we can achieve good performance: we can. Rather, the question is how difficult it is to achieve that performance. One of 10 GbEs greatest benefits is that it requires no changes to existing code bases for applications to realize multi-gigabit performance in any networking environment. The level of specialized optimization required to get competitive performance out of MPI-based applications, which potentially includes changes to the applications themselves, seriously threatens the benefits of running 10 GbE in a SAN environment.

Acknowledgments

We would like to thank our co-worker Jeremy Archuleta for his assistance configuring MPI to work in our testing environment, and for his continued help understanding its performance characteristics.

References

[1] W. Feng, G. Hurwitz, H. Newman, S. Ravot, L. Cottrell, O. Martin, F. Coccetti, C. Jin, D. Wei, S. Low, Optimizing 10-Gigabit Ethernet for Networks of Workstations, Clusters and Grids: a Case Study, Proceedings of SuperComputing 2003 (SC2003), November 2003.
[2] Fujitsu Ethernet Switch Crip, http://www.fma.fujitsu.com/10Gethernet/.
[3] G. Hurwitz, W. Feng, Initial End-to-End Performance Evaluation of 10-Gigabit Ethernet, Proceedings of Hot Interconnects 11 (HotI03), August 2003.
[4] G. Hurwitz, W. Feng, End-to-End Performance of 10-Gigabit Ethernet on Commodity Systems, IEEE Micro. 24 (1) (2004).
[5] J. Liu, B. Chandrasekaran, W. Yu, I. Wu, D. Buntinas, S. Kini, P. Wyckoff, D.K. Panda, Micro-Benchmark Level Performance Comparison of High-Speed Cluster Interconnects, Proceedings of Hot Interconnects 11 (HotI03), August 2003.
[6] LLCbench Home Page, http://icl.cs.utk.edu/projects/llcbench/.
[7] NetPIPE, http://www.scl.ameslab.gov/netpipe/.
[8] NTTCP: New TTCP program, http://www.leo.org/∼elmar/nttcp/.
[9] F. Petrini, D. Addison, J. Beecroft, D. Hewson, M. McLaren, Quadrics QsNet II: A Network for Supercomputing Applications, Proceedings of Hot Chips 14 (HotChips03), August 2003.
[10] A. Romanow, S. Bailey, An Overview of RDMA over IP, Proceedings of the First International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet 2003), February 2003.
[11] T. Shimizu, et al., A single chip shared memory switch with twelve 10Gb ethernet ports, Proceedings of Hot Chips 14 (HotChips03), August 2003.
[12] The Message Passing Interface (MPI) standard, http://www-unix.mcs.anl.gov/mpi/.
[13] W. Washington, C. Parkinson, An Introduction to Three-Dimensional Climate Modeling, University Science Books, 1991.

Justin (Gus) Hurwitz is a member of the Research & Development in Advanced Network Technology (RADIANT) team in the Computer & Computational Sciences Division at Los Alamos National Laboratory. His research interests include protocol and OS design for high-performance networking, and general approaches to fair utilization and optimization of finite resources. He has a Computer Science background and a Bachelor's degree in Liberal Arts from St. John's College. He is currently a student at the University of Chicago School of Law, where he is focusing on Intellectual Property law and the interaction between science, law, and policy.