Comparing Ethernet and Myrinet for MPI Communication

Supratik Majumder
Rice University
Houston, Texas
supratik@rice.edu

Scott Rixner
Rice University
Houston, Texas
rixner@rice.edu

ABSTRACT

This paper compares the performance of Myrinet and Ethernet as a communication substrate for MPI libraries. MPI library implementations for Myrinet utilize user-level communication protocols to provide low latency and high bandwidth MPI messaging. In contrast, MPI library implemenations for Ethernet utilize the operating system network protocol stack, leading to higher message latency and lower message bandwidth. However, on the NAS benchmarks, GM messaging over Myrinet only achieves 5% higher application performance than TCP messaging over Ethernet. Furthermore, efficient TCP messaging implmentations improve communication latency tolerance, which closes the performance gap between Myrinet and Ethernet to about 0.5% on the NAS benchmarks. This shows that commodity networking, if used efficiently, can be a viable alternative to specialized networking for high-performance message passing.

1. INTRODUCTION

Specialized networks, such as Myrinet and Quadrics, are typically used for high-performance MPI computing clusters because they provide higher bandwidth than commodity Ethernet networks [3, 15]. In addition, custom user-level communication protocols that bypass the operating system are used to avoid the memory copies between the user and kernel address spaces and the network I/O interrupts that are necessary in TCP-based solutions. These custom user-level protocols typically provide lower communication latency than TCP, making them attractive for MPI applications.

The advent of 1–10 Gbps Ethernet, however, minimizes the difference in bandwidth between specialized and commodity networks. Furthermore, the decreased latency of user-level protocols comes at a cost. The operating system provides several mechanisms to hide message latency by overlapping computation and communication. These mechanisms cannot be used efficiently by user-level protocols that bypass the operating system. In addition, TCP is a carefully developed network protocol that gracefully handles lost packets and flow control. User-level protocols can not hope to achieve the efficiency of TCP when dealing with these issues. Furthermore, there has been significant work in the network server domain to optimize the use of TCP for efficient communication.

This paper evaluates the performance differences between GM over 1.2 Gbps Myrinet and TCP over 1 Gbps Ethernet as communication substrates for the Los Alamos MPI (LA-MPI) library. The raw network unidirectional ping latency of Myrinet is lower than Ethernet by almost 40 μsec. However, a considerable fraction of this difference is due to interrupt coalescing employed by the Ethernet network interface driver. Disabling all interrupt coalescing at the driver level reduces this latency difference to about 10 μsec. The higher peak network bandwidth and lower raw network latency of GM over Myrinet translates into substantially lower message ping latency and higher message bandwidth than TCP over Ethernet. However, these metrics only consider communication performance in isolation, and therefore do not accurately predict application performance. This paper shows that despite the higher latency and lower bandwidth of Ethernet, the LA-MPI library using GM over Myrinet is only 5% faster than the LA-MPI library using TCP over Ethernet on 5 benchmarks from the NAS parallel benchmark suite. Furthermore, when TCP is used more efficiently, LA-MPI using TCP over Ethernet is able to come within 0.4% of the performance of LA-MPI using GM over Myrinet.

The rest of the paper is organized as follows. Section 2 gives an overview of the architecture of the LA-MPI library, as well as its TCP over Ethernet and GM over Myrinet paths. Section 3 compares the inherent communication characteristics of the Myrinet and Ethernet networks. Section 4 presents a comparison of the aforementioned MPI library versions on a set of application benchmarks taken from the NAS parallel benchmark suite. Section 5 discusses briefly about some other research works in the field of high-performance MPI computing. Finally, Section 6 concludes the paper.

2. MPI LIBRARY ARCHITECTURE

Most MPI libraries provide messaging support on a broad range of interconnect technologies, such as Ethernet, Myrinet, and Quadrics. This paper compares the performance of the Ethernet and Myrinet messaging layers in the

Los Alamos MPI (LA-MPI) library. The LA-MPI library is a high-performance, end-to-end, failure-tolerant MPI library developed at the Los Alamos National Laboratory (LANL) [7]. The LA-MPI version 1.4.5 library implementation consists of three almost independent layers: the *MPI Interface Layer*, the *Memory and Message Layer* (MML), and the *Send and Receive Layer* (SRL). The interface layer provides the API for the MPI standard version 1.2 specification. The MML provides memory management, storage of message status information, and support for concurrent and heterogeneous network interfaces. Finally, the SRL interfaces with the specific network hardware in the system, and is responsible for sending and receiving messages over the network. LA-MPI supports each type of network, including Myrinet and Ethernet, using independent *path* modules within the SRL for each type of network. LA-MPI provides a TCP path (LA-MPI-TCP) in order to communicate over Ethernet networks and a GM path (LA-MPI-GM) in order to communicate over Myrinet networks.

All communication within LA-MPI is controlled by the progress engine, which is responsible for making progress on pending requests in the library. The progress engine mainly belongs to the MML with helper routines in the SRL. The MML maintains a list of all pending messages—incomplete sends and receives—currently active in the library. Conceptually, the progress engine loops over all of these incomplete requests and attempts to send or receive as much of each message as possible. In order to make progress on each request, the engine invokes the send or receive routine of the appropriate path within the SRL to access the operating system and/or the networking hardware. Most calls into the interface layer of the library also invoke the progress engine. On a blocking MPI call, the progress engine continues to loop through all pending requests until the specified request is completed, and the blocking call can return. In contrast, on a non-blocking MPI call, the progress engine just loops through all of the pending requests once, and then returns regardless of the state of any of the requests.

An optimized, event-driven, TCP path (LA-MPI-TCP-ED) splits the library into two threads—a main thread to provide the core functionality through the functional interface of the library, and an event thread to handle message communication using TCP sockets over Ethernet [10]. This effectively offloads the progress engine to a separate event thread. LA-MPI-TCP-ED enables more effective overlap of computation and communication of an MPI application, leading to better performance of the application. However, the event-driven version of the LA-MPI library relies strongly on operating system support for detecting network events. A similar approach cannot be applied to Myrinet messaging since it uses a user-level GM communication library, which does not have the necessary facilities for detecting and delivering events.

2.1 The TCP Path

The TCP path supports TCP message communication over Ethernet using the socket interface to the operating system's network protocol stack. The TCP path utilizes events to manage communication. An event is any change in the state of the TCP sockets in use by the library. There are three types of events: read events, write events, and exception events. A read event occurs when there is incoming data on a socket, regardless of whether it is a connection request or a message itself. A write event occurs when there is space available in a socket for additional data to be sent. An exception event occurs when an exception condition occurs on the socket. Currently, the only exception event supported by the socket interface to the TCP/IP protocol stack is the notification of out-of-band data. The current LA-MPI library does not use any out-of-band communication, so exception events are only included to support possible future extensions.

The TCP path, like other paths supported by LA-MPI, provides its own progress routine which gets invoked by the library's main progress engine. The TCP path maintains three separate lists, one for each type of event, to keep track of all the events of interest. At various points of execution, the TCP path pushes these events of interest on to the respective event lists. In addition, callback routines are registered for each event. The progress mechanism of the TCP path uses these lists to inform the operating system that it would like to be notified when these events occur. The `select` system call is used for this purpose. In LA-MPI, these calls to `select` are non-blocking and return immediately. If any events of interest have occurred, they are each handled in turn by the appropriate callback routines and finally control is returned to the progress engine. The progress engine attempts to make progress on all active paths in the library in turn, and thus could invoke the TCP progress routine several times in the process of satisfying a blocking MPI request.

For communication in the TCP path, the library uses at least one bidirectional TCP connection between each pair of communicating nodes. These connections are not established during initialization. Rather, they are established only when there is an actual send request to a node that does not already have an open connection to the sending node. Once a connection is established, it remains open for future communication, unless an error on the socket causes the operating system to destroy it. In that case, a new connection would be established for future messages between those nodes.

The TCP path relies on the operating system for its memory buffering requirements. The socket buffers provide the necessary buffering for outgoing and incoming messages. The LA-MPI library copies data out of and into the socket buffer with the `read` and `write` system calls. However, the socket interface only allows for sequential access of data from the head of its buffer by the user application. Thus, a message has to be copied out of the socket buffer before the next message can be accessed. As a result, in the case of a message receive, which has not been posted in advance by the user application, the LA-MPI library needs to allocate temporary buffer space to hold this message after reading it out of the socket buffer. The only other instance when the library needs to provides its own message buffering is for buffered MPI send requests.

2.2 The Myrinet Path

The GM path of the LA-MPI library supports MPI communication over Myrinet. For this purpose the path utilizes the GM communication library, which is a message-based com-

munication system for Myrinet. The user-level GM communication library provides a low CPU overhead, portable, low latency, and high bandwidth communication substrate. GM allows memory-protected user-level OS-bypass network interface access to applications, thereby achieving zero-copy message sends and receives at the application level. However, to support this feature, GM requires the presence of DMA accessible memory on the host to send messages from, or receive messages into.

The GM communication library provides reliable, ordered delivery between communication endpoints, called *ports*. The communication model is connectionless: the LA-MPI library simply builds a message and sends it to any port in the network. The largest message GM can send or receive is limited to $2^{31} - 1$ bytes, but since send and receive buffers must reside in system memory, the maximum message size is limited by the amount of memory the GM driver is allowed to allocate by the operating system. During initialization, the LA-MPI library obtains memory from the operating system using the straightforward `gm_dma_malloc()` call. Memory management during the runtime of a MPI application is performed directly by the MML layer of the library, which is also responsible for managing network interface memory.

Message order is preserved only for messages of the same priority, from the same sending port and directed to the same receiving port. Messages with different priorities never block each other. The LA-MPI library only uses the `GM_LOW_PRIORITY` for all its message communication over Myrinet. The sends and receives in GM are regulated by implicit tokens, representing space allocated to the library in various internal GM queues. The LA-MPI library may call certain GM API functions only when it possesses an implicit send or receive token, and in calling that function, the library implicitly relinquishes the token. It is entirely the LA-MPI library's responsibility to keep track of the number of tokens of each type it possess. These tokens are implicitly possessed by the library at initialization.

Unlike the TCP path, the GM path of LA-MPI does not use the operating system to provide memory buffers. Furthermore, the GM communication library does not provide any memory buffer management routines, and holds the GM path of LA-MPI responsible for providing all buffer management. However, being a user-level communication library, GM allows the library to allocate buffers for message sends and receives directly on the network interface card. This helps to eliminate copies between user-space and kernel-space. The GM path also does not have to provide temporary buffering for unexpected messages since there is no restriction on access of data from the network interface buffers. These buffers on the network interface are managed through the use of the send and receive tokens by the GM path of the LA-MPI library.

3. NETWORK PERFORMANCE

Understanding the performance differences of MPI applications on Gigabit Ethernet and Myrinet requires a thorough comparison of both raw network properties and MPI library implementation characteristics. The raw network latency and bandwidth measurements show the inherent capabilities and limitations of the interconnect hardware. On the other hand, the MPI library characteristics provide a measure of the communication overhead added by the corresponding software support for a particular interconnect in the MPI library.

3.1 Raw Latency and Bandwidth

The raw network communication performance is usually evaluated with the aid of simple benchmarks which measure the latency and bandwidth of communication of different message sizes on a particular network. For the Myrinet network the communication performance is gathered from the *gm_allsize* benchmark, which is distributed as a part of the GM software distribution. This benchmark utilizes the GM communication API directly to send and receive data over the Myrinet network, and can be used to measure both communication latency and bandwidth. For Ethernet, a kernel-level benchmark is used for this purpose. This benchmark uses UDP datagrams for communication with zero-copy send and receive semantics. The use of the low overhead UDP protocol over IP ensures that the benchmark measures primarily raw network performance, rather than a combination of network and TCP/IP protocol stack performance. This ensures a fairer comparison between Myrinet and Ethernet networks. Latency is measured as the half of the average ping-pong latency for a particular message size. Bandwidth is measured by the total number of bytes transferred between two nodes divided by the total time taken for this transfer. Further details on the kernel-level benchmark used to measure Ethernet latency and bandwidth can be found in [13] and [14].

Figure 1 shows the raw network latency comparison between Ethernet and Myrinet across a range of message sizes. The topmost line corresponds to the latency of Gigabit Ethernet using Intel Pro/1000 MT Server Gigabit network interface cards (NIC) on both the receiver and the sender. The middle line shows the same data, but in this case all interrupt coalescing in the NIC driver is turned off. These two lines show that interrupt coalescing causes a significant increase in the latency of Ethernet communication, by as much as 30 μsec for 1 byte messages. Finally, the lowest line gives the Myrinet communication latency between two LANai7.2 Myrinet NIC equipped nodes. The figure shows that Myrinet has a significantly lower latency than Gigabit Ethernet without interrupt coalescing. Thus, Myrinet has around 40 μsec lower latency than Gigabit Ethernet, and around 10 μsec lower latency than Gigabit Ethernet without interrupt coalescing, in the range of message sizes shown.

Figure 2 shows the raw network unidirectional bandwidth comparison between Ethernet and Myrinet across a range of message sizes. The figure shows that the bandwidth obtained on Gigabit Ethernet with interrupt coalescing is remarkably similar in overall profile to that without interrupt coalescing. However, the obtained unidirectional bandwidth is slightly higher when the NIC driver does not coalesce any interrupts. For Gigabit Ethernet without interrupt coalescing, the unidirectional bandwidth saturates around 960 Mbps. When interrupts are coalesced, the bandwidth saturates about 30 Mbps lower. On the other hand, Myrinet bandwidth is substantially lower than Gigabit Ethernet for the same message size up to 1500-byte messages. This result is surprising since the version of Myrinet used here has

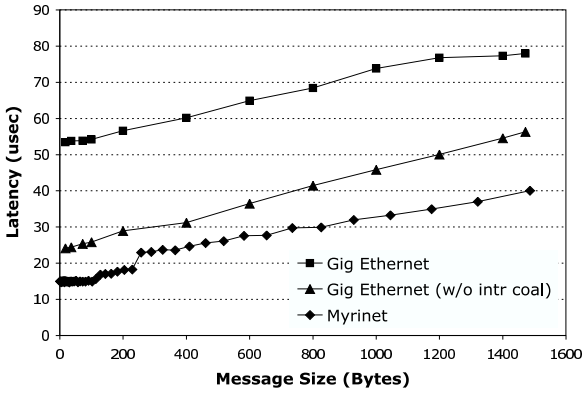

Figure 1: Comparison of raw network ping latencies for Myrinet and Ethernet

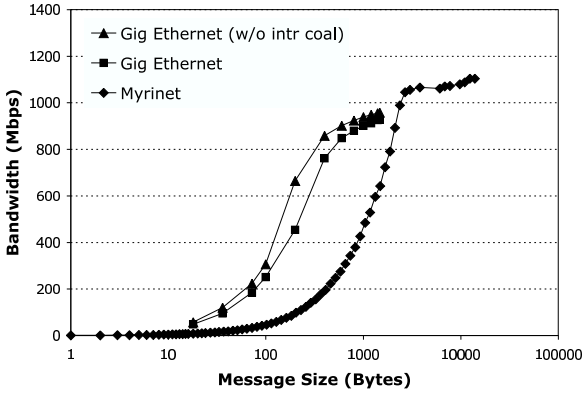

Figure 2: Comparison of raw unidirectional message bandwidths for Myrinet and Ethernet

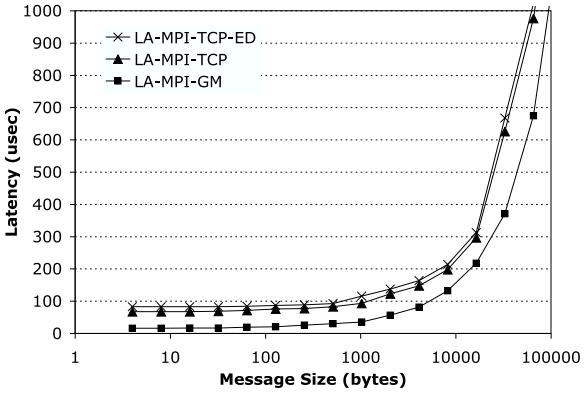

Figure 3: Comparison of message ping latencies for Myrinet and TCP over Ethernet

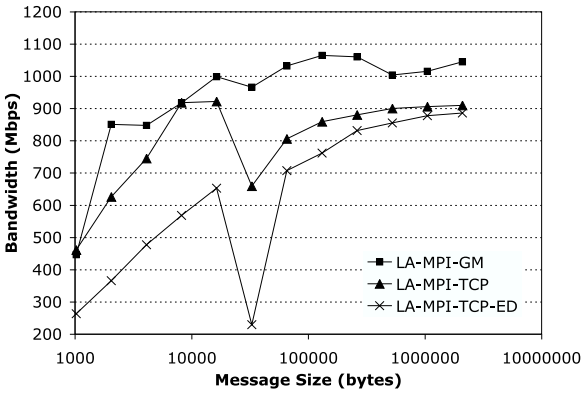

Figure 4: Comparison of unidirectional message bandwidths for Myrinet and TCP over Ethernet

a higher peak bandwidth (1.2 Gbps) than Gigabit Ethernet (1 Gbps). As a result, Myrinet bandwidth saturates at a higher 1100 Mbps, but at much larger message sizes than Ethernet.

3.2 MPI Latency and Bandwidth

While the raw latency and bandwidth of the network is important, the MPI library can add additional overhead. Simple latency and bandwidth microbenchmarks are typically used to assess the overall messaging performance of an MPI library, as these are perceived to be the characteristics most indicative of MPI application performance. Messaging latency can be measured by a simple microbenchmark with two communicating nodes. The first node simply sends a message to the second node, which returns the message back to the first node. The message latency is then half of the total time elapsed in this two-way message transfer. In this microbenchmark, the default blocking versions of send and receive are used for all MPI communication. Messaging bandwidth can also be measured by another simple microbenchmark with two communicating nodes. The first node repeatedly sends a fixed-size message to the second node as fast as it can. The second node simply receives these messages. The bandwidth is then the total number of bytes sent divided by the time elapsed to transfer all of the messages. Again, the default blocking versions of send and receive are used for all MPI communication.

Figure 3 shows message latency as a function of message size for LA-MPI using both GM over Myrinet and TCP over Ethernet as a communication medium. This plot shows that the ping latency of messages over Myrinet is substantially lower than that over Ethernet. LA-MPI using TCP, LA-MPI-TCP, consistently has a significantly higher latency; about 50 μsec higher than LA-MPI-GM for 4 B messages and increasing steadily to about 420 μsec higher than LA-MPI-GM for 64 KB messages. When TCP is used in an event-driven fashion, LA-MPI-TCP-ED, there is a further latency increase of 15 μsec above LA-MPI-TCP. This latency increase is the result of thread switching overhead between the main thread and the event thread within the library.

All of the library versions use the *eager* message transfer protocol for messages up to 16 KB, and then switch to the *rendezvous* protocol for larger messages. This results in as much as a two-fold increase in message latency for 32 KB messages compared to 16 KB messages, as shown in Figure 3. In the eager protocol, messages are sent immediately to the receiver, requiring the receiver to be able to receive these messages whether they are expected or not. In the rendezvous protocol, the sender only sends a small fragment of

the message at first, which serves as a *request-to-send* (RTS) message. The receiver must then respond with a *clear-to-send* (CTS) message before the sender can send the rest of the message. Typically, the receiver does not send the CTS until the corresponding receive gets posted by the application. This minimizes the required buffering on the receiver node, and prevents long messages from delaying other messages for which the receiver might be specifically waiting.

Figure 4 shows unidirectional messaging bandwidth as a function of message size for LA-MPI using both GM over Myrinet and TCP over Ethernet as a communication medium. The figure shows that the achieved messaging bandwidth for 1 KB messages is low for all of the library versions. For such small messages, the per-message overhead of both the MPI library and the communication substrate limit the achievable performance. As the message size increases beyond 1 KB, however, Myrinet enables the messaging bandwidth to increase faster than TCP over Ethernet. For *eager* messages ($\leq$ 16 KB), increasing message size increases achievable bandwidth until the network saturates: the TCP messaging bandwidth saturates at around 930 Mbps, whereas the higher theoretical peak bandwidth of Myrinet enables it to attain up to 1100 Mbps. There is a sharp drop in messaging bandwidth in all cases as the message size increases from 16 KB to 32 KB, because of the switch from the *eager* protocol to the *rendezvous* protocol. For messages larger than 32 KB, all library versions again show a consistent increase in bandwidth with message size. For *rendezvous* messages, LA-MPI-GM achieves a peak unidirectional bandwidth of about 1050 Mbps (with 2 MB messages). With the same message size, the TCP versions of LA-MPI, achieve slightly lower maximum bandwidths of around 900 Mbps, with LA-MPI-TCP-ED about 20 Mbps lower than LA-MPI-TCP. LA-MPI-TCP-ED consistently sustains less bandwidth than LA-MPI-TCP because of thread switching overhead, and because the increased responsiveness of LA-MPI-TCP-ED leads to 100% unexpected receives on this simple benchmark. However, as the figure shows, the difference in performance between the two consistently decreases with increasing message size.

4. NAS BENCHMARK PERFORMANCE

The simple microbenchmarks of the previous section show that Myrinet consistently enables lower latency and higher bandwidth MPI messaging. However, these microbenchmarks do not necessarily translate into overall application performance since they simply measure communication performance in isolation. In any real MPI application, communication occurs in parallel with computation, and most applications are written using non-blocking library calls to maximize the overlap between communication and computation.

The NAS parallel benchmarks (NPB) are a more realistic set of benchmarks that include both computation and communication [1]. NPB is a set of benchmarks, comprised of both application kernels and simulated computation fluid dynamics (CFD) applications. This paper uses five benchmarks from the NPB version 2.2 suite—BT, SP, LU, IS and MG—to provide a thorough comparison of the various LA-MPI versions using both Ethernet and Myrinet networks. Among the benchmarks, the first three are simulated CFD applications and the latter two are smaller application kernels. The NPB version 2.2 suite supports up to three precompiled data-sets, A, B, and C (in increasing order of size), for each benchmark.

The NAS benchmarks are run on a 4, 8, or 9 node cluster depending on whether the benchmark requires a squared or power-of-2 number of nodes. The 4-node experiments are run with the mid-size data-set (B) and the 8-node (or 9-node) experiments are run with both the B data-set and the larger C data-set. Each workstation node of the cluster has one AMD Athlon XP 2800+ processor, 1GB DDR SDRAM, and a 64bit/66Mhz PCI bus. Each of the nodes also has at least 40GB of hard drive capacity (none of the benchmarks are disk intensive). Each workstation also has one Intel Pro/1000 MT Server Gigabit Ethernet network adapter and a Myrinet LANai 7.2 network adapter. The systems are connected to each other using one 24-port Netgear Gigabit Ethernet switch and a 16-port Myrinet switch. Each node runs an unmodified FreeBSD-4.7 operating system with various socket and networking parameters tuned in order to provide maximum network performance. LA-MPI-TCP-ED uses the default POSIX thread library implementation in FreeBSD (`libc_r`).

Figure 5 compares the performance of different execution configurations of the five NAS benchmarks, achieved with LA-MPI-TCP, LA-MPI-TCP-ED, and LA-MPI-GM. Each set of three bars of the graph corresponds to one particular configuration of the benchmark and is named accordingly—the first part of name gives the benchmark name, the middle part conveys the data-set size used for that particular experiment, and the last part conveys the number of participating MPI nodes for the experiment. Each bar of the graph depicts the normalized execution time of the corresponding benchmark using a particular library version relative to LA-MPI-GM. Thus, a bar of length less than 1.0 indicates a performance gain over the LA-MPI-GM library. Each data point for the graph is generated by averaging the execution times over 10 runs of each experiment.

Figure 5 shows that LA-MPI using TCP over Ethernet (LA-MPI-TCP) consistently performs worse than LA-MPI using GM over Myrinet (LA-MPI-GM) on all of the 5 NAS benchmarks. Overall, across the 15 benchmark configurations, LA-MPI-TCP is 5.2% slower than LA-MPI-GM on average. However, when TCP is used in a more efficient manner (LA-MPI-TCP-ED), it reduces the overall performance advantage of LA-MPI-GM across the 15 benchmark configurations to just over 0.3% on average. More interestingly however, LA-MPI-TCP-ED is able to match or beat LA-MPI-GM on 6 out of the 15 benchmark configurations, with a peak speedup of 7% on IS.C.8. Also, moving to a bigger cluster, or a larger data-set improves LA-MPI-TCP-ED with respect to LA-MPI-GM. Thus, as the communication component of a MPI application increases, a properly designed application is able to extract greater benefits from the increased concurrency between communication and computation, that the event-driven library provides. To ensure that the results are not biased towards any particular Myrinet implementation of the MPI library, these same experiments were also run with the Myrinet port of the MPICH MPI library (MPICH-GM) [8]. On average, the performance of

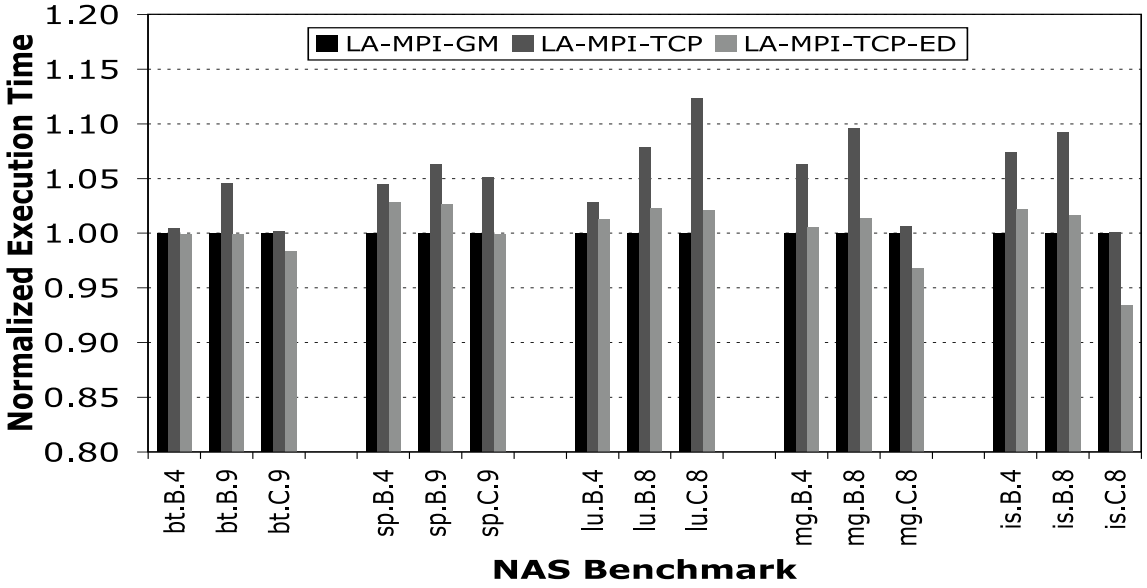

Figure 5: Execution times of NAS benchmarks for the different MPI library versions normalized to LA-MPI-GM (bars shorter than 1 represent a speedup over LA-MPI-GM).

these benchmarks with MPICH-GM differs from that with LA-MPI-GM by only 0.25%.

These results are especially interesting because they show that the LA-MPI-TCP-ED library using TCP messaging is able to match or outperform the Myrinet library version on several of the benchmark configurations, in spite of the significant latency and bandwidth advantage that Myrinet has over Gigabit Ethernet. LA-MPI-TCP-ED is able to outperform the other libraries because it is able to effectively overlap communication and computation. Longer messages using non-blocking MPI communication provide greater scope for an effective overlap, and thus the applications which use more of these messages show performance improvements with LA-MPI-TCP-ED.

5. RELATED WORK

One of the prime facilitators of high-performance message passing has been the use of specialized interconnect technologies, such as Quadrics and Myrinet [3, 15]. As explained in this paper, Myrinet is a high-performance, packet-communication, and switching technology that has gained considerable acceptance as a cluster interconnect [3]. It uses Myricom LANai programmable network interfaces, and user-level communication protocols to provide a low-latency high-bandwidth communication medium. Similarly, the Quadrics interconnect technology uses programmable network interfaces called Elan, along with special-purpose network switches and links to provide a high-performance messaging substrate on clusters [15]. The Quadrics network achieves high-performance by creating a global virtual-address space through hardware support to encompass the individual nodes' address spaces. User-level communication protocols and messaging libraries are implemented on the network interface itself, and enable the Quadrics network to provide even higher bandwidths and lower message latencies than Myrinet.

A subset of research efforts in high-performance message passing has concentrated on developing custom alternatives to the TCP/IP protocol stack for communication over Ethernet. Shivam et al. proposed a message layer called Ethernet Message Passing for enabling high-performance MPI [17]. EMP implements custom messaging protocols directly in the network interface, and bypasses the entire protocol stack of the operating system. EMP supports reliable messaging and provides zero-copy I/O to user MPI applications for pre-posted receives. Experimental results show that EMP provides significantly lower message latency than TCP over Ethernet. In a later work, Shivam et al. also extended EMP to parallelize the receive processing to take advantage of a multiple CPU network interface and, showed further gains in MPI performance [18].

TCP splintering is another research work aimed at improving the performance of TCP messaging over Ethernet [6]. This work focuses on the limitations of TCP's congestion-control and flow-control policies in a cluster environment, and proposes offloading parts of the protocol stack to the network interface to provide improved MPI performance. Specifically, it targets congestion control and acknowledgment generation for offloading to the network interface in order to reduce the CPU overhead of message-passing, and improve message latency.

6. CONCLUSIONS

This paper shows that while specialized networks may have lower latency and higher bandwidth than commodity Ethernet when measured in isolation, this does not necessarily translate into better application performance. As the NAS benchmarks show, the library's ability to allow overlapping communication and computation is equally important as raw latency and bandwidth. With comparable networking technologies, 1 Gbps Ethernet and 1.2 Gbps Myrinet, several (6 out of 15 configurations) of the NAS benchmarks

run faster (by as much as 7%) when the MPI library uses TCP over Ethernet for communication rather than GM over Myrinet. More importantly, the efficient use of TCP communication by the LA-MPI library reduces the performance advantage of GM over Ethernet from above 5% to around 0.3%. While the most recent Myrinet networks provide higher bandwidth, multiple Gigabit Ethernet links should provide competitive performance. Furthermore, the pending arrival of 10 Gbps Ethernet should extend Ethernet's performance advantage.

In addition, there are several other optimizations that could further improve the performance of TCP over Ethernet as a communication substrate for MPI messaging. The latency gap between Ethernet and Myrinet is largely due to the memory copies required to move data between the application and the kernel, and the overhead of interrupt processing. The user-level communication protocols employed by specialized networks, including Myrinet and Quadrics, avoid these copies by directly transferring data between the network interface and application buffers, resulting in lower communication latencies. However, these same techniques can be integrated into commodity protocols like TCP. For example, recent work looks to offload all or part of TCP processing onto an Ethernet network interface to reduce the processing load on the host processor, and reducing acknowledgment-related interrupts as a side-effect [2, 9]. Offloading TCP segmentation tasks can also result in significant performance benefits [11, 12]. The use of zero-copy sockets can improve TCP performance further by avoiding extraneous copies for message sends and posted message receives [4, 5, 16]. Since these techniques utilize the ubiquitous TCP/IP networking support present in all modern day operating systems, they can be easily deployed on a wide range of hardware and software platforms. Furthermore, techniques such as the event-driven LA-MPI library merely harness the capabilities of the TCP/IP protocol stack more efficiently, enabling complete independence of these techniques to the underlying TCP protocol stack implementation. The use of copy avoidance, interrupt avoidance, and TCP offload together with an efficient TCP path in the MPI library can make TCP a low-cost, portable, and reliable alternative to specialized networks. Moreover, these TCP enhancements would simultaneously benefit all networking applications that rely on TCP for communication.

7. ACKNOWLEDGMENTS

This work is supported in part by a donation from AMD and by the Department of Energy under Contract Nos. 03891-001-99-4G, 74837-001-03 49 and/or 86192-001-04 49 from Los Alamos National Laboratory.

8. REFERENCES

[1] D. H. Bailey, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinki, R. S. Schreiber, H. D. Simon, V. Venkatakrishnam, and S. K. Weeratunga. The NAS Parallel Benchmarks. *International Journal of Supercomputer Applications*, 5(3):63–73, 1991.

[2] H. Bilic, Y. Birk, I. Chirashnya, and Z. Machulsky. Deferred Segmentation for Wire-Speed Transmission of Large TCP Frames over Standard GbE Networks. In *Hot Interconnects IX*, pages 81–85, Aug. 2001.

[3] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. Su. Myrinet: A Gigabit-per-second Local Area Network. *IEEE MICRO*, 15(1):29–36, February 1995.

[4] J. S. Chase, A. J. Gallatin, and K. G. Yocum. End-system Optimizations for High-Speed TCP. *IEEE Communications, Special Issue on High-Speed TCP*, 39(4):68–74, April 2001.

[5] A. Gallatin, J. Chase, and K. Yocum. Trapeze/IP: TCP/IP at near-gigabit speeds. In *Proceedings of 1999 USENIX Technical Conference*, pages 109–120, June 1999.

[6] P. Gilfeather and A. B. Macabe. Making TCP Viable as a High Performance Computing Protocol. In *Proceedings of the Third LACSI Symposium*, Oct 2002.

[7] R. L. Graham, S.-E. Choi, D. J. Daniel, N. N. Desai, R. G. Minnich, C. E. Rasmussen, L. D. Risinger, and M. W. Sukalski. A Network-Failure-Tolerant Message-Passing System for Terascacle Clusters. In *Proceedings of the 16th Annual ACM International Conference on Supercomputing*, June 2002.

[8] 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, Sept. 1996.

[9] Y. Hoskote, B. A. Bloechel, G. E. Dermer, V. Erraguntla, D. Finan, J. Howard, D. Klowden, S. G. Narendra, G. Ruhl, J. W. Tschanz, S. Vangal, V. Veeramachaneni, H. Wilson, J. Xu, and N. Borkar. A TCP Offload Accelerator for 10 Gb/s Ethernet in 90-nm CMOS. *IEEE Journal of Solid-State Circuits*, 38(11):1866–1875, Nov. 2003.

[10] S. Majumder, S. Rixner, and V. S. Pai. An Event-driven Architecture for MPI Libraries. In *Proceedings of the Los ALamos Computer Science Institute Symposium (LACSI'04)*, October 2004.

[11] S. Makineni and R. Iyer. Architectural characterization of TCP/IP packet processing on the Pentium M microprocessor. In *International Symposium on High-Performance Computer Architecture*, pages 152–162, Feb. 2004.

[12] D. Minturn, G. Regnier, J. Krueger, R. Iyer, and S. Makineni. Addressing TCP/IP processing challenges using the IA and IXP processors. *Intel Technology Journal*, 7(4), 2003.

[13] V. S. Pai, S. Rixner, and H.-Y. Kim. Isolating the Performance Ipacts of Network Interface Cards through Microbenchmarks. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2004.

[14] V. S. Pai, S. Rixner, and H.-Y. Kim. Isolating the Performance Ipacts of Network Interface Cards through Microbenchmarks. Technical Report ee0401, Rice University, April 2004.

[15] F. Petrini, W. chun Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The QUADRICS Network: High-Performance Clustering Technology. *IEEE MICRO*, Jan 2002.

[16] S. H. Rodrigues, T. E. Anderson, and D. E. Culler. High-Performance Local Area Communication With Fast Sockets. In *Proceedings of the 1997 USENIX Technical Conference*, pages 257–274, Jan. 1997.

[17] P. Shivam, P. Wyckoff, and D. Panda. EMP: Zero-copy OS-bypass NIC-driven Gigabit Ethernet Message Passing. In *Proceedings of SC2001*, Nov 2001.

[18] P. Shivam, P. Wyckoff, and D. Panda. Can User Level Protocols Take Advantage of Multi-CPU NICs? In *Proceedings of IPDPS2002*, Apr 2002.