Open MPI's TEG Point-to-Point Communications Methodology: Comparison to Existing Implementations

T.S. Woodall[1], R.L. Graham[1], R.H. Castain[1], D.J. Daniel[1], M.W. Sukalski[2], G.E. Fagg[3], E. Gabriel[3], G. Bosilca[3], T. Angskun[3], J.J. Dongarra[3], J.M. Squyres[4], V. Sahay[4], P. Kambadur[4], B. Barrett[4], A. Lumsdaine[4]

[1] Los Alamos National Lab
{twoodall, rlgraham, rhc, ddd}@lanl.gov

[2] Sandia National Laboratories,
mwsukal@ca.sandia.gov

[3] University of Tennessee,
{fagg, egabriel, bosilca, anskun, dongarra}@cs.utk.edu

[4] Indiana University
{jsquyres, vsahay, pkambadu, brbarret, lums}@osl.iu.edu

Abstract. TEG is a new methodology for point-to-point messaging developed as a part of the Open MPI project. Initial performance measurements are presented, showing comparable ping-pong latencies in a single NIC configuration, but with bandwidths up to 30% higher than that achieved by other leading MPI implementations. Homogeneous dual-NIC configurations further improved performance, but the heterogeneous case requires continued investigation.

1 Introduction

Petascale computing is of increasing importance to the scientific community. Concurrently, the availability of small size clusters (on the order of tens to hundreds of CPUs) also continues to increase. Developing a message-passing system that efficiently deals with the challenges of performance and fault tolerance across this broad range represents a considerable challenge facing MPI developers.

The Open MPI project [3] —an ongoing collaboration between the Resilient Technologies Team at Los Alamos National Lab, the Open Systems Laboratory at Indiana University, and the Innovative Computing Laboratory at the University of Tennessee— is a new open-source implementation of the Message Passing Interface (MPI) standard for parallel programming on large-scale distributed systems [4,6] focused on addressing these problems. This paper presents initial results from Open MPI's new point-to-point communication methodology (code-named "TEG"[9]) that provides high-performance, fault tolerant message passing. A full description of the design is given in the accompanying paper [9].

Open MPI/TEG's provides an enhanced feature set with support for dropped packets, corrupt packets, and NIC failures; concurrent network types (e.g. Myrinet, Infini-

Band, etc.), in a single application run; single message fragmentation and delivery utilizing multiple NIC, including different NIC types, such as Myrinet and InfiniBand; and heterogeneous platform support within a single job, including different OS types, different addressing modes (32 vs 64 bit mode), and different endianess. All of these features have been adapted into a new modular component architecture that allows for both compile-time and runtime selection of options.

2 Results

The performance of Open MPI's TEG methodology was compared against that of several well-known existing MPI implementations: FT-MPI v1.0.2[2], LA-MPI v1.5.1[5], LAM/MPI v7.0.4[1], and MPICH2 v0.96p2[7]. Experiments were performed using a two processor system based on 2.0GHz Xeon processors sharing 512kB of cache and 2GB of RAM. The system utilized a 64-bit, 100MHz, PCI-X bus with two Intel Pro/1000 NICs (based on the Super P4Dp6, Intel E7500 chipset), and one Myricom PCI64C NIC running LANai 9.2 on a 66MHz PCI interface. A second PCI bus (64-bit, 133MHz PCI-X) hosted a second, identical Myricom NIC. The processors were running the Red Hat 9.0 Linux operating system based on the 2.4.20-6smp kernel. All measurements were made using TCP/IP running over both Gigabit Ethernet and Myrinet.

Data was collected for two critical parameters: latency, using a ping-pong test code (for both MPI blocking and non-blocking semantics) and measuring half round-trip time of zero byte messages; and single NIC bandwidth results, using NetPIPE v3.6 [8].

2.1 Latency

Latency was measured using a ping-pong test code for both blocking and non-blocking MPI semantics. Blocking semantics allow for special optimizations, so more mature implementations will often create specially optimized versions to take advantage of these capabilities. This optimization has not yet been performed for TEG.

Table 1 compares TEG's latencies with those of LAM/MPI 7, LA-MPI, FT-MPI, and MPICH2 for the non-blocking scenario. The latencies for all MPI implementations with TCP/IP over Myrinet are similar, at about 51 μs. Even when TEG makes asynchronous progress with a progress thread, thus greatly reducing the CPU cycles used, latency is basically unchanged.

When the tests are run using Gigabit Ethernet, however, the measured latencies are generally lower and show a greater range of values. TEG's polling mode and LAM/MPI 7 have the lowest latencies at just below 40 μs, or about 25% lower than was obtained over Myrinet. MPICH2 has slightly higher latency, followed by LA-MPI and FT-MPI. The asynchronous progress mode of Open MPI's TEG is much higher than the other entries, with values slightly lower than those obtained with TCP/IP over Myrinet.

Table 2 lists the latency data for the ping-pong results using blocking MPI semantics. In this case, TEG's latencies are not as good as was obtained when using non-blocking semantics, with LAM/MPI 7 and MPICH2 is outperforming TEG better, while LA-MPI and FT-MPI didn't perform as well as TEG. Again, TEG's asynchronous progress mode gives much higher latencies than the polling based approach.

Implementation	Myrinet Latency (μs)	GigE Latency (μs)
Open MPI/TEG (Polling)	51.5	39.7
Open MPI/TEG (Async)	51.2	49.9
LAM7	51.5	39.9
LA-MPI	51.6	42.9
FT-MPI	51.4	46.4
MPICH2	51.5	40.3

Table 1. Open MPI/TEG latency measurements compared to other MPI implementations (non-blocking MPI semantics—i.e., MPI_ISEND / MPI_IRECV).

Implementation	GigE Latency (μs)
Open MPI/TEG (Polling)	41.3
Open MPI/TEG (Async)	52.6
LAM7	36.0
MPICH2	39.0
LA-MPI	42.8
FT-MPI	46.8

Table 2. Open MPI/TEG latency measurements compared to other MPI implementations (blocking MPI semantics—i.e., MPI_SEND, MPI_RECV).

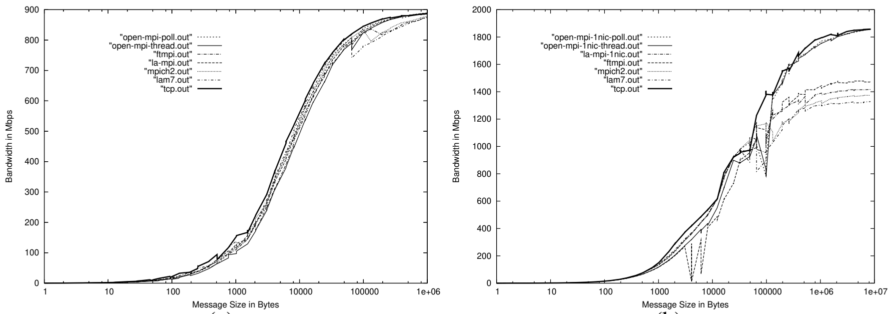
Overall, the initial results obtained with Open MPI/TEG show good latency performance characteristics when compared with more mature implementations. This performance is obtained despite the overhead required to: (a) stripe a single message across any number of different network types; (b) provide thread safety (MPI_THREAD_MULTIPLE); and (c) allow proper handling of various failure scenarios. The latency of the blocking calls is not as good as some of the other current implementations. This reflects the relative immaturity of the TEG module and the current lack of optimization.

2.2 Bandwidth

Open MPI/TEG's ability to run in a heterogeneous networking environment makes the study of bandwidth performance very interesting. To establish this condition, a single PTL implementation (TCP/IP) was used for this study and operated on networks with different physical characteristics for the underlying physical transport layer (Myrinet and Gigabit Ethernet). Bandwidths were measured using NetPIPE v3.6 [8] and compared to that obtained from the same set of MPI implementations used in the latency measurements.

Figure 1(a) shows the performance of TEG compared to that from other MPI implementations over Gigabit Ethernet. As the figure shows, TEG, MPICH2, FT-MPI, LA-MPI, and LAM/MPI all have similar performance characteristics, peaking out close to 900 Mb/sec, similar to that obtained from raw TCP. MPICH2, FT-MPI, and LAM/MPI exhibit some irregular behavior with message sizes around 100 Kbyte, but this smoothes out at higher bandwidths and is most likely an artifact of the rendezvous protocol.

However, as Figure 1(b) shows, when running the same tests on a higher bandwidth interconnect (Myrinet), TEG displays much better bandwidths than MPICH2, FT-MPI, LA-MPI, and LAM/MPI. TEG's performance closely tracks the raw TCP/IP benchmark results, except at just above a message size of 100 Kbyte, where the rendezvous protocol causes a temporary drop in measured bandwidth. Both TEG and raw TCP/IP peak out at a little above 1800 Kb/sec. In contrast, MPICH2, FT-MPI, LA-MPI, and LAM/MPI all peak out about 30% lower around 1400 Kb/sec. This is due to TEG's reduced overheads associated with multiple packet messages, including generation of the rendezvous protocol's ACK as soon as a match has been made on the receive side, and embedding a pointer to the receive object in the ACK so that subsequent packets can include this in their header for fast delivery on the receive side.



(a) (b)

Fig. 1. Open MPI/TEG point-to-point ping-pong bandwidth compared to with the MPICH2, FT-MPI, LA-MPI, and LAM/MPI implementations, and raw TCP/IP. (a) Gigabit Ethernet (b) Myrinet.

Figure 2 shows the results of running the same bandwidth tests over two Myrinet NICS, with TEG fragmenting and reassembling Open MPI's messages. The single NIC data is included for reference. As one would expect, the advantages of using multiple NIC's are not apparent in this particular test, until the message is large enough to be fragmented. At about an 8Mbyte message size, the dual NIC overall bandwidth is about 30% higher than the single NIC bandwidth. The dual NIC data does not appear to have peaked at the 8Mbyte maximum message size used in the experiment. However, we do not expect the dual NIC configuration to achieve double the rate of the single NIC data, since a single process is handling the TCP/IP stack. Parallelism, therefore, is only obtained over the network and by overlapping the send with the receive in the multiple fragment case.

The dual NIC implementation in Open MPI/TEG is about 10% more efficient than that in LA-MPI. Similar to the single NIC case, TEG benefits from the early generation of the rendezvous protocol ACK and embedding the receive object pointer information in the ACK message.

Figures 3 and 4 show the results of striping a single message over both Gigabit Ethernet and Myrinet using TEG. The single NIC Gigabit Ethernet and Myrinet are also

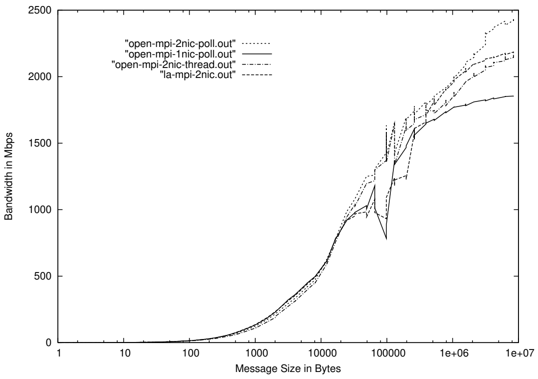


Fig. 2. Open MPI/TEG single and dual NIC point-to-point ping-pong bandwidth over Myrinet. Comparison with LA-MPI.

included for reference. The results alway fall between the single NIC Gigabit Ethernet at the low end, and the single NIC Myrinet data at the upper end. This appears to indicate that adding a Gigabit Ethernet NIC to a system with a single Myrinet NIC can result in degraded performance. In contrast, adding a Myrinet NIC to a system with a single Gigabit Ethernet NIC appears to improve performance. Future work will investigate possible explanations for the observed behavior.

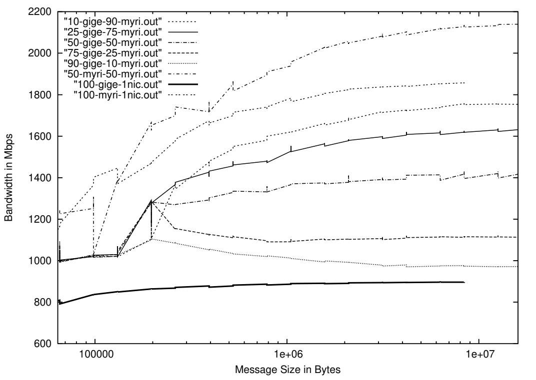


Fig. 3. Message striping across GigE and Myrinet with round-robin scheduling of first fragments across both interfaces and weighted scheduling of remaining data

Finally, Figure 5 shows the effect of varying the size of the first message fragment on the bandwidth profile of Open MPI/TEG. As one would expect, the asymptotic behavior is independent of the size of the first message fragment. First fragments smaller than 128 KBytes don't require enough processing at the destination to hide the latency of

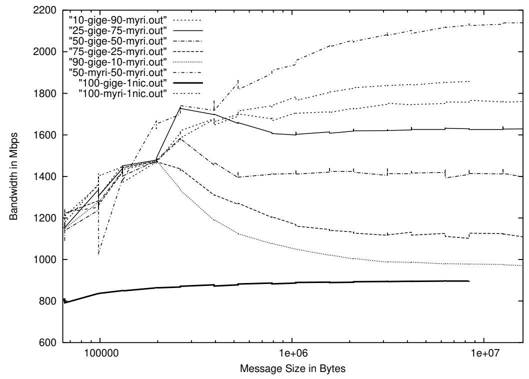
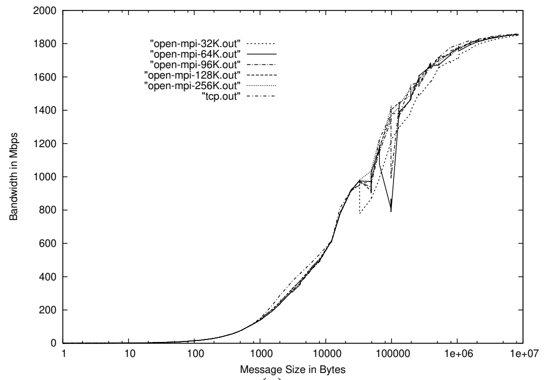


Fig. 4. Message striping across GigE and Myrinet with first fragments delivered via Myrinet and weighted scheduling of remaining data

the rendezvous protocol. The early transmission of the rendezvous ACK allows for this overlap to take place.



(a)

Fig. 5. Comparison of bandwidth as a function of first message fragment size for Open MPI/TEG

3 Summary

We have presented the results of running latency and bandwidth test with Open MPI with TCP/IP over Gigabit Ethernet and Myrinet. These results are compared with those obtained running LA-MPI, LAM/MPI, FT-MPI, and MPICH2, and show leading latency results when using non-blocking MPI semantics, and middle of the pack results with still to be optimized blocking MPI semantics. Open MPI's single bandwidths are better than those obtained with these same MPI's.

In addition multi-NIC bandwidth data is obtained with Open MPI. These bandwidths are about 30% better than the single NIC data over Myrinet, and about 10% better than those obtained by LA-MPI. Using Open MPI to send part of a message over networks with different characteristics (Gigabit Ethernet and Myrinet, in this case) needs further study at this early stage of Open MPI development.

4 Acknowledgments

This work was supported by a grant from the Lilly Endowment, National Science Foundation grants 0116050, EIA-0202048, EIA-9972889, and ANI-0330620, and Department of Energy Contract DE-FG02-02ER25536. Los Alamos National Laboratory is operated by the University of California for the National Nuclear Security Administration of the United States Department of Energy under contract W-7405-ENG-36. Project support was provided through ASCI/PSE and the Los Alamos Computer Science Institute, and the Center for Information Technology Research (CITR) of the University of Tennessee.

References

1. G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.
2. Graham E. Fagg, Edgar Gabriel, Zizhong Chen, Thara Angskun, George Bosilca, Antonin Bukovski, and Jack J. Dongarra. Fault tolerant communication library and applications for high perofrmance. In *Los Alamos Computer Science Institute Symposium*, Santa Fee, NM, October 27-29 2003.
3. E. Garbriel, G.E. Fagg, G. Bosilica, T. Angskun, J. J. Dongarra J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R.H. Castain, D.J. Daniel, R.L. Graham, and T.S. Woodall. Open mpi: Goals, concept, and design of a next generation mpi implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, 2004.
4. A. Geist, W. Gropp, S. Huss-Lederman, A. Lumsdaine, E. Lusk, W. Saphir, T. Skjellum, and M. Snir. MPI-2: Extending the Message-Passing Interface. In *Euro-Par '96 Parallel Processing*, pages 128–135. Springer Verlag, 1996.
5. R. L. Graham, S.-E. Choi, D. J. Daniel, N. N. Desai, R. G. Minnich, C. E. Rasmussen, L. D. Risinger, and M. W. Sukalksi. A network-failure-tolerant message-passing system for terascale clusters. *International Journal of Parallel Programming*, 31(4), August 2003.
6. Message Passing Interface Forum. MPI: A Message Passing Interface. In *Proc. of Supercomputing '93*, pages 878–883. IEEE Computer Society Press, November 1993.
7. Mpich2, argonne. `http://www-unix.mcs.anl.gov/mpi/mpich2/`.
8. Q.O. Snell, A.R. Mikler, and J.L. Gustafson. NetPIPE: A Network Protocol Independent Performace Evaluator. In *IASTED International Conference on Intelligent Information Management and Systems*, June 1996.
9. T.S. Woodall, R.L. Graham, R.H. Castain, D.J. Daniel, M.W. Sukalsi, G.E. Fagg, E. Garbriel, G. Bosilica, T. Angskun, J. J. Dongarra, J.M. Squyres, V. Sahay, P. Kambadur, B. Barrett, and A. Lumsdaine. Teg: A high-performance, scalable, multi-network point-to-point communications methodology. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, 2004.