Integrating New Capabilities into NetPIPE

Dave Turner, Adam Oline, Xuehua Chen, and Troy Benjegerdes

Ames Laboratory – Iowa State University
327 Wilhelm Hall, Ames, Iowa, 50011
turner@ameslab.gov

Abstract. The performance of the communication network can greatly affect the ability of scientific applications to make efficient use of the computational power available in high-performance computing systems. Many tools exist for analyzing network performance, but most concentrate on a single layer in the communication subsystem or on one type of network hardware. NetPIPE was developed to provide a complete and consistent set of analytical tools in a flexible framework that can be applied to analyze the message-passing libraries and the native software layers that they run on. Examples are given on how NetPIPE is being enhanced to enable research in channel bonding multiple Gigabit Ethernet interfaces, to analyze InfiniBand hardware and the MPI libraries being developed for it, and to optimize memory copy routines to make SMP message-passing more efficient.

1 Introduction

Performance losses can come from many sources in the communication network of clusters, Shared-memory Multi-Processor (SMP) and Massively Parallel Processing (MPP) systems. Internal limitations of the PCI and memory buses can restrict the rate that data can be transferred into and out of a node. The network hardware itself can impose limitations, as can improper tuning of the driver and OS parameters. The message-passing layer often requires tuning to achieve optimal performance, and the choice of a particular message-passing implementation can make a large difference.

Designing, building, and using high-performance computing systems requires careful measurement and tuning of the communication system to ensure that the processing power is being efficiently used. Many tools such as Netperf [1], Iperf [2], and the variants of ttcp are commonly used to analyze TCP performance between systems. Vendors often provide their own tools that allow users to measure the performance of the native software layer. Myricom provides a simple tool for analyzing the performance of Myrinet hardware at the GM layer, and Mellanox provides a tool for measuring the InfiniBand performance at the Verbs API (VAPI) layer.

While these tools are very good, they are somewhat limited in their scope. Most test only a single message size at a time, making it difficult to fully evaluate any communication system. All are aimed at testing only one layer, making it more difficult to directly compare performance at the native and message-passing layers.

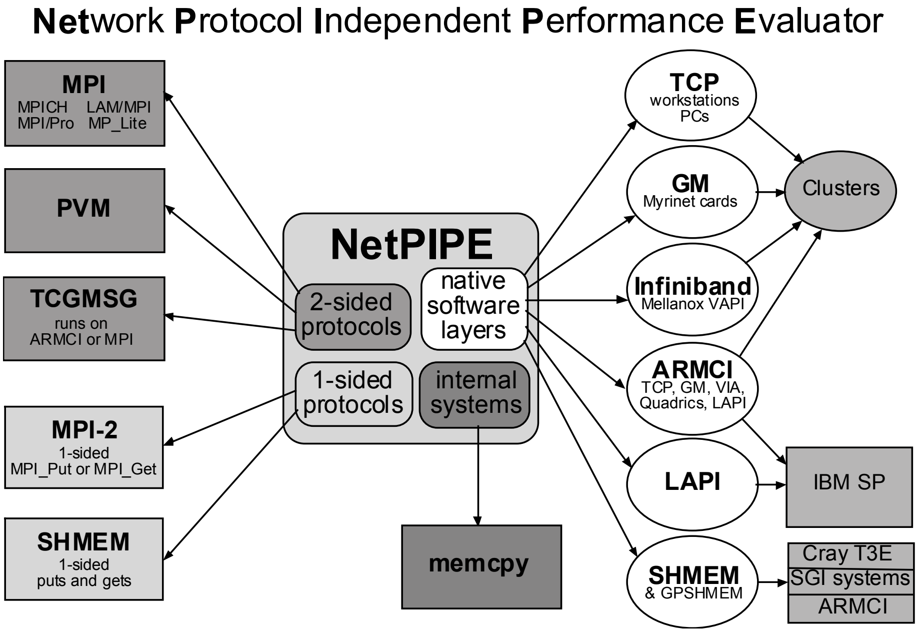

Fig. 1. A diagram showing the structure of NetPIPE and the modules developed for it.

The goal of the NetPIPE project is to provide a wide variety of features within a common framework that can be used to evaluate both message-passing libraries and the native software layers that they run on. Many new modules and features of NetPIPE will be introduced, with examples given of how they are being used to analyze cutting edge networking technology.

2 NetPIPE

NetPIPE is the **Net**work **P**rotocol **I**ndependent **P**erformance **E**valuator [3-5], a tool originally developed to provide a more complete measurement of the communication performance at both the TCP and MPI layers. A message is repeatedly bounced between nodes to provide an accurate measurement of the transmission time for each message size. Message sizes are chosen at regular intervals and at slight perturbations to more fully evaluate the communication hardware and software layers. This produces an accurate measurement of the small message latency, taken to be half the round trip time for an 8-byte message, and a graph of the throughput across a broad range of message sizes.

The authors have taken this framework, and greatly expanded the modules supported to allow measurements on more message-passing libraries and native software layers. Additional testing options have been built into the code to provide more insight into the performance bottlenecks. New modules are allowing NetPIPE to look at internal properties such as memory copy rates that affect SMP message-passing performance greatly. Current research is extending NetPIPE beyond point-to-point measurements, allowing it to address more global network performance issues.

The diagram in fig. 1 shows the current structure of NetPIPE. The code consists of one central program that provides the same testing framework for all environments. Modules have been developed to test the 2-sided communications of MPI [6-7] implementations, the PVM library [8-9], and the TCGMSG library [10]. The 1-sided *get* and *put* operations of the MPI-2 and SHMEM standards can be tested with or without the synchronization imposed by intervening *fence* calls. The native software layers that message-passing implementations are built upon can be tested using the TCP, GM, InfiniBand, ARMCI [11], LAPI, and SHMEM modules.

Having all these modules in the same framework allows for direct comparison between the various message-passing libraries. The efficiency of each message-passing library can also be measured by directly comparing its performance to that of the native software layer it runs on.

A ping-pong measurement across the full range of message sizes is ideal for identifying many deficiencies in the communication system. Latencies are typically limited by the network hardware, but may be hurt by poorly written or optimized drivers, or by the message-passing layer. Instabilities and dropouts in the performance curves may be affected by OS and network parameters such as the socket buffer sizes or the MTU size, or by problems in the message-passing layer. Limitations to the maximum throughput may be due to poor optimization at any level. The type of problem demonstrated by the performance measurement can help direct the user toward the proper solution.

NetPIPE also has a streaming mode where messages are sent in only one direction. The source node simply pushes messages over the network to the destination node in rapid succession. While the ping-pong tests apply more closely to scientific applications, the streaming mode can be useful since it puts more stress on the network. The OS can coalesce many small messages together into packets, so the transmission time for small messages should not be taken as the latency time. Streaming messages at high rates can cause the OS to adjust the window size down, which restricts any subsequent performance. The sockets must therefore be reset for each data point to prevent interference with subsequent measurements.

SMP message-passing performance depends greatly on whether the data starts in cache or main memory. Some networking hardware is also fast enough now that it may be affected by the starting location of the data. A complete understanding of the performance of these systems therefore requires testing with and without cache effects. The default configuration is to test using cache effects, where each node sends and receives the message from the same memory buffer each time. Testing without cache effects involves sending the message from a different location in main memory each time.

NetPIPE can now do an integrity check instead of measuring performance. In this mode, each message is fully tested to ensure it has not been corrupted during transmission. A bi-directional mode has been added to test the performance when messages flow in both directions at the same time. Real applications often produce message traffic in multiple directions through a network, so this provides another useful probe for the communication system. Future work will add the capability to perform multiple, synchronized, pair-wise measurements within NetPIPE. This should prove ideal for investigating global properties of networks such as the maximum throughput of the back plane in a switch.

3 Gigabit Ethernet Performance

Even though Gigabit Ethernet technology is fairly mature, care must still be taken in choosing the hardware, optimizing the driver and OS parameters, and evaluating the message-passing software. Instabilities caused by poor drivers can sometimes be overcome by optimizing the drivers themselves, choosing a better driver for that card, or simply adjusting the TCP socket buffer sizes. Limitations to the maximum throughput, typical of graphs where the performance flattens out for large messages, can often be optimized away by increasing the TCP socket buffer sizes or increasing the MTU size if supported.

The NetPIPE TCP module provides an easy way to measure the performance of a system while varying the socket buffer sizes using the –b flag. The maximum socket buffer size is often limited by the OS, but this may be a tunable parameter in itself. If the socket buffer size is found to be a limiting factor, you may be able to change the default socket buffer size in the OS and you may also need to tune the message-passing library (set P4_SOCKBUFSIZE for MPICH [12-13], for example).

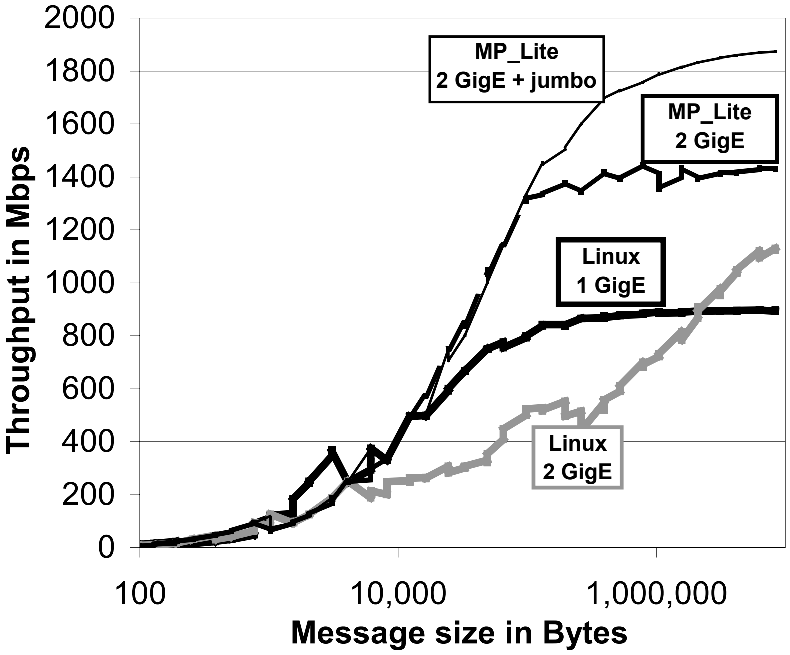

Fig. 2. The channel bonding performance using the built in Intel Pro/1000 ports on two SuperMicro X5DP8-G2 motherboards having 2.4 GHz Xeon processors running RedHat 7.3 Linux with the 2.4.18-10smp kernel.

The thick black line in fig. 2 shows the performance of the built in Intel Pro/1000 Gigabit Ethernet port between two SuperMicro X5DP8-G2 motherboards. This is good networking hardware, with a 62 µs latency and 900 Mbps throughput, but this lack of a nice smooth curve can sometimes indicate stability problems. Netgear GA302T Gigabit Ethernet cards have a lower latency at 24 µs and provide a much smoother performance curve. Both work well with the default socket buffer size, but can raise the throughput to 950 Mbps by setting the MTU size to 9000 Bytes.

Linux kernel level channel bonding across the two built in Gigabit Ethernet ports currently produces poorer performance than using just one of the ports. Using the MP_Lite message-passing library [14-16], channel bonding across the same two built in ports can be done with much greater efficiency by striping the data at the socket level. The benefit of using jumbo frames in this case is clear from the improvement from performance that flattens out at 1400 Mbps to a smoother curve that gets to nearly an ideal doubling of the single channel performance.

4 InfiniBand Research

InfiniBand [17] adapters are based on either the IBM or Mellanox chipsets. The current 4X technology is capable of operating at a maximum throughput of 10 Gbps. Many vendors such as Mellanox, DivergeNet, and JNI offer adaptors based on the Mellanox chipset, and programmed with the Mellanox VAPI or close variants.

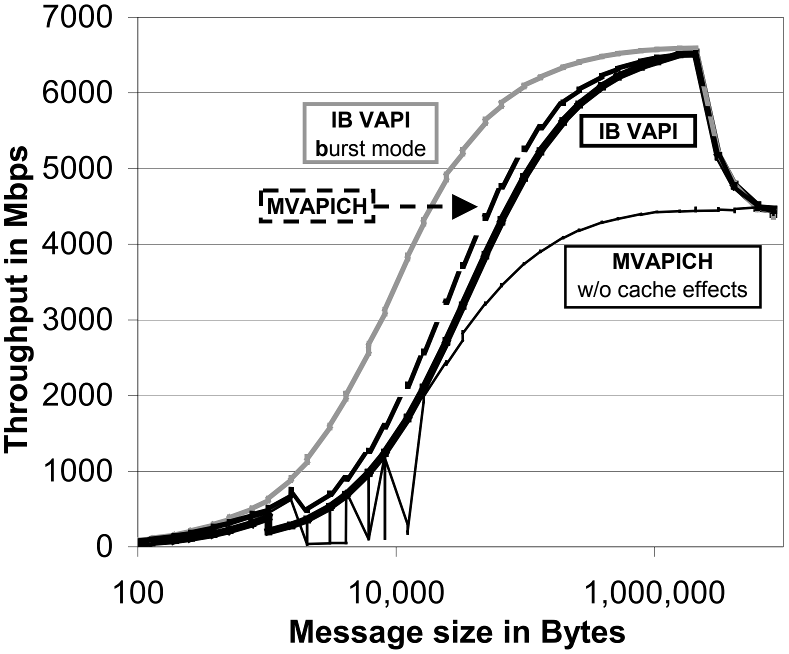

Fig. 3. The performance across Mellanox InfiniBand adapters between two 2.2 GHz Xeon systems running RedHat 7.3 Linux with the 2.4.18-10 kernel.

An InfiniBand module for NetPIPE was developed to assist in analyzing the characteristics of the hardware and Mellanox VAPI software. This also allows direct comparison with research versions of message-passing libraries such as MVAPICH 0.8 [18], an MPICH implementation for the Mellanox VAPI that grew out of an MPICH VIA module developed by the same group.

There are several unique features of InfiniBand that place new demands upon NetPIPE. The most obvious is that the transfer rates are much greater than previous technologies have delivered. Fig. 3 shows the communication performance reaching 6500 Mbps for message sizes that fit in cache, after which the performance tails off to 4400 Mbps. Running the NetPIPE tests without cache effects limits the performance to the same 4400 Mbps. This is approximately the memcpy rate for these systems. It is therefore not certain whether this is a fundamental limit of the DMA speed of the adapters in transferring data into and out of main memory, or whether the adapters are just tuned better for transfers from cache rather than main memory.

The performance tool that Mellanox distributes runs in a burst mode, where all receives are pre-posted before a trial starts, and are therefore excluded from the total communication time. While this is not representative of the performance applications would see, it is useful in determining the amount of time spent in posting a receive. The burst mode (-B) was added to NetPIPE to allow it to duplicate the measurements seen with the Mellanox tool. Fig. 3 shows that a significant amount of the communication time is spent in posting the receive, making it an attractive target for optimization. Future efforts to optimize message-passing libraries can use this information to concentrate efforts on reducing the need to pin memory buffers that have been previously used.

MVAPICH uses an RDMA mechanism to provide impressive small message latencies of around 8 μs. Pre-allocated buffers are used that avoid the need to perform memory pinning for each incoming message. However, the MVAPICH performance measured without cache effects shows severe problems between 1500 Bytes and 16 kB that will need to be addressed.

5 Memory Copy Rates

NetPIPE can also be used to probe the internal performance of a node. The memcpy module simply copies data between two memory locations within the same process rather than transferring data between 2 processes or 2 nodes. Cache effects obviously play a large role in these evaluations, as does the choice of a compiler.

The thin lines in fig. 4 show the performance of the GNU memcpy function from the 686 version of glibc using cache effects. The top of the spikes represents the good performance for transfers of data where the size is divisible by 4 Bytes. Memory transfers for sizes not divisible by 4 Bytes are handled with a byte-by-byte transfer that reduces the performance by as much as an order of magnitude. The Intel 7.1 compiler achieves better performance by simply transferring the body of data 4 bytes at a time, and handling any bytes at the end and any preceding bytes due to misalignment separately. The 386 version of glibc also uses this approach, so it is not clear why the 686 version does not.

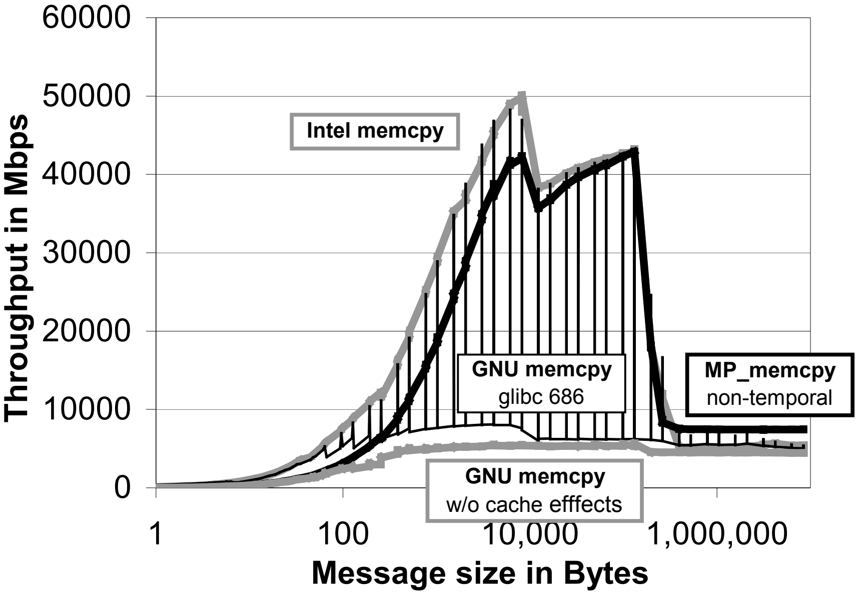

Fig. 4. The memory copy rates of the Intel compiler, the GNU compiler using the 386 and 686 versions of glibc, and an optimized routine using non-temporal memory copies.

An optimized memory copy routine is vital in SMP message-passing systems. These same spikes have been observed in performance tests on most MPI implementations. While most messages will be 4-Byte aligned, with sizes divisible by 4 Bytes, it is easy to write an optimized memory copy routine that produces optimal results for even the odd cases. The MP_memcpy curve in fig. 4 also shows that the copy rate from main memory can be improved by up to 50% by using the non-temporal copy techniques available on Pentium 4 chips.

6 Conclusions

The goal of the NetPIPE project is to provide a complete and consistent set of analytical tools within the same flexible framework to allow performance evaluations of both the message-passing libraries and the native software layers they run on. New modules have been developed to test 1-sided *get* and *put* operations of the MPI-2 and SHMEM interfaces, as well as native software layers such as GM, the Mellanox VAPI, ARMCI, and LAPI.

New capabilities have been built into the NetPIPE framework. Cache effects can now be fully investigated, and have been shown to play an important role in understanding InfiniBand and SMP message-passing performance. The streaming mode can now accurately handle the faster rates of cutting edge network hardware. Integrity tests can be used to determine if messages are being corrupted at some level in the communication system. A bi-directional mode allows testing of communications going in both directions at the same time, which more closely matches the message traffic in many applications. Adding the ability to do multiple pair-wise tests synchronized within NetPIPE will allow for a more global analysis of the capabilities of networks.

Acknowledgements

This project is funded by the DOE MICS department through the Applied Mathematical Sciences Program at Ames Laboratory. Ames Laboratory is operated for the U.S. Department of Energy by Iowa State University under Contract No. W-7405-Eng-82.

References

1. Netperf webpage: http://www.netperf.org/
2. Iperf webpage: http://dast.nlanr.net/Projects/Iperf/
3. NetPIPE webpage: http://www.scl.ameslab.gov/Projects/NetPIPE/
4. Snell, Q., Mikler, A., and Gustafson, J.: NetPIPE: A Network Protocol Independent Performance Evaluator. IASTED International Conference on Intelligent Management and Systems. (June 1996)
5. Turner, D., and Chen, X.: Protocol-Dependent Message-Passing Performance on Linux Clusters. Proceedings of the IEEE International Conference on Cluster Computing. (September 2002) 187-194
6. The MPI Standard: http://www.mcs.anl.gov/mpi/
7. MPI Forum. MPI: A Message-Passing Interface Standard. International Journal of Supercomputer Applications 8 (3/4). (1994) 165-416
8. PVM webpage: http://www.epm.ornl.gov/pvm/
9. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., and Sunderam, V.: PVM: Parallel Virtual Machine. The MIT Press (1994)
10. TCGMSG webpage: http://www.emsl.pnl.gov:2080/docs/parasoft/tcgmsg/tcgmsg.html
11. ARMCI webpage: http://www.emsl.pnl.gov:2080/docs/parasoft/armci/
12. MPICH webpage: http://www.mcs.anl.gov/mpi/mpich/
13. Gropp, W., Lusk, E., Doss, N., and Skjellum, A.: High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. Parallel Computing 22(6). (September 1996) 789-828
14. MP_Lite webpage: http://www.scl.ameslab.gov/Projects/MP_Lite/
15. Turner, D., Chen, W., and Kendall, R.: Performance of the MP_Lite Message-Passing Library on Linux Clusters. Linux Clusters: The HPC Revolution. University of Illinois, Urbana-Champaign. (June 25-27, 2001)
16. Turner, D., Selvarajan, S., Chen, X., and Chen, W.: The MP_Lite Message-Passing Library. Fourteenth IASTED International Conference on Parallel and Distributed Computing and Systems. Cambridge Massachusetts. (November 4-6, 2002) 434-439
17. InfiniBand webpage: http://www.infinibandta.org/
18. MVAPICH webpage: http://nowlab.cis.ohio-state.edu/projects/mpi-iba/