MPI and Embedded TCP/IP Gigabit Ethernet Cluster Computing

Neal Bierbaum
Sandia National Laboratories, Livermore, CA
nrbierb@sandia.gov

Abstract

A group of lower cost PC’s connected via Gigabit Ethernet and using MPI for communications between multiple parallel processes running simultaneously on all hosts provides a cost effective and powerful computing solution. The processing load for interprocess communications via TCP is significant when the parallel processes must exchange a large amount of data. When using a standard gigabit network interface card (NIC), TCP communications at near wire speed (above 800 Mbit/sec) use almost the entire processing capacity of a 1 GHz Pentium 3 processor or around 30% of a 2.4 GHz Pentium IV. This communications overhead significantly reduces the computational power of economical two processor systems. NICs that perform the protocol processing on the card offer the possibility of reducing this significantly. This study evaluates the performance and cost effectiveness of using a NIC with embedded TCP/IP processing to offload the network processing and allow more MPI processes per host.

How might a NIC with embedded TCP/IP protocol processing (hereafter referred to as an “embedded NIC”) improve the performance of the hosts within a computing cluster? Most importantly, it could eliminate the time that the host spends in the kernel processing each incoming packet and switching between kernel and user mode. It might also reduce memory load by directly moving the data to and from the user memory space to the interface without intervening copies. Finally, it might more quickly process the packets to decrease latency and increase the data transfer rate to the maximum allowed by the links’ Gigabit data rate. These benefits may allow each host to run another MPI process to either improve overall performance or allow equivalent performance with fewer hosts. It is this second option of reducing the number of hosts that will be used for the cost comparison.

The first portion of the study focused upon measuring the processing load and performance of a system with standard NICs to determine the theoretical gains possible. The actual communications load of computational processes using MPI varies widely depending upon the amount of data that must be exchanged. Synthetic tests which maximized network load were used to determine worst-case loading and give an upper bound on the gains that might be obtained with an embedded NIC. Then embedded NICs from Alacritech were used to directly compare loading, data rate, and latency with the reference systems. These cards are still in beta stage development. They showed multiple inconsistencies and problems that might be expected for a new technology. These problems have limited some tests and reduced some performance measurements, so the final results are not conclusive.

The embedded 1000BaseT NIC used in our study cost $795. To be economically justifiable the embedded NIC must provide performance improvements that allow a reduction in the number of hosts and network switch ports that at least offset this additional cost per host. Dual processor hosts with fast CPUs (2+ GHz P4) and larger amounts of memory (1+GB) are $3.5-4K per host with interface card. Wire interfaces and switches are far cheaper than the fiber optic versions; a smaller wire interface (24-48 ports) switch may cost $200 per port, a large switch still more. A first order comparison with the lowest per host costs shows that per host cost is $4500 with an embedded NIC vs. $3700 without, a ratio of 1.22 to 1. This implies that hosts with the embedded NIC must provide a 22% improvement in performance to allow the number of hosts to be cut to an equivalent total cost for constant performance. Can this be realized?

Several different forms of tests were used. Custom C++ classes were developed to support process load measurement, timing distribution and statistical analysis, and TCP connections with special timing, buffer, and options control. Those tests which used MPI were performed with MPICH and MPIpro. Five hosts were connected via a non-blocking Gigabit Ethernet switch. Each host contained two 1 GHz Pentium 3 processors and a 64 bit PCI interface slot for the NIC. Two Dell hosts with dual 2.4 GHZ Pentium IV processors were added later. These represent the current “best performance” lower cost hosts.

The first series of tests used the netperf TCP performance test software to determine if these systems could support
line rate data transfer under optimized conditions and to determine the processor load the transfers engendered. The second series of tests used the standard MPICH test program mpptest that combines a synthetic computational load with message passing in a synchronized loop. This was modified to support individual threads for computation and message passing, detailed loop timing measurements, and accurate system load measurement. The third series of tests used a custom program, latencyTest, to measure latency and system load for message passing with a simultaneous two-way synchronized exchange, ping-pong, and one-way scenarios with blocking and nonblocking MPI and direct TCP socket communications with a variety of options including direct and user copy sends with the optimized TCP sendfile for best performance with the Alacritech cards. The directly controlled TCP communications allowed testing TCP socket options and methods of transfer not directly realized in the current MPI software but which might yield improved performance with the embedded NIC.

Analysis of the testing with standard NICs indicated several important results:

1. Full rate one-way TCP message transfer with larger message sizes nearly uses the computational capacity of a 1GHz Pentium 3. In one test using netperf with two receivers and one transmit running simultaneously, the two processor system had both processors running at 93% total load. All of this is system load (kernel processing); most is TCP/IP and buffer copying. With the 2.4 GHz host the loads were not so significant. A single max rate transfer used only 10% total load, the simultaneous transfers only 30%. Thus only older hosts might benefit significantly from an embedded card.

2. The maximum packet size as determined by the Gigabit Ethernet interface’s MTU and the buffer size for the TCP receive buffer were the most critical factors for performance in all tests of data transfer rates. The key lessons from these tests were two: the MTU for the Gigabit Ethernet card should be set to “jumbo frame” size of 9000 bytes and the default buffer size for MPI communications should be changed from 16K to 64K bytes. For MPICH this requires a slight modification of code; MPIpro requires specific runtime arguments. These two simple changes almost quadruple the performance of larger size data transfers between MPI processes.

3. If a second processor or embedded processing is available, the program will benefit greatly from simply separating the MPI data transfer and the local computation into unique threads. The speed-up will depend upon the ratio of computation to communication, but can be nearly twofold if both are matched. MPICH is not thread safe so all MPI actions must be performed within a single thread with the explicitly programmed use of mutexes and condition variables to control synchronization between the MPI communication thread and other threads. The processing load created by the MPI data transfers can be reduced by splitting the MPI communications into still further threads and using more efficient blocking sends and receives. MPIpro is fully multithreaded and uses multiple threads for MPI communications without any further user code in the program using MPI.

4. Finally, some MPI transactions involve exchanges of only very small amounts of data so end-to-end latency rather than top data rate is critical. Latency is dominated by factors other than network data rate and is just about identical for 100MBit and 1Gbit LANs.

The NICs were replaced with Alacritech 1000BaseT NICs with on-card hardware and firmware support for TCP/IP protocol processing and the kernel was modified with a small set of patches to support the Alacritech driver’s bypass of the kernel’s TCP/IP stack for certain network operations. This allows all software to be run with varying degrees of offload by the interface card with no change in program code. The most efficient offload is performed during blocking sends and receives with the receive buffer space preassigned. MPI_Isend and MPI_Irecv are, in general, the optimum MPI commands for data transfer because they do not immediately block the calling process. The MPIpro implementation of these commands uses threads and blocking IO. This allows the Alacritech card to more fully process TCP/IP streams onboard the card and results in the largest gain. MPICH uses nonblocking IO; the gains are not as significant because the kernel must still do some work.

The Alacritech NICs are not yet commercially released and the preliminary status of the software/firmware showed up repeatedly with inconsistencies in operation and performance. As is, the cards cannot be used in any production environment but the performance in some tests show a reduction of the network processing load. Currently the cards do not support jumbo frames which reduces their maximum single stream data rate; later software revisions are planned to support this and may result in important improvements. Latency tests with small message sizes show per packet latency reasonably similar to that seen with the standard NIC. Some tests on the older hosts showed promise. For example, simultaneous two way message passing with MPIpro left 81% of the total processing capacity available vs. 66% with a standard NIC, a gain of 1.34 to 1. Because the network load was not as great a percentage on the new hosts, such gains were not possible and little change was seen.

A perfected gigabit TCP/IP NIC may be worthwhile for older clusters with slower processors at current prices, but the price will need drop significantly before it is cost effective for new clusters. Special purpose cards which integrate higher level functions such as I-SCSI are also attractive. Finally, the next generation 10 gigabit ethernet will almost certainly require embedded protocol support.