

MPI and Embedded TCP/IP Gigabit Ethernet Cluster Computing

Neal Bierbaum

Sandia National Labs

nrbierb@sandia.gov

Abstract

A group of lower cost PCs 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. TCP communications at near wire speed (above 800 Mbit/sec) use almost the entire processing capacity of a 1 GHz Pentium 3 processor with a standard network interface card (NIC), effectively reducing the performance of an economical two processor host by almost half. 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.

Introduction

Mid-size clusters (10-100) of computers connected by high speed LANs provide a cost effective resource for many large problems. These generally use Linux as the operating system and MPI to integrate parallel processes on multiple hosts. A switched Gigabit Ethernet (1000BT) LAN connecting all hosts in a cluster provides high bandwidth interconnectivity between processes that need to exchange large amounts of data. In this environment the TCP/IP network protocol processing consumes an important amount of the computing resources of each node. Network interface cards that perform the protocol processing on the card offer the possibility of reducing this significantly so that the host's processing capacity may be applied to the computational tasks. How large can the gains be with this approach? Do these much more expensive cards (now \$795 compared with standard 64-bit wide PCI gigabit cards at \$75 for 1000BaseT) justify their premium price?

Design of Study

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. This provides an upper bound on the gains that might be obtained with an embedded NIC. These tests showed that TCP/IP protocol processing at near line-rate generated a load almost equivalent to the capacity of one of the two 1GHz Pentium 3 processors in the test host. This heavy load shows that the gain from using an embedded NIC could be very significant.

The second portion of the study used embedded NICs from Alacritech 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. Yet some of the results showed a promise that has made further testing and analysis worthwhile.

Cost Analysis

The embedded 1000BaseT NIC used in our study cost ~\$725 more than a standard NIC. 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 servers 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 costs are \$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?

Test Design

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, v.1.2.2, the most widely used open source MPI implementation, and MPIpro, a commercial MPI implementation that uses multiple threads and blocking IO. Five hosts were connected via a nonblocking gigabit Ethernet switch. Each host contained two 1 GHz Pentium 3 processors and a 64 bit wide PCI interface slot with a 3Com "Alteon" Ethernet NIC. Each host used the Linux 2.4.16 kernel.

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. Packet size, network buffer size, and message size were varied.

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* function. The directly controlled TCP communications allowed testing options and methods of transfer not directly realized in the current MPI software but which might yield improved performance with the embedded NIC.

First Phase Base Results

Analysis of the testing during the first phase indicated several important results:

1. More than the full processing power of a 1 GHz Pentium III can be consumed with just a pair of TCP connections performing send and receive at maximum rate over the Alteon Gigabit interface. In one test using *netperf* with two receives and one transmit running simultaneously (equivalent to multiple instances of an MPI process with major data exchange running on the host), the two processor system had both processors running at 93% total load -- this simple network action consumed 186% of a single processor's capacity! All of this is system load (kernel processing); most is TCP/IP and buffer copying. Receiving a transfer uses about twice the processing of sending. Maximum rate message relay with large messages generally used at least 60% to 80% of a single CPU. This shows that removing the protocol processing load from the hosts CPUs has the potential to free a large amount of computing resource for direct computational work.
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 if possible. 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. But if buffer sizes cannot be increased, use a smaller MTU (2100 bytes is about optimum for 16K buffers).

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. Some MPI transactions involve exchanges of only very small amounts of data, especially when they are used for process synchronization. The end-to-end latency is critical for such actions, especially if they involve a large number of processors. Latency, the time between one process sending the packet and another receiving it, is dominated by factors other than network data rate. In fact, the latency for small packet sizes either by MPI or just with TCP/IP sockets was almost identical with 100MBit and 1Gbit LANs.

Second Phase Preliminary Results

The NICs in two of the test hosts were replaced by the Alacritech 1000BaseT card with on-card

hardware and firmware support for TCP/IP protocol processing. The Linux 2.4.16 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 for Linux and the preliminary status of the Linux version of software/firmware showed up repeatedly with inconsistencies in operation and fairly common full system crashes by the Linux kernel during heavy testing. As is, these NICs cannot be used in any production environment. But the performance in some tests shows that they may well be effective offloading much of the network processing load. Currently the cards do not support jumbo frames but only the standard 1500 byte MTU which somewhat reduces their maximum single stream data rate -- especially if they are communicating with a standard NIC. Latency tests with small message sizes show per packet latency reasonably similar to that seen with the standard NICs.

Analysis and Results

The potential gain can be shown by comparison of simultaneous two way message passing with MPIpro between pairs of hosts using the standard NICs and the embedded NICs. 2500 130KB messages were sent each way simultaneously with synchrony between each transfer with nearly equivalent total time. 81% of the total processor time was unused on the hosts with embedded NICs vs. 66% for the standard NICs, a ratio of 1.34 to 1. Other tests did not show such gains.

Conclusions

The beta level Alacritech cards are not yet ready for Linux in other than test environments. Yet our initial results show that these or similar cards that may be released by other manufacturers in the near future will be economically justifiable

even at their current prices once perfected. Anyone planning a moderately sized cluster, especially one which will use MPIpro with Gigabit Ethernet should evaluate a NIC with embedded TCP/IP.