# GigaBit Performance under NT

Mark Baker
*University of Portsmouth*
*Hants, UK, PO4 8JF, UK*
Mark.Baker@computer.org

Stephen Scott and Al Geist
*Oak Ridge National Laboratory*
*Oak Ridge, TN 37831-6367, USA*
{scottsl,gst}@ornl.gov

Logan Browne
*Hiram College*
*Hiram, OH44234, USA*
*BrownLC@hiram.edu*

January 13, 2000

## Abstract

The recent interest and growing popularity of commodity-based cluster computing has created a demand for low-latency, high-bandwidth interconnect technologies. Early cluster systems have used expensive but fast interconnects such as Myrinet or SCI. Even though these technologies provide low-latency, high-bandwidth communications, the cost of an interface card almost matches that of individual computers in the cluster. Even though these specialist technologies are popular, there is a growing demand for Ethernet which can provide a low-risk and upgradeable path with which to link clusters together. In this paper we compare and contrast the low-level performance of a range of Giganet network cards under Windows NT using MPI and PVM. In the first part of the paper we discuss our motivation and rationale for undertaking this work. We then move on to discuss the systems that we are using and our methods for assessing these technologies. In the second half of the paper we present our results and discuss our findings. In the final section of the paper we summarize our experiences and then briefly mention further work we intend to undertake.

**Keywords**: cluster interconnect, communication network, Gigabit Ethernet, PVM, MPI, performance evaluation.

# 1. Introduction

The concept of a cluster of computers as a distinguished type of computing platform evolved during the early 1990's[1]. Prior to that time, the development of computing platforms composed of multiple processors was typically accomplished with custom-designed systems consisting of proprietary hardware and software. Supercomputers, or high-performance multiprocessor computers, were designed, developed, and marketed to customers for specialized grand challenge applications. Typically, the applications that ran on these supercomputers were written in Fortran or C, but used proprietary numerical or messaging libraries, that were generally not portable.

However, rapid advances in commercial off-the-shelf (COTS) hardware and the shortening of the design cycle for COTS components made the design of custom hardware cost-ineffective. By the time a company designed and developed a supercomputer, the processor speed and capability was out-paced by commercial processing components. In addition to the rapid increase in COTS hardware capability that led to increased cluster performance, software capability and portability increased rapidly during the 1990's. A number of software systems that were originally built as academic projects led to the development of standard portable languages and new standard communication protocols for cluster computing.

The programming paradigm for cluster computing falls primarily into two categories: message passing and distributed shared memory (DSM). Although DSM is claimed to be an easier programming paradigm as the programmer has a global view of all the memory, early efforts instead focused on message passing systems. Parallel Virtual Machine[2] (PVM) started as a message passing research tool in 1989 at Oak Ridge National Laboratory (ORNL). Version 2, written at the University of Tennessee, was publicly released in early 1991. As a result of this effort and other message passing schemes, there became a push for a standardized message passing interface. Thus, in 1994 MPI Version 1 was approved as a *de jure* standard for message-passing parallel applications[3]. Many implementations of MPI-1 have been developed. Some implementations, such as MPICH, are freely available. Others are commercial products optimized for a particular system, such as SUN HPC MPI. Generally, each MPI implementation is built over faster and less functional low-level interfaces, such as BSD Sockets, or the SGI SHMEM interface.

# 2. Message Passing

## 2.1 MPI Overview

The MPI standard[4] is the amalgamation of what were considered the best aspects of the most popular message-passing systems at the time of its conception. The standard only defines a message passing library and leaves, amongst other things,

process initialisation and control to individual developers to define. MPI is available on a wide range of platforms and is fast becoming the *de facto* standard for message passing.

The design goals of the MPI were portability, efficiency and functionality. Commercial and public domain implementations of MPI exist. These run on a range of systems from tightly coupled, massively-parallel machines, through to networks of workstations. MPI has a range of features including: point-to-point, with synchronous and asynchronous communication modes; and collective communication (barrier, broadcast, reduce).

MPICH[5,6] developed by Argonne National Laboratory and Mississippi State University, is probably the most popular of the current, free, implementations of MPI. MPICH is a version of MPI built on top of Chameleon[7]. MPICH and its variants are available for most commonly used distributed and parallel platforms.

## 2.2 PVM Overview

The Parallel Virtual Machine[8] (PVM) system provides an environment within which parallel programs can be developed and run. PVM is a continuing research and development project between ORNL, Emory University and the University of Tennessee.

PVM transparently handles all message routing, data conversion and task scheduling across a network of heterogeneous computer architectures. PVM is available for most computer architectures, including Linux and NT. The PVM system consists of:

- A PVM daemon (or NT service) which is installed on each PVM host computer – this daemon is used to initiate and manipulate the PVM environment.
- A set of libraries to perform parallel communication between PVM tasks, an initiation method for the parallel environment.
- A console that allows users to manipulate their PVM environment by, for example, adding, deleting hosts as well as starting and monitoring, and stopping PVM programs.
- A set of functions for debugging both the PVM environment and a PVM program.

# 3. Gigabit Ethernet

Gigabit Ethernet offers an upgrade path for current Ethernet installations and allows existing installed stations, management tools and training to be reused. It is anticipated that the initial applications for Gigabit Ethernet are for campuses or buildings requiring greater bandwidth between routers, switches, hubs, repeaters and servers[9]. At some time in the near future Gigabit Ethernet will be used by high-end desktop computers requiring a higher bandwidth than Fast Ethernet can offer.

Gigabit Ethernet is an extension of the standard (10 MBps) Ethernet and Fast Ethernet (100 MBps) for network connectivity. The Gigabit Ethernet standard, IEEE 802.3z, was officially approved by the IEEE standards board in June 1998. Gigabit Ethernet employs the same Carrier Sense Multiple Access with Collision Detection (CSMA/CD) protocol, frame format and size as its predecessors.

Much of the IEEE 802.3z standard is devoted to the definition of physical layer of the network architecture. For Gigabit Ethernet communications, several physical layer standards are emerging from the IEEE 802.3z effort – these standards are for different link technologies as well as short and long distant interconnects. The differences between the technologies are shown in Table 1[10].

| | **Ethernet 10 BaseT** | **Fast Ethernet 100 BaseT** | **Gigabit Ethernet 1000 Base X** |
|---|---|---|---|
| **Data Rate** | 10 Mbps | 100 Mbps | 1000 Mbps |
| **Cat 5 UTP** | 100 m (min) | 100 m | 100 m |
| **STP/Coax** | 500 m | 100 m | 25 m |
| **Multimode Fiber** | 2 km | 412 m (half duplex)<br>2 km (full duplex) | 550 m |
| **Single-mode Fiber** | 25 km | 20 km | 5 km |

**Table 1**: Ethernet segment limitations

## 4. MPI NT Environments

There are now six MPI environments for NT[11]. These range from commercial products, such a MPI/Pro and PaTENT, to the standard release of MPICH with a WinSock devise. The MPI environments used to evaluate Gigabit network performance are described briefly in sections 4.1 – 4.3.

### 4.1 MPI/PRO for Windows NT

MPI/Pro[12] is a commercial environment released in April 1998 by MPI Software Technology, Inc. The current version of MPI/Pro is based on WinMPIch[13] but has been fairly radically redesigned to remove the bottlenecks and other problems that were present. MPI/Pro supports both Intel and Alpha processors and is released to be used with Microsoft Visual C++ and Digital Visual Fortran. The MPI/Pro developers are currently working on a new source base for MPI that does not include any MPICH code and supports the VI Architecture[14].

### 4.2 PaTENT WMPI 4.0

PaTENT[15] is the commercial version of WMPI funded by the European project WINPAR[16]. PaTENT differs from WMPI in a number of small ways which includes: sanitized release, easier installation, better documentation and full user support. PaTENT is available for Microsoft Visual C++ and Digital Visual Fortran and consists of libraries, header files, examples and daemons for remote

starting. PaTENT includes ROMIO, ANL's implementation of MPI-IO, configured for UFS. PaTENT uses the Installshield software mechanisms for installation and configuration.

### 4.3 WMPI

WMPI[17] from the Department of Informatics Engineering of the University of Coimbra, Portugal is a full implementation of MPI for Microsoft Win32 platforms. WMPI is based on MPICH and includes a P4[18] device. P4 provides the communication internals and a startup mechanism (that are not specified in the MPI standard). For this reason WMPI also supports the P4 API. The WMPI package is a set of libraries (for Borland C++, Microsoft Visual C++ and Microsoft Visual FORTRAN). The release of WMPI provides libraries, header files, examples and daemons for remote starting.

## 5. Performance Tests

### 5.1 Test Equipment

The aim of these tests is restricted to gathering data that helps indicate the expected communications performance (peak bandwidth and message latency) of MPI on NT. The benchmark environment consisted of two dual-processor Pentium's (450 MHz PIII) with 512 MBytes of DRAM running NT 4 (SP5), Windows 2000β3[1] with individual links between each pair of network cards. The technical details of the network cards assessed is given in Table 2.

| Card Make | Technical Details | Cost |
|---|---|---|
| NetGear[19] FA310TX 100Mbps | IEEE 802.3u 100BASE-TX Fast Ethernet and 802.3i 1 | MSRP $24.95 ($17.50 in qty 50) |
| GigaNet[20] Clan GNN1000 | 32/64-bit 33MHz, PCI 2.1 compliant, 1.25Gbps full duplex[2]. | MSRP $795 |
| Packet Engine[21] GNIC II | 32/64-bit 33MHz, PCI 2.1 compliant, 2 Gbps full duplex | $995 No longer available - out of NIC business |
| SysKonnect[22] SK-9841 | 32/64-bit 33/66MHz PCI 2.2 complaint, 2 Gbps full duplex | MSRP $729 |
| NetGear GA620 | 32/64-bit 33/66MHz PCI 2.1 complaint, 2 Gbps full duplex | MSRP $299.99 |

**Table 2**: Network Card Specification

### 5.2 Multi-processor Benchmark - PingPong

In this program, increasing sized messages are sent back and forth between processes. PingPong is an SPMD program written in C using the PVM, MPI and WinSock message passing APIs. These codes have been carefully developed so that all three versions as closely as possible match each others behaviour. PingPong provides information about the latency of send/receive operations and

[1] Our references to NT 5 and Windows 2000 are synonymous.
[2] GigaNet uses a proprietary protocol for communications, rather than Ethernet

the uni-directional bandwidth that can be attained on a link. To ensure that anomalies in message timings do not occur the PingPong is repeated for all message lengths.

### 5.2.1 MPI Version

The MPI version of the code uses the blocking send/receive on both processes.

```
MPI_Send(A,nbyte,MPI_BYTE,0,10,MPI_COMM_WORLD);
MPI_Recv(A,nbyte,MPI_BYTE,0,20,MPI_COMM_WORLD, &status);
```

### 5.2.2 PVM Version

The PVM version of the code is slightly more complicated as data needs to be packed into buffers before being sent and unpacked at the receiving end.

*Master:*

```
pvm_initsend(ENCODING);
for (length = 0, length < maximum; increment message length)
{
   pvm_pkbyte(send buffer, length, 1);
   pvm_send(slave ID, 1)
   pvm_recv(-1, -1)
}
```

*Slave:*

```
pvm_initsend(ENCODING);
while (true) {
    bufid = pvm_recv(-1, -1);
    pvm_bufinfo(bufid, (int*)0, (int*)0, &dtid);
    pvm_send(parent ID, 2);
  }}
```

### 5.3 Differences of the MPI and PVM versions of PingPong

A comparison of the MPI and PVM codes shows that there are some potential differences in how user data is handled and this may cause some performance differences. The one obvious difference is the way user data is handled. In particular the PVM Master leaves the received user data in a temporary buffer space. This and other effects will be investigated and reported upon in the final workshop presentation.

## 6. Results

### 6.1 Introduction

In this section we present and discuss the results that were obtained from running the various performance tests under MPI and PVM. It should be noted that not all the PVM results were available at the time of submission of this paper – but will be available for the actual workshop. It should also be noted that due to design restrictions, PaTENT or WMPI are unable to use alternative network interfaces,

other than that pointed at by the local host name. This problem was pointed out to both sets of developers (Genias and Coimbra), but unfortunately a "fix" was provided in time to incorporate the results in this paper.

| | System | Latency (μs) |
|---|---|---|
| 1. | MPI/Pro 1.2.3, SMP NT4 | 106.3 |
| 2. | WSOCK 32, SMP NT4 | 74.0 |
| 3. | WMPI 1.2, SMP NT4 | 44.2 |
| 4. | PaTENT 4.014, SMP NT4 | 32.8 |
| 5. | MPI/Pro 1.2.3, SMP NT5 | 98.2 |
| 6. | WSOCK 32, SMP NT5 | 76.4 |
| 7. | PaTENT SMP NT5 | 35.5 |
| 8. | MPI/Pro 1.2.3, TCP 100 Mbps | 207.6 |
| 9. | WSOCK 32, TCP 100 Mbps | 97.5 |
| 10. | WMPI 1.2, TCP 100 Mbps | 283.4 |
| 11. | MPI/Pro 1.2.3, TCP NT5 100 Mbps | 244.1 |
| 12. | WSOCK 32 TCP NT5 100 Mbps | 112.7 |
| 13. | MPI/Pro 1.2.3, TCP GigaNet | 207.8 |
| 14. | WSOCK 32, GigaNet | 96.9 |
| 15. | MPI/Pro 1.2.3, TCP Packet Engine | 335.6 |
| 16. | WSOCK 32, TCP Packet Engine | 298.4 |
| 17. | MPI/Pro 1.2.3, TCP SysKonnect | 178.8 |
| 18. | WSOCK 32, TCP SysKonnect | 90.6 |
| 19. | MPI/Pro 1.2.3, TCP NetGear | 585.5 |
| 20. | WSOCK 32, NetGear | 666.2 |

**Table 3**: Measured 1 Byte Message Latency

### 6.2 Latency Results (Table 3)

**SM**[3] – PaTENT and WMPI clearly have the lowest latencies under NT4 – approximately half the time taken by WinSock and MPI/Pro. Under NT5 WinSock and PaTENT latencies are slightly slower than under NT4 (~8%). However, MPI/Pro under NT5 is slightly faster (~8%) than under NT4.

**TCP**[4] **(100 Mbps)** – WinSock has more than half the latency of the MPI environments – both under NT4 and NT5. MPI/Pro is about 25% faster than WMPI. Under NT5 all systems exhibit a 10 – 15% increase in latency.

TCP (GigaBit) – The WinSock results for GigaNet (53%), Packet Engine (11%) and SysKonnect (50%) network cards are all faster than the MPI/Pro results. However, for NetGear performance WinSock (14%) is slower than MPI/Pro. This particular result is unexpected as MPI/Pro is built on top of the WinSock API. Overall, the SysKonnect card exhibits the lowest latencies, closely followed by GigaNet and Packet Engine. The latencies for NetGear are more than double of those for the other network cards.

[3] SM is where two processes are running on one computer and potentially communicating via Shared-Memory.

[4] TCP is where two processes are running on separate computers and communicating via TCP/IP.

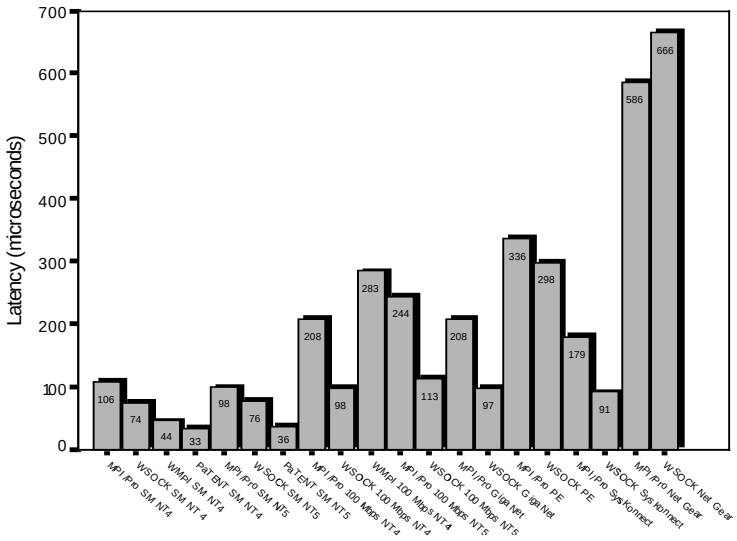


**Figure 1** - One Byte Network Latencies

## 6.3 Network Bandwidths

### 6.3.1 Shared Memory Results (Figure 2)

PaTENT and WMPI exhibit the best overall performance under NT4 and NT5. Under NT4, PaTENT and WMPI have a peak bandwidth of just over 100 MBytes/s and under NT5 PaTENT peaks at 122 Mbytes/s. MPI/Pro under NT 4 and NT5 has a similar bandwidth to WinSock up until message lengths of 8K. MPI/Pros bandwidth then continues to increase, peaking at 107 Mbytes/s under NT4 and at 122 Mbytes/s under NT5. Winsock peaks at 31 Mbytes/s under NT 4 and 39 Mbytes/s under NT 5 – here it also exhibits a huge performance dip between 16K and 64K message lengths. It should be noted that higher peak bandwidths were achieved under NT5 compared to NT 4.

### 6.3.2 Distributed Memory

#### MPI/Pro Results (Figure 3)

The bandwidth results from the 100 Mbps and GigaNet network cards between 1 and 512 Bytes are very similar. Thereafter the GigaNet results continue to increase up to 256K length messages where a peak of 37 Mbytes/s is reached. The 100 Mbps network card outperforms the Packet Engine, SysKonnect and NetGear network cards up until message lengths of about 1K. The 100 Mbps technology peaks at 8.8 Mbytes/s. The bandwidth of NetGear is much poorer than all the other technologies up until 2K message lengths. The peak bandwidths for Packet Engine, SysKonnect and NetGear are 12 Mbytes/s, 17 Mbytes/s and 19 Mbytes/s respectively.

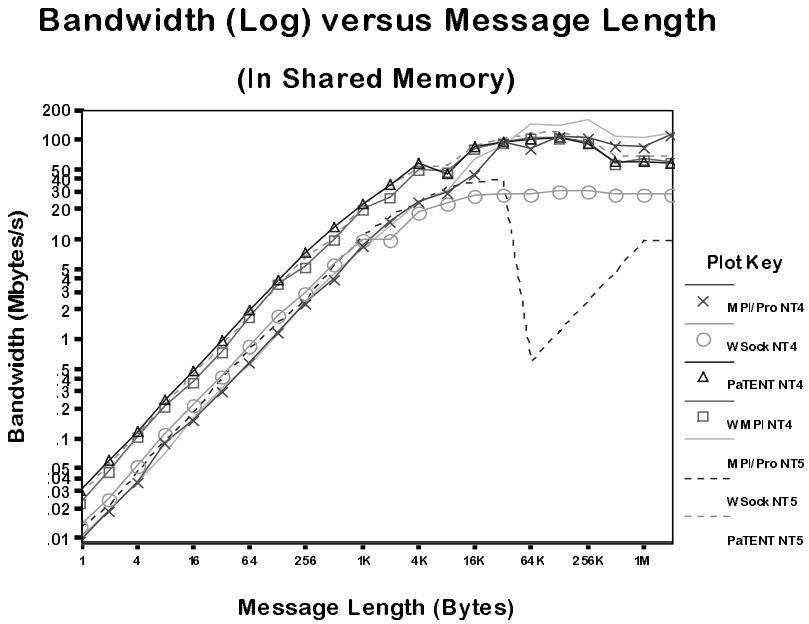


Figure 2 - **PingPong Shared Memory Results**

### WinSock Results (Figure 4)

The bandwidth results from the 100 Mbps and GigaNet network cards between 1 and 128 Bytes are very similar. Thereafter the GigaNet results continue to increase up to 8K length messages where a peak of 38 Mbytes/s is reached. The 100 Mbps network card outperforms the SysKonnect network card up until 256 Bytes message length. The 100 Mbps network card outperforms the Packet Engine and NetGear network cards up until 4K message lengths. The 100 Mbps technology peaks at 10 Mbytes/s. The bandwidth of NetGear is much poorer than all the other technologies up until 8K message lengths. The peak bandwidths for Packet Engine, SysKonnect and NetGear are 10.6 Mbytes/s, 17.4 Mbytes/s and 17 Mbytes/s respectively.

## 7. Summary and Conclusions

### 7.1 Summary

In this paper we have presented and discussed the results from our simple network performance tests on NT using the MPI, PVM and WinSock message passing APIs on six different network interface technologies. At the date of submission, we have been unable to complete the PVM tests, so our discussion on the performance differences is limited at this moment.

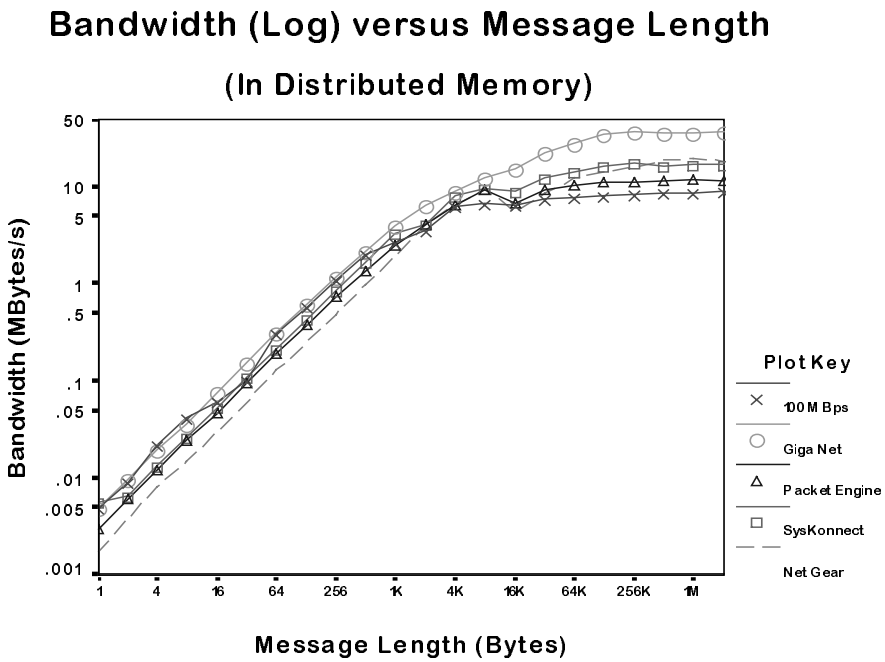


**Figure 3** - MPI/Pro Bandwidth Results

Our experiences with the performance of MPI under NT 4 and Windows 2000 are inconclusive. Currently, it appears that in shared-memory mode that the latencies under Windows 2000 may be marginally lower than NT 4. The measured peak bandwidths of Windows 2000 were greater than NT4. In distributed-memory mode the measured latencies under Windows 2000 were approximately 20% higher than the equivalent under NT 4. The measured bandwidths for Windows 2000 and NT 4 were very similar however.

It is interesting to note that the measured network latencies for 100 Mbps Ethernet cards and Giga Net under WinSock and MPI/Pro are almost equivalent. The performance of the Packet Engine Gigbit card is between 7% and 13% faster respectively. However, the performance of the SysKonnect and Net Gear cards are significantly slower that standard 100 Mbps Ethernet.

### 7.2 Price/Performance Considerations

Table 4 shows the price/performance ratios calculated using the network card costs in September 1999 versus the peak measured bandwidth and minimum latency. It should be noted that the calculated ratios shown are only an approximate indicator as the price of the network cards varies significantly based on the quantity bought and the discounts given. The smaller the price/performance ratio the better value for money that can be expected from a network card. The choice of what is the most appropriate card is often not based

solely on the price/performance, but also other factor such as desired performance, compatibility or availability.

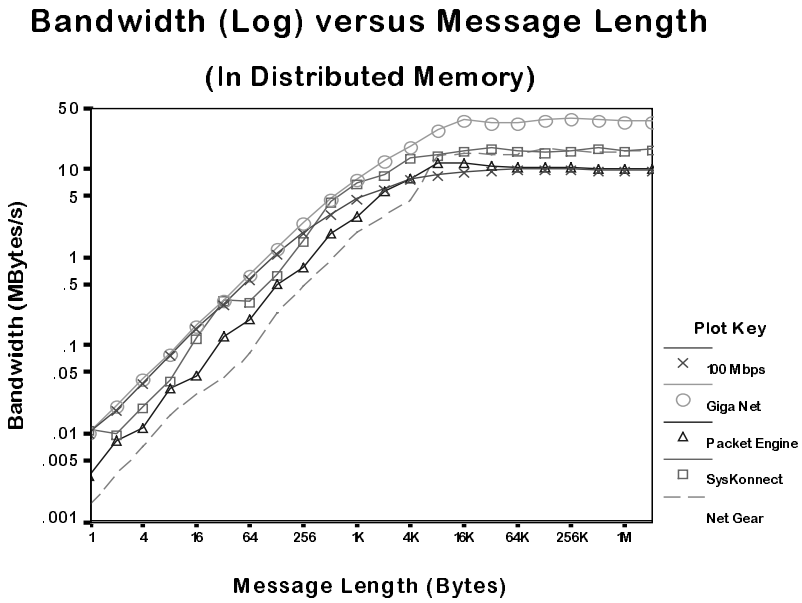


**Figure 4** - WinSock Bandwidth Results

The ratios shown in Table 4 indicate that the 100 Mbps Fast Ethernet cards provide significantly better price/performance than the other network cards. However, the ratios for the NetGear Gigabit card are significantly better than the other price/performance ratios available.

| Card Make and speed | Price/Performance ($/Mbytes/s) | Price/Performance ($/µs) |
|---|---|---|
| NetGear FA310TX 100Mbps | $24.95/8.8 = 2.835 | $24.95/208 = 0.12 |
| GigaNet - Clan GNN1000 | $795/37 = 2149 | $$795/208 = 3.82 |
| Packet Engine – GNIC II | $995/12 = 82.92 | $995/336 = 2.96 |
| SysKonnect – SK-9841 | $729/17 = 42.88 | $729/179 = 4.07 |
| NetGear - GA620 | $299.99/19 = 15.79 | $299.99/585 = 0.51 |

**Table 4**: Network Card Cost versus Performance (MPI/Pro)

### 7.3 Summary of Conclusions

Our work has shown that release 1.2.3 of MPI/Pro imposes an approximate additional 1 Byte latency of 25% and 50% over WinSock under shared and distributed-memory modes respectively. We have shown that the Giga Net Gigabit Ethernet provides the highest bandwidth of those tested. We suspect, as currently we do not have a concrete price for this card, that the price/performance of this card will be poorer that that of Net Gear but better than Packet Engine and NetGear. Our price/performance figures do, however, strongly suggest that the current performance and costs of the Gigabits cards makes standard 100 Mbps a much sounder technology investment at the moment. Obviously, other

factors, like required peak bandwidth, may make the decision of what technology to choose not one purely based on price/performance. Another factor that puts the Gigabit Ethernet at a disadvantage compared to other network technologies, such as Myrinet[23] and SCI[24], is the relatively high start up latencies – approximately an order of magnitude higher. These high latencies are being addressed with the new VIA interfaces and drivers being developed for Ethernet.

### 7.4 Future Work

This work is part of an on going effort to investigate the performance of a range of cluster-based technologies. The next phase of our work will involve comparing the performance of different network technologies under NT and Linux.

### References


[1] A. Geist, Cluster Computing: The Wave of the future, *Springer Verlag Lecture Notes in Computer Science*, May 1994.

[2] The PVM project - http://www.epm.ornl.gov/pvm/

[3] MPI Forum - http://www.mpi-forum.org/docs/docs.html

[4] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, *University of Tennessee, Knoxville, Report No. CS-94-230*, May 5, 1994

[5] MPICH - http://www.mcs.anl.gov/mpi/mpich/

[6] W. Gropp, et. al., A high-performance, portable implementation of the MPI message passing interface standard - http://www-c.mcs.anl.gov/mpi/mpicharticle/paper.html

[7] W. Gropp and B. Smith, Chameleon parallel programming tools users manual. *Technical Report ANL-93/23*, Argonne National Laboratory, March 1993.

[8] PVM: A Users' Guide and Tutorial For Networked Parallel Computing – http://www.netlib.org/pvm3/book/pvm-book.html

[9] Gigabit Ethernet Alliance - Gigabit Ethernet: Accelerating the standard for speed, http://www.gigabit-ethernet.org/technology/whitepapers, September 1999.

[10] Ethernet Segment Limits. - http://www.gigabit-ethernet.org/technology/

[11] TOPIC – http://www.dcs.port.ac.uk/~mab/TOPIC/

[12] MPI Software Technology, Inc. – http://www.mpi-softtech.com/

[13] WinMPICh - http://www.erc.msstate.edu/mpi/mpiNT.html

[14] VIA – http://www.viaarch.com

[15] PaTENT - http://www.genias.de/products/patent/

[16] WINdows based PARallel computing - http://www.genias.de/

[17] WMPI - http://dsg.dei.uc.pt/w32mpi/

[18] R. Buttler and E. Lusk, User's Guide to the p4 Parallel Programming System, *ANL-92/17*, Argonne National Laboratory, October 1992.

[19] NetGear - http://netgear.baynetworks.com/

[20] GigaNet - http://www.giga-net.com/

[21] Packet Engine - http://www.packetengines.com/index4.html

[22] SysKonnect - http://www.syskonnect.de/

[23] N. Boden, et. al. Myrinet - A Gbps LAN. *IEEE Micro*, Vol. 15, No.1, February 1995. http://www.myri.com/

[24] Dolphin Interconnect Solutions - http://www.dolphinics.no/