

NETWORK ARCHITECTURES FOR CLUSTER COMPUTING

prepared by

Jerry Dinkel

April 24, 2000

prepared for

Dr. Maples

CECS 572

Distributed Computing Systems and Networking

MW 7:00 PM

California State University Long Beach

1. Introduction

The need to provide computational solutions to a wide range of scientific, engineering, and business problems has driven the evolution of several distinct classes of computers. On the low end of the computer spectrum are the personal computers which are designed to provide low cost solutions to a wide range of general purpose problems. The high end of the spectrum includes the supercomputers which are tailored to solving the most complex and computationally intensive problems with little, or no regard to cost. The middle of the spectrum is populated by workstations, minicomputers, and mainframes. Over the years, the boundaries defining these classes have grown less distinct as advancements in semiconductor and microprocessor technologies have driven down costs and improved computational performance.

Supercomputers, which are the most specialized of all the computer architectures, remain the only solution for many of today's most difficult computational problems. The parallel architecture of modern supercomputers are highly optimized for floating point calculations and high sustained throughput of data between mass storage and main memory. To achieve this level of performance, supercomputers must be custom built using the latest semiconductor materials, processor architectures and communication bus designs. Supercomputers take a no compromise approach to optimizing overall system performance. The penalty for this approach is that supercomputers are very expensive to produce and are thus very few in number. User access to these machines remains very limited and very expensive.

The need to gain access to the computational power of supercomputers led researchers to explore the use of alternative lower cost hardware solutions. The concept of computer clusters was the direct result of the rapid advancements being made in the performance of the microprocessors used in personal computers. Computer clustering involves harnessing the combined computational power of several low cost personal computers by connecting them together through a common communications network to solve a problem in parallel. The goal of clustering is to achieve near-supercomputing performance at personal computer prices.

The implementation of a computer cluster requires the combined support of both software and hardware. The operating systems of the cluster machines must support message passing, distributed shared memory, and a parallel file system. To allow the exchange of messages and data between each machine in the cluster, a common communications network must be employed.

Beowulf, which was funded by NASA, was one of the first research projects to investigate the use of low cost personal computers in a cluster architecture. The initial project effort was conducted using a 16 node cluster composed of identical computers based on the 100 MHz Intel 80486 DX4 processor. The computers used the Linux operating system and were connected using a dual channel network constructed from 10 Mbps Ethernet. The performance of the initial

Beowulf cluster, although below supercomputing levels, was nonetheless encouraging. These results led to the continued development of the Beowulf architecture using newer, and faster, processors.

Analysis of the Beowulf cluster indicated that the overall system performance was limited by the data throughput of the communications network. This limitation served to restrict the transfer of data between remote disk drives and a node’s memory, and between memories on separate nodes. Data starvation resulted in processor stalling which reduced the overall performance of the cluster. Subsequent work with the Beowulf communications network has identified several ways to increase the network bandwidth and the overall data throughput.

This paper will examine several approaches for improving the performance and capacity of the communications network used for cluster computing. Although the primary focus will be on Beowulf, the network architectures presented are directly applicable to other types of computer clusters. The paper will examine several network architectures suitable for a 16 node cluster which were developed and tested using the early generation Beowulf clusters.

2. Beowulf Architecture

Since its inception at NASA’s Goddard Space Flight Center, the Beowulf project has evolved through several generations of computer and networking hardware. The project goal was to maximize computational performance using commodity hardware, without exceeding a cost target of \$50,000. Table 1 summarizes the major hardware components utilized in the early generation Beowulf clusters. In addition to those built by NASA, many other Beowulf clusters have been built by various government, research, and commercial institutions. The largest of these clusters contain in excess of 250 individual nodes.

Generation	Nodes	Processor (per node)	Memory (per node)	Storage (per node)	Network Type	Performance
1	16	100 MHz DX4	16 MB	540 MB	10 Mbps Ethernet	42 MFLOP
2	16	100 MHz Pentium	32 MB	1.2 GB	100 Mbps Ethernet	280 MFLOP
3	16	200 MHz Pentium Pro	64 MB	6 GB	100 Mbps Ethernet	2 GFLOP
4	64	Dual 200 MHz P6	128 MB	21 GB	100 Mbps Ethernet	10.2 GFLOP

Table 1. Beowulf Hardware Components

To insure the maximum utilization of the available hardware and communication resources, Beowulf clusters are designed to operate as an autonomous environment. Each cluster typically has one node designated as the “master”, which is responsible for coordinating and allocating the

computational activities of the other “slave” nodes. The master node has the only communications link to an external network and thus serves as the sole interface between the cluster and the outside world. All user interaction and program execution is initiated through the master node. This architecture allows the cluster to be fully dedicated to the execution of a single program without having to handle other processing and communications tasks.

All the nodes in the Beowulf cluster run the Linux operating system. Linux is an open-source UNIX clone that is freely distributed under the GNU Public License. Linux is a fully POSIX compliant operating system that supports a TCP/IP protocol stack with a BSD-compatible socket interface, very broad device support, dynamically linked shared libraries, interprocess communication, and an efficient virtual memory subsystem with unified buffer cache. Linux also supports many of the most popular application programming interfaces for distributed computing such as Remote Procedure Calls (RPC), the Parallel Virtual Machine (PVM) library, and the Message Passing Interface (MPI).

Due to its open source heritage, Linux was ideally suited for the highly experimental and developmental nature of the Beowulf project. With all of the source code available in the public domain, modifications could be made to the core functionality of the operating system kernel, device drivers, and I/O subsystems. This provided the system developers with the ability to tailor the operating system to support the architectural and performance requirements of the Beowulf cluster.

3. Channel Bonded Networks

All of the nodes in a Beowulf cluster are connected through a common communications network. This network provides the sole means for transferring data and messages between the individual nodes. The communications network serves the same purpose within the cluster as the data bus on a supercomputer. The difference lies in the fact that the bus of a supercomputer is contained on a common backplane which, by design, provides an extremely high, non-contentious bandwidth. In comparison, the Beowulf cluster must make due with commodity networking links that are essentially de-coupled from the computer hardware. This design choice results in a significantly lower bandwidth potential and thus poses a major bottleneck for data transfer between nodes. A major area of research in Beowulf clusters has centered around optimizing the utilization of available network bandwidth.

The first generation Beowulf cluster was designed using two channel bonded Ethernet networks connecting each of the 16 processing nodes. Each network channel consisted of a 10 Mbps Ethernet link. One of the channels used twisted pair 10BaseT with a hub, and the other used multidrop 10Base2 thin-net. Each node was equipped with a separate network adapter for each of the Ethernet channels. Communication between processes executing on individual nodes

was accomplished through the standard BSD socket interface which utilized a modified network interface.

The network interface was specially modified to make the dual channel Ethernet appear as a single channel to a running application. This was accomplished by establishing an internal pseudo-interface through which the application sent and received packets over the network. This pseudo-interface was assigned the hardware address of the primary network adapter. The network layer of the protocol stack was responsible for routing packets to and from the dual Ethernet channels through this interface. Packets received from either of the channels were modified so as to appear to have come from the primary network adapter. Packets sent were alternately distributed to the primary and secondary adapters. If the sending buffer of either of the network adapters became full, then all additional packets were routed to the other adapter. Although this simple strategy for routing packets through the two adapters does not exploit possible traffic patterns on the two networks, it did yield a maximum throughput of 13.6 Mbps from the two channels. This equates to 68% of the maximum available bandwidth.

To evaluate the performance of the channel bonded network, tests of the network throughput were conducted by varying the amount and size of the data traffic exchanged between nodes in the cluster. Rather than try to characterize the network performance using real world applications, in which many independent variables may effect the throughput, test applications were written that generate predetermined patterns of message traffic. From these experiments, the network throughput was measured as the number of packets transferred multiplied by the packet size and divided by the transfer time.

The first experiment used a program to exchange tokens (i.e., messages) between pairs of nodes in the cluster. In the test, one node generates a fixed size token, comprised of one or more UDP packets, and transmits it to a second node. The second node stores the token in a buffer and then transmit it back to the first node when the receipt is complete. To maximize the network throughput, all sockets used for token exchange were made non-blocking by using the Linux implementation of the POSIX *fcntl()* system call. During each exchange, the minimum transfer time between nodes was “determined by hardware and software considerations and accurately reflects all factors contributing to sustained communication performance.” [2]

To insure that the throughput of the network was being tested and not that of the node processors, all the nodes in the cluster were partitioned into eight sending/receiving pairs. Since no node participated in more than one pair, there was no possible contention for a single processor. By changing the number of pairs exchanging tokens at any one time, the load on the network could be varied between 1 and 8 tokens. Network traffic could also be altered by varying the size of the tokens being exchanged. Tokens larger than 1500 bytes—the maximum Ethernet frame size—were broken into separate UDP packets.

The results of the token exchange experiment using the 10 Mbps channel bonded Ethernet are shown in Figure 1. The results were generated from a series of test cases in which the number of tokens, the token size, and the number of channels was varied. The number of tokens was varied between 1 and 8. The token sizes were 2^n bytes, where n was varied from 2 to 13. The number of channels was either 1 or 2. For each combination of token rate, token size, and number of channels, 1000 token exchanges between send/receive pairs were used to generate each data point. The use of 1000 exchanges per data point was determined to be sufficient to minimize any timing errors.

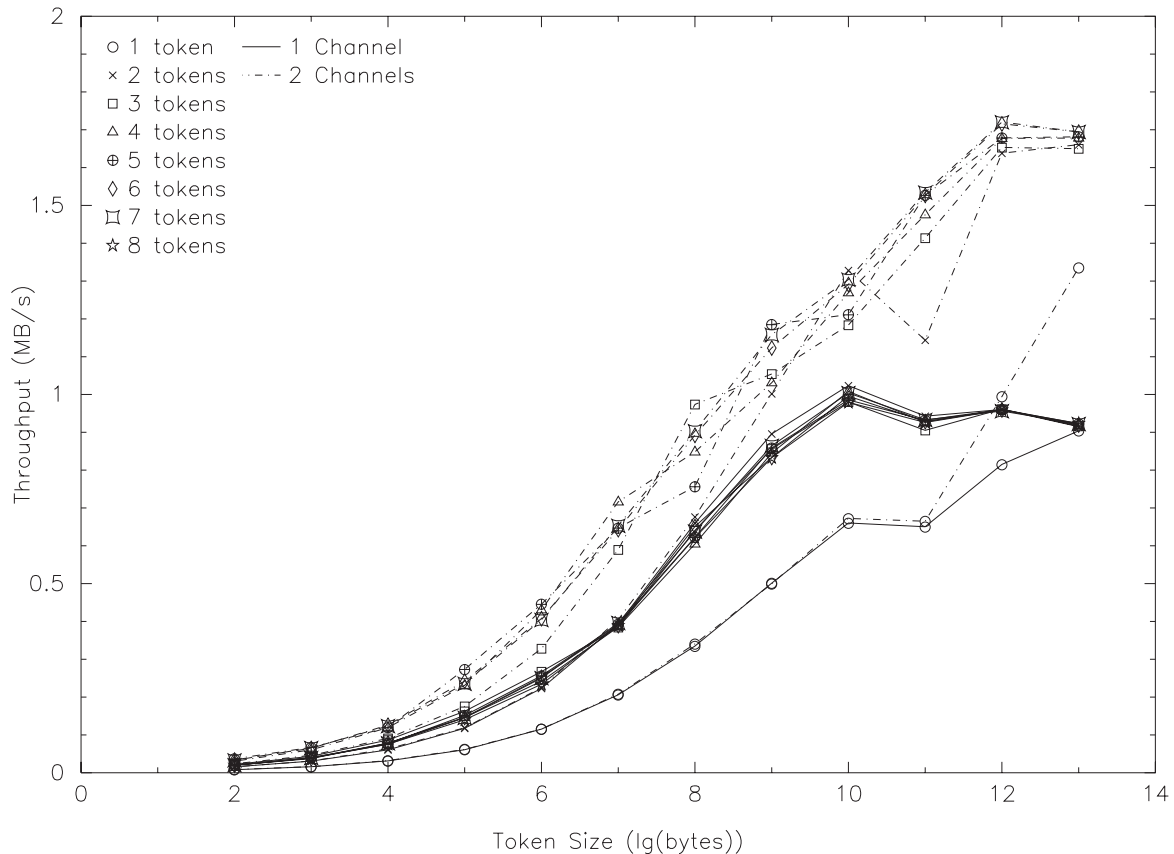


Figure 1. 10 Mbps Channel Bonded Network Throughput During Token Exchange

The experimental results indicate that the data throughput is a function of the token size and is relatively insensitive to the number of tokens exchanged at any one time. Excluding the case when the token rate is 1, all other cases are tightly clustered and have about the same throughput for the same token size. This phenomenon is a result of network contention that serves to serialize the data transfer on the available channels. Due to the implementation of the token exchange program, a token rate of 1 does not stress the network capabilities since at most 1 token can be “in-flight” at any one time. However, a token rate of 2 or more results in multiple tokens contending for the same network bandwidth and thus causes Ethernet’s contention arbitration strategy to sequence the packet delivery.

For token sizes less than or equal to 1024 bytes, both 1 and 2 channel networks are able to sustain the network traffic regardless of the number of tokens exchanged. However, as the token size increases to 2048 bytes, the 1 channel network becomes saturated and is unable to attain a higher throughput. This results from the need to break the token into two or more UDP packets, which increases the contention for the network and thus yields a higher collision rate. As the token size continues to be increased to the 8192 byte maximum, the network throughput deteriorates further. At its peak, the 1 channel network has a throughput of about 1 MBps, or 80% of the peak 1.25 MBps possible on 10 Mbps Ethernet.

Compared to the 1 channel network, the 2 channel network is able to sustain a higher throughput for a given token size, regardless of the number of tokens being exchanged. The onset of saturation for the 2 channel network does not occur until the token size increases above 4048 bytes. Degradation of the network throughput is evident at the maximum token size of 8192 bytes. However, the rate at which the throughput degrades can not be determined from the existing experimental data. At its peak, the 2 channel network has a throughput of about 1.7 MBps, or 68% of the peak 2.5 MBps possible on channel bonded 10 Mbps Ethernet.

The second experiment conducted with the 10 Mbps channel bonded Ethernet network was designed to test the I/O performance of the cluster by evaluating the network response to sustained data transfers between nodes. To mimic the operation of the Beowulf cluster during normal program execution, the data transfers performed by the test program included a mix of intraprocessor and interprocessor copies. Intraprocessor copies are local file transfers in which a single processor copies a file from its local disk to another file on its local disk. Interprocessor copies are remote file transfers between two processors and their associated local hard disks.

The test program was designed so that at any one time, each node was involved in either a local disk file copy or a remote file transfer. For remote file transfers, the cluster nodes were divided into send/receive pairs, as had been done with the token exchange experiment. This insured that the I/O performance of the cluster was being measured without the adverse effects of processor or disk contention. All file copies were performed using the UNIX *read()* and *write()* system calls. All remote file transfers were executed through the socket interface using UDP packets to transmit data over the network. To insure that all file transfers involved only uncached files, a dummy 32 MB file was copied prior to each test run.

The results of the file copying experiment using the 10 Mbps channel bonded Ethernet are shown in Figure 2. The results were generated from a series of test cases in which the mix of file copies, as well as the file size was varied. The mix of file copies ranged from 0, in which 14 nodes copied files locally, to 7, in which 7 send/receive pairs copied files remotely. At the time of the experiment, only 15 of the cluster's 16 nodes was available for testing, thus the maximum number

of send/receive pairs was limited to 7. The file sizes were 2^n MB, where n was varied from 0 to 4. All of the test cases were run using both of the available network channels.

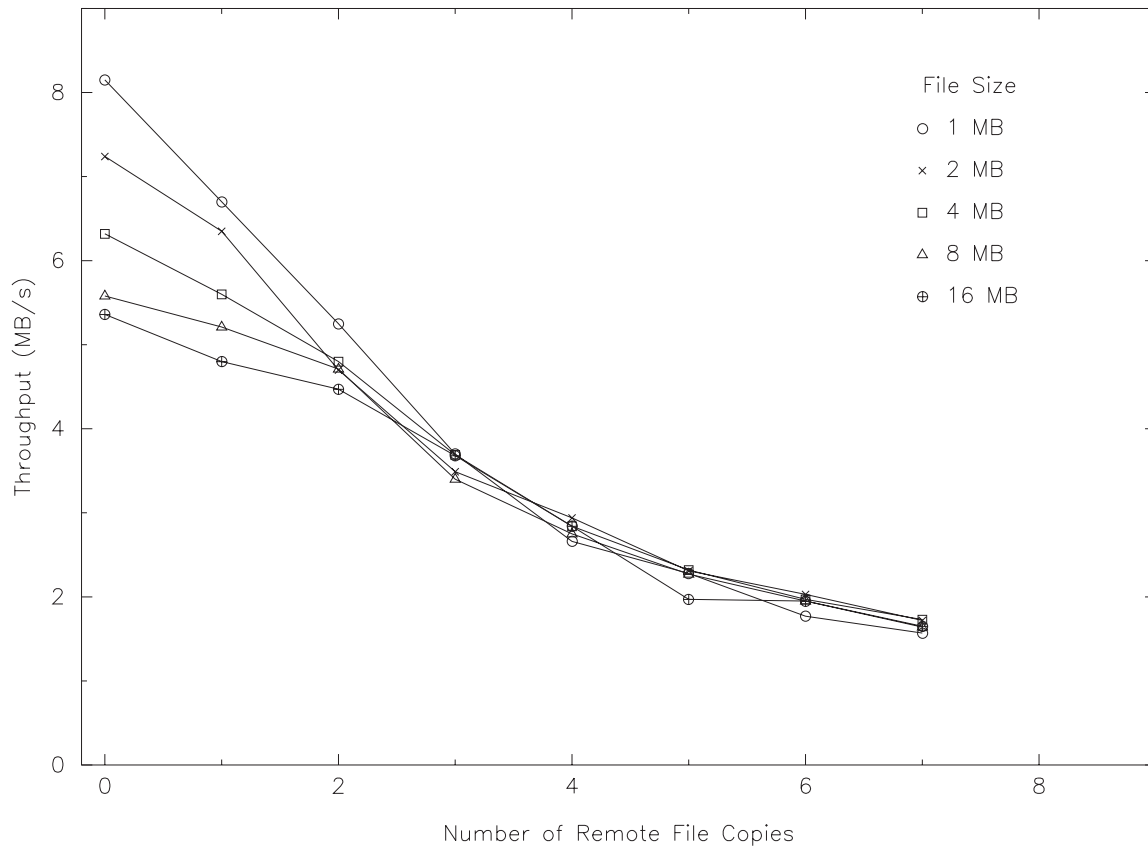


Figure 2. 10 Mbps Channel Bonded Network Throughput During File Copy

The experimental results show a progressive degradation in network throughput as the number and size of remote file transfers increases. When all file copies are done locally, the data throughput exceeds 8 MBps. This value decreases for larger file sizes due to buffering policies between the processor and the local hard disk. When the number of remote file copies is increased to 1, the extra overhead of transmitting data over the network degrades overall throughput by about 15%, in the worst case. As the number of remote copies is increased above 2, the data points for each test case become clustered closely together. This indicates that regardless of the file size, the network throughput is being constrained by the available bandwidth. When all file transfers are done remotely, the network throughput has decreased to 1.7 MBps, or 68% of the peak 2.5 MBps possible on channel bonded 10 Mbps Ethernet. This is the same throughput measured in the token exchange experiment using 2 channels.

The results of the network performance experiments using the first generation Beowulf cluster clearly showed that the channel bonded 10 Mbps Ethernet was constraining the communications throughput of the system. This is quite evident when noting that the throughput of an all local file copy is a factor of 4 greater than an all remote file transfer. Since the network of the cluster

was meant to serve the same purpose as the backplane in a supercomputer, this communications constraint serves to diminish the computational performance of the entire system. As work began on assembling the second generation Beowulf cluster, this situation was aggravated by the increased computational power of the updated hardware. To alleviate this interprocessor communications bottleneck, the bandwidth was increased by upgrading the network to 100 Mbps Ethernet.

The second generation Beowulf cluster was designed using two channel bonded 100 Mbps Ethernet networks connecting each of the 16 processing nodes. As in the first generation cluster, each node was equipped with a separate network adapter for each of the Ethernet channels. The two channels were transparently bonded by the network interface to appear as a single channel to a running application. Following the completion of the updated cluster, the network was subjected to the same performance experiments as the first generation cluster.

The results of the token exchange experiment using the 100 Mbps channel bonded Ethernet are shown in Figure 3. The results were generated using the same set of test cases from the first generation cluster in which the number of tokens, the token size, and the number of channels was varied. The only difference being that at the time of the experiment, only 15 of the cluster's 16 nodes was available for testing, thus the maximum number of send/receive pairs was limited to 7.

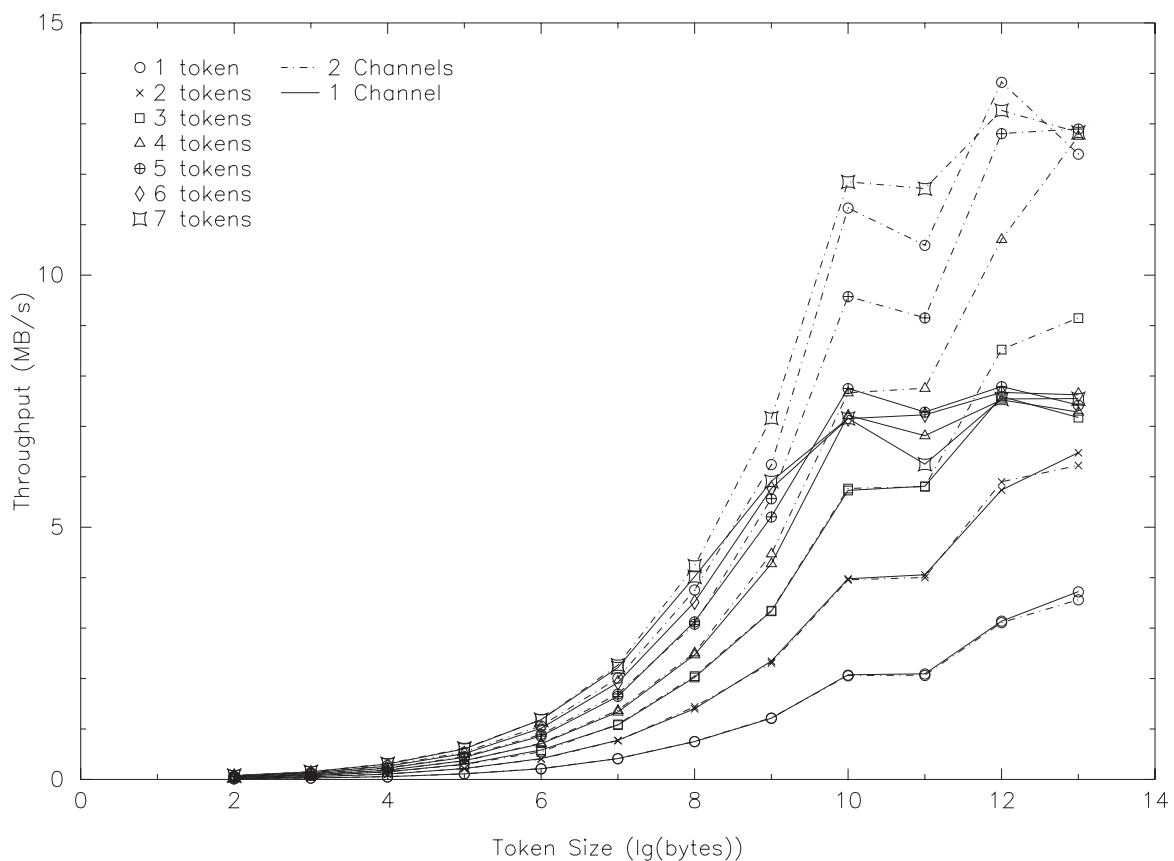


Figure 3. 100 Mbps Channel Bonded Network Throughput During Token Exchange

The most interesting result from the 100 Mbps token exchange experiment is that the data throughput is a function of both the token size and the number of tokens exchanged. This is in sharp contrast to the 10 Mbps experiment where throughput was a function of token size only. The results indicate that, in general, the throughput increases with the number of tokens and the token size up to the point of network saturation for both 1 and 2 channel networks.

The throughput of the 1 and 2 channel networks follow the same general growth pattern as the number of tokens and token size is increased. The major difference being that the throughput for the 2 channel network is always equal to, or higher, than the 1 channel network for a given number of tokens and token size. Both networks exhibit exponential growth up to a token size of 1024 bytes. As the token size is increased from 1024 to 2048 bytes, the throughput of both networks flatten out into a stair step. This is a result of the extra overhead of having to handle two UDP packets for every token transmitted. Between 2048 and 4096 bytes the throughput of both networks again begins to increase. Above a token size of 4096 bytes, both networks exhibit throughput degradation for the test cases with the greater number of tokens.

The token exchange experiment shows that a channel bonded 100 Mbps Ethernet network scales comparably to a channel bonded 10 Mbps Ethernet network. At its peak the 1 channel network has a throughput of 7.8 MBps, or 62% of the peak 12.5 Mbps possible on 100 Mbps. The 2 channel network achieves a throughput of 13.8 MBps, or 55% of the 25 MBps possible on channel bonded 100 Mbps Ethernet. Thus, the increase in throughput that can be achieved by going from 1 to 2 channels is 77% for 100 Mbps Ethernet and 70% for 10 Mbps Ethernet. In terms of bandwidth utilization, the 10 Mbps network is superior, achieving 80% of peak compared to the 100 Mbps network's 62% of peak for a single channel. However, more importantly, there is a factor of 8 increase in usable throughput between a channel bonded 10 Mbps network and a channel bonded 100 Mbps network.

The results of the file copying experiment using the 100 Mbps channel bonded Ethernet are shown in Figure 4. The results were generated using the same set of test cases from the first generation cluster in which the mix of file copies, as well as the file size was varied. As in the original experiment, only 15 of the cluster's 16 nodes was available for testing, thus the maximum number of send/receive pairs was limited to 7.

The experimental results demonstrate that the 100 Mbps channel bonded Ethernet network provides sufficient bandwidth to avoid constraining communications between nodes in the cluster. As the number of remote file transfers is increased, the throughput no longer converges on a maximum sustainable value for all file sizes. Instead, the throughput remains relatively constant, only degrading by about 3.5% for each additional remote file transfer added. This is in sharp contrast to the first generation network in which each new remote file transfer degraded the

throughput by about 15%. Another important point to note is that, in the worst case, the throughput of an all local file copy is only a factor of 1.4 greater than an all remote file transfer.

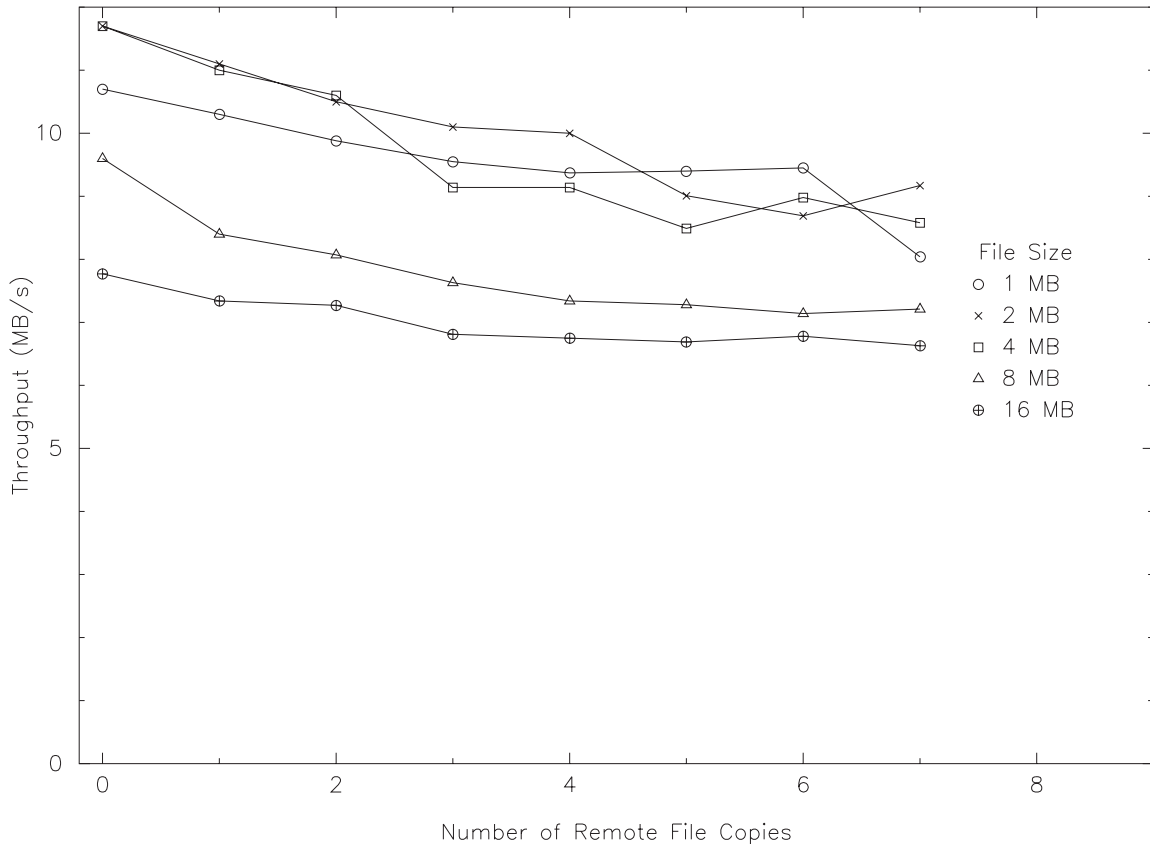


Figure 4. 100 Mbps Channel Bonded Network Throughput During File Copy

4. Meshed Network Topologies

Network performance experiments using the first and second generation Beowulf clusters demonstrated that the use of channel bonded 100 Mbps Ethernet is sufficient to provide a communications backbone for parallel computation. However, the research team also realized that the use of more complex network topologies had the potential for further improving network performance. This assumption was based on the fact that in a multiple-access Ethernet network, only one packet can be transferred over a link at a time. Channel bonding theoretically allowed the number of simultaneous packet transmissions to be equal to the number of network channels available. However, the scalability of this approach is constrained by the maximum number of network adapters that can be installed in any one machine, and the maximum number of nodes that can be supported on any one link. A solution to this problem involves breaking the network into a set of interconnected segments.

Using the computer hardware from the first generation Beowulf cluster, two different segmented network architectures were implemented using 10 Mbps Ethernet. Figure 5 illustrates a software routed mesh architecture in which each cluster node has access to two separate network links. Any node in the cluster can communicate directly with six other nodes in the cluster that share the two common links. The six nodes that are directly accessible over these two links are called local nodes. For a node to communicate with the remaining nine nodes in the cluster, an intermediate node, which is local to both the sender and receiver, must act as a software router to forward the data packet between network links. Any two nodes can communicate using no more than one intermediate router node. The nine nodes that must be reached through a router node are called remote nodes.

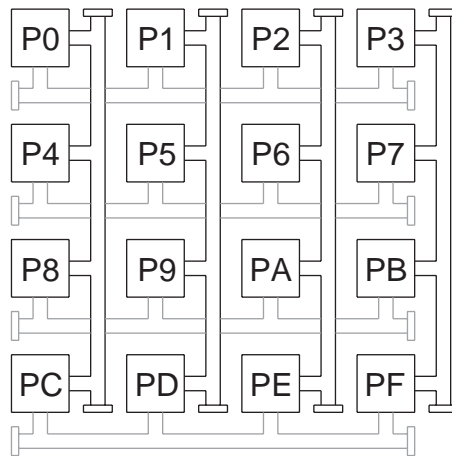


Figure 5. Software Routed Mesh Architecture

The software routed mesh architecture has the advantage over a channel bonded network of having no more than four nodes contending for the bandwidth of any one link. However, this decrease in bandwidth contention does come with a price. A routed packet is replicated on two separate links and thus consumes twice the aggregate bandwidth of non-routed packets. The process of routing a packet through a node consumes processor cycles and incurs an additional latency penalty on the transmission time. However, this cost is minimized by using the IP packet forwarding feature included in the Linux kernel.

The second segmented network architecture implemented using 10 Mbps Ethernet was a switch routed mesh architecture illustrated in Figure 6. This network functions identically to the software routed mesh architecture except that all routing to foreign links is handled by two four-port Ethernet switches. This architecture has the same advantages and disadvantages as the software routed mesh architecture except that no CPU cycles are lost to routing, and the additional latency in transmission is introduced by the Ethernet switch, not by a routing node.

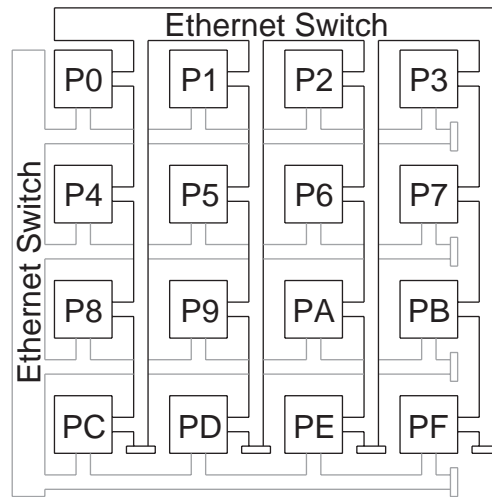


Figure 6. Switch Routed Mesh Architecture

As was done with the channel bonded networks, the routed mesh networks were evaluated using the token exchange and the file copy experiments. All the parameters for the token exchange experiment were the same except that throughput results were only collected for the test case involving 7 sending/receiving pairs. All the parameters for the file copy experiment were the same except the throughput results were only collected for the test case involving a 2 MB file size. The major difference in the way the routed mesh experiments were conducted involved the method for routing data packets between the sending and receiving pairs.

As a consequence of the mesh topology, there are always two possible routes between any two nodes in the cluster. One routing method used in the experiments utilized a static routing table to forward data packets to the proper network link along a predetermined path. The other routing method had the originating node alternately send data packets on one of the available links and then the other. The intermediate routing nodes in the software routed mesh and the Ethernet switches in the switch routed mesh always used a routing table to forward the packets along a predetermined path.

The results of the token exchange experiment using the 10 Mbps meshed Ethernet topologies are shown in Figure 7. The throughput graph summarizes the results for both channel bonded and mesh topologies. Single net refers to data transfers over a 1 channel multiple-access link connecting all 14 nodes in the cluster. Bonded dual net refers to data transfers over a channel bonded multiple-access link connecting all 14 nodes in the cluster. Switched mesh, local-net refers to data transfers between local nodes in a switch routed mesh architecture using static routing. Switch mesh, remote-net refers to data transfers between remote nodes in a switch routed mesh architecture using static routing. Routed mesh, local-net refers to data transfers between local nodes in a software routed mesh architecture using static routing. Routed mesh, remote-net refers to data transfers between remote nodes in a software routed mesh architecture using static

routing. Bonded switched mesh, local-net refers to data transfers between local nodes in a switch routed mesh architecture with an alternating routing path used by the originating node. Bonded switched mesh, remote-net refers to data transfers between remote nodes in a switch routed mesh architecture with an alternating routing path used by the originating node.

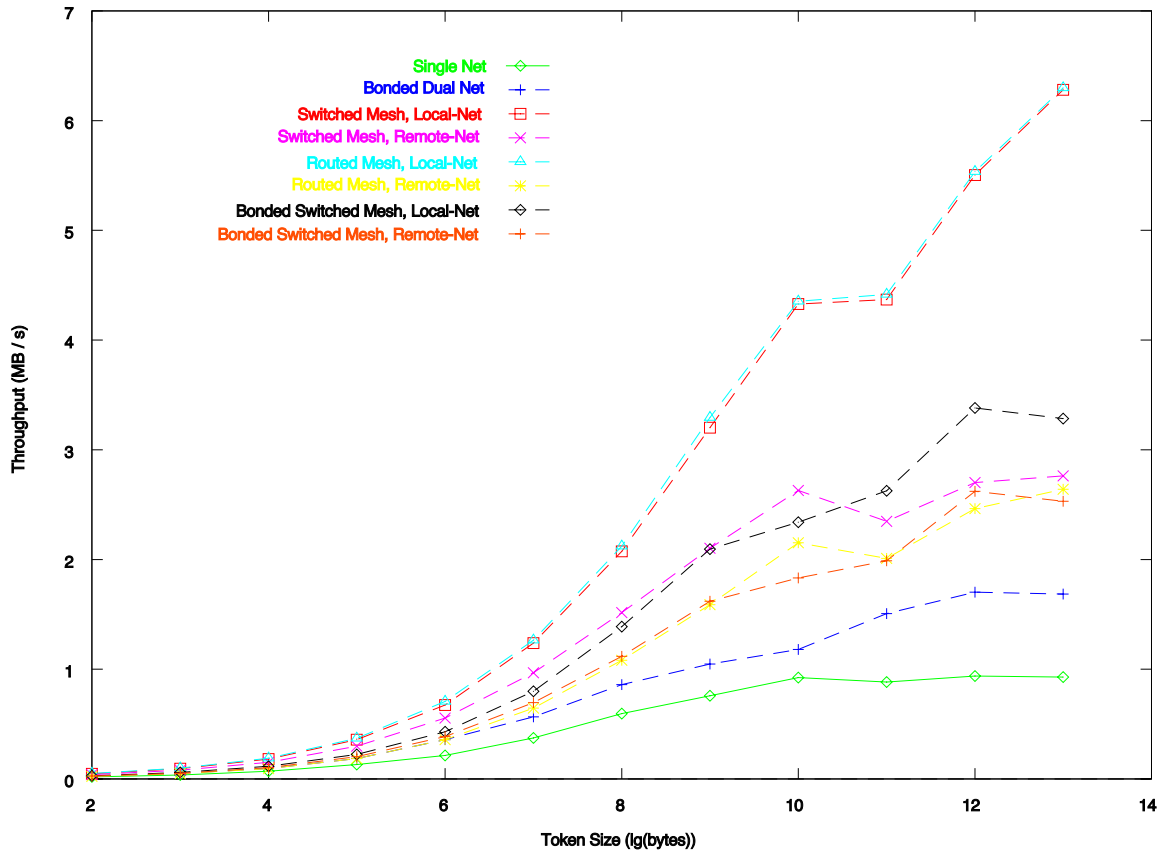


Figure 7. Meshed Network Throughput During Token Exchange

One of the most striking results from the token exchange experiment using the mesh topologies was the increase in throughput experienced for local node transfers. Although it is not surprising that the local node transfers yielded the highest throughput, what was unexpected was the improvement over the bonded dual net results. The peak throughput achieved for local node transfers in the mesh topologies was 6.3 MBps for an 8192 byte token. This is a factor of 3.7 improvement over the original channel bonded 10 Mbps Ethernet network. This result clearly demonstrates that by spreading network traffic over multiple independent Ethernet segments, the level of contention can be reduced and the bandwidth utilization can be greatly improved.

While the throughput for local node transfers was virtually identical for both the software routed and switch routed networks, there was a significant difference in the performance of remote node transfers. Unlike the local node transfers which occurred on a common network segment, remote node transfers required that the data packet be forwarded through a router. The process of storing and forwarding a packet, whether done by a routing node or by a dedicated

switch, incurs a significant penalty to the throughput of the network. The throughput differs by a factor of about 2.3 between local and remote transfers of an 8192 byte token for the switch routed network. This difference is somewhat greater for the software routed network which suffers from a higher latency.

The use of a bonded switched mesh network was explored to determine if any performance improvement would result from alternating the path used by the sending node to transmit data packets. This approach was an attempt to meld the benefits experienced with channel bonding onto the routed mesh topologies. For local node transfers, the bonded switched mesh network performed better than the bonded dual net network due to a reduction in network contention resulting from multiple independent Ethernet segments. However, when compared to the switched mesh network, the bonded switch mesh network yielded a much lower throughput. This results from the sending node alternately placing data packets on a local network segment and then a remote network segment. Packets placed on the remote network segment must be routed back to the local segment via the Ethernet switch. This longer route combined with the latency associated with routing serves to degrade the overall throughput.

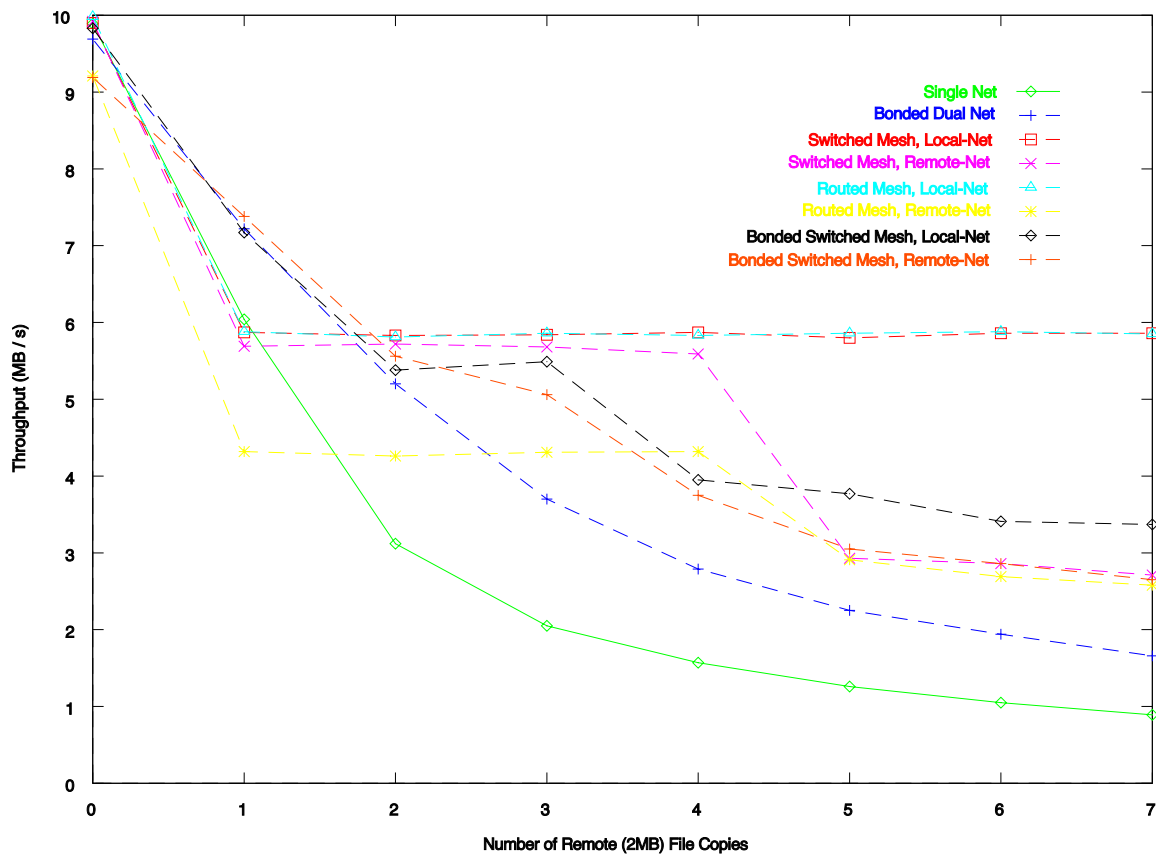


Figure 8. Meshed Network Throughput During File Copy

When applied to remote node transfers, the bonded switched mesh network’s performance was marginally lower than the switched mesh network’s. The theory behind the bonded

approach suggested that distributing data packets over two different network segments would tend to balance out the overall data traffic. However, in practice, the overhead and latency associated with the additional packet routing resulted in a net decrease in throughput. This was only somewhat offset at higher traffic rates where the two configuration's performance were more closely matched.

The results of the file copying experiment using the 10 Mbps meshed Ethernet topologies are shown in Figure 8. The labeling of the throughput curves is the same as described for Figure 7.

The file copying experiment serves to reinforce the same basic conclusions drawn from the token exchange experiment. The only major difference relates to the relative performance of the software routed mesh network to the switch routed mesh network when performing a remote node transfer. Due to the additional overhead and latency involved in routing packets through a processing node, the software routed network performed significantly poorer than the switch routed network when there were four or less remote copies. However, as the number of remote copies is increased above four, the increased traffic density causes contention for the available bandwidth of the Ethernet switches. This contention manifests itself in the degradation of the system throughput.

5. Conclusions

From its inception, cluster computing was designed to provide a low cost alternative to the supercomputer. This goal was made possible by capitalizing on the rapid increase in the computational power of personal computers over the last ten years. However, to effectively harness the combined parallel power of a cluster of personal computers requires the services of a high capacity communications network to perform the necessary interprocess data transfers. Whereas computer performance has continued to increase in accordance with Moore's Law, communication networks have seen a much more conservative increase in available bandwidth over the same period. Thus, the early history of cluster computing can be defined by the methods used to optimize the available network resources.

The first generation Beowulf cluster gave a hint of the computational performance that could be achieved using inexpensive commodity hardware. However, the use of a channel bonded 10 Mbps Ethernet proved to be largely inadequate for handling the required data traffic load of a 16 node cluster. In all but the most lightly loaded conditions, network contention resulted in a serialization of data traffic that served to greatly diminish the overall throughput. The use of channel bonding did help balance the traffic load across the two available Ethernet segments which resulted in a throughput improvement. However, the most significant problem encountered with this network design was the difference in throughput achieved between local and remote file transfers. In order to sustain the processing capability of the cluster, especially when using higher performance hardware, the available network bandwidth had to be improved.

The incorporation of a channel bonded 100 Mbps Ethernet network into the second generation Beowulf cluster solved many of the bandwidth problems experienced with the 10 Mbps system. The 100 Mbps network was capable of sustained data throughput even when heavily loaded. More importantly, the faster network significantly reduced the differential between local and remote file transfers. This improvement allowed the network to more closely emulate the functionality provided by a common backplane in a supercomputer.

Although the use of 100 Mbps Ethernet appeared to solve most of the communications problems posed by the clustered architecture, the results of the meshed topology experiments have more far-reaching ramifications. By breaking the network into a set of independent segments, data contention on any one segment was greatly diminished and thus the overall network throughput increased. The price paid for this improvement differed between the two mesh topologies. For the software routed mesh network, the price came in the form of the CPU overhead necessary to route packets between network segments. For the switch routed mesh network, the price paid was the cost of the additional Ethernet switches needed to route data packets. Obviously the software routed approach is preferred when system cost is the driving factor and the switch routed approach is preferred when performance is most important.

Looking beyond a 16 node cluster, the mesh topologies clearly provide the greater opportunity for scaling. The channel bonded architecture is limited in two important ways as the number of nodes in the cluster is increased. First, there is a maximum number of nodes that can be placed on a single Ethernet link. Second, as more nodes are added to a link, the contention for the bandwidth increases resulting in an overall degradation in network performance. Both of these problems can be easily addressed using a mesh topology. Instead of looking at cluster growth as the addition of nodes to individual links, it should instead be looked at as the addition of new meshes to an existing network. By providing a separate high-speed communications link between meshes, the cluster can be theoretically scaled into the thousands of nodes. This is in fact the same approach used to scale all large networks.

The goal of cluster computing is to achieve supercomputer performance at personal computer prices. Although not fully realized, great strides have been made, in a very short time, in reaching this goal. As new computer hardware is introduced which helps narrow the gap in performance, the challenge will remain how to most efficiently tie the components together. The communications network of a computer cluster has, and will likely remain, the choke point for improving overall system performance. As was shown with the mesh topologies, the careful application of design principles can lead to efficient and cost-effective solutions to the communication demands of future computer cluster configurations.

6. References

1. Becker, D. J.; Sterling, T.; Savarese, D.; Dorband, J. E.; Ranawak, U. A.; Packer, C. V., "Beowulf: A Parallel Workstation for Scientific Computation" in *Proceedings International Conference on Parallel Processing*, 1995.
2. Becker, D. J.; Sterling, T.; Savarese, D.; Fryxell, B.; Olson, K., "Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation" in *Proceedings High Performance and Distributed Computing*, 1995.
3. Sterling, T.; Becker, D. J.; Savarese, D.; Berry, M. R.; Reschke, C., "Achieving a Balanced Low-Cost Architecture for Mass Storage Management through Multiple Fast Ethernet Channels on the Beowulf Parallel Workstation" in *Proceedings International Parallel Processing Symposium*, 1996.
4. Reschke, C.; Sterling, T.; Ridge, D.; Savarese, D.; Becker, D.; Merkey, P., "A Design Study of Alternative Network Topologies for the Beowulf Parallel Workstation" in *Proceedings High Performance and Distributed Computing*, 1996.
5. Ridge, D.; Becker, D. J.; Merkey, P.; Sterling, T., "Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs" in *Proceedings IEEE Aerospace*, 1997.
6. Salmon, J.; Stein, C.; Sterling, T., "Scaling of Beowulf-Class Distributed Systems" in *Proceedings of ACM/IEEE Supercomputing 98, High Performance Networking and Computer Conference*, IEEE Computer Society, November 1998.
7. Savarese, D.; Sterling, T., "Beowulf" in *High Performance Cluster Computing, Volume 1*, (edited by) Rajkumar, B., Prentice Hall PTR, Upper Saddle River, N.J., pp. 625-645, 1999.