Scaling of Beowulf-class Distributed Systems

John Salmon Christopher Stein Thomas Sterling

Abstract

Beowulf-class systems employ inexpensive commodity processors, open source operating systems and communication libraries and commodity networking hardware to deliver supercomputer performance at the lowest possible price. Small to medium sized Beowulf systems are installed or planned at dozens of universities, laboratories and industrial sites around the world. The design space for larger systems, however, is largely unexplored.

We investigate two interconnection techniques that would allow the scaling of a Beowulf class system to unprecedented sizes (hundreds or thousands of processors). The first is a commercially available high-speed backplane with fast-ethernet and gigabit ethernet switch modules. The second is a network that combines point-to-point connections and commodity switches, employing the Beowulf nodes as IP routers as well as computational engines. The first method has the drawback of a relatively high price tag - a significant but not overwhelming fraction of the cost of the system must be devoted to the switch. The second approach is inexpensive, but introduces extra latency and may restrict bandwidth depending on the topology employed. Our tests are designed to quantify these effects.

We perform three sets of tests. Low-level tests measure the hardware performance of the individual components, i.e., bandwidths and latencies through various combinations of network interface cards, node-based routers and switches. System-level tests measure the bisection bandwidth and the ability of the system to communicate and compute simultaneously, under conditions designed to generate contention and heavy loads. Finally, application benchmarks measure the performance delivered to a subset of the NAS parallel benchmark suite.

Low-level measurements indicate that latencies are only weakly affected by introduction of node-based routers, and that bandwidths are completely unaffected. Switches introduce between 19 and 30 microseconds of latency and linux-based routers introduce 75 microseconds.

System level measurements indicate that even under load, the routed network delivers a bisection bandwidth near what one would expect by simply summing the capacities of the individual components. A very large scatter is introduced into latencies, however.

Application level measurements reveal that the routed network is acceptable for some applications, but can have significant deleterious effect on others. Successful use of routed networks for very large beowulf systems will requires a deeper understanding of this issue.

1 Introduction

In the last few years, clusters of PCs including Beowulf-class systems have emerged as an important and rapidly expanding approach to parallel computing. Beowulf-class systems [3, 4] are a natural consequence of several trends in computing: powerful and inexpensive mass-market microprocessors; powerful, robust and freely available operating systems; widespread adoption of message-passing APIs for engineering and scientific supercomputing applications; and high-speed, low-cost networking. Price-performance ratios for sustained execution has been achieved at $15 per Mflops for favorable (but non-trivial) scientific and engineering applications. LINPACK performance of over 20 Gflops has been reported, and over 10 Gflops has been sustained for scientific applications. Beowulf-class and other forms of clustered PC systems are becoming established as a new platform for accomplishing real-world moderate-scale (a few Gigaflops) computation in science, engineering, and commerce.

Beowulf-class systems are on the threshold of becoming supercomputers, but systems larger than a few dozen processors are still experimental. The future of Beowulf-class PC clusters for large-scale applications will will be determined by three issues: 1) architectural scalability, 2) development of latency tolerant applications, and 3) software environments and tools. This paper focuses on the first of these. Despite the success of many small (4 to 48 processor) Beowulfs, few systems of more than 100 processors have been installed and the design of such systems is largely unexplored. That such machines can be constructed and effectively applied using low cost COTS technology has yet to be established.

We consider the scalability of system designs capable of integrating hundreds or thousands of processors. We study the systems at the component level, the system level, and from the perspective of overall application performance. Both low-level synthetic benchmarks and real-world application benchmarks are employed to characterize the network components (hardware and software) and the overall system scaling characteristics. The purpose of this paper is to take an initial look at possible implementation strategies for large-scale Beowulf-class systems.

Beowulf designs are necessarily highly restricted. The individual components are constrained by the characteristics of available commodity networking systems. The particular choices and made here are driven by the characteristics of fast ethernet based technologies which has been widely adopted systems due to its excellent cost and adequate performance.

2 Practical Scalable Networks

The ultimate network would offer high performance and reliability, be inexpensive, and would be scalable at linearly increasing cost up to thousands of processors without significant degradation in performance. Unfortunately, such a network is not likely to exist any time soon. Fast ethernet constitutes a significant compromise, with its greatest advantage being low cost and its greatest drawbacks being high latency, modest bandwidth and limited scalability.

It is not appropriate to include a detailed tutorial on fast ethernet technology here. However, a few details are crucial to an understanding of the networks we propose below. We refer to the computational elements in our system as *nodes*. Practically speaking, a node is a single motherboard/memory bus, which may have one or more processors. Nodes are also equipped with one or more *Network Interface Controllers* (NICs). Good quality NICs cost less than $50 and can send and receive data simultaneously at 100Mbit/s with end-to-end latencies in the neighborhood of 100-200 microseconds.
Nodes can be connected in three ways:

point-to-point An inexpensive cable simply connects the ports on two NICs.

hub An inexpensive component that serves as a *shared* medium and connects several NICs. Any two NICs connected by a hub may communicate, but they compete for a common 100Mbps medium.

switched Switches are similar to hubs, but allow more than one pair at a time to communicate without interference. A wide variety of switches is commercially available, in sizes ranging up to at least 120 ports. We will denote the number of ports on a switch as S. The price-per-port of ethernet switches increases dramatically with increasing S. Small switches (48 or fewer ports) cost between $100-$200 per port. Larger switches, constructed using expensive backplanes and pluggable modules, cost considerably more per port.

Switches are the de facto standard connection component for small Beowulf systems. Although switches can be connected to one another with little or no difficulty, one of the architectural restrictions of ethernet is that the resulting network may not contain any closed loops. This severely limits our ability to construct a network purely from interconnected switches. A possible configuration is shown in Figure 1. This is the least expensive configuration capable of scaling Beowulf systems up to multiples of the size of an individual switch. For the cost of L switches with P ports, and one root switch with L ports one can interconnect up to L*P nodes in this way, but the the per-processor bisection bandwidth through the top switch is u/P, where u is the uplink bandwidth.

One possible solution to this problem is to use a higher bandwidth technology for the links between switches. In fact, several switches available today offer a small number (typically one or two) of “Gigabit uplinks” in addition to the standard 100Mbps ports. Gigabit ethernet (1000baseT) is still a very new technology, but there is every reason to believe that ever-larger gigabit switches will become available as moderate-cost commodity components in coming years. This might seem to suggest that 100Mbit switches with gigabit uplinks offer a path to scalable Beowulf systems. Unfortunately, in the same time frame, processor performance is likely to increase as well, and we will likely begin to exploit gigabit NICs and switches at the nodes of future Beowulf systems. At that point, the gigabit uplinks offer no advantage, and they will quickly be saturated by even a small fraction of non-local communication.

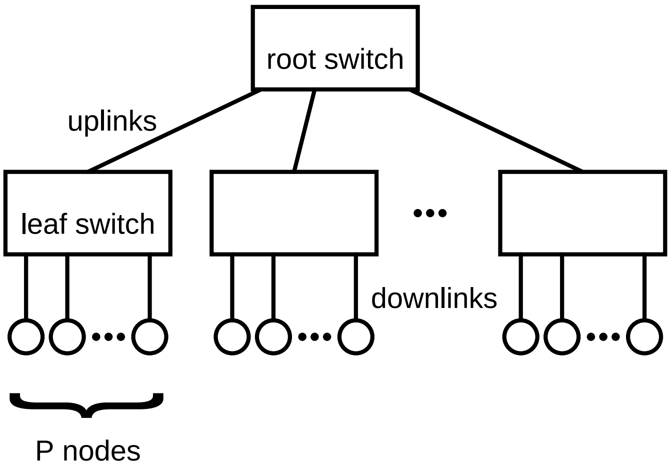

Figure 1: Tree of switches configuration showing a root switch connecting L leaf switches, each of which also links P nodes. In principle, the uplinks can use a different, higher-speed technology than the downlinks. At the moment, 100Mbit downlinks with Gigabit uplinks are becoming practical, but it is likely that we will see Gigabit ethernet in the downlinks as well in the not-too-distant future.

2.1 Routed Networks

The second way to overcome the limitation on closed loops is to do routing. If a "router" is interposed between two switches, they can be isolated from one another, and not run afoul of the "spanning tree algorithm" that effectively enforces the no-closed-loops rule. Dedicated, standalone routers can be expensive, but it is also possible to use a coventional computer, i.e., a node, as a router. That is, the nodes of the Beowulf themselves can act as routers in the network. One simply installs more than one NIC in each routing node and the OS can be directed to forward messages by tranferring data between NICs. Routing entails software overhead which is manifested as an additional latency associated with multi-hop messages. It may also impact the rate at which intermediate nodes carry out user-level computations.

The ability to route between NICs on the same node opens an enormous class of network topologies for consideration. We abstract some key features of these topologies as follows. First, we assume that the each switch has P ports. We note that NICs are much less expensive than switch ports (by a factor of three or more), so we restrict our attention to networks in which each node has only one NIC connected to a switch and C NICs connected directly to other NICs/processors using point-to-point links. The more general case would allow more than one NIC per processor to be connected to switch ports, perhaps on different switches, and offers even greater flexibility at some increase in cost.

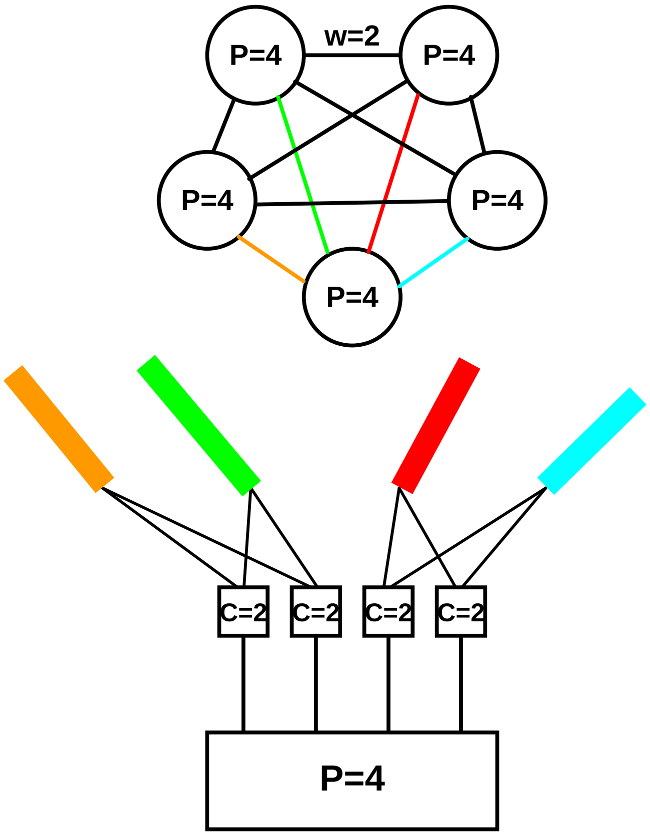

Figure 2: The upper figure is a graph representing a routed network consisting of 20 nodes arranged in L=5 groups of P=4 nodes each. The lower figure shows detail of one of the groups. Each node has C=2 NICs available to communicate with other groups, so the weight of each edge in the graph is P*C/(L-1)=2, i.e., there are 2 channels available between any two groups. The bisection bandwidth implied by the graph is obtained by noting that any bisection cuts four links, each of weight 2, and also cuts a single vertex in half (with internal bandwidth of 4 connections). Therefore, the per-node bisection bandwidth is: (4*2 + 4)/20 = 3/5. So the bisection bandwidth available to each node is slightly more than half that of a single dedicated, unshared connection.

Such networks can be further abstracted. Notice that each switch effectively defines a meta-node in the form of a "traditional" Beowulf system, i.e., a switch and a collection of P nodes. In addition to the normal internal connections within this meta-node, there are also P*C external connections which connect meta-nodes to one another. If we neglect differences that arise from the choices of exactly which nodes are connected between meta-nodes, we can describe the general case as a weighted graph consisting of L vertices (corresponding to meta-nodes), with the edges weighted so that the sum of the weights incident on each vertex is equal to P*C. That is, the weight of an edge corresponds to the number of point-to-point links that exist between the vertices (meta-nodes) connected by the edge. The total number of nodes in the system is, of course, N=L*P. A fully connected routed network with P=4, L=5, C=2 and N=20 nodes, is illustrated in Figure 2.

It is interesting to examine the diameter and bisection bandwidth implied by such a network. The *diameter* is the maximum number of edges between any two vertices. It is related to the maximum latency experienced by messages traversing the system. Such messages will be routed from vertex to vertex, incurring additional latency at each hop. Therefore, low diameter networks are probably preferable. Fortunately, the degree of the vertices is rather large, so there is little practical value in studying graphs with diameter greater than one. Communication networks are also characterized by their *bisection bandwidth*, i.e., the maximum bandwidth available between arbitrarily chosen halves of the system. The correspondence between the weight of each edge of the graph, and the number of individual network connections linking its vertices, allows us to compute the theoretical bisection bandwidth by summing the weights of edges.

The simplest graphs to consider are homogeneous and *fully connected*, i.e., every meta-node is connected to every other one by an equal number of point-to-point links. The weight of each edge is then P*C/(L-1), and there are $(L/2)^2$ such edges crossing any bisection (for even L). Therefore, the bisection bandwidth per node is (C/4)(L/(L-1)). The diameter, of course, is one. With C=3, the bisection bandwidth per node is over 75% of the bandwidth of a single point-to-point connection, suggesting that bandwidth-limited applications may be able to scale up from smaller Beowulf systems with only moderate degradation in performance.

In a fully connected network L cannot be larger than (P*C+1), so the overall size of the system is limited to N < P(P*C+1) - theoretically scalable up to thousands of processors with currently available commodity components. Of course, the are a host of other problems that bear on the viability of such large systems, but at least they are buildable in principle.

3 Measurements

We report three classes of measurements designed to evaluate network choices from the perspective of elementary building block performance, aggregate network performance, and application performance.

3.1 Elementary network components

We have run a series of low-level tests designed to measure latency and bandwidth in switched networks of commodity components. In particular, we wish to measure the latency and bandwidth of communication through a variety of paths routed through multiple switches and nodes. We may expect that the latency will grow as more intermediate elements are introduced, but the rate of growth needs to be established. We also expect that the bandwidth will be insensitive to intermediate elements, but this too needs to be quantitatively established.

We have constructed a small set of elementary mini-networks and run a selection of the `lmbench`[2]. suite of networking tests on them. In particular, we have measured TCP and UDP latencies and socket bandwidths

Table 1 shows results from `lmbench`. These network configurations were chosen because they constitute elementary components of the larger point-to-point networks under consideration. We also measured the performance of one and two Lucent (formerly Prominet) P-550 Cajun switches with a maximum internal switching capacity of 22.88Gbit/s, and six available modular slots. We used modules with 20 switched fast ethernet ports and linked the two Cajun switches together via trunked gigabit ethernet modules. In each test, the network was unloaded except for the processors involved in the test. Thus, these tests reveal the best-case behavior when there is no contention for resources. The data in the table are closely fit by the following rules:

$$\begin{aligned}
\text{UDP Latency} &= 135\mu\text{s} + 19\mu\text{s}(\#\text{ Bay Networks}) + 70\mu\text{s}(\#\text{ Linux routers}) \\
\text{TCP Latency} &= \text{UDP Latency} + 19\mu\text{s} \\
\text{Bandwidth} &= 11.38\text{MByte/s}
\end{aligned}$$

A single Lucent/Prominet switch introduces an additional latency of 35 μswhile a second one adds only 5 μsmore.

3.2 Aggregate network measurements

At least two factors undermine our ability to predict application performance from the latency and bandwidths reported by lmbench. First, one must account for additional layers of overhead imposed by application level communication software, e.g., MPI. Second, contention for resources can result in significantly degraded performance when the system is run under load.

3.2.1 Code for synthetic load generation

We have implemented a heavily instrumented synthetic load generator in C with MPI to attempt to assess these two effects. The code is fairly compact, and should compile without difficult on any ANSI C, MPI platform. It is included as an appendix to this paper for those who wish to try it themselves, or those who wish to know exactly what we are reporting in this section. It carries out a mixture of floating-point "calculation" and network communication, and

Connection	TCP Latency μs	UDP Latency μs	Socket Bandwidth MByte/s
e-e	154	118	11.39
e-L-e	214	176	11.39
e-L-L-e	287	240	11.38
e-B-e	173	137	11.38
e-L-B-e	232	197	11.38
e-B-L-L-B-e	324	281	11.38
e-B-B-e	192	156	11.39
e-B-B-B-e	210	175	11.37
e-P-e	189	142	11.32
e-P-P-e	194	154	11.39

Table 1: Latencies and bandwidths reported by `lmbench`. Connection types are abbreviated as follows: e=communication end-point, a 200Mhz PentiumPro running Linux, L=Linux kernel-based IP routing, B=Bay Networks 350T switch, P=Lucent/Prominet Cajun P-550 switch. The link between Lucent/Prominet switches was implemented by trunking together two 1000baseT links provided gigabit ethernet modules.

uses the PentiumPro hardware cycle counters (when available) to measure time intervals with very low granularity (a few cycles). The communication pattern can be fixed (processors communicate only with one chosen neighbor) or random (a random list of partners is drawn up before measurements commence, and communication cycles through the list). Message lengths and the lengths of intervals of floating point computation are set on the command line. When a range is chosen, the values are uniformly distributed in logarithm, so that plots can effectively cover a wide dynamic range. Our experience has shown that the distribution of times can be highly irregular, so that simple statistical aggregations like arithmetic means, linear regressions, etc. can be grossly misleading, or can obscure significant features of the true behavior of the system. Therefore, the program itself simply reports each time interval it measures, and it is the responsibility of post-processing to plot or otherwise characterize the results.

The figures in this section are all scatterplots of time vs. message length for 9000 "ping-pong" message transfers carried out between 36 processes in one of our test configurations. Each process sent and received approximately 250 messages of varying sizes. For message traffic, the time displayed on the vertical axis is actually half of the out/back time measured by the code, i.e., it is an estimate of the time required for a unidirectional message of the given length. All processes were running simultaneously, subject to our ability to ensure this by starting measurements immediately after a call to MPI_Barrier.

3.2.2 Hardware testbed

Our testbed system consists of 36 nodes. Each node has 3 fast ethernet interface cards. The first interface (eth0) connects a node to a Cajun switch. The second interface (eth1) connects to a port on a 16 port Bay Networks 350T switch. The 36 nodes are distributed evenly, i.e., P=9, across L=4 bay switches. The third interface (eth2) connects a node directly to another node's eth2 over a point to point link, i.e., C=1.

There are 18 point to point eth2 connections. Nodes are grouped together into switch groups 1 - 4 according to the bay switch they are connected to. Each group has 3 nodes connected by point to point eth2 to 3 nodes on every other switch group. On any given bay switch L and remote switch R there are six nodes on L not connected to a node on R by eth2. These use a connected node on L that is connected to R as a gateway to all other nodes on R. The gatewaying is distributed across the connected nodes by source, so that each of the three connected nodes gateways for two of the non-connected nodes. From source to destination, an IP packet will traverse at most 2 gateways. Of the 35 remote nodes in the system, 9 are reachable directly in one hop, 14 are reachable in two hops, and 12 are reachable in three hops.

A script configures the kernel IP routing table. The correctness of the routing was verified with a sequence of traceroute tests and analysis of the routing table.

We used two basic configurations to study two different classes of networks:

- The routed network described above with C=1 point-to-point NIC per node, L=4 meta-nodes, each consisting of a Bay Networks 350T switch and P=12 Pentium Pro nodes.
- A network consisting of two Lucent/Prominet Cajun P-550 switches connected by a pair of trunked 1000baseT fiber links. We operated this system in two configurations - one in which all 36 nodes were connected to the same switch and the other in which 18 nodes were connected to one switch and 18 were connected to the other.

3.2.3 Results

Figure 3 shows MPI performance with two different fixed patterns of communication using the Cajun switches – one in which all communication takes place within a single switch, and the other in which all communication takes place over the gigabit fiber links connecting two switches. It is reassuring that the two configurations appear to be indistinguishable, confirming that the trunk connection between the two switches is not a bottleneck. In addition, this is our first indication that the conventional formula relating communication time to message length may be inadequate:

$$t_{comm} = t_{latency} + length/bandwidth. \quad (1)$$

The solid line in the figure plots Equation 1 with $t_{latency} = 260\,\mu$s and bandwidth=9.8MB/s. Clearly the line does not capture the nuances of the actual data. The features in the actual data at multiples of 1500 bytes are related to the ethernet MTU, i.e., the maximum amount of data that can be transmitted with a single ethernet frame. The feature at 16kbytes may be cache related.

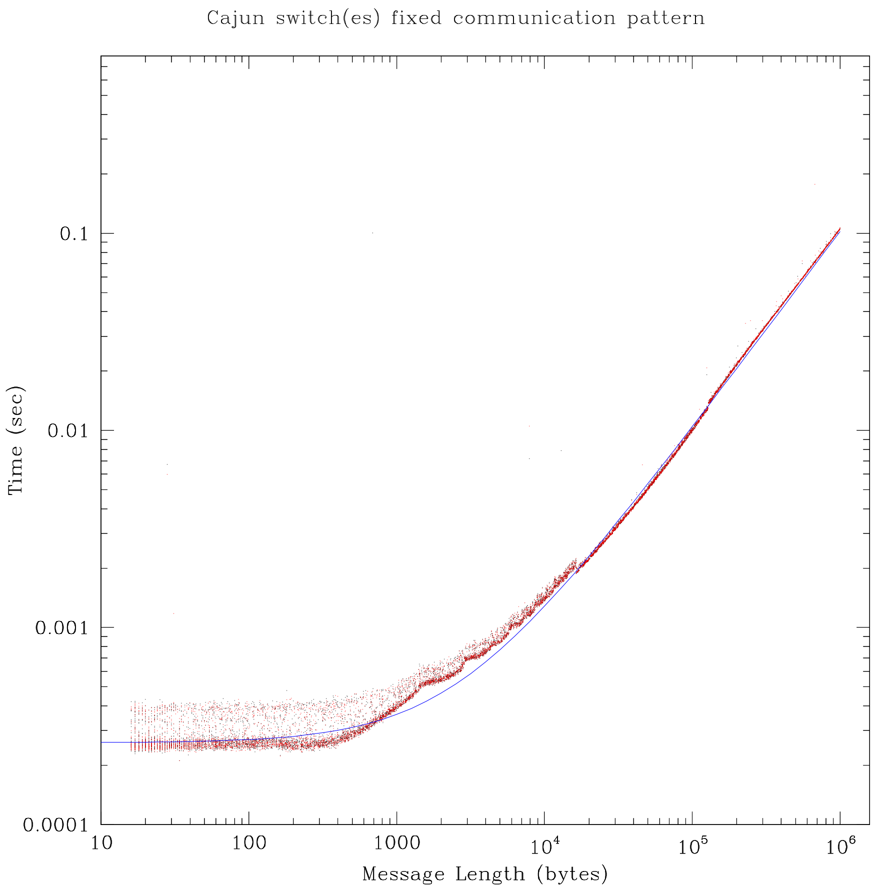

Figure 3: Time vs. message length for two different fixed communication patterns using the Cajun switches. Black dots represent a configuration in which all nodes are connected to the same switch. Red dots are for a configuration in which all nodes are paired up so that one member of the pair is on one switch and the other member of the pair is on the other switch. That is, all communication must take place over the trunked gigabit fiber connection between switches. The two colors of dots are virtually indistinguishable. The blue line shows the time predicted by Equation 1 with $t_{latency}$=260 μs and bandwidth=9.8MB/s.

While interesting, these features are probably not of great significance to overall application performance. Of much greater importance is the spread in latencies for short messages - frequently a factor of two or more, and the occasional outliers - above 0.1 seconds in a few cases.

Figure 4 shows the effect of using a random communication pattern. Again, we have two configurations - one in which all nodes are on the same switch, and the other in which the nodes are divided equally between switches and again, the point sets are indistinguishable. However, in this case, there is a new branch with latencies in the neighborhood of $10000\,\mu$s. In addition, there is much more scatter in the more common branch with latencies between 200 and $800\,\mu$s. Clearly, it would be misleading to characterize this data as being described by a single latency and bandwidth.

We now turn our attention to the routed networks discussed in Section 2.1. Figure 5 shows the results of four separate runs each using a different fixed communication pattern. The different communication patterns exercise the point-to-point links, the in-switch links, two-hop and three-hop virtual connections. The two-hop and three-hop connections correspond to the e-L-B-e and the e-B-L-L-B-e lines of Table 1. There are marked differences between the different classes of connection. Not only is the latency much higher for the multi-hop connections, but the bandwidth is also considerably reduced. Furthermore, there is considerably more spread in the actual times, for messages of all lengths. The reduction in bandwidth is not surprising, since we established in Section 2.1 that the maximum theoretical per-node bisection bandwidth for a network with L=4, C=1 is only 1/3 of a single 100Mbps link. We actually see performance of more than 33Mbps because our test code only transfers data in one direction at a time between any two nodes, while the hardware can transfer 100Mbps in both directions simultaneously.

Figure 6 shows the result of performing random communication on a routed network. The two colors show the behavior with and without interleaved floating point calculation. They are apparently indistinguishable, indicating that there is minimal degradation in communication performance when intermediate processors may be busy with other user tasks. Furthermore, results are very similar to Figure 5, and the four different classes of connections are clearly visible even though they have not been highlighted by color. Just as in Figure 4, there is a second branch with an apparent latency of $10000\,\mu$s and rare outliers which require more than 0.1sec. Since these features occur in both graphs, they are probably unrelated to the particular network, and are the result of interactions with other processes on the system, or subtle details of the MPI and TCP implementations.

It is not surprising that user-level computations do not impact communication. The converse is, unfortunately, not the case. Figure 7 shows the effect of routing on computational performance. The black dots represent the rate at which the processors evaluate the logistic function (a very tight, unrolled, pipelined loop, running at about 125Mflop/s) with no communication. The red dots show the effect of communication and routing. Computations will frequently be interrupted by up to $10\,\mu$s, and occasionally by as much as $100\,\mu$s.

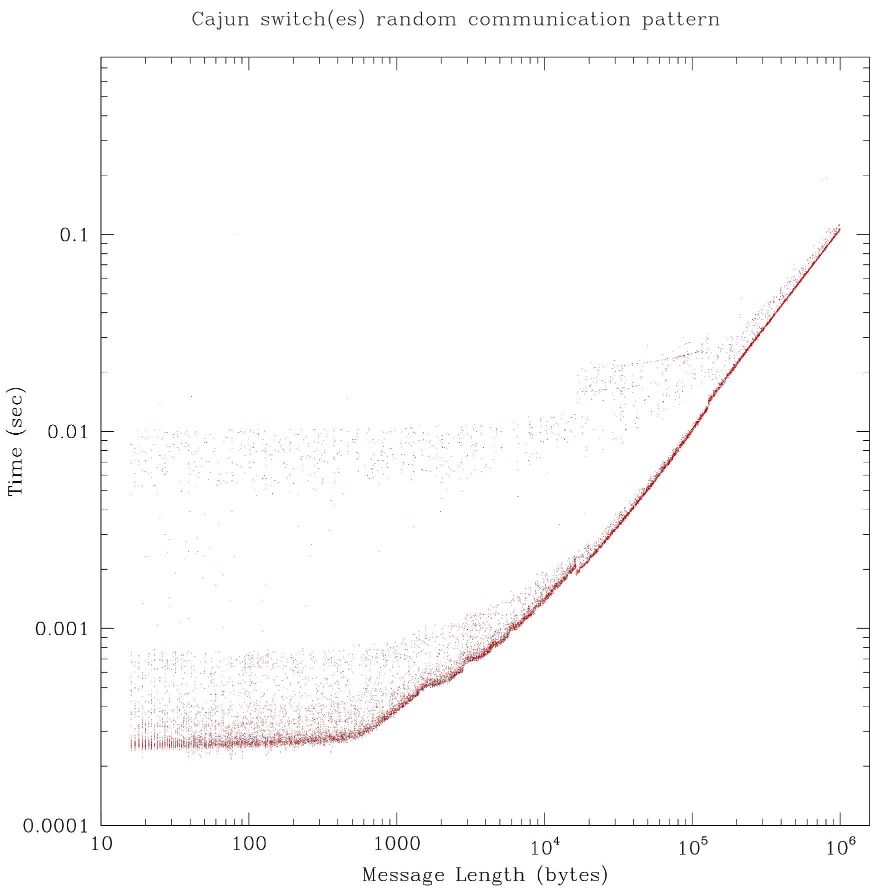

Figure 4: Time vs. message length for random communication patterns on two different network configurations. Black dots represent a configuration in which all nodes are connected to the same switch. Red dots are for a configuration in which nodes are split equally between two switches. The two colors of dots are virtually indistinguishable, but the branch of values with latencies approaching 0.01s is worrisome.

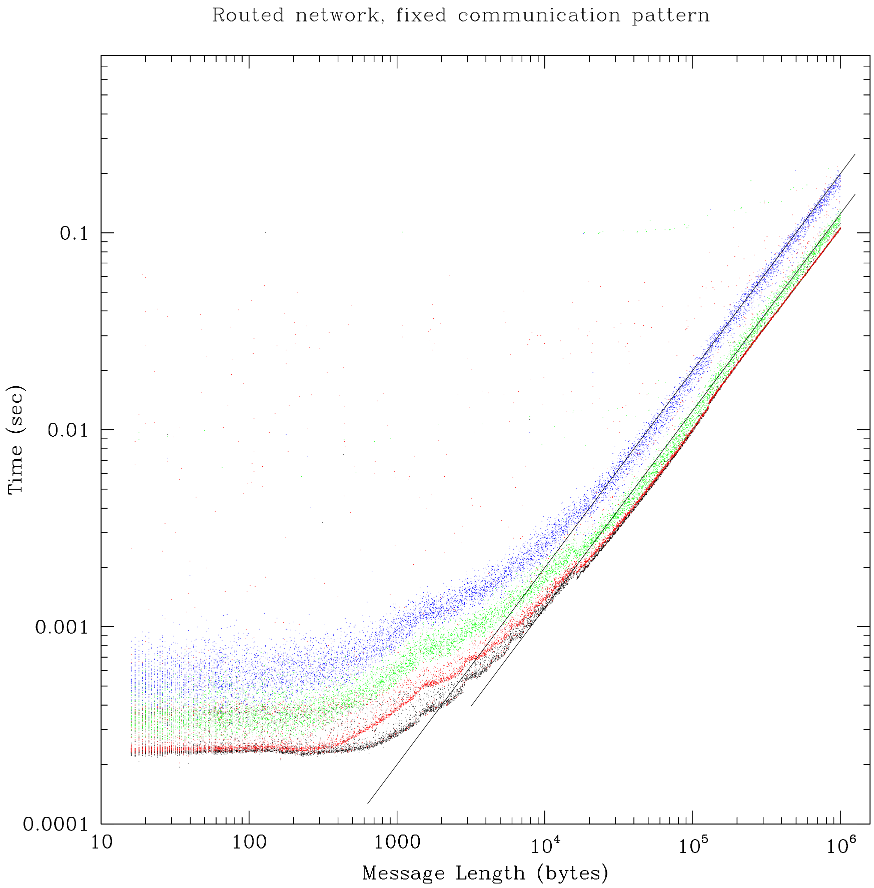

Figure 5: Time vs. message length for fixed communication patterns on the routed network. Four colors represent four different ways of pairing off the nodes. Black dots show times over the point-to-point links. Red dots show times between nodes on the same switch (since switches have an odd number of processors, there are also two point-to-point links included here). Green dots show times for nodes that are all exactly two hops apart (one point-to-point link, and one switch link). Finally, blue dots are for pairings involving maximally separated nodes, two switch links and one point-to-point link. The two solid lines correspond to bandwidths of 8MB/s (lower) and 5MB/s (upper).

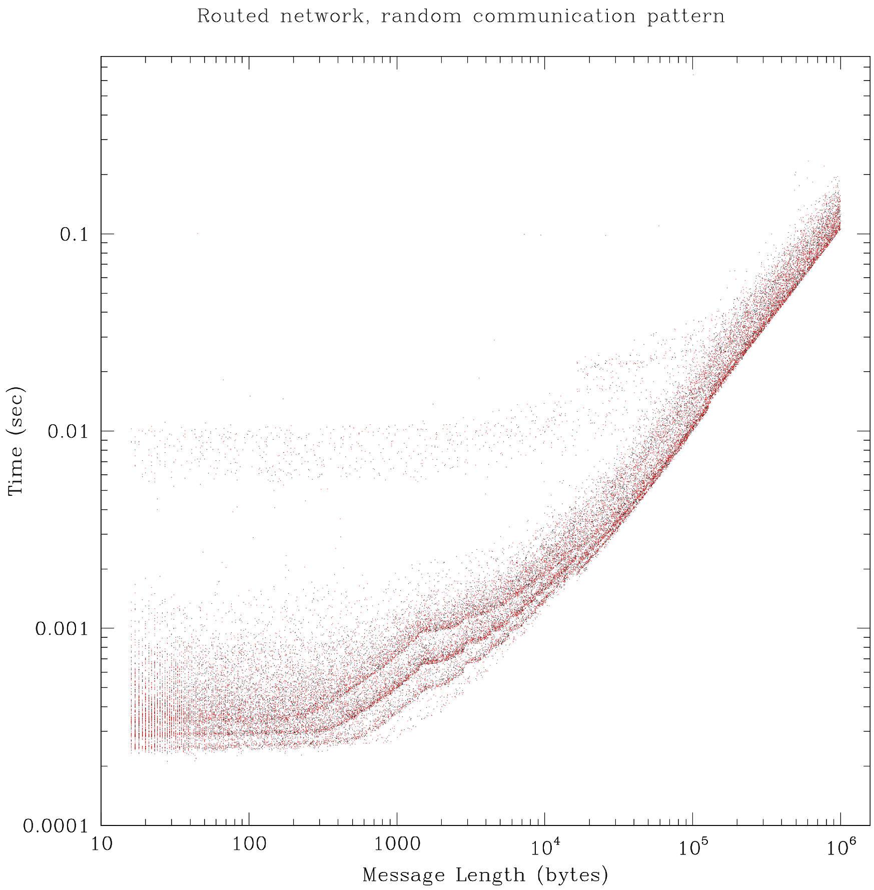

Figure 6: Time vs. message length for random communication patterns on the routed network. The colors represent times for a run that does nothing but communication (black) and a separate run (red) in which processors alternate between communication and floating point calculations.

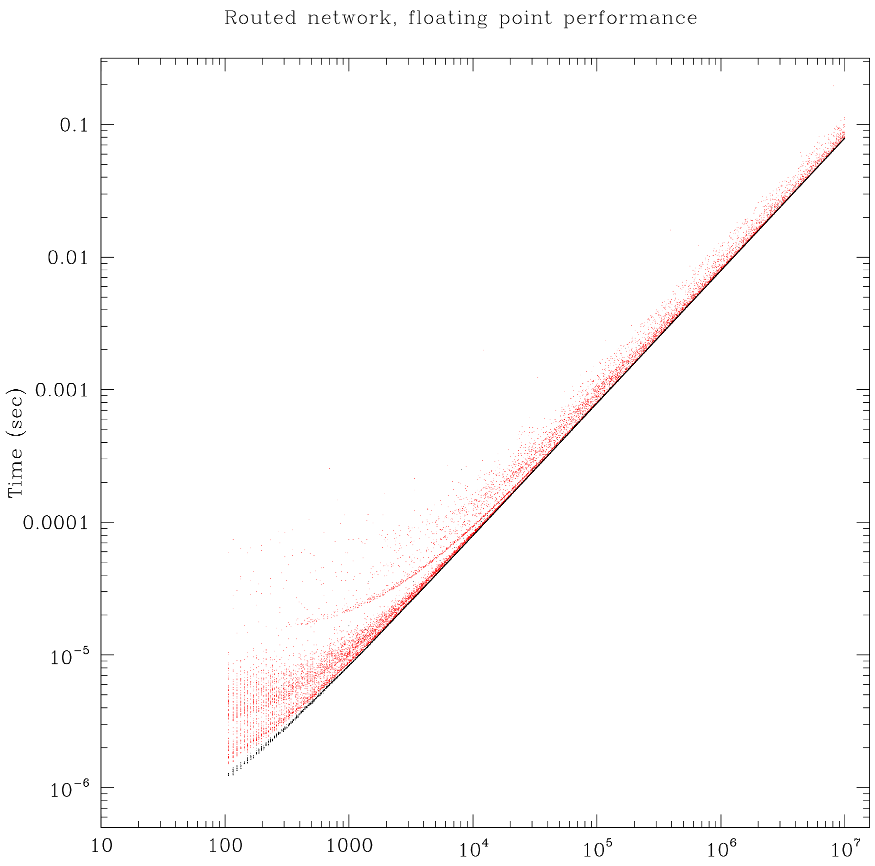

Figure 7: Time vs. number of floating point operations for routed networks with (red) and without (black) communication traffic.

test	class W NP=1	class A NP=16		class A NP=32	
		Cajun	Routed	Cajun	Routed
cg	19.95	7.94	5.04	5.74	2.97
ep	0.55	0.55	0.55	0.55	0.55
ft	27.17	11.20	9.19	8.46	7.59
is	3.14	x	0.11	0.33	x
lu	38.32	32.45	31.60	27.70	23.94
mg	21.06	16.46	15.21	13.91	x

Table 2: Results for selected NAS Parallel Benchmarks, version 2.3. All values are Mops/s/processor, as reported by the benchmark suite. Entries marked with an x did not complete for unknown reasons.

Long computations (0.1 sec) suffer a performance degradation of up to 30%, and the effect can be much worse on short bursts of computation.

3.3 Application level benchmarks

Finally, we ran several of the NAS parallel benchmarks [1] on our Cajun network and our routed network. The tests were performed with unmodified source code from version NPB2.3 obtained from the NAS web site. We used a commercial compiler from the Portland Group, pgf77 Rel 1.7-4 to compile the FORTRAN modules, and egcs snapshot egcs-19980803 was used to compile the C modules.

We were unable to run several codes to completion, suggesting that there may be lingering bugs in our system. Nevertheless, those results that were successfully verified are shown in Table 2. Note that no effort was made to map the calculations onto the hardware topology. The very dramatic differences in latencies and bandwidths across different communication paths (c.f. Figure 5) suggests that this could make a significant difference in the overall outcome. The results in Table 2 are encouraging in the sense that the routed network does not greatly underperform the more expensive Cajun network. However, it is clear that there is a real performance degradation even on relatively small networks.

4 Summary

We have examined possible architectural approaches to the design of Beowulf-class PC clustered systems with more processors than can be connected to a mass-market COTS switch. Two classes of networks were investigated in detail: trunked high bandwidth backplanes, and routed topologies. The former is expensive, but perhaps practical as costs are dropping rapidly due to market forces, while the latter is a low-cost alternative, and is ultimately probably necessary if one is to build systems much larger than the largest available switch.

The scalable networks proposed here deliver component-level latencies and bandwidths comparable to those seen in much smaller switched networks, and scale easily to several hundred nodes.

Contention for resources, both network and computation, introduces additional effects beyond the component-level measurements. Our results indicate that the trunked Cajun switches hold up very well under load, while the routed networks are more problematical. Under load, a large variance is introduced into the time required for message delivery. While most messages are delivered within a time consistent with the component-level measurements, a non-trivial number suffer a delay of $10000\,\mu$s or more, and there are even rare instances of delays approaching $0.1\,\mu$s. The resulting network performance is not adequately characterized by a single latency and bandwidth. There are important topological effects (some virtual connections have three times the average latency of others), as well as a large spread in observed message times, even between topologically equivalent pairs.

The effects of this complex network behavior on a small suite of application benchmarks reveals that there is a significant cost to using routed networks. The overall effect is different for each of the programs in the NAS benchmark suite, and in fact, there are probably still opportunities for improving the results by altering the mapping from logical connectivity (expressed in the software) to physical connectivity (expressed in the wiring of point-to-point links). The initial results here are grounds for cautious optimism that routed networks can be used for larger systems, but further study is required to determine if the observed problems can be reduced or overcome.

A C Implementation of Synthetic Load Generator

The measurement code used to generate the data in Section 3.2 is available electronically at http://www.cacr.caltech.edu/ johns/papers/sc98/.

The program is written in ANSI C with calls to MPI libraries. It should be reasonably portable, but it has only been compiled and run on PentiumPro Linux systems with egcs, mpich-1.1.1 and glibc.

References

[1] David Bailey, Tim Harris, William Saphir, Rob van der Wijngaart, Alex Woo, and Maurice Yarrow. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, Numerical Aerodynamic Simulation Facility, NASA Ames, 1995. http://science.nas.nasa.gov/Software/NPB/.

[2] Larry McVoy and Carl Staelin. lmbench: Portable tools for performance analysis. In *USENIX Winter Conference*, January 1996. http://www.usenix.org/publications/library/proceedings/sd96/mcvoy.html.

[3] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer". BEOWULF: A parallel workstation for scientific computation. In *International Conference on Scientific Computation*, 1995.

[4] Michael S. Warren, Timothy C. Germann, Peter S. Lomdahl, David M. Beazley, and John K. Salmon. Avalon: An Alpha/Linux cluster achieves 10 Gflops for $150k. In *Supercomputing '98*, Los Alamitos, 1998. IEEE Comp. Soc. (Elsewhere in this proceedings).