Cost/Performance Evaluation of Gigabit Ethernet and Myrinet as Cluster Interconnect

Helen Chen, Pete Wyckoff, and Katie Moor

Sandia National Laboratories
PO Box 969, MS9011
7011 East Avenue, Livermore, CA 94550

Abstract-- The idea of cluster computing is to aggregate machine room full of relatively cheap hardware, connected with some sort of network, and apply the combined power of the individual machines on a single calculation. Because this architecture consists of distributed memory, parallel processes communicate using a message-passing paradigm. High-end cluster users typically rely on special purpose hardware such as Myrinet [1], HiPPI [2], or ServerNet [3] for their message passing infrastructures, thereby losing the cost benefit offered by the commodity market. Riding on the wave of Ethernet popularity, Gigabit Ethernet [4] is fast becoming a commodity item. We evaluated its performance by simulating core algorithms from real parallel applications. We compared raw performance figures such as bandwidth and latency, as well as more complex parameters such as jitter, routing, and points of congestion in the fabric against similar studies conducted on Avici and Myrinet technology.

***Index terms*—Simulation, Gigabit Ethernet, Myrinet, topology**

A. INTRODUCTION

The number of enthusiasts of commodity high-performance computing platforms is growing following the general demise of the massively parallel processor (MPP) manufacturers. Sandia National Laboratories, a United States Department of Energy research facility, has also joined the fray because it can no longer satisfy the thirst for FLOPS by buying monolithic multi-million dollar machines, as there is not sufficient market demand to keep vendors in business.

The hardware employed in a cluster is generally the most readily available in the volume personal computing market, so as to leverage the cost advantages of buying commodity hardware. Unfortunately, a critical piece of the hardware for cluster computing, namely the highspeed interconnect, is completely irrelevant for the mass market. High-end cluster users, therefore, have to rely on special purpose hardware such as Myrinet, HiPPI, or ServerNet for their message passing infrastructures, thus losing the cost benefit offered by the volume market.

The purpose of this study is to identify a cost-effective alternative to traditional interconnects. Riding on the wave of Ethernet popularity, Gigabit Ethernet is fast becoming a commodity item. In addition to bandwidth enhancement, the full-duplex mode of Gigabit Ethernet [4] allows switched access at full channel capacity without the limitation of CSMA/CD. Therefore, we believe, given a scalable switching architecture, Gigabit Ethernet can be a cost-effective interconnection solution for cluster computing.

The remainder of this paper discusses the simulation results of a conventional Gigabit-Ethernet switch. We describe the methods we used, which involve a mix of "artificial" basic tests and simulations of core algorithms from real parallel applications. Results are compared with two identical studies conducted on the Avici Terabit Switch Router and the Myrinet solution. We then tally the positive and negative aspects of each technology.

B. INTERCONNECT TECHNOLOGIES

The following subsections describe the network fabrics we considered for the three simulation studies. In each subsection we calculate the current pricing for a prototypical 256-node cluster, a size which is feasible for most high-end cluster builders.

1. Conventional Gigabit Ethernet (GigE) Switch

Conventional GigE switches use designs based on a backplane bus or crossbar. The largest non-blocking switch available today supports only 64 nodes and therefore, cascading is required to build a cluster beyond that size. These switches use the spanning tree algorithm to calculate a loop-free tree that has only a single path for each destination, using the redundant paths as hot stand-by links [5]. This precludes the use of a mesh topology, making the network vulnerable to bottlenecks due to output port contention. Therefore we believe that interconnection fabrics using conventional GigE switches are limited to building small parallel systems of no more than 256 nodes.

Nevertheless, we decided to conduct a simulation study of a 256-node cluster in order to evaluate the effects of ethernet's packet framing, inter-frame gap, maximum and minimum packet size, and store-and-forward switching mechanism on the performance of parallel applications. Network cards for Gigabit Ethernet are approximately $700, and 64-port switches can be found for $50K. Including 20 extra fiber cables for inter-switch trunking, this configuration cost 256 x $700 + 5 x $50,000 + (256 + 4 x 6) x $75 = $450K.

2. Myrinet

Myricom's Myrinet is a cost-effective, special purpose communication and switching technology. It interconnects hosts and switches using 1.28 Gb/s full-duplex links. The Myrinet PCI host adapter can be programmed to interact directly with the host processors for low-latency communications and the network to send, receive, and buffer packets.

Work supported by the US Department of Energy under contract DE-AC04-94AL85000

All Myrinet packets carry a source-based routing header to provide intermediate switches with forwarding directions. The current Myrinet switch is a 16-port crossbar. These ports can be used to interconnect either switches or processors, allowing arbitrary network topologies. However, the severe cable length restriction of 35 feet impedes the realization of complex topologies. Thus, for our large-scale simulations, we chose a two dimensional torus, with dual interswitch connections, as the best tradeoff in terms of area and cost. Myrinet sells a network interface card for $1700, 16-port switches for $5000, and cables for $200. For the topology described above, the total cost for a 256-node cluster is 256 x $1700 + 32 x $5000 + 12 x 32 x $200 = $670K.

3. Avici Terabit Switch Router [6]

The Avici switch router uses two direct-connect networks [7] as its switching fabric to achieve high performance, scalability, and robustness. The dual fabric connects switching nodes (or line cards) using twelve 20 Gb/s full-duplex links to form two three-dimensional toroidal meshes. As such, the Avici switch router can be incrementally expanded to include up to 1120 cards without blocking. At 16 GigE ports per card, this configuration can interconnect up to 17,920 compute nodes.

Similar to Myrinet, the Avici switch uses wormhole routing inside the fabric to achieve low latency. Unlike Myrinet, however, rather than buffering the entire message inside the network (causing path blocking for the duration of the message), the Avici router segments its messages into 72-byte scheduling units (flits) and uses credit-based flow control to prevent flit loss. The Avici router eliminates the blocking characteristics of wormhole routing by using per-connection buffer management and over-provisioned fabric links. Because of the huge speed mismatch between Gigabit Ethernet and the fabric link (1:20), the Avici router stores each incoming Gigabit Ethernet packet before forwarding to prevent buffer underrun. Gigabit Ethernet cards cost around $700 each, and fiber cables $75 each. Avici projects a 256-port switch for $350,000. The total cost for a 256-node cluster is 256 x ($700 + $75) + $350,000 = $550K.

C. SIMULATION METHODOLOGY

Due to time constraints, we adapted existing simulation packages to capture the important characteristics of each technology, such as its link level protocol and switch architecture. Considering that these characteristics are unique to the particular technologies, we were not concerned with the effects due to differences in implementation. Instead, we ensured the fidelity of our simulation results by extending the packages to use the same set of parallel algorithms to generate traffic. We also coded an identical interface layer to handle details of packet transmission and reception. We used OPNET to simulate a 256-node cluster connected by conventional GigE switches [8]. These OPNET models illustrate the simulation methodology we adopted on all three technologies. We will compare the OPNET results with similar studies performed on the Avici and the Myrinet interconnects in a later section.

1. The OPNET Models

Using OPNET's network editor, we composed our computational cluster using components such as switches, nodes, and links. The interconnection network consisted of five conventional Gigabit Ethernet switches, 256 compute nodes, and full-duplex point-to-point links to interconnect them. We populated four switches each with 64 compute nodes, which were in turn connected via a fifth switch. We chose a star topology because it offered the lowest hop count between the most distant nodes in the network.

At the next lower level, we used OPNET's node editor to construct our compute and switch nodes. While the switch nodes were composed entirely of existing OPNET models, we wrote the traffic generators (i.e. parallel applications) and an Ethernet Transport Adaptation Layer (ETAL) to construct our compute nodes (Fig. 1).

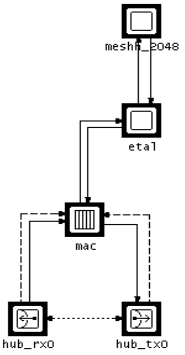

Figure 1. Compute Node model

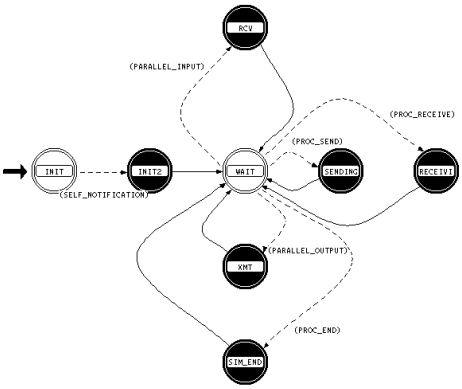

Figure 2. State diagram for the mesh algorithm

Our parallel application model contained a state transition diagram (Fig. 2) that represented the parallel code. Each of the 256 nodes in the system ran the same state machine, which transitioned between sending, receiving, and computing states.

We will defer the description of our parallel applications until the "Parallel Code Algorithms" subsection.

As mentioned earlier, the ETAL module provided application programmers with a uniform interface to the network; it handled the details of packet transmission and reception, where packets were segmented and reassembled when necessary in order to fit the Ethernet MTU (1500-byte) limitation. ETAL segmented the application's messages into Ethernet packets using the following OPNET kernel routines: *Op_sar_segbuf_create ()* allocated a segmentation-buffer; *Op_sar_segbuf_pk_insert ()* tagged and inserted a message to be segmented into the pre-allocated segmentation-buffer; *Op_sar_srcbuf_seg_remove ()* subsequently removed a segment from the buffer suitable for transmission inside an Ethernet frame. At the receiving end, the ETAL reassembled the original message from its tagged Ethernet fragments. We created a reassembly-buffer using *op_sar_rsmbuf_create ()*. We then inserted incoming Ethernet packets into the pre-allocated reassembly-buffer using *op_sar_rsmbuf_insert ()*. Finally, we detected and removed a completely reassembled messages from the reassembly-buffer using *op_sar_rsmbuf_pk_count ()* and *op_sar_rsmbuf_pk_remove ()* respectively.

The ETAL module also kept track of in-flight messages and their creation time, which we used to generate performance statistics. We logged statistics using identical "printf" formats for all three simulation studies, such that we could use the same set of parsing tools to deduce and plot statistics from the output of the OPNET as well as the Avici and Myrinet simulation runs.

2. Parallel Code Algorithms

Accurate characterization of network performance is a complex task. Simple numbers such as minimum latency or maximum bandwidth are not sufficient metrics to enable cross-technology comparisons. We augmented these basic numbers with results from computational core algorithms from real parallel codes.

A code entitled *token_pass* was our simplest test. It arranged the participating processors in a virtual loop that iterated the passing of a "token" around the loop a certain number of times. Each processor awaited a message from its neighbor to the left, then delayed a bit to simulate processing time, before sending a message to its neighbor on the right. By increasing the size of the token, we can perform accurate bandwidth measurements. By setting the payload to zero, we can find the minimum message latency. Since only two processors at a time were ever involved in a communication, there were no contention effects to filter out from the results.

The code *mesh* simulated a computational kernel from a two-dimensional finite element calculation. This class of structured grid codes is very common among the large-scale calculations being performed today at the laboratories. The processors are laid out in a virtual two-dimensional mesh, where each processor communicates with its immediate neighbors in both the *x* and *y* directions. The code performed a number of iterations of computation and communication cycles, which represented the real code's explicit time stepping algorithm as it solved a generalized partial differential equation.

3. Host Overhead

End-to-end latency in passing a point-to-point message includes: the processing overhead of some message passing interface (e.g., MPI), the processing of the operating system's transport/network protocols and their associated memory copies, the network interface card's medium access overhead, and the network's transmission delay. We combined all but the network transmission delay into the host overhead abstraction. We wrote an MPI-based "ping-pong" program that measures one-way latency of various length messages. One node starts its timer, and sends a message to the other node, then receives a copy of the same message from the other node, and stops its timer. The recorded value (i.e. the one-way delay) is half of this round-trip timing period, and to ensure accuracy, 200 trials were averaged together.

Our simulator extrapolated one-way latency from the linearly fitted plots of the experimental measurements (Fig. 3). Software overhead was calculated by subtracting the transmission delay from the derived one-way latency. We further divided the results by two, to account for processing delay incurred at both the sender and the receiver, and these values were plotted in Fig. 4.

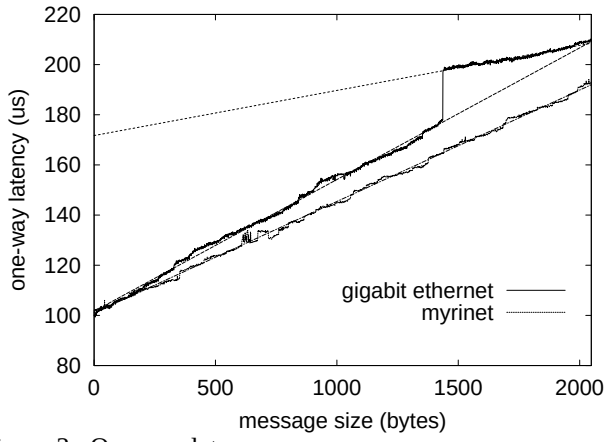

Figure 3. One way latency

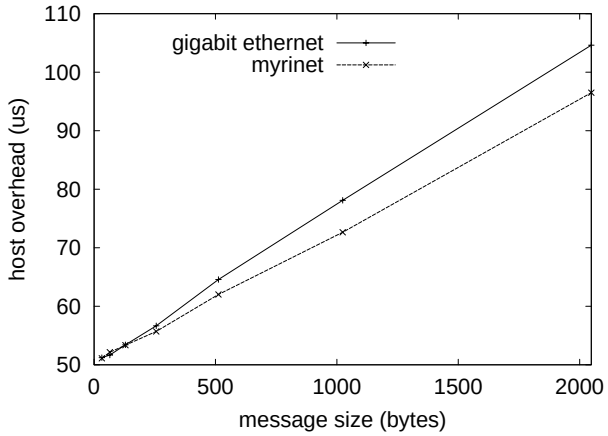

Figure 4. Host overhead (μs)

D. RESULTS

Since the goal of our study is to identify potential interconnect technologies, we did not consider host overhead in the first set of our simulations. With the knowledge of the network potential in hand, we then included the host overhead to evaluate its impact on the overall performance of parallel applications.

1. Network Performance

Token_pass

We ran the *token_pass* code with a one-byte payload to determine the minimum message latency between neighboring nodes in a 256-member virtual ring for all three technologies. As mentioned earlier, since only two processors at a time were involved in communication, there were no contention effects. We compiled our results and listed the minimum, maximum, average, and standard deviation for each study in Table 1.

	Min	Avg	Max	σ^2
Myrinet	0.388	0.427	0.869	0.093
Avici GigE	1.380	1.386	1.530	0.024
Conventional GigE	1.564	1.595	3.532	0.244

Table 1. Minimum message latency results, in μs.

As shown, the Myrinet technology delivered very good latency. The Avici and Conventional GigE latency suffered because of the store-and-then-forward mechanism implemented at the switch and host ingress points. Jitter, the variation in latency, is purely a function of network topology in the absence of congestion; it reflects the difference in distance between *token_pass* neighbors in the network. The conventional GigE topology faired worst in this category because the most distant neighbors have three additional store-and-forward delays in their communication path.

Using *token_pass* and a 15-MB message, we measured throughput for each technology to verify the correctness of our simulation code. We chose this message size because it was large enough to fill the end-to-end communication pipe, a criterion necessary for throughput measurements. The end-to-end communication pipe is the product of the theoretical bandwidth and the round-trip time. Our simulation throughput values are within half a percent of their corresponding theoretical bandwidth.

Mesh

The data in Table 2 lists the average time for a message to pass through the respective network, along with the standard deviation of the measurements and the maximum and minimum times. By comparing the average and maximum values for each technology, we observed that the maximum message-transfer-time of large messages could be up to an order of magnitude more than the average in the case of Myrinet and conventional GigE. This is due to the presence of link contention in those technologies. Only with the Avici switch were the numbers more comparable. These maximum numbers tend to pull up the averages.

In order to visualize the effects of link contention to the application's overall performance, we plotted, in Fig. 5, the *mesh* code completion times for messages ranging 32, 64, 128, 256, 512, 1024, and 2048 bytes, for all three technologies. Looking at the Avici results, we observed that the total time to completion gradually increased, from 140 μs for 32-byte messages, to 900 μs at 2-kilobyte messages. In the Myrinet case, total time to completion for these simulations was the same as for Avici at small message sizes, to about three times longer in the large message case. The results for the conventional GigE cascade of switches featured completion times, for small messages, that were consistently three to six times larger than both the Avici and Myrinet. With messages larger than 1-kilobyte, however, we observed better performance than Myrinet. It seemed that the scheduling algorithm in the conventional GigE switch could handle congestion better than Myrinet's wormhole routing mechanism.

	Avici			
Size	Min	Avg	Max	σ^2
32	1.38	2.57	5.28	0.87
64	1.65	3.27	7.11	1.10
128	2.76	5.34	9.72	1.75
256	4.86	9.18	17.37	3.02
512	9.09	16.97	30.93	5.44
1024	17.49	32.56	55.95	10.25
2048	22.20	59.20	112.59	20.21

	Myrinet			
Size	Min	Avg	Max	σ^2
32	0.58	1.57	5.74	0.90
64	0.78	2.23	9.85	1.44
128	1.18	3.74	17.58	2.54
256	1.98	7.85	55.75	7.07
512	3.58	16.73	226.71	18.83
1024	6.78	36.54	441.90	43.75
2048	13.18	75.89	906.86	93.33

	Ethernet			
Size	Min	Avg	Max	σ^2
32	1.56	53.77	80.64	11.30
64	1.85	53.86	80.55	11.37
128	2.88	53.47	81.19	12.81
256	4.92	50.85	85.45	13.67
512	9.02	57.28	118.22	27.03
1024	17.21	85.60	254.26	68.44
2048	29.52	151.92	378.46	94.66

Table 2. Latency simulation results from mesh algorithm. Size in bytes, all other fields in μs.

In summary, the average message latency delivered by the three network technologies is affected both by available bandwidth and the presence of bottlenecks. Fabric blocking is bad for parallel algorithms in that it increases the maximum latency seen by any particular message. Since all messages must eventually reach their destination before the code can

complete, that maximum latency value is crucial to the wall-clock performance of a code. The Avici switch was seen to have the smallest amount of fabric blocking, while the Myrinet and conventional GigE fabric offered potential blocking points at every switch along the path of a message.

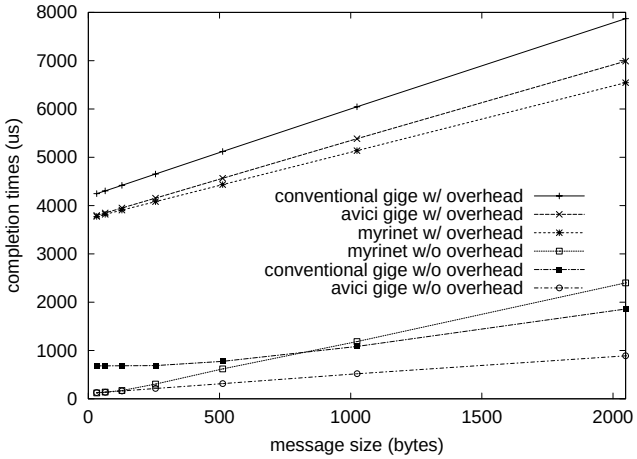

Figure 5. *Mesh* completion times verses message sizes, with and without software overhead

2. Performance with Host Overhead

In this study, we ran the same *mesh* simulations, adding host overhead. We plotted the completion times of these runs in Fig. 5 as a function of message size and compared them with those obtained from the network-only runs. As shown, while the Avici hardware demonstrated better completion times than Myrinet due to its non-blocking architecture, it lost its advantage when the host overhead was taken into account. In fact, it fared worse than Myrinet due to the approximately 10% higher overhead measured in the Gigabit Ethernet hosts, which clearly indicated that the bottleneck was at the compute nodes, and not in the network. With identical host overhead (same GigE adapter and TCP/IP processing [8]), the performance degradation in conventional GigE paralleled that in the Avici system.

Many cluster computing practitioners have noticed that host overhead is a barrier to increasing the performance of message-passing parallel programs, and "operating system bypass" schemes have thus become a topic of recent interest. To predict the beneficial effects of research in this area, our next set of experiments explored the performance characteristics of the three interconnects with progressively reduced host overhead.

In Fig. 6 we plotted the completion times of the *mesh* algorithm for many message sizes, as the host overhead was reduced from its current experimentally determined value (normalized to 1.0) down to no overhead at all. With the exception of the conventional GigE fabric, for sufficiently small messages (under 128 bytes), the realizable performance gain was strictly proportional to the host overhead and the effect of the network hardware was not evident. At large message sizes, however, there existed a point in the curve for each message size where improvements in host overhead can

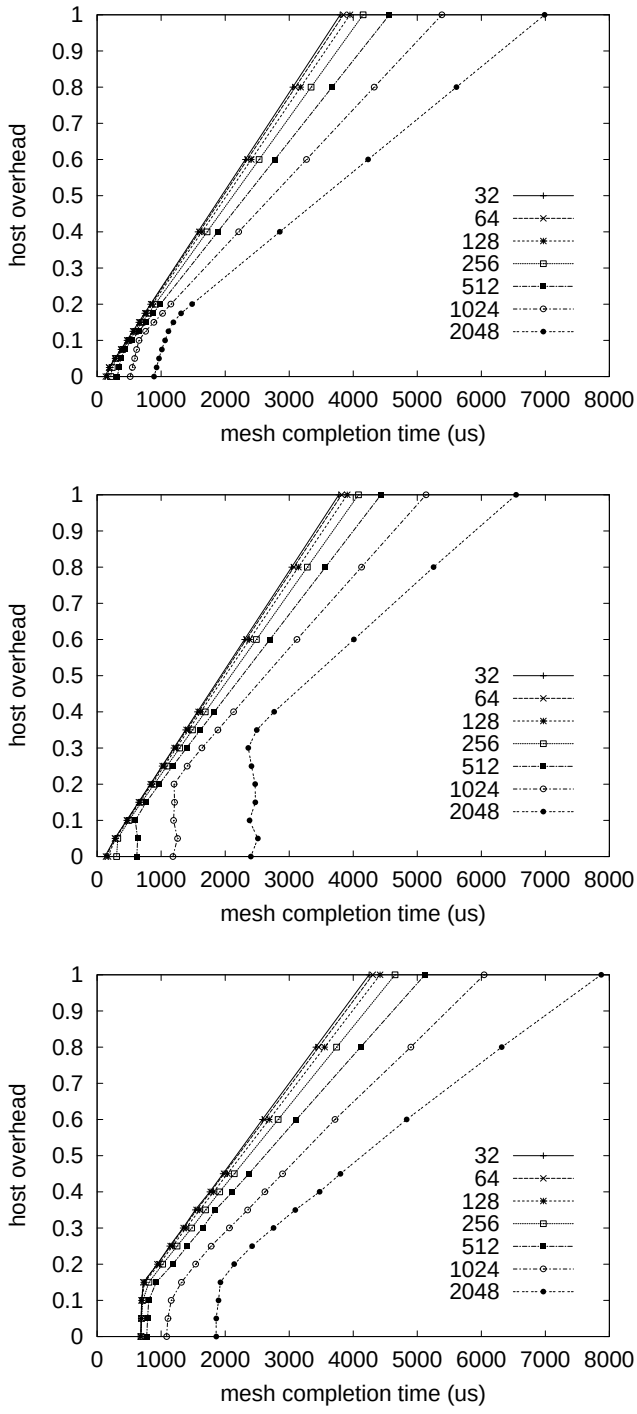

Figure 6. *Mesh* completion time as a function of software overhead : (a) Avici GigE, (b) Myrinet, and (c) Conventional GigE

no longer reduce the *mesh* completion time, because the performance bottleneck had shifted from the host to the network. For Myrinet, these points were roughly at 5% of overhead for 256-byte messages, 10% for 512, 20% for 1024, and 33% for 2048. In the Avici case, the larger messages also shifted the responsibility to the network as the host overhead improved, but not until it was dropped to 15% of its current value for the largest (2048-byte) messages. In addition, we

noted that even at these points, the completion times continued to improve, although at a much slower rate.

In the case of conventional GigE, the curves indicate that, regardless of message sizes, the network always bottlenecks when the host overhead had been reduced to 15% of its current value. We believe this was caused by the severe interswitch bottleneck coupled with multiple store-and-forwards incurred in the cascaded switch topology (see the TECHNOLOGY section). Again, we observed that, at 2048 bytes and sufficiently low host overhead, conventional GigE demonstrated better performance than Myrinet, indicating Ethernet protocol, with its MTU limitation, allows fairer scheduling during congestion than Myrinet's wormhole routing algorithm. Since the slowest parallel process dominates the parallel code completion time, unfair sharing of limited network bandwidth seemed to have adverse effects on Myrinet's performance.

E. CONCLUSIONS

We have presented the results of the analysis for three different major network architectures for parallel commodity computing. It is important to choose the network correctly because it can have a large impact on all but the most embarrassingly parallel applications, and may be the source of up to half of the cost of the entire machine. Important factors to consider are raw performance figures such as bandwidth and latency, as well as more complex parameters such as jitter, routing, multicast support, and distribution of blocking in the fabric.

Since our network design goal is to facilitate the performance of real applications, we evaluated the performance of the three network technologies when applied to specific application cores important to our users. In this context we analyzed timing results gathered from the networks and drew conclusions from our knowledge of the network about its effect on performance of the application.

Our simulation results show that Myrinet behaved well in the absence of congestion. Under heavy load, its latency suffered due to blocking in wormhole routing. Also Myrinet is limited from scaling too far due to the short cable length problem. Future development by Myricom may alleviate this constraint, although the cost to latency or budgets is unknown. The simplicity in the Myrinet switch results in low per-connection cost; however, the non-commodity network adapter cards keep the host side of the connection expensive.

Conventional Gigabit Ethernet switches currently do not scale to support more than 64 GigE ports, which leads to the introduction of a topology that involves cascading multiple stages of small switches. The presence of multiple hops in a path between hosts, and the store-and-forward nature of legacy Ethernet leads to longer message delays. Bandwidth bottlenecks at the topmost switch in the cascade may also be a problem.

The Avici terabit switch router has an internal fabric, which is quite similar to Myrinet, in that it is a very high-bandwidth three-dimensional torus using source routing and simple non-buffering switches. The line cards present standard GigE connections to hosts, though, in keeping with the current commodity favorite. Our simulations showed that the Avici switch outperformed Myrinet on messages that were 256 bytes and above, and was comparable in the small-message regime. From a cost standpoint, Avici is only slightly cheaper than Myrinet for a comparable topology, but is expected to reduce in price with further penetration of Gigabit Ethernet into the market.

Unfortunately, today's commodity Operating Systems and network interface cards incur too much latency, offsetting any performance advantages that might come from a more capable network. The most crucial factors in building high performance, distributed memory, parallel systems, therefore, are: the use of an Operating System bypass scheme to reach the network directly, and of pipelined network card design. As research improves in this area, the selection of network hardware will become more important. We hope to see end-to-end one-way latencies on the order of 10 μs in the near term, which corresponds to a tenfold reduction compared to the current state of affairs. Fig. 5 shows that, in this future scenario, the Avici fabric has sufficient network bandwidth to handle the load offered by our parallel codes. On the other hand, a network infrastructure built on Myrinet and conventional GigE technology will suffer saturation when message sizes are 512 bytes and larger. We conclude that when making network selections, one should take into consideration the performance requirements necessary to balance the demands of future, low latency compute nodes.

REFERENCES

[1] http://www.myri.com/myrinet/|overview/index.html, 1999

[2] http://www.hnf.org/, 1998

[3] http://www.tandem.com/pres_rel/sandnpl.htm, 1999

[4] R. Seifert, "Gigabit Ethernet: technology and applications for high speed LANs," Addison-Wesley, pp. 141-280, 1998

[5] G. Held, "Ethernet networks: design, implementation, operation, management," John Wiley & Sons, Inc., pp. 78—95, 1998

[6] W. Dally, "Scalable Switching Fabrics for Internet Routers," Computer Systems Laboratory, Stanford University and Avici Systems, July 1999.

[7] J. Duato, S. Yalmanchili, and L. Ni, "Interconnection Networks: an Engineering Approach," IEEE Computer Society Press, pp. 11—16, 1997

[8] http://www.mil3.com/products/| |modeler/home.html, 1999