Performance evaluation of a Gigabit Ethernet switch and Myrinet using real application cores

Helen Chen and Pete Wyckoff
Sandia National Laboratories
7011 East Avenue, Livermore, CA 94550

***Abstract*— Traditionally, high-end clusters use special purpose network hardware, thereby losing the cost benefit offered in mass market. Riding on the wave of Ethernet popularity, Gigabit Ethernet is fast becoming a commodity item. Given a scalable switching architecture, we believe it can be a cost-effective alternative to interconnect parallel systems. This study evaluates the performance of the Avici Gigabit Ethernet switch against Myrinet. We simulate a 256-node cluster running core algorithms from real parallel applications, and then compare 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.**

***Index terms*—Gigabit Ethernet, Terabit Switch Router, simulation, parallel algorithms**

A. Introduction

The idea of cluster computing is to aggregate machine rooms full of relatively cheap hardware, connected with some sort of network, and apply the combined force of the individual machines to a single calculation. The hardware employed in a cluster is generally from the volume personal computer market, so as to leverage the cost advantages of buying commodity hardware. As this architecture has distributed memory, parallel processes communicate using a message-passing paradigm. Unfortunately, highspeed interconnect is completely irrelevant for the mass market. High-end cluster users, therefore, have to rely on special purpose hardware such as Myrinet [1], HiPPI [2], or ServerNet [3] for their message passing infrastructures, losing the cost benefit offered by the commodity market.

The purpose of this study is to identify a scalable and cost-effective alternative to the traditionally expensive 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 that, given a scalable switching architecture, Gigabit Ethernet can be a cost-effective solution for cluster computing.

The remainder of this paper discusses the simulation results of the Avici based Gigabit Ethernet switch, and the methods we used, which involve a mix of "artificial" basic tests and simulations of core algorithms from real parallel applications. We compare our results against an identical study of Myrinet, a popular interconnect technology, and tally the positive and negative aspects of each.

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

B. Interconnect Technologies

The following paragraphs describe the network technologies we considered in the simulations that will be discussed in Section C. In each section we calculate the current pricing for a prototypical 256-node cluster, a size which is feasible for most high-end cluster builders.

1. Myrinet

Myricom's Myrinet is one of the more cost-effective, special purpose interconnection technologies. 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 with the network to send, receive, and buffer packets. All Myrinet packets carry a source-based routing header to provide intermediate switches with forwarding directions. Myrinet packets can be of arbitrary size.

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 construction 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 256 nodes is 256 x $1700 + 32 x $5000 + 12 x 32 x $200 = $670K.

2. Avici terabit switch router [5]

The Avici switch router uses two direct-connect networks [6] 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 (Fig. 1). As such, the Avici switch router can be incrementally expanded to include up to 1120 cards without blocking. At 16

Gigabit-Ethernet ports per card, this configuration can interconnect up to 17,920 compute nodes.

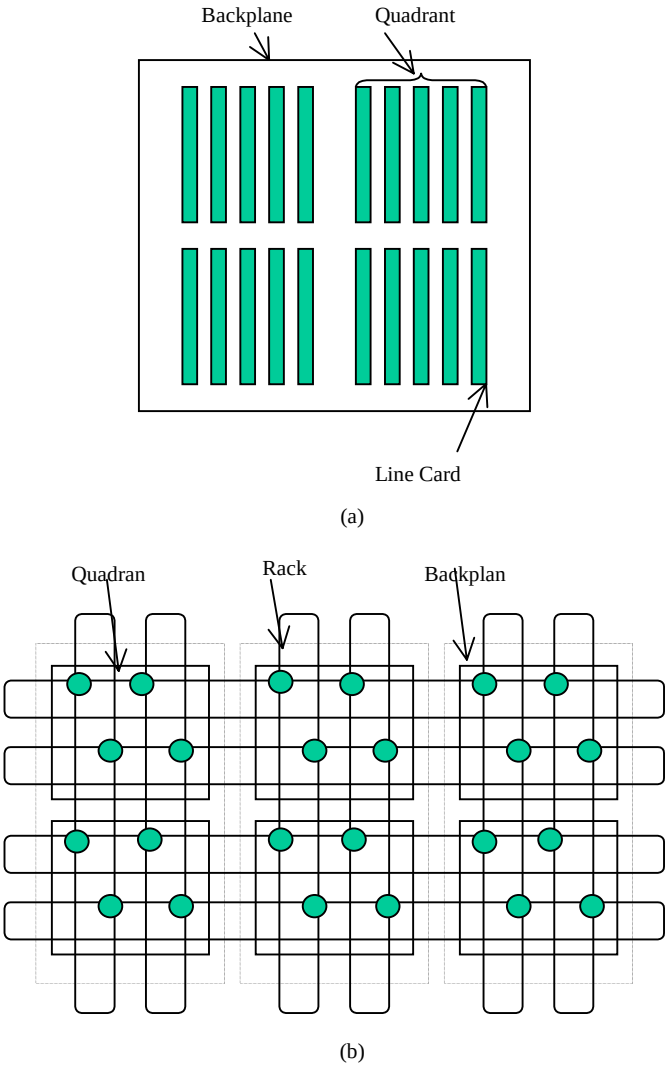


Figure 1. Avici switch router packaging: (a) backplane and (b) system.

Similar to Myrinet, the Avici router uses wormhole routing inside the fabric to achieve low latency. Unlike Myrinet, however, rather than buffering the entire message inside the network, the Avici router segments its messages into 72-byte scheduling units (flits) and exercises credit-based flow control to prevent flit loss. The Avici router eliminates blocking doing wormhole routing using per-connection buffer management and scheduling, and over-provisioning 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. A Gigabit Ethernet card costs around $700, its fiber cable $75 each. Avici projects a 256-port switch for $350,000. The total cost for a 256-node cluster is, therefore, 256 x ($700 + $75) + $350,000 = $550K. We anticipate further price reduction, as Gigabit Ethernet becomes more widely deployed.

C. Simulation Methodologies

A previous study [7] demonstrated that a realistic model of a distributed memory system must strike a balance between detail and simplicity in order to reveal important bottlenecks without making analysis of interesting problems intractable. This model, called LogP, describes a parallel system using the following four parameters: the computing bandwidth, the communication bandwidth, the communication delay, and the efficiency of coupling communication and computation. We present the methodology for the abstraction of the computation bandwidth of compute nodes in section C.1. Sections C.2 and C.3 describe two existing simulation packages that we used to derive the respective communication bandwidth and delay of the Myrinet and the Avici technologies. Finally, we present, in Section C.4, the computation representation of a kernel parallel code used at Sandia to provide the parameters that couple communication and computation.

1. Host overhead abstraction

The end-to-end latency in passing a point-to-point message includes 1) the processing overhead of some message-passing interface (e.g., MPI). 2) The processing of the Operating System's transport/network protocols (e.g., portals or TCP/IP) and their associated memory copies. 3) The network interface card's medium access overhead. 4) The network's transmission delay. We combine 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 on our testbed. Node A starts its timer, sends a message to Node B, then receives a copy of the message from Node B, and stops its timer. The recorded value (i.e., the one-way latency) is half of this round-trip timing period, and to ensure accuracy, 200 trials are averaged together. Our simulator extrapolates one-way latency from the linearly fitted plots of the experimental measurements (Fig. 2). Software overhead was calculated by subtracting the transmission delay from the derived one-way latency. We further divide the results by two, to account for processing delay incurred at the sender and the receiver, and they are plotted, as a function of message sizes, in Fig. 3.

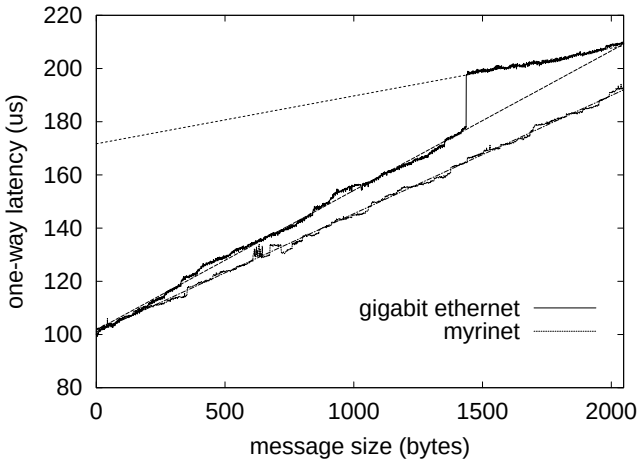


Figure 2. One-way latency

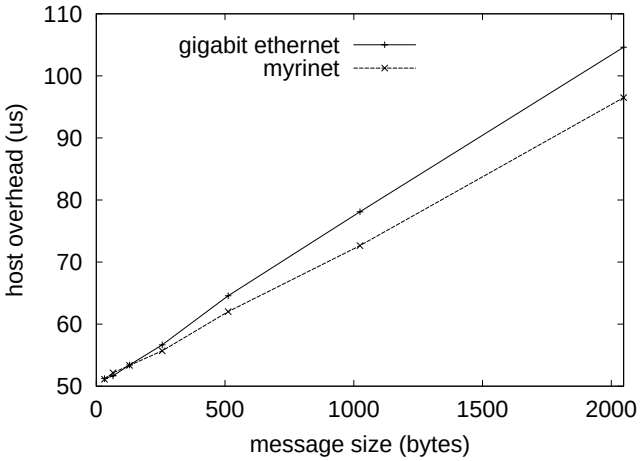


Figure 3. Host overhead (µs)

2. *Avici simulator*

Allen King and coworkers at Avici wrote a simulator [8] to be used in planning the switch hardware they built. We inserted hooks into the simulator by which we could feed our own traffic patterns into the Avici fabric. In addition, various other calls are used by the fabric and the application code modules to notify each other of initialization, completion, and to acquire or change fabric parameters. We also added the modeling of a line card, which connects 16 Gigabit-Ethernet ports into the Avici backplane. An interface layer handles the details of segmentation and reassembly so that an application is insulated from the transport details.

3. *Myrinet simulator*

The Myrinet simulator [9] was initially developed by Chen-Chi Kuo as a graduate student in the Computer Science Department at the University of Utah to be used in their full system simulator. We adapted just the Myrinet part of the simulator, and added an event handling mechanism along with the packet tracking and upper layer frameworks written for the Avici simulator interface. We generated hardware parameters from Myricom's documentation or by performing empirical tests on our cluster. Our connectivity topology is a two-dimensional (4 x 8), wrapped torus of switches (Fig. 4), with eight nodes to a switch, and was chosen for its scalability properties over more complex topologies. The links between switches consist of two parallel cables, giving a doubled hop-to-hop bandwidth of 2.56 Gb/s, and we generate routes using a utility provided by Myricom which implements the up*/down* algorithm. The same application codes designed for use with the Avici simulator couple directly to the interface we wrote to communicate with Utah's flit-level Myrinet simulator, similar parsing tools are used to deduce statistics from the output of simulation runs.

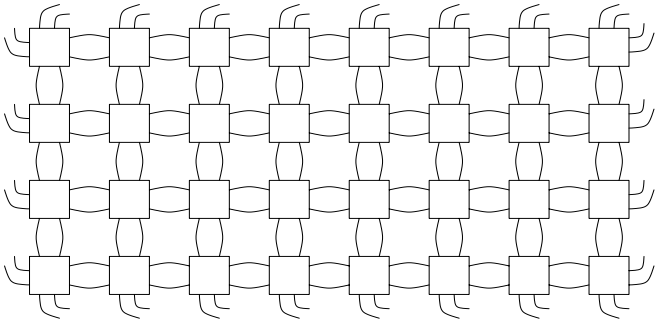


Figure 4. Myrinet 2d Torus Topology

4. *Parallel code algorithm*

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 augment these basic numbers with the results from the computational core algorithm of real parallel codes used at Sandia.

The code *mesh* simulates 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, and each processor will communicate with its immediate neighbors in both the *x* and *y* directions. The code performs a number of iterations of computation and communication cycles, which represents the explicit time stepping algorithm of the real code as it solves a generalized partial differential equation.

D. Results and Analysis

Since the goal of our study is to identify potential interconnects we explored the raw capacity of each technology without considering the host overhead. We did, however, add the host overhead in our subsequent experiments in order to evaluate its impact on the overall performance of parallel applications.

1. Interconnect Performance

Message latency

The data in Table 1 show 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.

Table 1. Latency simulation results from *mesh* algorithm. Size in bytes, all other fields in μs.

	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

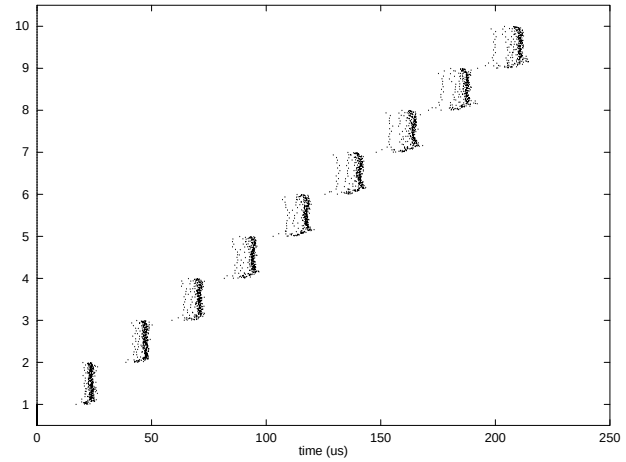
	Myrinet			
Size	Min	Avg	Max	σ^2
32	0.58	1.57	5.74	0.90
64	0.78	2.23	9.85	1.44
128	1.18	3.74	17.58	2.54
256	1.98	7.85	55.75	7.07
512	3.58	16.73	226.71	18.83
1024	6.78	36.54	441.90	43.75
2048	13.18	75.89	905.86	906.96

Up to a message size of 256 bytes, the Myrinet network delivers average latencies about 1*μs* lower than does the Avici Ethernet. This is due to the latency introduced in the Avici switch fabric, which we measure to be the same amount. Myrinet switches add about 300 *ns* per hop, with an average of 3 hops per route in an 8x4 two-dimensional torus, to give the 1 *us* latency to transit Myrinet switch.

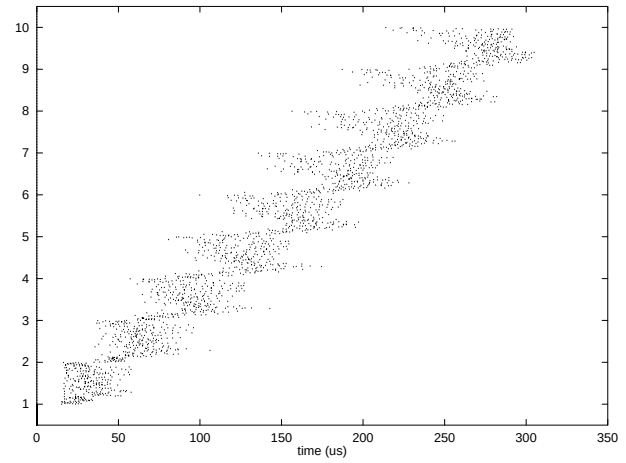
In the large message extreme, a 2-kB packet takes, on average, 59 *μs* to traverse the Avici Ethernet network, versus 76 *μs* to traverse the Myrinet network. However, the worst case transfer time is a factor of eight greater for the Myrinet. Note that these are not raw transfer times, but the result of the interactions with transfers between other pairs of nodes on the network. This leads us to conclude that the effect is from the blocking induced by obstructing messages in the network traffic. The Avici switch is designed to be non-blocking with its extreme path redundancy and large fabric speedup. Therefore, we conclude that the smaller difference between its maximum and average message transfer time is due to output port contention, i.e., when multiple messages are waiting to enter a single destination host.

Completion time

The second data analysis we perform takes into account more of the details of the algorithm. Fig. 5 shows plots of the results at a 256-byte message size, for both technologies.



(a) Avici GigE



(b) Myrinet

Figure 5. Mesh completion times 256-byte messages

Each plot shows, for each iteration and processor, the time when that processor completed that iteration. The unlabeled vertical axis is the iteration number of the algorithm, from 1 to 10. The horizontal axis is the global time, in microseconds, and varies from plot to plot, as the completion times are quite different with respect to both message size and to network technology.

Each integral band of *y*-axis is broken up into 256 points, one for each processor. A dot is placed in a processor's strip in a given iteration number at the time that processor has sent and received all messages

necessary to proceed with the calculation of that timestep.

One thing to notice in the plots is that, within each iteration, some processors always complete much earlier than others. For the *mesh* case, these are usually the ones on the corner that have fewer neighbors and thus fewer messages to exchange. There are also apparent stripes in the iteration bunches reflecting discrepancies in transfer-time between topologically nearby and distant *mesh* neighbors

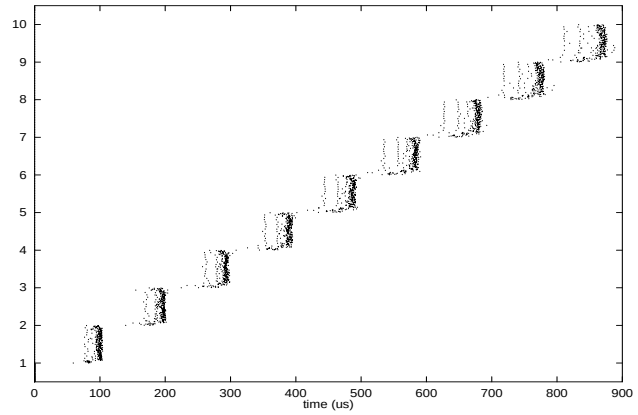
At the relatively small 256-byte message size shown in Fig. 5, the iteration bunches are well separated from each other as most of the time to completion of each iteration is taken up by computation time, represented in our simulation by a sleep of 10 μs. An ideal network, that used no time to transfer messages, would show perfectly vertical lines at each of the iterations, with the last line (between 9 and 10) at 90 μs. Anything more than 90 μs is the effect of communication delay.

The plots for the rest of the studied ranges of message sizes are not shown. From the Avici results, we observe that the total time to completion gradually increases, from 140 μs for 32-byte messages, to 900 μs for 2-kB messages. We also note that each iteration-bunch is becoming more separated into individual stripes, with the edge processors finishing progressively earlier than the bulk in the center (Fig. 6).

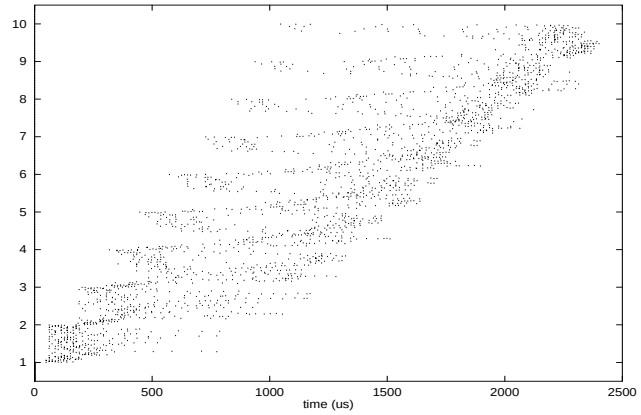
In the Myrinet network case, the iteration groups are fuzzier. By comparing Fig. 5 and Fig. 6, we observe that the fuzziness increases with increasing message size, which is a direct result of the progressively larger variance in communication times listed in Table 1. At large message sizes, the discrepancy in transfer-time for both topologically nearby and distant neighbors are greater, because messages sent through the network are subject to more potential points of blocking. Total time to completion for these simulations are the same as for the Avici at small message sizes, up to about three times longer in the large message case.

2. Performance with host overhead

In this study, we ran the same *mesh* simulations, adding the experimentally extrapolated host overhead (Section C.1) at the sender and the receiver. We plot the completion-times of these runs as a function of message size and compare them with those obtained from the network-only runs (Fig. 7). As shown, while the Avici hardware demonstrated better completion-times than Myrinet due to its non-blocking architecture, it loses the advantage when the host overhead was taken into account. In fact, it fared worse than the Myrinet solution because of the approximately 10% higher overhead measured in the Gigabit ethernet hosts (Fig. 3). This clearly indicates that we are bottlenecked at the compute nodes, and not the network fabric. Fig. 8 plots the per processor *mesh* completion-times for 2-kB messages as in Fig. 6, but with host overhead. It is interesting to note that the completion-times spreads of the iterations are smaller in comparison to those in Fig. 6, especially in the case of Myrinet. Obviously, the presence of large host overhead has reduced jitter within the network fabric. The reason for this phenomenon is not clear and is currently under investigation.



(a) Avici



(b) Myrinet

Figure 6. *Mesh* completion times, 2048-byte message size

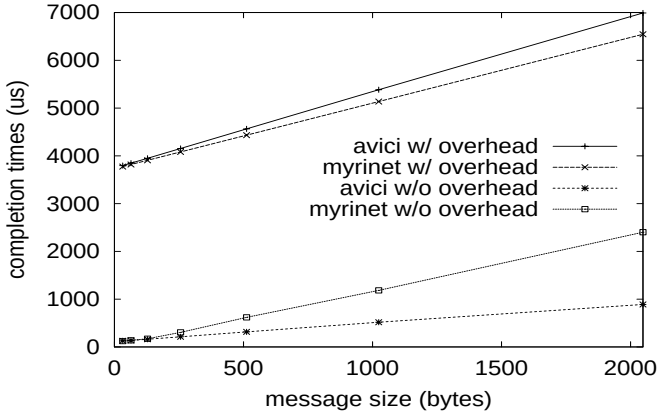
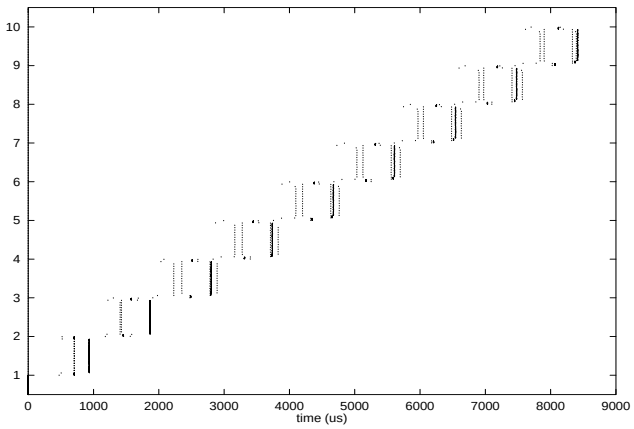
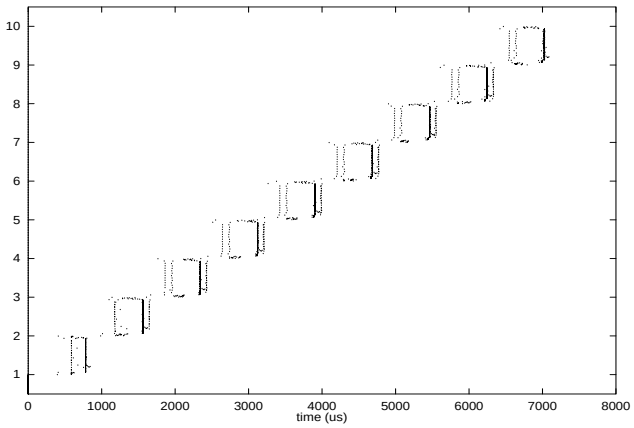


Figure 7. *Mesh* completion times verses message sizes, with and without software overhead



(a) Avici

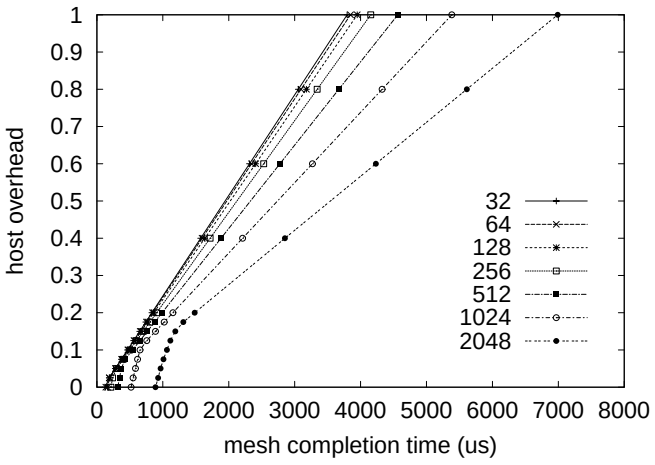


(b) Myrinet

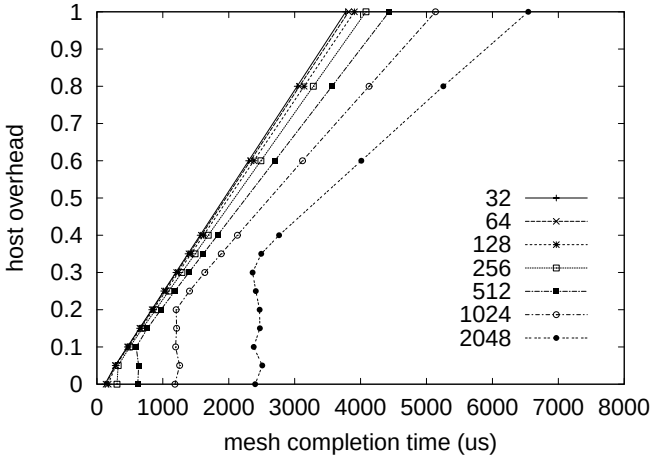
Figure 8. *Mesh* completion time with host overhead, 2048-byte messages

Many cluster computing practitioners have noticed that software 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 [10, 11]. These schemes involve user memory mapped network interfaces and zero-copy transfers within the message passing primitives. To predict the beneficial effects of research in this area, our next set of experiments explores the performance characteristics with progressively reduced host overhead. In Fig. 9 we plot completion times of the *mesh* algorithm for many message sizes, as the host overhead is reduced from its current experimentally derived value (normalized to 1.0) down to no overhead at all. For sufficiently small messages (under 128 bytes), the realizable performance gain is strictly proportional to the host overhead and the effect of either hardware interconnect technology is not evident.

At large message sizes, however, there exists a point in the curve for each message size where improvements in host overhead can no longer reduce the *mesh* completion time, indicating the shift of performance bottleneck from the host to the network. For Myrinet, these points are 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 shift the responsibility to the network as the host overhead improves, although at a later stage. For example, the inflection point for the 2048-byte message did not occur until its host overhead has dropped to 15% of its current value, as compared to 33 % in the Myrinet case. In addition, we note that, even after the inflection points, the completion times continued to improve, although at a much slower rate.



(a) Avici



(b) Myrinet

Figure 9. Message completion times as a function of software overhead

In summary, our results demonstrate that the relatively large host overhead in cluster computing can offset the performance advantages of more capable network technologies. However, significant progress has already been made in the design and implementation of OS-

bypass mechanism [12]. In fact, Myricom reported a 13-21 μs of one-way latencies transferring small MPI messages over GM; GM is Myricom's implementation of OS-bypass available on free BSD, LINUX, and Solaris [13]. These progresses will soon approach the point where Myrinet would become the performance bottleneck in the 256-node configuration we simulated. Therefore, proper selection of an interconnection network must consider the network architecture such that it can balance the bandwidth requirements of future parallel systems.

E. Conclusions

We have presented the results of the analysis for two different 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 performance factors to consider are bandwidth, latency, jitter, routing, 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 two 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 behaves well in the absence of congestion. Under heavy load, its latency suffers due to blocking in the fabric. Also Myrinet is limited from scaling too far because of cable length limitations. Future development by Myricom may alleviate this constraint, although the cost to latency or budget 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.

The Avici terabit switch router has an internal fabric with some similarities to Myrinet, in that it uses source routing and non-buffering switches. Unlike Myrinet, it achieves non-blocking via extremely high path-redundancy, high-bandwidth internal links and per-connection buffer management and scheduling. The line cards present standard GigE connections to hosts, in keeping with the current commodity favorite. Our simulations showed that the Avici switch outperformed Myrinet on large messages (256 bytes and above), and was comparable in the small-message regime. Avici is only slightly cheaper than Myrinet for the 256-node topology we simulated, but would be increasingly more cost effective as the cluster size increases. Moreover, we expect reduction in cost 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 technology. The most crucial factors in building high performance, distributed memory, parallel systems, therefore, are not only the selection of a capable network technology, but also methods of reducing host overhead such as an Operating System bypass scheme. As research improves the state of the art 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 future, which corresponds to a tenfold reduction compared to the current state of affairs. Our results show that, in this future scenario, the Avici fabric should have sufficient network bandwidth to handle the load offered by our parallel codes. On the other hand, a network infrastructure built on Myrinet technology could suffer saturation for message sizes greater than 512 bytes.

Acknowlegment

The authors would like to thank James Brandt for his technical input. We would also like to extend our appreciation to Nina Berry, David Evensky and Ann Gentile for their editorial comments.

References

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

[2] HiPPI, http://www.hnf.org, 1999.

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

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

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

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

[7] D. Culler, et al "LogP: Towards a Realistic Model of Parallel Computation", Proceedings of the Fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, May 1993.

[8] A. King, "Terasim: the Simulator for Avici TSR", Avici Systems, Inc., 1997.

[9] C. C. Kuo, "The Avalanche Myrinet Simulation Package", Department of Computer Science, University of Utah, 1997.

[10] F. O'Correll, et al., "The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network," International Conference on Supercomputing '98, pp. 243-250, ACM, July 1998.

[11] T. Takahashi, et al., “Implementation and Evaluation of MPI on an SMP Cluster,” IPPS/SPDP ’99, Workshops, Vol. 1586 of Lecture Notes in Computer Science, pp. 1178-1192, Springer-Verlag, April 1999.
[12] MPI-BIP, http://lhpca.univ-lyonl.fr/mpibip.html
[13] MPICH-GM, http://www.myri.com/