A Performance Analysis of the Ammasso RDMA Enabled Ethernet Adapter and its iWARP API

Dennis Dalessandro
Ohio Supercomputer Center
1 South Limestone St., Suite 310
Springfield, OH 45502
dennis@osc.edu

Pete Wyckoff
Ohio Supercomputer Center
1224 Kinnear Road
Columbus, OH 43212
pw@osc.edu

Abstract

Network speeds are increasing well beyond the capabilities of today's CPUs to efficiently handle the traffic. This bottleneck at the CPU causes the processor to spend more of its time handling communication and less time on actual processing. As network speeds reach 10 Gb/s and more, the CPU simply can not keep up with the data. Various methods have been proposed to solve this problem. High performance interconnects, such as Infiniband, have been developed that rely on RDMA and protocol offload in order to achieve higher throughput and lower latency. In this paper we evaluate the feasibility of a similar approach which, unlike existing high performance interconnects, requires no special infrastructure. RDMA over Ethernet, otherwise known as iWARP, facilitates the zero copy exchange of data over ordinary local area networks. Since it is based on TCP, iWARP enables RDMA in the wide area network as well. This paper provides a look into the performance of one of the earliest commodity implementations of this emerging technology, the Ammasso 1100 RNIC.

Keywords: *RDMA, iWARP, TCP*

1 Introduction

As network speeds increase the CPU must spend more time working to service network requests. This takes time away from the real work that needs to be done and becomes a bottleneck to performance. The problem becomes especially evident when network speeds are increased to 10 Gb/s and beyond. The real problem is the TCP/IP stack, which must be processed by the host CPU. To avoid this problem, technology such as Infiniband has been developed that not only allows for a very fast interconnect, but also employs a mechanism known as Remote Direct Memory Access (RDMA) [26] to bypass the operating system and CPU in order to directly move data into application memory.

Support for this project was provided to OSC by the Department of Energy ASC program.

This may be a fine approach, and in fact is quite common. The problem with technology such as Myrinet [21], Infiniband [12], or Quadrics [25], is that they all employ a special purpose interconnect. It is an undeniable fact that Ethernet is the dominant existing network infrastructure. In order to take advantage of this, RDMA over TCP/IP has been developed [26]. While the special purpose interconnects are limited in scope to specific high performance computing (HPC) deployments, RDMA over TCP/IP, which is known as iWARP [10], has the potential to be easily integrated into our everyday networks. One major reason that iWARP is interesting to the HPC community is cost. Due to the high cost of special purpose interconnects, it is still common to have a cluster connected with Ethernet. Though 10 Gb/s Ethernet is extremely expensive and not widely used at present, 1 Gb/s Ethernet is moderately priced, and is commonly used. Despite the current high cost of 10 Gb/s Ethernet, it is likely that the price will drastically decrease in the near future [33].

If we consider the way Ethernet has been adopted in the past, it is pretty clear that the price will drop considerably with time. It should be clear that TCP/IP based RDMA implementations can greatly improve performance of the system without greatly increasing the cost.

Some would argue that Infiniband is on its way to becoming a commodity interconnect. This may very well be true. Despite being a commodity product, Infiniband will remain at a disadvantage due to its local area network limitation. iWARP [26], on the other hand, running over TCP/IP is not limited to the local area network environment, but is capable of functioning in a wide area environment as well.

The sections that follow provide insight into the performance implications of iWARP as well as the motivation for iWARP. In Section 2 we shall explore the basic reasons why to use RDMA in the first place, followed by a performance analysis of an actual RDMA implementation. We will look at the feasibility of using iWARP as the basis for message passing, as well as the impact of the verbs level API. In Section 4 we examine some of the limitations to iWARP, then

follow with some insight into related work, and our intended future use of iWARP in section 5.

2 Why RDMA?

The obvious question is, why use RDMA? In traditional TCP/IP-based networks, the operating system must handle processing of the network stack. This makes other processes wanting to use the processor have to wait. It may seem intuitive that to add extra processing of RDMA on top of this would only slow things down. This, however, is not the case. For RDMA over Ethernet to work correctly the processing is done on-board an RDMA enabled Network Interface Card (RNIC). This alleviates the CPU from having to process the network stack. In [11] we clearly see the need to eliminate the role of the CPU in network processing.

Offloading of the network stack can be accomplished by itself, such as what happens with TCP offload engines, or TOE cards. An RNIC is more than a TOE. The RNIC supports an API that provides for the direct placement of incoming data to the memory of the application. It is this process that makes the RNIC capable of a true zero-copy data transfer. Operating system bypass is another technique that is offered by an RNIC but not a TOE. This avoids some indirection and context switches that affect particularly small data transfers. U-net [32] and EMP [29] are two examples that recognize the need both to bypass the kernel and to minimize memory copies. Other attempts [13] at zero-copy are not general enough to function in a real-world environment where packets are dropped and arrive out of order.

RDMA also has the advantage of a verbs level API for direct access to the hardware. This means applications can be written to completely bypass the kernel and achieve much higher network performance. The advantage to a verbs API layer is more important than just for performance of applications. This also means that it is possible to have a common API which can allow programmers to write software for one RDMA device and have it run on another RDMA device. The Direct Access Transport (DAT) Collaborative aims to do just this, with DAPL [8]. For instance, in the case of iWARP, each RNIC may have its own API, but all iWARP RNICs follow the same verbs [26]. Compatibility will not be limited only to iWARP RNIC's from different vendors, rather code written in DAPL will function on iWARP RNICs, Infiniband Host Channel Adapters (HCA), and other special purpose interconnects as well.

It has been argued that there is a need to enable ordinary sockets applications to take advantage of modern high performance user level APIs [2], and as it turns out, DAPL [8] is not the only effort to abstract the verbs layer of RDMA devices. Sockets Direct Protocol (SDP) [24, 23] is a way to allow applications written to use TCP/IP sockets to take advantage of the RDMA type of transport. This means existing applications can take advantage of RDMA without being rewritten; a recompile or relinking with the correct libraries is all that is needed, the socket semantics are the same. Other approaches include extending the existing socket interface [9], and another approach to abstract the interface API of an RDMA device [4]. It is not clear what trade-offs in performance will arise through use of these APIs. In at least two instances as shown in [7] the TCP performance over sufficiently fast interconnects such as Infiniband[12] was much lower than native communication. In fact with a maximum bandwidth of 10 Gb/s available, only around 2 Gb/s was achieved.

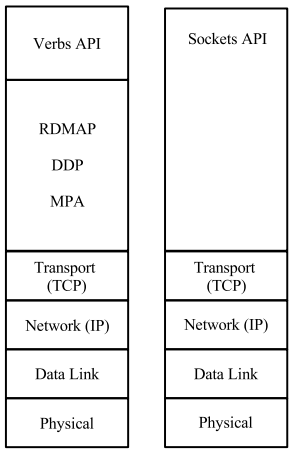

Figure 1. iWarp and TCP Protocol Stacks.

Other advantages of RDMA, specifically those related to performance, will be highlighted throughout the remainder of this paper. The diagram in Fig 1 shows the protocol stack of iWARP compared with traditional TCP, and it is important to understand, in order to look at experimental performance in the next section. At the top of the stack we find the verbs level iWARP API [10]. The iWARP API interfaces with the RDMAP [27], DDP [28], and MPA [6] layers to create TCP packets which can be sent into the network and handled like any other packet. In an ideal situation everything below the iWARP API layer would be offloaded to hardware. At the very least the Transport layer and below must be offloaded to hardware, in the case of TCP only the Data Link and Physical layers are handled by hardware.

3 Experiments

The results that follow were gathered using two RNIC Ethernet server adapters from Ammasso Inc. [1]. Two IBM Xseries servers were outfitted with the Ammasso cards, each running RedHat Enterprise Linux and the 2.4.21-15 kernel. Both servers contain a single P4 Xeon chip running at 2.8GHz, and 1.5GB RAM. The same base version of MPICH [19] is used for both TCP and iWARP, that is MPICH 1.2.5, the only difference is TCP's MPICH is im-

plemented as sockets and iWARP's MPICH is implemented with Ammasso's API calls. Since standard gigabit Ethernet switches can be expected to add around 5 microseconds of latency, the two RNICs were connected back to back, via a CAT5e Ethernet crossover cable. This also eliminates any skew in the results due to other network traffic.

All data, including ordinary TCP socket based transfers, was gathered using the Ammasso cards rather than the onboard Ethernet of the servers. This is made possible through the dual data paths of the Ammasso 1100. Traffic intended to be used in an RDMA operation between two RNICs is processed on the card by the RDMA data path, while traffic meant to be processed as regular TCP, such as that from an NFS server, is handled by another data path. This path forwards the incoming data along to the usual TCP/IP stack, as any other network card would.

The RNIC has two IP addresses and two MAC addresses. This allows the RNIC to determine what traffic is meant for the TCP data path and what traffic is meant for the RDMA data path. This dual use of an RNIC is an important advantage. With special purpose networks, there is usually a need to use a regular Ethernet device as well. Often special purpose networks are capable of handling TCP traffic, but this does not solve all the problems. For instance, the special purpose interface can not be established until the operating system is running; in a cluster which utilizes network booting there needs to be a way of using standard TCP to boot the operating system.

In the following sections we shall refer to RDMA over Ethernet as simply iWARP and common TCP socket based communication as TCP.

3.1 Message Passing Capability

We have chosen to examine the latency and bandwidth for MPI [20] operations largely in part because MPI is the dominant form of communication in the HPC realm. Latency and bandwidth are extremely important because they are the most basic metrics and give us the best overall view of the performance related to the interconnect. It is for this reason that we should consider iWARP's capability to facilitate message passing.

The goal is not to evaluate a particular implementation of MPI, rather we want to look at some basic performance metrics that are indicative of how well suited the technology is to message passing. We do this by looking at iWARP compared with TCP. We do not consider Infiniband due to the fact that Infiniband is a 10 gigabit per second interconnect and our Ethernet hardware is only 1 gigabit. For a detailed analysis of MPI over Infiniband's performance, see [16].

The latency and bandwidth tests in Figures 2, 3, 4, and 5 were conducted with NetPIPE 3.x [31], a protocol independent network performance analyzer. We use a version of NetPIPE compiled for MPI over TCP and a version compiled for MPI over iWARP. The latency and bandwidth results are shown below. Basically, NetPIPE conducts a series of ping-pongs between two MPI processes.

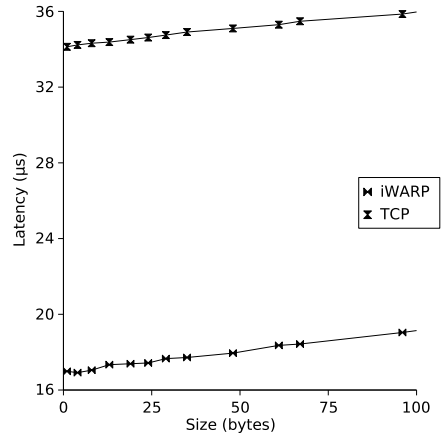

Figure 2. Latency for small message sizes

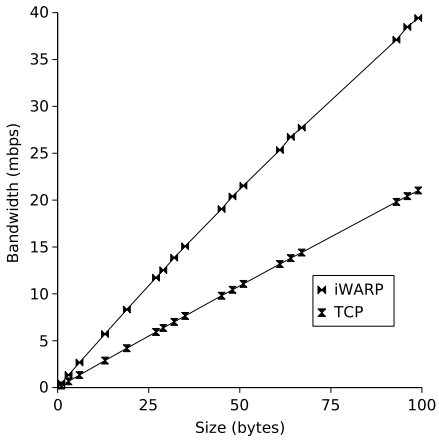

Figure 3. Bandwidth for small message sizes

In Figure 2 we are looking at only tests smaller than 100 bytes. It is important to look at what happens for small messages for a number of reasons. First and foremost the details of what happens with small messages are washed out in graphs which cover as large of a range as do Figs. 4 and 5. We also should have an understanding on how small messages behave since small messages are often used for control in applications. Latency is key for these small control messages.

We see that in Figure 2 the latency for iWARP is about half that of the latency for TCP. It is also the case that iWARP has less latency for 100 bytes than TCP does for 0 byte messages. In fact, for a given message size iWARP is almost twice as fast as TCP. It is worth observing that both lines in Figure 2 have relatively the same rate of change,

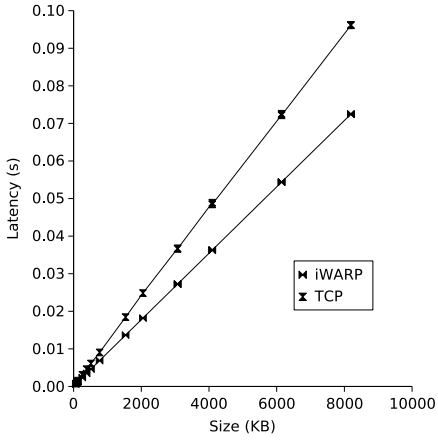

Figure 4. Latency, the big picture

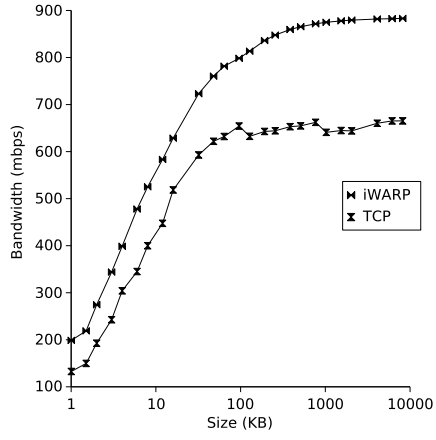

Figure 5. Bandwidth, the big picture

about 2 microseconds from 1 to 100 bytes. This is most likely explained by the fact that iWARP is really built on top of TCP, thus the relative changes occur at the same size messages.

Figure 3 shows us the bandwidth of both iWARP and TCP for small message sizes. We see in Figure 3 that iWARP's bandwidth increases much more rapidly than that of TCP, yet in Figure 5 the overall trend of TCP is the same as with iWARP. Yet, iWARP does not continually increase in bandwidth relative to TCP. This reveals a key point about iWARP and TCP. iWARP will start off with higher bandwidth, due to the low latency, but iWARP is still limited by the absolute bandwidth of the underlying transport, which as mentioned before is TCP.

Thus, as Figure 5 indicates, iWARP and TCP experience the same relative changes, but iWARP reaches the physical limit of the network, while TCP can not. The major advantage to iWARP as indicated by Figure 3 is that the difference in latency is widening due to the decreased bandwidth of TCP for large messages.

Another important characteristic of networks is their ability to process simultaneous bidirectional traffic, this is important for message passing and asynchronous communication. In Figure 6 we see iWARP as having a large increase in bi-directional bandwidth over one way bandwidth. While, on the other hand, TCP bi-directional bandwidth is not a significant advantage, in fact with TCP the bidirectional bandwidth is actually slightly less than the one-way bandwidth. The program used to generate the data in Figure 6 was from the OSU MPI benchmarks [17, 14]. It is for this reason that the bandwidth shown in Figure 6 can not be compared to the bandwidths in the previous figures, the two tests are different. Due to the RNIC's capability to offload processing of the TCP/IP stack and move data directly between user memory and the RNIC, send and receive operations can occur at the same time. This is what gives the increase in bi-directional bandwidth over one-way bandwidth.

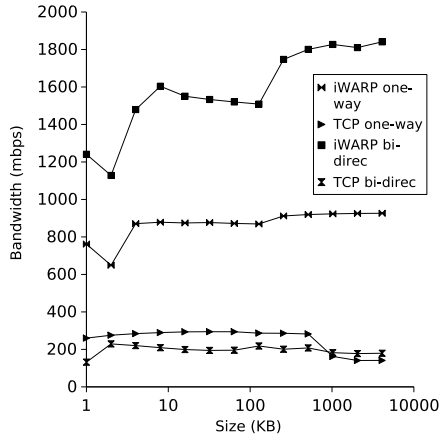

Figure 6. One-way and bi-directional bandwidth

3.2 MPI Details

Since MPI [20] is one of the main uses that iWARP will be targeted for, we should look at what overhead is associated with the MPI layer. It is obvious that there has to be some overhead. By looking at Figure 7, we see that the overhead associated with MPI in iWARP is actually less than the overhead associated with MPI by TCP. The data in Figure 7 was obtained through NetPIPE [31], for all TCP, and for iWARP MPI, while the data for native, or raw iWARP was gathered using custom software written specifically to mimic the behavior of NetPIPE, keeping the semantics the same, but implemented with an iWARP API.

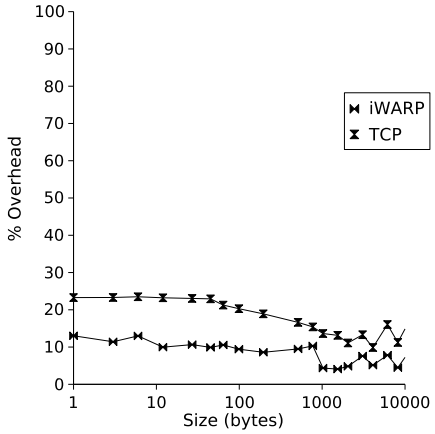

Figure 7. MPI bandwidth overhead

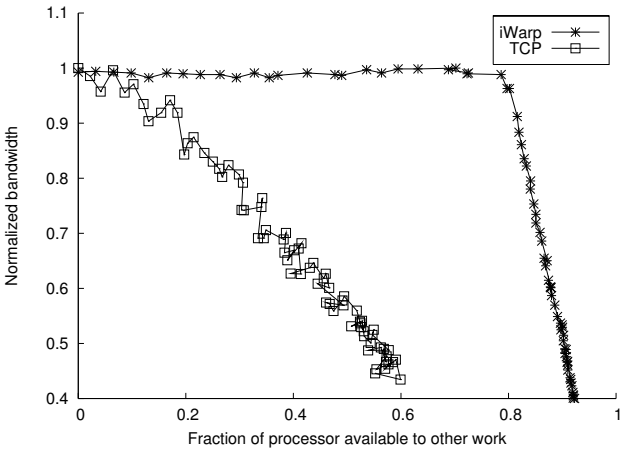

Figure 8. Normalized bandwidth as amount of non-communication work increases

In order to calculate the amount of overhead formula 1 was used.

$$BWOverhead = \frac{raw - mpi}{raw} \times 100 \quad (1)$$

We can see from Figure 7 that the overhead associated with bandwidth is around 10% for iWARP, compared to 20% for TCP. This difference is likely due to differences in the programming interfaces for iWarp and TCP. The iWarp API provides higher-level functions that allow for significant offload and overlap unlike the sockets interface used with TCP. We do not show a graph for latency but the results are the same as we would expect.

3.3 Impact of the CPU

One of the biggest advantages of RDMA is that the CPU does not have to be involved in as much of the communication of data, at least in a true zero copy implementation this is the case. Using COMB [15] to overlap MPI communication and computation, we can look at what happens as the CPU is needed for computational tasks. COMB uses a busy loop to perform a known amount of work while sending and receiving data with a remote host. This amount of work increases and measures of bandwidth are taken at various intervals. This MPI-based benchmark provides the data used in Figure 8.

From Figure 8 we see how unaffected iWARP is as the amount of CPU time used for computation is increased. iWARP achieves a higher bandwidth than TCP as far out as 80%. This means that with only 20% of the processor dedicated to network communication, iWARP has a bandwidth as good as TCP using 100% of the processor for network communication. According to Figure 8 TCP bandwidth begins to drop noticeably right away, while iWARP holds steady as more of the CPU is dedicated to computation. The reason for the steep decline in iWARP bandwidth is that the CPU must do some work to progress the application. The CPU is responsible for such activities as posting Send Work Requests and checking for Completion Queue Events, although the CPU is not involved in data path operations as it is with TCP. Toward the right side of Figure 8, the short intervals that are allocated to message passing work are too far apart, effectively starving the NIC by leaving its queues empty.

3.4 iWARP API Impact

We have so far shown the performance benefits of RDMA, now we turn to the costs associated with the API of our particular RNIC implementation.

API Call	Time (μs)
RNIC Open	156.15
RNIC Close	81.91
Register Async Event Handler	1.96
Register CQE Handler	694.31
Allocate Protection Domain	1.13
Create Queue Pair	78.21
Post Work Request - RQ	4.63
Post Work Request - SQ	4.87
Connect Queue Pair	305536.34
Poll CQ - Empty	0.12
Poll CQ - Non-Empty	1.96

Table 1. Individual API function timings

The data in Table 1 was gathered through the use of custom software designed to test each entry in the table. These API calls represent some of the most common that will be used in most iWARP applications. Due to space requirements we shall not go into details on the API calls here, we

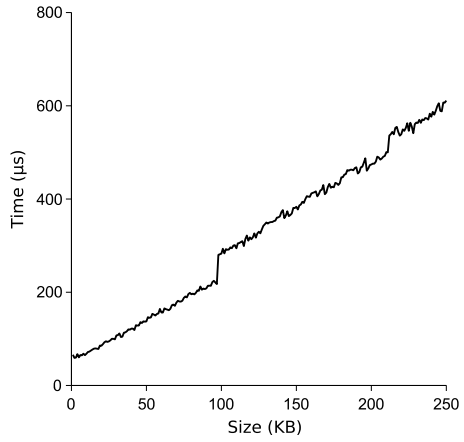

Figure 9. Memory registration costs

suggest referring to [10] for more information on what each call does.

It is, however, worth noting that posting work requests is quite efficient. Less than five microseconds to post a send or a receive work request. These are the values of send and receive overhead in a LogP [5] model of iWARP. It is also the case that the overhead incurred in the send or receive operation has a large impact on the performance of the application [18].

Another very important aspect that we should look at is the cost of registering or reserving memory for the buffers of data we want to transfer. These are costs not directly incurred in the TCP case. Figure 9 shows us the costs associated with registering memory regions. There are two types of memory regions in iWARP [26], tagged and untagged. The costs associated with these two types are relatively the same which is why Figure 9 does not make a distinction between the two. It appears that the cost of registering memory scales very well. To register 1 kB of memory takes only around 60 microseconds. The somewhat periodic jumps in registration time, especially pronounced at around 100K and 225K, are due to communication protocol issues between the host driver and the RNIC which are necessary to convey details of the physical buffer list.

4 iWarp Limitations

iWARP may be a better solution than traditional TCP/IP communication for many applications, but it is not without limitations. Currently 10 Gb/s Ethernet is very expensive and not commonplace, thus implementations today are limited to a wire speed of only 1 Gb/s. On the other hand, special purpose networks are commonly in the 10 Gb/s range. iWARP is also very new, Ammasso [1] was the first to produce such a product, while specialty interconnects such as Infiniband [12], Myrinet [21], and Quadrics [25] are already established and commonplace, particularly in the HPC arena. As mentioned earlier this is likely to change in the near future, but until then iWARP can not compete on a level playing field with the likes of Infiniband, Myrinet, and Quadrics.

iWARP suffers, as do the specialty interconnects mentioned previously, from the problem of the application supplying enough data to take advantage of the full bandwidth. A 10 Gb/s link does not help if the average application can only generate 1 Gb/s worth of traffic. Another limitation of iWARP is, of course, how fast the RNIC can process incoming traffic. The same is true of any RDMA implementation. Another advantage specialty purpose interconnects have over iWARP is the capability to implement hardware specific functionality, such as the reliable hardware multicast mechanism in Quadrics. This kind of feature is not possible with iWARP because iWARP is built on an existing infrastructure of Ethernet switches that do not support such features (although one might imagine using the non-reliable multicast in some fashion). On the other hand, iWARP can run over any TCP/IP based network, which is really the biggest benefit of all.

5 Related and Future Work

There has been much work done in the field of zero-copy techniques and protocol offload [29, 32, 30, 3]. Unlike these, our work is an in-depth study of a commodity iWarp implementation.

In the future we are planning to look into the feasibility of RDMA in the wide area network, utilizing the Third Frontier Network [22] already in place in Ohio. We also plan to evaluate soon to appear 10 Gb/s iWarp implementations and to evaluate the scaling effects of multiple-speed iWarp adapters in a single network.

6 Conclusion

In this paper we have shown the performance of iWARP looking at both the message passing capabilities as well as the underlying API performance. It is evident that iWARP is a viable and attractive vessel for message passing, and as such is a good fit for cost sensitive high performance computing where specialty interconnects are not always an option.

7 Acknowledgments

We would like to thank Ammasso Inc. for the evaluation hardware, specifically Michael Cucchi who was instrumental in getting the hardware to us and directing our technical inquires to the right people. We would also like to acknowledge the anonymous referees for reviewing this paper and providing valuable feedback.

References

[1] Ammasso Inc. Ammasso 1100. URL

http://www.ammasso.com/products.htm.

[2] Pavan Balaji, Piyush Shivam, Pete Wyckoff, and D. K. Panda. High performance user-level sockets over gigabit ethernet. In *Proceedings of Cluster '02*, Chicago, IL, September 2002.

[3] R. Brightwell and K. Underwood. An analysis of NIC resource usage for offloading MPI. In *18th International Parallel and Distributed Processing Symposium*, 2004.

[4] The Interconnect Software Consortium. Interconnect transport API (IT-API) issue 1.0. URL http://www.opengroup.org/bookstore/catalog/c040.htm.

[5] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauser, E. Santos, R. Subramoniam, and T. Eicken. LogP: Towards a realistic model of parallel computation. In *Fourth ACM SIGPLAN symposium on Principles and practice of parallel programming*.

[6] P. Culley, U. Elzur, R. Recio, S. Bailey, and J. Carrier. Marker PDU aligned framing for TCP specification, February 2004. URL http://www.ietf.org/internet-drafts/draft-ietf-rddp-mpa-02.txt.

[7] Dennis Dalessandro. Why RDMA? URL http://www.osc.edu/~dennis/rdma/rdma.html.

[8] DAT Collaborative. Direct access transport. URL http://www.datcollaborative.org.

[9] D. Edmondson. ICSC socket API extensions working group. URL http://www.opengroup.org/austin/docs/austin_105.pdf.

[10] Jeff Hilland, Paul Culley, Jim Pinkerton, and Renato Recio. RDMA Protocol Verbs Specification, April 2003. URL http://www.rdmaconsortium.org/home/draft-hilland-iwarp-verbs-v1.0-RDMAC%.pdf.

[11] J. Hurwitz and W. Feng. Initial end-to-end performance evaluation of 10-gigabit ethernet. In *IEEE Hot Interconnects: 11th Symposium on High-Performance Interconnects*, August 2003.

[12] *InfiniBand Architecture Specification*. InfiniBand Trade Association, October 2004. URL http://www.infinibandta.org/specs/.

[13] C. Kurmann, M. Muller, F. Rauch, and T. Stricker. Speculative defragmentation: A technique to improve the communication software efficiency for gigabit ethernet. In *HPDC 2000*, 2000.

[14] Network-Based Computing Laboratory. MPI over Infiniband project. URL http://nowlab.cis.ohio-state.edu/projects/mpi-iba/index.html.

[15] W. Lawry, C. Wilson, A. Maccabe, and R. Brightwell. COMB: A portable benchmark suite for assessing MPI overlap. In *Cluster 2000*, 2000.

[16] J. Liu, J. Wu, S. Kini, P. Wyckoff, and D. K. Panda. High performance RDMA-based MPI implementation over InfiniBand. In *Proceedings of ICS '03*, San Francisco, CA, June 2003.

[17] J. Liu, B. Chandrasekaran, W. Yu, J. Wu, D. Buntinas, S. Kini, D. K. Panda, and P. Wyckoff. Microbenchmark performance comparison of high-speed cluster interconnects. *IEEE Micro*, 24(1):42–51, January/February 2004.

[18] R. Martin, A. Vahdat, D. Culler, and T. Anderson. Effects of communication latency, overhead, and bandwidth in a cluster architecture. In *The 24th Annual International Symposium on Computer Architecutre*, 1997.

[19] MCS/ANL. *MPICH—A Portable Implementation of MPI*. URL http://www-unix.mcs.anl.gov/mpi/mpich/.

[20] *MPI: A Message-Passing Interface Standard*. MPI Forum, March 1994.

[21] Myricom. Myrinet home page. URL http://www.myri.com/.

[22] OSC. Third frontier network: Ohio lights the way, 2005. URL http://www.osc.edu/oarnet/tfn/.

[23] J. Pinkerton. Sockets direct protocol v1.0, October 2003. URL http://www.rdmaconsortium.org/home/SDP_tutorial_v1.0d.pdf.

[24] James Pinkerton, Ellen Deleganes, and Michael Krause. Sockets Direct Protocol (SDP) for iWARP over TCP (v1.0), October 2003. URL http://www.rdmaconsortium.org/home/draft-pinkerton-iwarp-sdp-v1.0.pdf.

[25] Quadrics. Quadrics corporate website. URL http://doc.quadrics.com.

[26] RDMA Consortium. Architectural specifications for RDMA over TCP/IP. URL http://www.rdmaconsortium.org/.

[27] R. Recio, P. Culley, D. Garcia, J. Hilland, and B. Metzler. An RDMA protocol specification, April 2005. URL http://www.ietf.org/internet-drafts/draft-ietf-rddp-rdmap-04.txt.

[28] Hemal Shah, James Pinkerton, Renato Recio, and Paul Culley. Direct data placement over reliable transports, February 2005. URL http://www.ietf.org/internet-drafts/draft-ietf-rddp-ddp-04.txt.

[29] Piyush Shivam, Pete Wyckoff, and D. K. Panda. EMP: zero-copy OS-bypass NIC-driven gigabit ethernet message passing. In *Proceedings of SC '01*, Denver, CO, November 2001.

[30] K. Skevik, T. Plagemann, and V. Groebel. Evaluation of a zero-copy protocol implementation. In *27th Euromicro Conference*, 2001.

[31] Q. Snell, A. Mikler, and J. Gustafson. NetPIPE: A network protocol independent performance evaluator. URL http://www.scl.ameslab.gov/netpipe.

[32] T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: A user-level network interface for parallel and distributed computing. In *Proceedings of SOSP'95*, Copper Mountain, CO, December 1995.

[33] D. Zabrowski. 10 gigabit ethernet market opportunities. URL http://www.s2io.com/news/events/webex/DaveZabrowski.pdf.