Initial Performance Evaluation of the NetEffect 10 Gigabit iWARP Adapter

To appear in Proceedings of IEEE Cluster 2006, RAIT Workshop, Barcelona, Spain, September 2006.

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

Gary Montry
NetEffect Inc.
9211 Waterford Centre Blvd., Suite 100
Austin, TX 78758
gmontry@NetEffect.com

Abstract

Interconnect speeds currently surpass the abilities of today's processors to satisfy their demands. The throughput rate provided by the network simply generates too much protocol work for the processor to keep up with. Remote Direct Memory Access has long been studied as a way to alleviate the strain from the processors. The problem is that until recently RDMA interconnects were limited to proprietary or specialty interconnects that are incompatible with existing networking hardware. iWARP, or RDMA over TCP/IP, changes this situation. iWARP brings all of the advantages of RDMA, but is compatible with existing network infrastructure, namely TCP/IP over Ethernet. The drawback to iWARP up to now has been the lack of availability of hardware capable of meeting the performance of specialty RDMA interconnects. Recently, however, 10 Gigabit iWARP adapters are beginning to appear on the market. This paper demonstrates the performance of one such 10 Gigabit iWARP implementation and compares it to a popular specialty RDMA interconnect, InfiniBand.

Keywords: *RDMA, iWARP, InfiniBand, network performance*

1 Introduction

In the past, the speed of the network has been the major performance bottleneck. With the advent of new networking technologies, and the push toward 10 gigabits per second, the bottleneck is no longer the network. The CPU itself is the reason for limited performance. To process network requests for the de facto networking standard, TCP/IP, the processor must dedicate a large number of cycles and resources to facilitate data transfers. This problem is magnified as the capability of the network expands. While capable of handing network traffic at 1 Gb/s, it is very difficult for a processor to keep up with 10 Gb/s of network processing. At such high data rates, even if the CPU can handle the network processing, the overhead induced by the network limits the availability of the CPU to do other work. This need to dedicate the processor to networking constrains typical designs that overlap communication and computation, such as in message passing parallel programs and multi-client web servers.

Over the past few years there have been a number of approaches to reduce the demands of high-speed networks on the host processor. One such approach is the TCP Offload Engine, or TOE. A TOE simply offloads all network processing to the network adapter, from TCP on down to the bottom of the stack. This is naturally a large help, as it alleviates much of the strain on the processor. Yet the CPU is still responsible for making copies of data and passing it through the operating system boundary, due to the design of the programming interface of such devices, the sockets API. This design thus still has a severe bottleneck, and represents only half of the solution.

A better approach is to offload not only the network processing, but also to provide facilities for user applications to place data directly into the network adapter's buffers, and vice versa. This "zero-copy" mechanism is an integral part of what is known as Remote Direct Memory Access, or RDMA, along with "OS bypass", indicating that the operating system is not involved in the critical path of packet processing. RDMA has long been shown to be an effective solution for high performance interconnects [1, 2, 3, 4]. In fact RDMA is the basis for interconnects such as InfiniBand, Quadrics, and Myrinet [5, 6, 7].

The problem in the past has been the specialized nature of RDMA. The specialty interconnects mentioned previously are built on hardware that is incompatible with the de facto networking protocol today, TCP/IP. This means that these interconnects can not be used in existing networks, including the Internet, without costly protocol conversion or encapsulation. It also means that to utilize these hardware components an entirely new network infrastructure must be adopted. This is the reason that it is highly unlikely that these RDMA interconnects will ever see usage beyond high-end computational clusters. The good news is, a recent set of specifications [8, 9, 10] from the IETF has paved the way for RDMA to work over TCP/IP. These specifications, collectively known as iWARP, enable a standards-based alternative to specialty interconnects such as InfiniBand, while remaining compatible with existing networking protocols and equipment.

This work examines iWARP and demonstrates the performance capabilities of the NetEffect iWARP adapter. We begin by describing iWARP in Section 2, then in Section 3 we explore the background of iWARP in hardware and provide information on the hardware used in this work. We finish that Section with a look at what the future holds for iWARP in Section 3.2. The experimental set up and results are presented in Section 4, along with details on performance considerations in Section 4.1. Finally in Section 5, we explore potential future work, and conclude the paper in Section 6.

2 iWARP Overview

iWARP may, or may not stand for anything, depending on who is asked. It is a convenient name to refer to the set of four main components needed to implement RDMA on TCP/IP.

At the very top of the iWARP stack is the verbs or API layer. There exists a specification [11] from the RDMA Consortium [12] that can be used as a guide to create an API. As is typical in recent networking protocols, there is no exact programming interface, but rather guidelines on the major elements and actions involved in communicating. Ammasso and NetEffect, the first companies to bring an iWARP adapter to market, have created their own APIs based on this verbs specification. However, there are numerous choices for other APIs to interact with an iWARP device, such as DAPL. With the recent inclusion of iWARP into the OpenFabrics project, formerly known as OpenIB, direct support for iWARP devices is available in the Linux kernel, and a common API for both InfiniBand and iWARP devices is made possible. This path for future API standardization seems to be the most promising.

The other three components of the iWARP protocol connect the programming interface to a reliable transport protocol. iWARP relies on an underlying transport, and specifications for using two have been written. Both TCP and SCTP offer the transport functionality needed by iWARP, although with different adaptation requirements as discussed below. Reusing an existing transport layer is one of the major advantages of iWARP, as it permits direct use of the iWARP protocol over the overwhelming majority of existing networks.

The RDMA Protocol layer (RDMAP) [8], sits below the verbs layer. It is responsible for coordinating RDMA data transactions and supplies primitives for remote put and get operations. In order to place data into user application buffers without the need to make a copy of the data, RDMAP relies on the next layer. The Direct Data Placement layer (DDP) [9] moves data between the application's memory space and the iWARP device, without passing the data through the kernel, or making a copy of the data. It uses steering tags provided by the application to refer to memory segments on the remote machine. DDP relies on a reliable, though not necessarily in-order, transport layer that delivers messages intact without requiring reassembly. While SCTP offers this functionality directly, TCP/IP is a byte-streaming protocol that does not have a concept of message boundaries. For DDP to be able to reconstruct the segments, it uses the Marker PDU Aligned (MPA) framing layer [10]. This layer inserts small tags in the data stream at every 512-byte boundary, allowing a receiving DDP layer to discover message boundaries and hence place data directly into user buffers.

With all of these new protocol layers and additions to the network processing stack, it may seem that iWARP would end up being slower and more resource intensive, but all aspects of the iWARP protocol are handled on-board an RDMA Network Interface Controller (RNIC), and not by the host operating system, resulting in very fast processing. In addition to the processing of the RDMA protocol stack, the TCP/IP processing must also be offloaded to the RNIC as well, just as in a TOE.

3 iWARP in Hardware

The first iWARP product to appear on the commodity market was the Ammasso 1100 RNIC, a 1 Gigabit adapter. The Ammasso RNIC was a giant step forward for iWARP, and led to a number of research works [13, 14, 15, 16], including outfitting a cluster with 41 iWARP nodes at OSC's Springfield facility. This unique iWARP platform enables researchers to experiment with iWARP at moderate scale. Unable to compete with the likes of InfiniBand, due to being only a Gigabit adapter, Ammasso has since ceased operations, although development of software to support the Ammasso RNIC adapter continues in the OpenFabrics project.

Next to arrive on the scene is the NetEffect 10 Gigabit iWARP adapter [17]. With this RNIC, iWARP is now able to compete with InfiniBand on equal footing. The performance of these devices is the subject of this paper.

3.1 NetEffect RNIC Description

NetEffects first product is a single-port 10 Gigabit Ethernet channel adapter with a PCI-X 64/133 interface implemented in a structured ASIC with either CX-4 (copper) or SR (fiber) board connectivity. The NetEffect ASIC integrates NIC, TOE and iWARP logic entirely in hardware. This enables the NetEffect adapter to support RDMA communication and OS by-pass through the TCP/IP-based iWARP standards. The ASIC is deeply pipelined with independent transmit and receive engines mainly implemented in hardware state machines. This approach provides maximum simultaneous bandwidth in both directions with relatively low power and clock rate logic. The memory-based switch inside the chip implies that packet data motion is kept to a minimum during processing to optimize packet throughput. Separation of header and payload on packet ingress allows the engine to always have the connection information available in cache. These hardware features allow the NetEffect adapter to attain very impressive packet throughput rates for Ethernet protocols.

3.2 Future of iWARP

As the following sections will show, iWARP is clearly a viable interconnect, although at present the cost of iWARP adapters and 10 Gigabit Ethernet switches prohibits its use in many scenarios. Yet despite the high initial cost of iWARP, particularly the switch cost, it is important to recall the oft-demonstrated nature of Ethernet technology costs to decrease rapidly over a short time period after introduction.

One area that iWARP certainly has an advantage in is wide-area networking. Since WANs are built on TCP/IP infrastructure, iWARP is designed directly to work in the WAN. Early adopters of iWARP for use in the WAN will likely be enterprise-level servers where the performance benefits of iWARP warrant the high cost, such as in web servers or database engines. Previous work with clients using software to communicate with hardware iWARP [16, 15, 18] has shown it is possible for clients to emulate iWARP in software. This enables a server equipped with an iWARP adapter to still take advantage of the RNIC, even if the client side is not so equipped. By using a software implementation on the clients, the server is able to reduce its processing load per connection and can thus handle many more clients. Freely available software exists [19], easing the widespread adoption of iWARP technology in incremental steps.

4 Experimental Results

This section explains the experiments conducted, and the results obtained, to gain initial insight into the performance of the NetEffect 10 Gigabit iWARP adapter. We have chosen InfiniBand to compare iWARP with as InfiniBand is one of the most popular commodity cluster interconnects, and hardware was easily obtained. The test environment consists of two severs running Fedora Core 4 Linux. The hardware specifications of both servers are the same, using a Supermicro H8DC8 motherboard with dual Opteron 246 processors and 3 gigabytes of RAM. The NetEffect RNIC is a PCI-X card running at 133MHz, and the InfiniBand is a Mellanox 4X memfree HCA (MHES14), in an x8 PCI express slot.

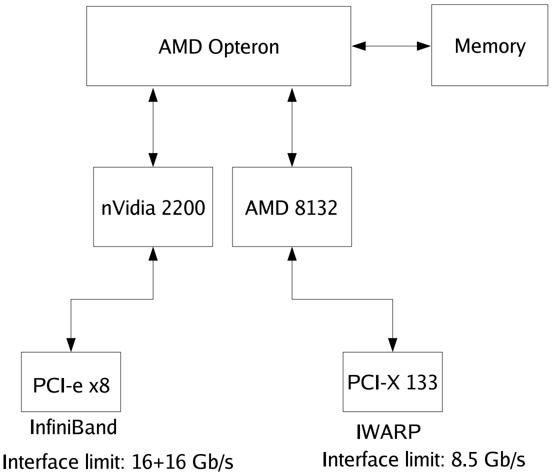

Figure 1. Hardware configuration

4.1 Performance Considerations

There are a few hardware issues to be kept in mind when considering the following experiments. The iWARP cards are connected back to back via a CX4 (copper) cable, while the InfiniBand cards are connected through an InfiniBand switch via a CX4 cable. We use a 24-port OEM switch based on InfiniScale III silicon. InfiniBand switches add very little overhead, less than 200 nanoseconds usually. While an Ethernet switch was not available at the time of this study, there are at least two on the market that feature cut-through operation and similar per-packet latencies. We plan to rerun the tests with one switch shortly after the paper deadline.

Figure 1 shows the internal system configuration. The InfiniBand cards are connected to the host bus via a PCI-Express x8 interface that allows 16 Gb/s of data transfer in each direction. The NetEffect card, however, is connected via a PCI-X 133 MHz interface that limits transfers to 8.5 Gb/s and only in one direction at a time. We also expect the latency of bus transfers on PCI-X to exceed their counterparts on PCI-Express by a few hundred nanoseconds.

Another difference between the two devices is the wire speed. While InfiniBand 4x is marketed as a 10 Gigabit device, the user payload present on the network is only 8 Gb/s due to the use of 8b/10b encoding. 10 Gigabit ethernet, however, delivers 10 Gb/s of user payload while the signaling speed on the network is 12.5 Gb/s. (Both networks do impose a small and similar amount of packet overhead in the user payload.)

Software versions used for the tests are: NetEffect Alpha 2 device library, Mellanox IBGD-1.8.2 device library. The MPI libraries are mpich2 version 1.0.3 with a device written by NetEffect for their RNIC, and mvapich-0.9.5-mlx1.0.3 for InfiniBand.

4.2 Benchmark Software

The software use to gather the data presented in sections 4.3 and 4.4 is part of a package known as iWarpPerf. iWarpPerf is freely available [20], and includes native support for the NetEffect RNIC, as well as the Ammasso RNIC, and even includes MPI support. Since there is an MPI version we can conduct the same tests on InfiniBand as well. The two main tests that this software conducts are a bandwidth and latency test.

The bandwidth test entails sending a configurable number of messages (window size) via RDMA write, of a particular size. Once these have completed, the recipient replies with a small (1 byte) reply. Based on the amount of data sent and the time it took, we calculate the bandwidth. This is a better representation of bandwidth than calculating the inverse from latency. For the native iWARP versions we post window size number of RDMA write calls and poll on the last byte of the buffer. In the MPI version we use MPI_ISend (and receives) to accomplish the same thing, and rather than polling on the buffer we use MPI's built in capabilities (MPI_Waitall). As in the native version, a small 1 byte reply is sent, this time with regular MPI_Send and receive.

Latency, as we define it, is half of the round-trip time to send a message and get a response of equal size. We implement a ping-pong type benchmark using RDMA writes for the native versions, and again polling on the last byte of the buffer. As for the MPI version, we use MPI_Send and receive to coordinate which works well for the latency test.

In order to gage CPU utilization in Section 4.5 we use a tool known as COMB [21], commonly referred to as busymove. This tool generates a known quantity of work and conducts concurrent network transfers. Based on the amount of work done, COMB is able to calculate the amount of CPU available for processes other than network communication related. Since COMB is implemented in MPI it can be used for both iWARP and InfiniBand tests.

The last piece of software used in the experiments is iWarpMemReg [20]. This test, despite the name is also available for InfiniBand. iWarpMemReg conducts a number of memory registration tests and computes an average and standard deviation. There are a number of modes, which are designed to test registration and deregistration only, as well as the combination of registration and deregistration. This software is written in native NetEffect verbs API, as well as the native InfiniBand API, commonly known as VAPI.

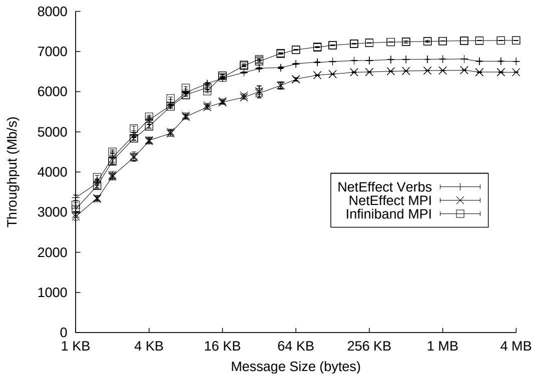

Figure 2. Bandwidth Comparisons

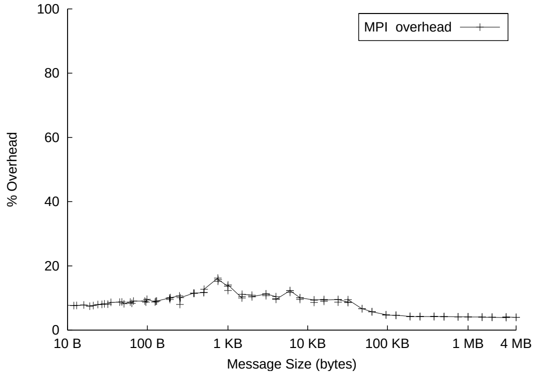

Figure 3. MPI Overhead

4.3 Bandwidth

One of the most basic performance metrics is the throughput or bandwidth. For many, this is the number they look for first when gauging the performance of an interconnect. We present the bandwidth of iWARP and InfiniBand below.

Looking at Figure 2 we see that the bandwidth for the code written in the NetEffect verbs API is pretty comparable to the bandwidth for InfiniBand. Infact bandwidth is basically the same up to 16KB messages. We show only messages of 1KB and larger. The rationale for this is at smaller message sizes, latency is the dominant factor, not how much data can be pushed through. Even though iWARP in this case is 10 gigabit, due to PCI-X bus limitations, mentioned previously, it is not possible to attain this. Since the InfiniBand device is plugged into a bus which has more bandwdith than the device can supply, the InfiniBand device is able to reach full bandwidth. The interesting fact is that even on a slower, more limited bus, comparable performance can be realized with iWARP.

The NetEffect verbs code is a bit better performing than the iWARP MPI, which makes perfect sense, since MPI is

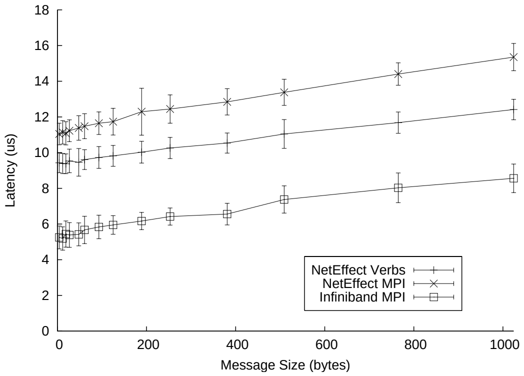

Figure 4. Latency Comparisons

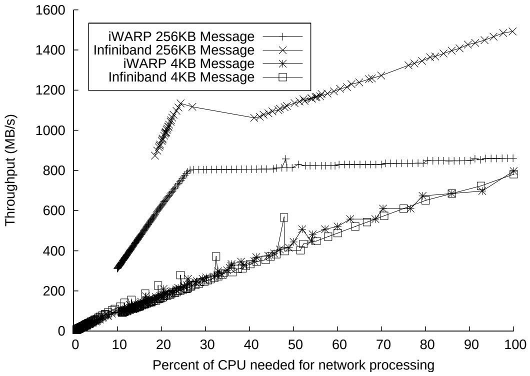

Figure 5. CPU Usage

yet another layer on top of the verbs API. We show the overhead associated with MPI in Figure 3. In previous work with gigabit iWARP [13], we found the overhead for MPI to be relatively the same, around 10%. We also found that TCP at 1 Gigabit had an MPI overhead that was considerably higher, around 20%. One can infer similar overheads for 10 Gigabit TCP.

4.4 Latency

The other common benchmark that many are interested in is latency. As Figure 4 shows, InfiniBand has quite an advantage where latency is concerned, at least for small messages. Based on the data in the previous section, we know that when it comes to larger message sizes, iWARP will eventually perform about as good as InfiniBand. If iWARP were compared in a siliar bus it is possbile iWARP could even outperform InfiniBand. This larger latency may be less attractive for clustering, but has little impact when running in the WAN over large distances. This small, on the order of microsecond, latency gets washed out by the many-millisecond delays encountered in the WAN.

4.5 CPU Utilization

Since one of the main benefits that RDMA brings is reduced strain on the server, we should now turn our attention to the CPU utilization. It has been shown in our previous work [13] that the bandwidth of TCP decreases linearly as the amount of CPU used for other work increases. In agreement with what we found in that previous work, in Figure 5 we see that iWARP only needs around 80% of the CPU in order to maintain full bandwidth. Here we show the amount of CPU required to attain a certain bandwidth, and we see that once iWARP has 20% of the available CPU time, iWARP is able to maintain excellent bandwidth rates. Even with only 10% of the processor iWARP is still able to maintain 400 MB/s. These observations apply to large messages, of 256 kB. When we consider small messages, we end up with a fairly linear progression, which is to be expected as the adapter handles most of the work for large messages. At the smaller message end, both iWARP and InfiniBand require at least some CPU time for posting descriptors, clearing completion events, registering memory, and other operations.

COMB, the software used for this experiment, involves sending data in a bi-directional manner, explaining why the InfiniBand throughput exceeds the theoretical unidirectional maximum. The reason that we see no bi-directional benefit for iWARP here is due to the PCI-X bus limitation mentioned earlier. Why the InfiniBand bi-directional throughput has a slight downward slope as CPU time decreases is not yet know, and we plan to explore this in more detail.

4.6 Memory Registration

Lastly we should pay attention to another important performance consideration, and that is the cost of registering memory. Every application must register memory, and the costs associated with this are very important in the design and implementation of software. Looking at Figure 6, we see that iWARP imposes less overhead for registering memory than does InfiniBand. The InfiniBand line shows the familiar stair-step shape seen in previous work [13, 22]. The reason for the jumps in the graph is the need for extra bus messages needed to send the growing physical buffer list down to the network adapter. The iWARP trace shows the same jumps although they are not as pronounced at the resolution in the figure.

We also see that memory deregistration is nearly constant and unaffected by the size of the region. This is also not surprising as previous work has shown similar results. Again, there is less overhead for the iWARP device.

5 Future Work

This work represents the first evaluation of an actual 10 Gigabit iWARP device and lays the ground work for continued usage of iWARP. We are planning to pursue an eval-

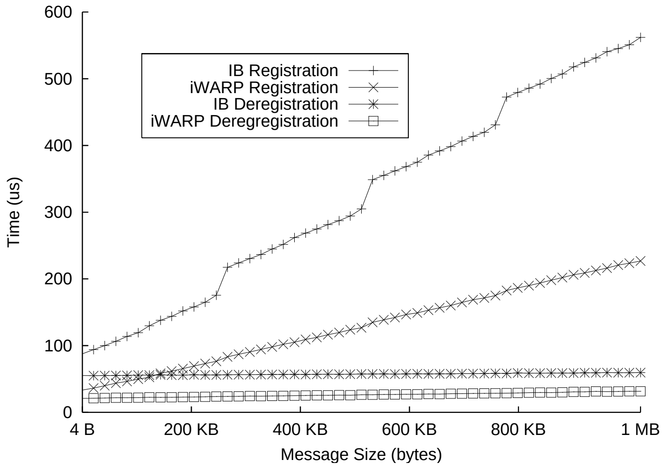

Figure 6. Memory registration

uation of iWARP as a clustering interconnect, as well as a number of other research projects. Plans are in progress to place 10 Gigabit iWARP hardware at both of OSC's centers and connect over a high speed (10 Gigabit) WAN. We are also interested in interoperability, both with other iWARP hardware and software iWARP.

One particular aspect of the NetEffect iWARP adapter that we plan to investigate in the near future is the ability of iWARP to perform exceptionally well with multiple connections. Early tests show iWARP to have a significant advantage over InfiniBand in this respect. The capability for iWARP to handle many connections is quite important for the role iWARP is likely to play both in clustering architectures and in high-end servers with thousands of connections.

In directly related work, we plan to pursue evalautions of the NetEffect adapter against other iWARP adapters as they become available. We are also interested in looking at the performance benefit gained by iWARP over a TCP Offload Engine (TOE), as well as more detailed comparisons with InfiniBand.

6 Conclusion

In this paper we have shown the basic performance of 10 Gigabit iWARP and that it is a worthy alternative to InfiniBand. While InfiniBand may start with lower latency, iWARP wins the battle of bandwidth for larger data sizes. Given the low CPU demands of iWARP it is hopeful that iWARP will relieve the problem of Ethernet scaling beyond 1 Gb/s.

References

[1] Dennis Dalessandro and Pete Wyckoff. Fast Scalable File Distribution Over InfiniBand. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Workshop on System Management Tools for Large Scale Parallel Systems*, Denver, CO, April 2005.

[2] P. Buonadonna, A. Geweke, and D. Culler. An implementation and analysis of the Virtual Interface Architecture. In *Proceedings of SC'98*, San Jose, CA, November 1998.

[3] C. Csanady and P. Wyckoff. Bobnet: High-performance message passing for commodity networking components. In *Proceedings of PDCN*, December 1998.

[4] 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.

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

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

[7] InfiniBand Trade Association. *InfiniBand Architecture Specification*, October 2004.

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

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

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

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

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

[13] Dennis Dalessandro, Pete Wyckoff. A Performance Analysis of the Ammasso RDMA Enabled Ethernet Adapter and its iWARP API. In *Proceedings of the IEEE Cluster 2005 Conference, RAIT Workshop*, Boston, MA, September 2005.

[14] Dennis Dalessandro. RDMA Over TCP/IP: The Next

Step in Ethernet Technology. In *Proceedings of the 2005 Commodity Cluster Symposium: On the Use of Commodity Clusters for Large-Scale Scientific Applications*, Greenbelt, MD, July 2005.

[15] Dennis Dalessandro, Pete Wyckoff, Ananth Devulapalli. iWarp Protocol Kernel Space Software Implementation. In *Proceedings of the 20th IEEE International Parallel and Distributed Processing Symposium (IPDPS '06), Communication Architectures for Clusters Workshop*, Rhodes, Greece, April 2006.

[16] Dennis Dalessandro, Pete Wyckoff, Ananth Devulapalli. Design and Implementation of the iWarp Protocol in Software. In *Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and Systems*, Phoenix, AZ, November 2005.

[17] NetEffect Inc. Neteffect corporate website. http://www.neteffect.com, 2006.

[18] P. Balaji, H.-W. Jin, K. Vaidyanathan, and D. K. Panda. Supporting iWARP compatibility and features for regular network adapters. In *Proceedings of the IEEE Cluster 2005 Conference, RAIT workshop*, Boston, MA, September 2005.

[19] Dennis Dalessandro and Ananth Devulapalli and Pete Wyckoff. Software iWarp user and kernel software distribution. http://www.osc.edu/research/network_file/projects/iwarp/index.shtml, 2005.

[20] Dennis Dalessandro. iWarp resources. http://www.osc.edu/~dennis/iwarp/index.html, 2005.

[21] W. Lawry, C. Wilson, A. Maccabe, and R. Brightwell. COMB: A portable benchmark suite for assessing MPI overlap. In *Proceedings of the IEEE Cluster 2000 Conference*, September 2000.

[22] Pete Wyckoff and Jiesheng Wu. Memory registration caching correctness. In *Proceedings of CCGrid'05*, Cardiff, UK, May 2005.