Proc. of the 13th IEEE Symp. on High-Performance Interconnects (Hot Interconnects 2005), Palo Alto, CA, August 2005.
Also available as Los Alamos technical report LA-UR-05-2635.

Performance Characterization of a 10-Gigabit Ethernet TOE*

W. Feng[†] P. Balaji[‡] C. Baron[§] L. N. Bhuyan[§] D. K. Panda[‡]

[†]Advanced Computing Lab, Los Alamos National Lab
feng@lanl.gov

[‡]Comp. Sci. and Engg. Ohio State University
{balaji, panda}@cse.ohio-state.edu

[§]Comp. Sci. and Engg. UC Riverside
{cbaron, bhuyan}@cs.ucr.edu

Abstract

Though traditional Ethernet based network architectures such as Gigabit Ethernet have suffered from a huge performance difference as compared to other high performance networks (e.g, InfiniBand, Quadrics, Myrinet), Ethernet has continued to be the most widely used network architecture today. This trend is mainly attributed to the low cost of the network components and their backward compatibility with the existing Ethernet infrastructure. With the advent of 10-Gigabit Ethernet and TCP Offload Engines (TOEs), whether this performance gap be bridged is an open question.

In this paper, we present a detailed performance evaluation of the Chelsio T110 10-Gigabit Ethernet adapter with TOE. We have done performance evaluations in three broad categories: (i) detailed micro-benchmark performance evaluation at the sockets layer, (ii) performance evaluation of the Message Passing Interface (MPI) stack atop the sockets interface, and (iii) application-level evaluations using the Apache web server. Our experimental results demonstrate latency as low as 8.9 μs and throughput of nearly 7.6 Gbps for these adapters. Further, we see an order-of-magnitude improvement in the performance of the Apache web server while utilizing the TOE as compared to the basic 10-Gigabit Ethernet adapter without TOE.

1 Introduction

Despite the performance criticisms of Ethernet for high-performance computing (HPC), the Top500 Supercomputer List [2] continues to move towards more commodity-based Ethernet clusters. Just three years ago, there were *zero* Gigabit Ethernet-based clusters in the Top500 list; now, Gigabit Ethernet-based clusters make up 176 (or 35.2%) of these. The primary drivers of this Ethernet trend are ease of deployment and cost. So, even though the end-to-end throughput and latency of Gigabit Ethernet (GigE) lags exotic high-speed networks such as Quadrics [20], Myrinet [9], and InfiniBand [4] by as much as ten-fold, the current trend indicates that GigE-based clusters will soon make up over half of the Top500 (as early as November 2005). Further, Ethernet is already the ubiquitous interconnect technology for commodity grid computing because it leverages the legacy Ethernet/IP infrastructure whose roots date back to the mid-1970s. Its ubiquity will become even more widespread as long-haul network providers move towards 10-Gigabit Ethernet (10GigE) [14, 13] backbones, as recently demonstrated by the longest continuous 10GigE connection between Tokyo, Japan and Geneva, Switzerland via Canada and the United States [12]. Specifically, in late 2004, researchers from Japan, Canada, the United States, and Europe completed an 18,500-km 10GigE connection between the Japanese Data Reservoir project in Tokyo and the CERN particle physical laboratory in Geneva; a connection that used 10GigE WAN PHY technology to set-up a *local-area network* at the University of Tokyo that appeared to include systems at CERN, which were 17 time zones away.

Given that GigE is so far behind the curve with respect to network performance, can 10GigE bridge the performance divide while achieving the ease of deployment and cost of GigE? Arguably yes. The IEEE 802.3-ae 10-Gb/s standard, supported by the 10GigE Alliance, already ensures interoperability with existing Ethernet/IP infrastructures, and the manufacturing volume of 10GigE is already driving costs down exponentially, just as it did for Fast Ethernet and Gigabit Ethernet[1]. This leaves us with the "performance divide" between 10GigE and the more exotic network technologies.

In a distributed grid environment, the performance difference is a non-issue mainly because of the ubiquity of Ethernet and IP as the routing language of choice for local-, metropolitan, and wide-area networks in support of grid computing. Ethernet has become synonymous with IP for these environments, allowing complete compatibility for clusters using Ethernet to communicate over these environments. On the other hand, networks such as Quadrics, Myrinet, and InfiniBand are unusable in such environments due to their incompatibility with Ethernet and due to their limitations against using the IP stack in order to maintain a high performance.

With respect to the cluster environment, Gigabit Ethernet suffers from an order-of-magnitude performance penalty when compared to networks such as Quadrics and InfiniBand. In our previous work [14, 13, 6], we had demonstrated the capabilities of the basic 10GigE adapters in bridging this gap. In this paper, we take the next step by demonstrating the capabilities of the Chelsio T110 10GigE adapter with TCP Offload Engine (TOE). We present performance evaluations in three broad categories: (i) detailed micro-benchmark performance evaluation at the sockets layer, (ii) performance evaluation of the Message Passing Interface (MPI) [19] stack atop the sockets interface, and (iii) application-level evaluation using the Apache web server [1]. Our experimental results demonstrate

*This work was supported by the DOE ASC Program through Los Alamos National Laboratory contract W-7405-ENG-36, and technical and equipment support from Chelsio Communications and Fujitsu.

[1]Per-port costs for 10GigE have dropped nearly ten-fold in two years.

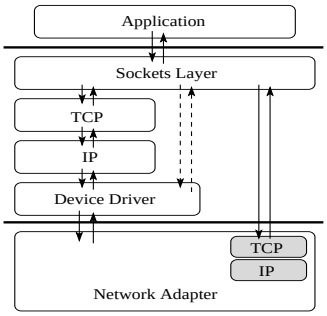


Figure 1. TCP Offload Engines

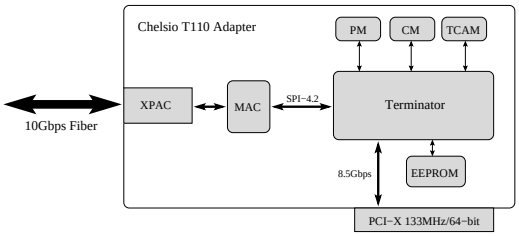


Figure 2. Chelsio T110 Adapter Architecture

latency as low as 8.9 μs and throughput of nearly 7.6 Gbps for these adapters. Further, we see an order-of-magnitude improvement in the performance of the Apache web server while utilizing the TOE as compared to a 10GigE adapter without TOE.

2 Background

In this section, we briefly discuss the TOE architecture and provide an overview of the Chelsio T110 10GigE adapter.

2.1 Overview of TCP Offload Engines (TOEs)

The processing of traditional protocols such as TCP/IP and UDP/IP is accomplished by software running on the central processor, CPU or microprocessor, of the server. As network connections scale beyond GigE speeds, the CPU becomes burdened with the large amount of protocol processing required. Resource-intensive memory copies, checksum computation, interrupts, and reassembling of out-of-order packets put a tremendous amount of load on the host CPU. In high-speed networks, the CPU has to dedicate more processing to handle the network traffic than to the applications it is running. TCP Offload Engines (TOEs) are emerging as a solution to limit the processing required by CPUs for networking.

The basic idea of a TOE is to offload the processing of protocols from the host processor to the hardware on the adapter or in the system (Figure 1). A TOE can be implemented with a network processor and firmware, specialized ASICs, or a combination of both. Most TOE implementations available in the market concentrate on offloading the TCP and IP processing, while a few of them focus on other protocols such as UDP/IP.

As a precursor to complete protocol offloading, some operating systems started incorporating support for features to offload some compute-intensive features from the host to the underlying adapter, e.g., TCP/UDP and IP checksum offload. But as Ethernet speeds increased beyond 100 Mbps, the need for further protocol processing offload became a clear requirement. Some GigE adapters complemented this requirement by offloading TCP/IP and UDP/IP segmentation or even the whole protocol stack onto the network adapter [15, 11].

2.2 Chelsio 10-Gigabit Ethernet TOE

The Chelsio T110 is a PCI-X network adapter capable of supporting full TCP/IP offloading from a host system at line speeds of 10 Gbps. The adapter consists of multiple components: the Terminator which provides the basis for offloading, separate memory systems each designed for holding particular types of data, and a MAC and XPAC Optical Transceiver for physically transferring data over the line. An overview of the T110's architecture can be seen in Figure 2.

Context (CM) and Packet (PM) memory are available on-board as well as a 64 KB EEPROM. A 4.5 MB TCAM is used to store a Layer 3 routing table and can filter out invalid segments for non-offloaded connections. The T110 is a Terminator ASIC, which is the core of the offload engine, capable of handling 64,000 connections at once, with a setup and tear-down rate of about 3 million connections per second.

Memory Layout: Two types of on-board memory are available to the Terminator. 256 MB of EFF FCRAM Context Memory stores TCP state information for each offloaded and protected non-offloaded connection as well as a Layer 3 routing table and its associated structures. Each connection uses 128 bytes of memory to store state information in a TCP Control Block. For payload (packets), standard ECC SDRAM (PC2700) can be used, ranging from 128 MB to 4 GB.

Terminator Core: The Terminator sits between a systems host and its Ethernet interface. When offloading a TCP/IP connection, it can handle such tasks as connection management, checksums, route lookup from the TCAM, congestion control, and most other TCP/IP processing. When offloading is not desired, a connection can be tunneled directly to the host's TCP/IP stack. In most cases, the PCI-X interface is used to send both data and control messages between the host, but an SPI-4.2 interface can be used to pass data to and from a network processor (NPU) for further processing.

3 Interfacing with the TOE

Since the Linux kernel does not currently support TCP Offload Engines (TOEs), there are various approaches researchers have taken in order to allow applications to interface with TOEs. The two predominant approaches are High Performance Sockets (HPS) [21, 17, 18, 7, 8, 5, 16] and TCP Stack Override. The Chelsio T110 adapter uses the latter approach.

In this approach, the kernel-based sockets layer is retained and used by the applications. However, the TCP/IP stack is overridden, and the data is pushed directly to the offloaded protocol stack, bypassing the host TCP/IP stack implementation. One of Chelsio's goals in constructing a TOE was to keep it from being too invasive to the current structure of the

system. By adding kernel hooks inside the TCP/IP stack and avoiding actual code changes, the current TCP/IP stack remains usable for all other network interfaces, including loopback.

The architecture used by Chelsio essentially has two software components: the TCP Offload Module and the Offload driver.

TCP Offload Module: As mentioned earlier, the Linux operating system lacks support for TOE devices. Chelsio provides a framework of a TCP offload module (TOM) and a thin layer known as the *toedev* which decides whether a connection needs to be handed over to the TOM or to the traditional host-based TCP/IP stack. The TOM can be thought of as the upper layer of the TOE stack. It is responsible for implementing portions of TCP processing that cannot be done on the TOE (e.g., TCP TIME_WAIT processing). The state of all offloaded connections is also maintained by the TOM. Not all of the Linux network API calls (e.g., tcp_sendmsg, tcp_recvmsg) are compatible with offloading to the TOE. Such a requirement would result in extensive changes in the TCP/IP stack. To avoid this, the TOM implements its own subset of the transport layer API. TCP connections that are offloaded have certain function pointers redirected to the TOM's functions. Thus, non-offloaded connections can continue through the network stack normally.

Offload Driver: The offload driver is the lower layer of the TOE stack. It is directly responsible for manipulating the Terminator and its associated resources. TOEs have a many-to-one relationship with a TOM. A TOM can support multiple TOEs as long as it provides all functionality required by each. Each TOE can only be assigned one TOM. More than one driver may be associated with a single TOE device. If a TOE wishes to act as a normal Ethernet device (capable of only inputting/outputting Layer 2 level packets), a separate device driver may be required.

4 Experimental Evaluation

In this section, we evaluate the performance achieved by the Chelsio T110 10GigE adapter with TOE. In Section 4.1, we perform evaluations on the native sockets layer; in Section 4.2, we perform evaluations of the Message Passing Interface (MPI) stack atop the sockets interface; and in Section 4.3, we evaluate the Apache web server as an end application.

We used two clusters for the experimental evaluation. **Cluster 1** consists of two Opteron 248 nodes, each with a 2.2-GHz CPU along with 1 GB of 400-MHz DDR SDRAM and 1 MB of L2-Cache. These nodes are connected back-to-back with Chelsio T110 10GigE adapters with TOEs. **Cluster 2** consists of four Opteron 846 nodes, each with four 2.0-GHz CPUs (quad systems) along with 4 GB of 333-MHz DDR SDRAM and 1 MB of L2-Cache. It is connected with similar network adapters (Chelsio T110 10GigE-based TOEs) but via a 12-port Fujitsu XG1200 10GigE switch (with a latency of approximately 450 ns and capable of up to 240 Gbps of aggregate throughput). The experiments on both the clusters were performed with the SuSE Linux distribution installed with kernel.org kernel 2.6.6 (patched with Chelsio TCP Offload modules). In general, we have used Cluster 1 for all experiments requiring only two nodes and Cluster 2 for all experiments requiring more nodes. We will be pointing out the cluster used for each experiment throughout this section.

For optimizing the performance of the network adapters, we have modified several settings on the hardware as well as the software systems, e.g., (i) increased PCI burst size to 2 KB, (ii) increased send and receive socket buffer sizes to 512 KB each, and (iii) increased window size to 10 MB. Detailed descriptions about these optimizations and their impacts can be found in our previous work [14, 13, 6].

4.1 Sockets-level Evaluation

In this section, we evaluate the performance of the native sockets layer atop the TOEs as compared to the native host-based TCP/IP stack. We perform micro-benchmark level evaluations in two sub-categories. First, we perform evaluations based on a single connection measuring the point-to-point latency and uni-directional throughput together with the CPU utilization. Second, we perform evaluations based on multiple connections using the multi-stream, hot-spot, fan-in and fan-out tests.

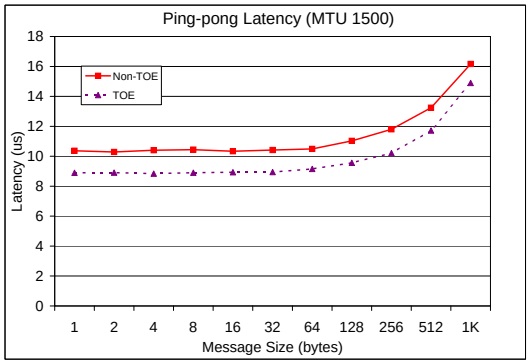
4.1.1 Single Connection Micro-Benchmarks

Figures 3 and 4 show the basic single-stream performance of the 10GigE TOE as compared to the traditional host-based TCP/IP stack. All experiments in this section have been performed on Cluster 1 (described in Section 4).

Figure 3a shows that the TCP Offload Engines (TOE) can achieve a point-to-point latency of about 8.9 μs as compared to the 10.37 μs achievable by the host-based TCP/IP stack (non-TOE); an improvement of about 14.2%. Figure 3b shows the uni-directional throughput achieved by the TOE as compared to the non-TOE. As shown in the figure, the TOE achieves a throughput of up to 7.6 Gbps as compared to the 5 Gbps achievable by the non-TOE stack (improvement of about 52%). Throughput results presented throughout this paper refer to the application data transferred per second and do not include the TCP/IP/Ethernet headers.

Increasing the MTU size of the network adapter to 9 KB (Jumbo frames) improves the performance of the non-TOE stack to 7.2 Gbps (Figure 4b). There is no additional improvement for the TOE due to the way it handles the message transmission. For the TOE, the device driver hands over large message chunks (16 KB) to be sent out. The actual segmentation of the message chunk to MTU-sized frames is carried out by the network adapter. Thus, the TOE shields the host from the overheads associated with smaller MTU sizes. On the other hand, for the host-based TCP/IP stack (non-TOE), an MTU of 1500 bytes results in more segments and correspondingly more interrupts to be handled for every message causing a lower performance as compared to Jumbo frames.

We also show the CPU utilization for the different stacks. For TOE, the CPU remains close to 35% for large messages. However, for the non-TOE, the CPU utilization increases



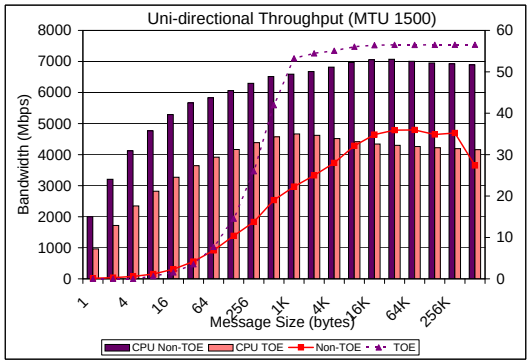
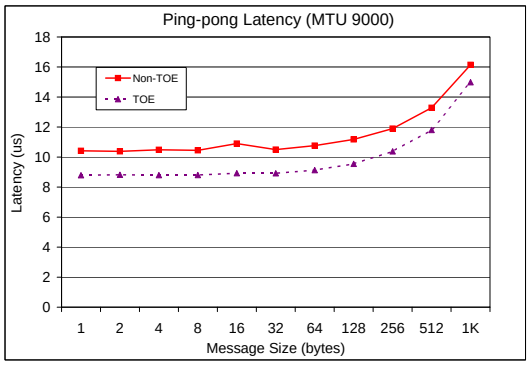


Figure 3. Sockets-level Micro-Benchmarks (MTU 1500): (a) Latency and (b) Throughput



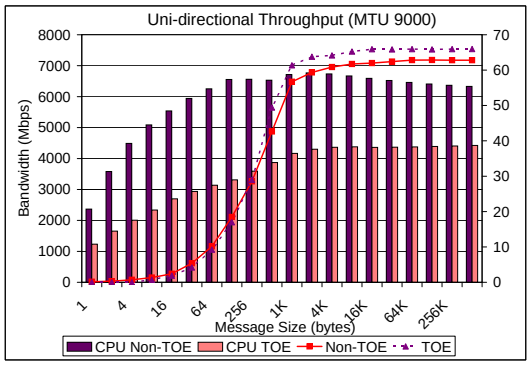


Figure 4. Sockets-level Micro-Benchmarks (MTU 9000): (a) Latency and (b) Throughput

slightly on using jumbo frames. To understand this behavior, we reiterate on the implementation of these stacks. When the application calls a `write()` call, the host CPU copies the data into the socket buffer. If there is no space in the socket buffer, the CPU waits for the network adapter to complete sending out the existing data and creating space for the new data to be copied. Once the data is copied, the underlying TCP/IP stack handles the actual data transmission. Now, if the network adapter pushes the data out faster, space is created in the socket buffer faster and the host CPU spends a larger fraction of its time in copying data to the socket buffer than waiting for space to be created in the socket buffer. Thus, in general when the performance increases, we expect the host CPU to be spending a larger fraction of time copying data and burning CPU cycles. However, the usage of Jumbo frames reduces the CPU overhead for the host-based TCP/IP stack due to reduced number of interrupts. With these two conditions, on the whole, we see about a 10% increase in the CPU usage with Jumbo frames.

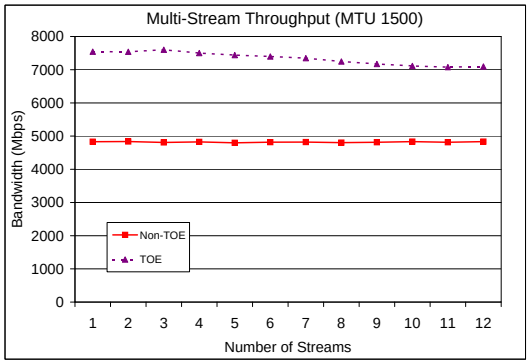
4.1.2 Multiple Connection Micro-Benchmarks

Here we evaluate the TOE and non-TOE stacks with micro-benchmarks utilizing multiple simultaneous connections. For all experiments in this section, we utilize an MTU of 1500 bytes in order to stick to the standard Ethernet frame size.

Multi-stream Throughput Test: Figure 5a shows the aggregate throughput achieved by two nodes (in Cluster 1) performing multiple instances of uni-directional throughput tests. We see that the TOE achieves a throughput of 7.1 to 7.6 Gbps. The non-TOE stack gets saturated at about 4.9 Gbps. These results are similar to the single stream results; thus using multiple simultaneous streams to transfer data does not seem to make much difference.

Hot-Spot Latency Test: Figure 5b shows the impact of multiple connections on small message transactions. In this experiment, a number of client nodes perform a point-to-point latency test with the same server forming a hot-spot on the server. We performed this experiment on Cluster 2 with one node acting as a server node and each of the other three 4-processor nodes hosting totally 12 client processes. The clients are alloted in a cyclic manner, so 3 clients refers to 1 client on each node, 6 clients refers to 2 clients on each node and so on. As seen in the figure, both the non-TOE as well as the TOE stacks show similar scalability with increasing number of clients, i.e., the performance difference seen with just one client continues with increasing number of clients. This shows that the look-up time for connection related data-structures is performed efficiently enough on the TOE and does not form a significant bottleneck.

Fan-out and Fan-in Tests: With the hot-spot test, we have shown that the lookup time for connection related data-structures is quite efficient on the TOE. However, the hot-spot test does not stress the other resources on the network adapter such as management of memory regions for buffering data during transmission and reception. In order to stress such resources, we have designed two other tests namely fan-out and fan-in. In both these tests, one server process carries out uni-directional throughput tests simultaneously with a num-



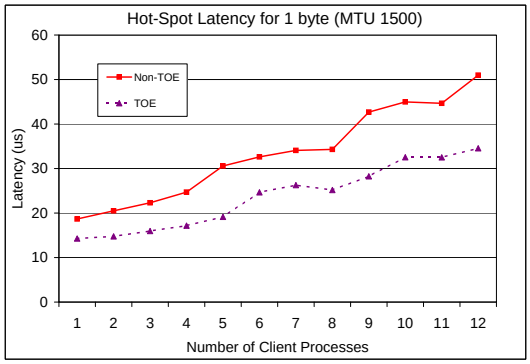
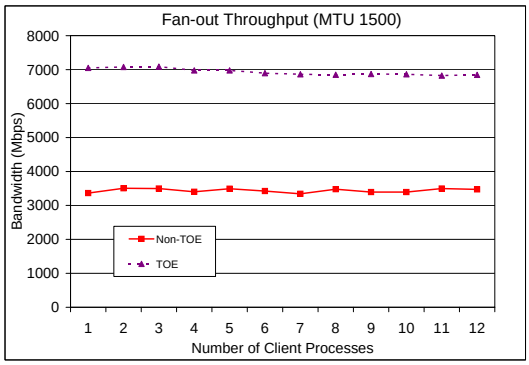


Figure 5. (a) Multi-stream Throughput and (b) Hot-Spot Latency



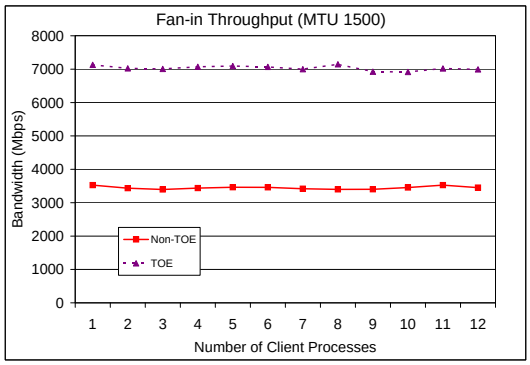


Figure 6. (a) Fan-out Test and (b) Fan-in Test

ber of client threads (performed on Cluster 2). The difference being that in a fan-out test the server pushes data to the different clients (stressing the transmission path on the network adapter) and in a fan-in test the clients push data to the server process (stressing the receive path on the network adapter). Figure 6 shows the performance of the TOE stack as compared to the non-TOE stack for both these tests. As seen in the figure, the performance for both the fan-out and the fan-in tests is quite consistent with increasing number of clients suggesting an efficient transmission and receive path implementation.

4.2 MPI-level Evaluation

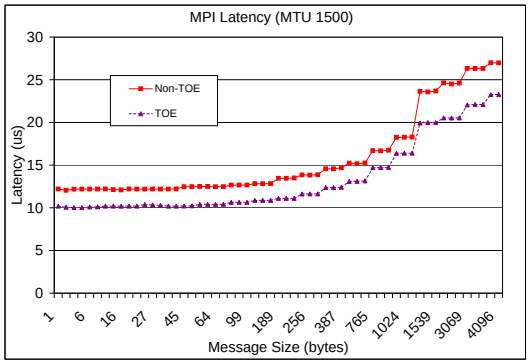
In this section, we evaluate the Message Passing Interface (MPI) stack written using the sockets interface on the TOE and non-TOE stacks. MPI is considered the *de facto* standard programming model for scientific applications; thus this evaluation would allow us to understand the implications of the TOE stack for such applications. We used the LAM [10] implementation of MPI for this evaluation.

Figure 7 illustrates the point-to-point latency and uni-directional throughput achievable with the TOE and non-TOE stacks for an MTU size of 1500 bytes. As shown in Figure 7a, MPI over the TOE stack achieves a latency of about 10.2 μs compared to the 12.2 μs latency achieved by the non-TOE stack. The increased point-to-point latency of the MPI stack as compared to that of the native sockets layer (8.9 μs) is attributed to the overhead of the MPI implementation. Figure 7b shows the uni-directional throughput achieved by the two stacks. TOE achieves a throughput of about 6.9 Gbps as compared to the 3.1 Gbps achieved by the non-TOE stack.

4.3 Application-level Evaluation

In this section, we evaluate the performance of the stacks using a real application, namely the Apache web server. One node is used as a web-server and three nodes to host up to 24 client processes.

In the first experiment (Figure 8a), we use a simulated trace consisting of only one file. Evaluating the stacks with various sizes for this file lets us understand their performance without being diluted by other system parameters. As seen in the figure, the TOE achieves a significantly better performance as compared to the non-TOE especially for large files. In the next experiment (Figure 8b), we build a trace based on the popular Zipf [22] file request distribution. The Zipf distribution states that the probability of requesting the I^{th} most popular document is inversely proportional to a constant power α of I. α denotes the temporal locality in the trace (close to *one* represents a high temporal locality). We used the World-Cup trace [3] to associate file sizes with the Zipf pattern; like several other traces, this trace associates small files to be the most popular ones while larger files tend to be less popular. Thus, when the α value is very close to *one*, a lot of small files tend to be accessed and when the α value becomes smaller, the requests are more spread out to the larger files as well. Accordingly, the percentage improvement in performance for the TOE seems to be lesser for high α values as compared to small α values.



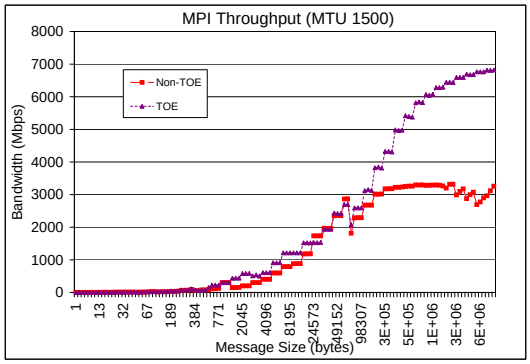
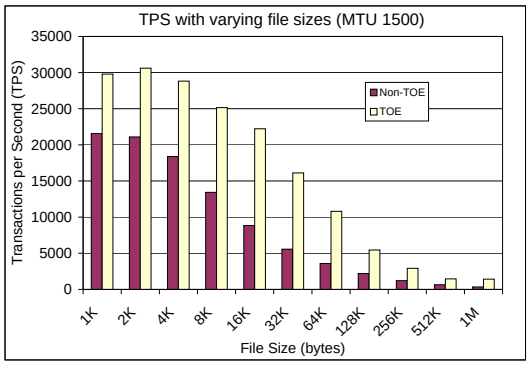


Figure 7. MPI-level Micro-Benchmarks (MTU 1500): (a) Latency and (b) Throughput



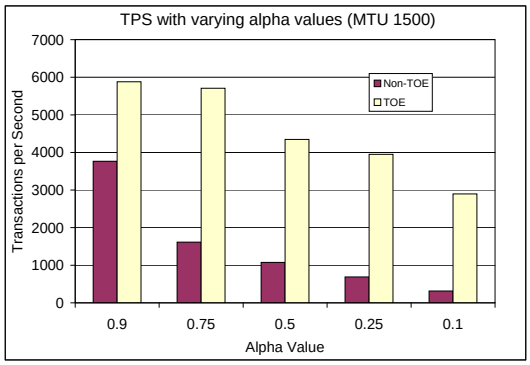


Figure 8. Apache Web-Server Performance: (a) Single File Traces and (b) Zipf Traces

5 Concluding Remarks

In this paper, we presented a detailed performance evaluation of the Chelsio T110 10GigE adapter with TOE. We have performed evaluations in three categories: (i) detailed micro-benchmark level evaluation of the native sockets layer, (ii) evaluation of the Message Passing Interface (MPI) stack over the sockets interface, and (iii) application-level evaluation of the Apache web server. These experimental evaluations provide several useful insights into the effectiveness of the TOE stack in scientific as well as commercial domains.

References

[1] The Apache Web Server. http://www.apache.org/.

[2] Top500 supercomputer list. http://www.top500.org, November 2004.

[3] The Internet Traffic Archive. http://ita.ee.lbl.gov/html/traces.html.

[4] Infiniband Trade Association. http://www.infinibandta.org.

[5] P. Balaji, S. Narravula, K. Vaidyanathan, S. Krishnamoorthy, J. Wu, and D. K. Panda. Sockets Direct Protocol over InfiniBand in Clusters: Is it Beneficial? In *ISPASS '04*.

[6] P. Balaji, H. V. Shah, and D. K. Panda. Sockets vs RDMA Interface over 10-Gigabit Networks: An In-depth analysis of the Memory Traffic Bottleneck. In *RAIT workshop '04*.

[7] P. Balaji, P. Shivam, P. Wyckoff, and D. K. Panda. High Performance User Level Sockets over Gigabit Ethernet. In *Cluster Computing '02*.

[8] P. Balaji, J. Wu, T. Kurc, U. Catalyurek, D. K. Panda, and J. Saltz. Impact of High Performance Sockets on Data Intensive Applications. In *HPDC '03*.

[9] N. J. Boden, D. Cohen, R. E. Felderman, A. E. Kulawik, C. L. Seitz, J. N. Seizovic, and W. K. Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro '95*.

[10] Greg Burns, Raja Daoud, and James Vaigl. LAM: An Open Cluster Environment for MPI. In *Supercomputing Symposium*.

[11] Chelsio Communications. http://www.chelsio.com/.

[12] Chelsio Communications. http://www.gridtoday.com/04/1206/104373.html, December 2004.

[13] W. Feng, J. Hurwitz, H. Newman, S. Ravot, L. Cottrell, O. Martin, F. Coccetti, C. Jin, D. Wei, and S. Low. Optimizing 10-Gigabit Ethernet for Networks of Workstations, Clusters and Grids: A Case Study. In *SC '03*.

[14] J. Hurwitz and W. Feng. End-to-End Performance of 10-Gigabit Ethernet on Commodity Systems. *IEEE Micro '04*.

[15] Ammasso Incorporation. http://www.ammasso.com/.

[16] H. W. Jin, P. Balaji, C. Yoo, J . Y. Choi, and D. K. Panda. Exploiting NIC Architectural Support for Enhancing IP based Protocols on High Performance Networks. *JPDC '05*.

[17] J. S. Kim, K. Kim, and S. I. Jung. Building a High-Performance Communication Layer over Virtual Interface Architecture on Linux Clusters. In *ICS '01*.

[18] J. S. Kim, K. Kim, and S. I. Jung. SOVIA: A User-level Sockets Layer Over Virtual Interface Architecture. In *Cluster Computing '01*.

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

[20] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network (QsNet): High-Performance Clustering Technology. In *HotI '01*.

[21] H. V. Shah, C. Pu, and R. S. Madukkarumukumana. High Performance Sockets and RPC over Virtual Interface (VI) Architecture. In *CANPC workshop '99*.

[22] George Kingsley Zipf. Human Behavior and the Principle of Least Effort. Addison-Wesley Press, 1949.