The Convergence of Ethernet and Ethernot: A 10-Gigabit Ethernet Perspective

P. BALAJI, W. FENG AND D. K. PANDA

Technical Report
Ohio State University (OSU-CISRC-1/06-TR10)

The Convergence of Ethernet and Ethernot: A 10-Gigabit Ethernet Perspective*

P. Balaji,
Computer Science and Engg.,
Ohio State University
balaji@cse.ohio-state.edu

W. Feng,
Computer Science,
Virginia Polytechnic Institute
feng@cs.vt.edu

D. K. Panda,
Computer Science and Engg.,
Ohio State University,
panda@cse.ohio-state.edu

Abstract

Recently, a vast number of interconnect technologies such as InfiniBand, Myrinet and Quadrics have been introduced into the system-area network (SAN) environment; the primary driving requirements of this environment being high-performance and a feature-rich interface. Ethernet, on the other hand, is already the ubiquitous technology for wide-area network (WAN) environments. Traditionally, SAN technologies had been shut off from the WAN environment due to their incompatibility with the existing Ethernet compatible infrastructure. Similarly, Ethernet has traditionally not been considered a SAN interconnect due to its close to order-of-magnitude performance gap compared to other SAN interconnects such as InfiniBand, Myrinet and Quadrics (informally called Ethernot networks). However, recently there has been a significant push from both ends of the Ethernet-Ethernot spectrum leading to a convergence of Ethernet and Ethernot. For example, Ethernot networks such as Myrinet and Quadrics are attempting work-arounds in their technologies to allow Ethernet and hence WAN compatibility. Ethernet, on the other hand, is focusing on reaching the performance capabilities of Ethernot networks by utilizing the recently introduced 10-Gigabit Ethernet (10GigE) network and providing hardware offloaded protocol stacks (in particular hardware offloaded TCP/IP or TOE) similar to the other Ethernot networks. In this paper, we focus on the later approach and try to study the potential of 10GigE TOEs in bridging the Ethernet-Ethernot performance gap; specifically compared to the InfiniBand and Myrinet networks.

We carry out this evaluation in two stages. First, we compare the performance of the host-based TCP/IP stack over 10GigE to that of 10GigE TOEs in order to understand the performance gains achievable through the use of hardware offloaded protocol stacks for 10GigE. Next, we compare the performance of 10GigE TOEs with that of the other Ethernot networks providing similar hardware offloaded protocol stacks such as InfiniBand and Myrinet. We carry out this evaluation using an extensive micro-benchmark suite as well as several sample real applications from various domains. Our experimental results demonstrate that not only can the performance of basic 10GigE be significantly improved by utilizing the hardware offloaded protocol stack, but also the aforementioned Ethernet-Ethernot performance gap can be largely bridged between 10GigE, IBA, and Myrinet via the sockets interface.

1 Introduction

In the past decade, a vast number of interconnect technologies have been introduced into the system-area network (SAN) environment. Infi niBand (IBA) [5], Myrinet [15], Quadrics [35], etc., are amongst the more widely used ones in this environment today. The primary drivers for the success of these networks are considered to be: (i) their high-performance due to hardware offbaded protocol stacks and (ii) the feature-rich interface they expose to end applications. Traditionally, these networks did not address the wide-area network (WAN) environment due to their incompatibility with the existing infrastructure.

*This work was supported by Los Alamos National Laboratory contract W-7405-ENG-36, DOE Grant #DE-FC02-01ER25506, NSF grants #CCR-0204429 and #CCR-0311542, and technical and equipment support from Chelsio Communications and Fujitsu Networks.

Ethernet, on the other hand, is already the ubiquitous interconnect technology for wide-area networks (WANs) in support of grids because it leverages the legacy IP/Ethernet infrastructure, which has been around since the mid-1970s. Its ubiquity will become even more prominent as long-haul network providers move away from the more expensive (but Ethernet-compatible) SONET technology towards 10-Gigabit Ethernet (10GigE) [26, 22, 9, 21, 7] backbones, as recently demonstrated by the longest continuous 10GigE connection between Tokyo, Japan and Geneva, Switzerland via Canada and the United States [20] in late 2004; an 18,500km connection that used the 10GigE WAN PHY technology to set-up a *pseudo local-area network* at the University of Tokyo that appeared to include systems at CERN, which were 17 time zones away. Inspite of dominating the Top500 supercomputer list [3] with a 49.8% share of the clusters, Gigabit Ethernet is still not considered a SAN interconnect because of a close to order-of-magnitude performance gap it has, when compared to traditional SAN technologies such as Infiniband, Myrinet and Quadrics.

Broadly, based on the environment they were initially developed for (SAN or WAN), networks are informally broken down into two classes, namely Ethernet and non-Ethernet (or Ethernot) networks. However, recently there has been a significant push towards the convergence of these two kinds of networking technologies. There have been initiatives from both ends of the Ethernet-Ethernot spectrum to reach a convergence point; a convergent networking technology which can address both the high-performance and rich feature-set requirements of the Ethernot environment as well as the WAN compatibility requirements of the Ethernet environment.

From the Ethernot technologies' side, Myricom Incorporation (the vendors for the Myrinet network) has recently introduced network adapters which utilize the Myrinet interconnect technology while using Ethernet as the underlying wire protocol. Similarly, Quadrics Limited (the vendors for the Quadrics network) has introduced network switches which utilize the Quadrics technology while talking Ethernet on the wire. Such initiatives from traditional Ethernot networks shows the convergence of Ethernot technologies towards the Ethernet market.

On the other end of the spectrum, with the introduction of 10GigE TCP Offbad Engines (TOEs) [39], Ethernet is attempting to converge towards the performance capabilities of traditional Ethernot networks. However, given that GigE is so far behind the curve with respect to network performance, can 10GigE bridge the Ethernet-Ethernot performance gap 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 IP/Ethernet 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 question about the *performance gap* between 10GigE and the traditional Ethernot network technologies such as Infiniband (IBA) and Myrinet. This question is the core of the study carried out in this paper.

1.1 Characterizing the Performance Gap

With several networking technologies in the high-speed networks market today, each exposing its own communication interface, characterizing the *performance gap* between these networks is no longer a straightforward task. This issue is not unique to only lower-level performance characterization; it is also a major issue for application developers. Due to the increasingly divergent communication interfaces exposed by the networks, application developers demand a common interface that they can utilize in order to achieve portability across the various networks. The Message Passing Interface (MPI) [31, 25, 16] and the sockets interface have been two

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

of the most popular choices towards achieving such portability. MPI has been the *de facto* standard for scientific applications, while sockets has been more prominent in legacy scientific applications as well as grid-based or heterogeneous-computing applications, file and storage systems, and other commercial applications. Because traditional sockets over host-based TCP/IP has not been able to cope with the exponentially increasing network speeds, IBA and other network technologies proposed a high-performance sockets interface, known as the Sockets Direct Protocol (SDP) [2]. SDP is a mechanism to allow existing sockets-based applications to transparently take advantage of the hardware-offloaded protocol stack provided by these exotic networks. As a result, Chelsio and other 10GigE vendors have recently released adapters that deliver hardware-offloaded TCP/IP protocol stacks (TOEs) to provide high-performance support for existing sockets-based applications.

In this paper, we concentrate on the sockets interface to perform two kinds of studies. First, we compare the performance of the host-based TCP/IP stack over 10GigE to that of 10GigE TOEs in order to understand the performance gains achievable through the use of hardware offloaded protocol stacks for 10GigE (in this case, hardware offloaded TCP/IP protocol stack). Next, we compare the performance of 10GigE TOEs with that of other interconnects providing similar hardware offloaded protocol stacks such as InfiniBand and Myrinet. We present performance evaluations at two levels: (i) a detailed micro-benchmark evaluation and (ii) an application-level evaluation with sample applications from multiple domains, including a bio-medical image visualization tool known as the Virtual Microscope [4], an iso-surface oil reservoir simulator called Iso-Surface [13], a cluster file-system known as the Parallel Virtual File-System (PVFS) [17], and a popular cluster management tool named Ganglia [1].

Our results demonstrate that TCP offload can not only give 10GigE a significant push in the performance it can achieve, but also allow it to perform quite comparably to other traditional Ethernot networks such as InfiniBand and Myrinet for sockets-based applications.

The remainder of the paper is organized as follows. In Section 2, we provide background about Protocol-Offload Engines (POEs) and the network interconnects used in this paper, namely 10GigE, IBA and Myrinet. In Section 3, we describe the approaches used by each of the networks to access their respective POEs while maintaining the sockets interface. We provide details about the experimental testbeds used for the evaluation in Section 4. Sections 5 and 6 deal with the experimental comparisons for 10GigE TOE vs. 10GigE host-based TCP/IP and 10GigE TOE vs. IBA and Myrinet, respectively. Finally, we discuss prior related work in Section 7, and conclude the paper in Section 8.

2 Background

In this section, we first provide a brief overview of hardware-offloaded protocol stacks, known as Protocol-Offload Engines (POEs), provided by networks such as 10GigE, IBA, and Myrinet. Next, we briefly describe the architectures and capabilities of the aforementioned high-performance networks considered in this paper.

2.1 Overview of Protocol Offload Engines

Traditionally, the processing of protocols such as TCP/IP is accomplished via software running on the host CPU. As network speeds scale beyond a gigabit per second (Gbps), the CPU becomes overburdened with the large amount of protocol processing required. Resource-intensive memory copies, checksum computation, interrupts, and reassembly of out-of-order packets impose a

heavy load on the host CPU. In high-speed networks, the CPU has to dedicate more cycles to handle the network traffic than to the application(s) it is running. Protocol-Offbad Engines (POEs) are emerging as a solution to limit the processing required by CPUs for networking.

The basic idea of a POE is to offbad the processing of protocols from the host CPU to the network adapter. A POE can be implemented with a network processor and firmware, specialized ASICs, or a combination of both. High-performance networks such as IBA and Myrinet provide their own protocol stacks that are offbaded onto the network-adapter hardware. Many 10GigE vendors, on the other hand, have chosen to offbad the ubiquitous TCP/IP protocol stack in order to maintain compatibility with legacy IP/Ethernet infrastructure, particularly over the WAN. Consequently, this offbading is more popularly known as a TCP Offbad Engine (TOE).

2.2 Overview of High-Speed Networks

In this section, we provide an overview of the high-speed networks that are used in this work: 10GigE, IBA, and Myrinet.

2.2.1 10-Gigabit Ethernet

The 10GigE infrastructure that we used in this paper is built on two basic hardware blocks, viz., the Chelsio T110 TOE-based network adapter and the Fujitsu XG800 virtual cut-through switch.

Chelsio T110 TOE-based Network Adapter: The Chelsio T110, as shown in Figure 1a, is a PCI-X network adapter capable of supporting full TCP/IP offbading from a host system at line speeds of 10 Gbps. The adapter consists of multiple components: (i) the terminator which provides the basis for offbading, (ii) separate memory systems each designed for holding particular types of data, and (iii) a MAC and XPAC optical transceiver for the physical transfer of data over the line.

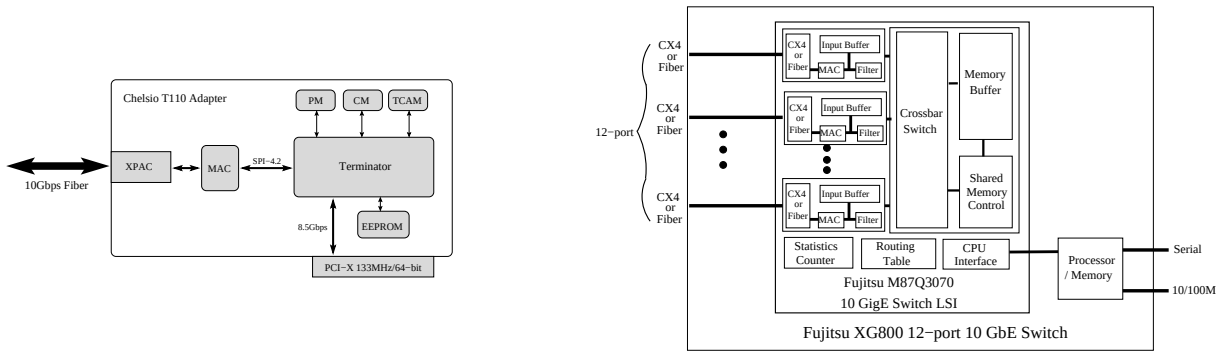

Figure 1. 10-Gigabit Ethernet Network: (a) Chelsio T110 adapter architecture and (b) Fujitsu XG800 switch architecture

- *Terminator Core:* The Terminator ASIC in the T110 forms the core of the offbad engine, capable of handling 64,000 connections at once and with a set-up and tear-down rate of about three million connections per second. It sits between the host and its Ethernet interface. When offbading a TCP/IP connection, it can handle tasks such as connection management, checksums, route lookup from the Ternary Content Addressable Memory (TCAM), congestion control, and most other TCP/IP processing. When offbading 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.

- *Memory Layout:* A 4.5MB TCAM (Ternary Content Addressable Memory) is used to store a Layer 3 routing table and can filter out invalid segments for non-offloaded connections. A 256-MB EFF FCRAM Context Memory (CM) 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 (TCB). Packet Memory (PM) stores the payload of packets and uses standard ECC SDRAM (PC2700), ranging in size from 128 MB to 4 GB.

Fujitsu XG800 Virtual Cut-through Switch: In our 10GigE network test-bed, the above mentioned Chelsio T110 network adapters are interconnected using a Fujitsu XG800 virtual cut-through switch.

Figure 1b shows a functional block diagram of the XG800 switch, which features non-blocking layer-2 switching for twelve 10GigE ports with a 450-nanosecond fall-through latency. The XG800 switch is also unique in that it uses the Fujitsu MB87Q3070 switch-on-a-chip, which significantly reduces the switch footprint. The reduced switch size is particularly important in size sensitive applications such as High-performance Computing (HPC) and Grid Computing applications. This switch also offers the advantage of a great media interfacing flexibility through a selection of XENPAK 10GigE interfaces for CX4 copper, and SR, LR, ER and LW fiber operation.

The core technology of XG800 switch is the Fujitsu MB87Q3070 single chip 10GigE switch LSI. The LSI offers four major technical features as listed below.

1. *12-port 10-gigabit Ethernet switch integrated in a single chip:* Existing 10GigE switch equipments are extremely large in size as they have been intended for general telecommunication equipment, and support various interfaces, layer-3 and higher functions, and large buffer memory for long distance transmission making it difficult to build a single chip switch LSI. By narrowing down the features to the functions needed for interconnects in HPC systems and the interface to 10GigE, Fujitsu has been able to realize a low-cost, low-power, high-performance single chip switch LSI. The switch supports layer-2 switching that is achieved through a new buffer memory sub-system, along with a control scheme for the crossbar switch that interconnects the shared buffer memory with the ports. A total of 12 10GigE ports are integrated in a single chip, along with high-speed buffer memories and high-speed I/O macros for switching.

2. *Higher bandwidth of 240G-bit/sec:* The Fujitsu XG800 uses a new on-chip memory sub-system, named *Multi-port Stream Memory* to effectively utilize multiple memory blocks in the chip and implement the high-throughput, large-volume, and multi-port shared memory in the chip. This high-performance on-chip shared memory achieves a high bandwidth of 240Gbps, which allows the 12 10-gigabit Ethernet ports to simultaneously read and write in a completely non-blocking manner.

3. *Substantially reducing fall-through latency:* The XG800 also uses a new scheduling control scheme that has been developed for the shared memory to forward the incoming packets to the output ports with a shorter fall-through latency. This contributes to substantially reducing the conventional switching latency of several microseconds to a short 450ns.

4. *Enabling CX4 copper cable transfer with high speed I/O circuit:* The XG800 uses a high speed I/O circuit (eXAUI: enhanced 10-gigabit Attachment Unit Interface) with equalization circuits in both the transmitter and receiver to compensate frequency-dependent losses. This I/O circuit enabled electrical transfer through CX4 copper cables between systems, and through backplane

PCB within a system.

2.2.2 InfiniBand

The InfiniBand Architecture (IBA) [5] defines a switched network fabric for interconnecting processing and I/O nodes. It provides the communication and management infrastructure for inter-processor communication and I/O. In an IBA network, nodes are connected to the fabric via Host-Channel Adapters (HCAs) that reside in the processing or I/O nodes.

Our IBA platform consists of InfiniHost HCAs and an InfiniScale switch from Mellanox [30]. InfiniScale is a full wire-speed switch with eight 10-Gbps ports. There is also support for link packet buffering, inbound and outbound partition checking, and auto-negotiation of link speed. The switch has an embedded RISC processor for exception handling, out-of-band data management support, and counter support for performance monitoring. The InfiniHost MT23108 HCA connects to the host through the PCI-X bus. It allows for a bandwidth of up to 10 Gbps over its ports. Memory protection along with address translation is implemented in hardware. The HCA supports on-board DDR memory up to 1GB.

2.2.3 Myrinet

Myrinet [14] is a high-speed interconnect technology using wormhole-routed crossbar switches to connect all the NICs. MX and GM [33] are the low-level messaging layers for Myrinet clusters. They provide protected user-level access to the network interface card and ensures reliable and in-order message delivery. They also provide a connectionless communication model to the upper layer, i.e., there is no connection setup phase between the ports before communication, and each port can send messages to or receive messages from any other port on a remote node.

Our Myrinet network consists of Myrinet-2000 'E' cards connected by a Myrinet-2000 switch. Each card has two ports with the link bandwidth for each port being 2 Gbps. Thus the network card can support an aggregate of 4 Gbps in each direction using both the ports. The Myrinet-2000 switch is a 16-port crossbar switch. The network interface card connects to a 133-MHz/64-bit PCI-X interface on the host. It has a programmable Lanai-XP processor running at 333 MHz with 2-MB on-board SRAM. The Lanai processor on the NIC can access host memory via the PCI-X bus through the DMA controller.

3 Interfacing with POEs

Since the Linux kernel does not currently support Protocol Offbad Engines (POEs), researchers have taken a number of approaches to enable applications to interface with POEs. The two predominant approaches are high-performance sockets implementations such as the Sockets Direct Protocol (SDP) and TCP Stack Override.

3.1 High-Performance Sockets

High-performance sockets are pseudo-sockets implementations that are built around two goals: (a) to provide a smooth transition to deploy existing sockets-based applications on clusters connected with networks using offbaded protocol stacks and (b) to sustain most of the network performance by utilizing the offbaded stack for protocol processing. These sockets layers override the existing kernel-based sockets and force the data to be transferred directly to the offbaded stack (Figure 2a). The Sockets Direct Protocol (SDP) is an industry-standard specification for high-performance sockets implementations.

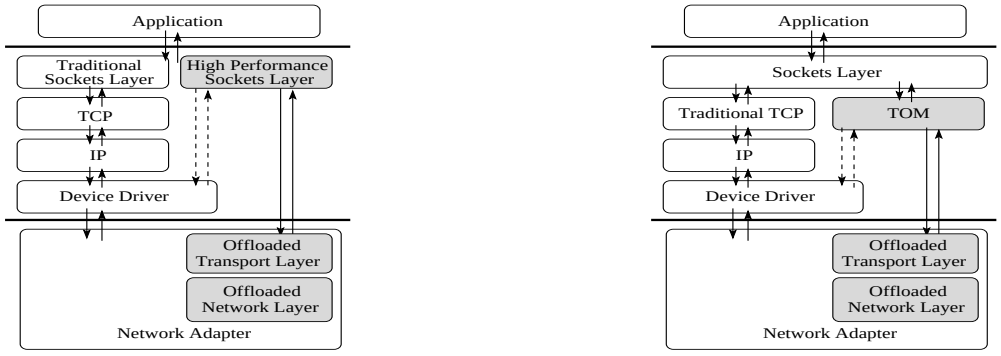

Figure 2. Interfacing with POEs: (a) High Performance Sockets and (b) TCP Stack Override

In the High Performance Sockets based approach, the TCP/IP stack in the kernel does not have to be touched at all since all the data communication calls such as `read(), write()`, etc., are trapped and directly mapped to the offbaded protocol stack. However, this requires several aspects that are handled by the sockets layer (e.g., buffer management for data retransmission and pinning of buffers) to be duplicated in the SDP implementation. IBA and Myrinet use this approach to allow sockets-based applications to utilize their offbaded protocol stacks.

3.2 TCP Stack Override

This approach retains the kernel-based sockets layer. However, the TCP/IP stack is overridden and the data is pushed directly to the offbaded protocol stack in order to bypass the host TCP/IP stack implementation (see Figure 2b). The Chelsio T110 10GigE adapter studied in this paper follows this approach. The software architecture used by Chelsio essentially has two components: the TCP offbad module (TOM) and the offbad driver.

TCP Offload Module: As mentioned earlier, the Linux operating system lacks support for TOE devices. Chelsio provides a framework of a TCP offbad 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. The state of all offbaded connections is also maintained by the TOM. Not all of the Linux network API calls (e.g., tcp_sendmsg, tcp_recvmsg) are compatible with the TOE. Modifying these 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 offbaded have certain function pointers redirected to the TOM's functions. Thus, non-offbaded connections can continue through the network stack normally.

Offload Driver: The offbad 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 the 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 handling only Layer 2 packets), a separate device driver may be required.

The advantage of this approach is that it does not require any duplication in the functionality of the sockets layer. The disadvantage, however, is that the kernel to be modified, forcing data directly from the sockets layer to the offbaded protocol stack, for an end application to be able to utilize this approach. As mentioned earlier, the Chelsio 10GigE TOEs use this approach.

4 Experimental Testbed

For experimentally evaluating the performance of the three networks, we performed experiments on three kinds of experimental test-beds.

Cluster 1: A cluster system consisting 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 the Chelsio T110 10GigE adapters.

Cluster 2: A cluster system consisting 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. These nodes were connected using the 10GigE network as described later.

The experiments on clusters 1 and 2 were performed with the SuSE Linux distribution installed with kernel.org kernel 2.6.6 (patched with Chelsio TCP Offbad modules). These clusters we used for comparing the performance of the host-based TCP/IP stack on 10GigE with that of the 10GigE TOEs as described in Section 5. For this comparison, 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.

Cluster 3: A cluster of four nodes built around SuperMicro SUPER X5DL8-GG motherboards with ServerWorks GC LE chipsets, which include 64-bit, 133-MHz PCI-X interfaces. Each node has two Intel Xeon 3.0 GHz processors with a 512-kB L2 cache and a 533-MHz front-side bus and 2 GB of 266-MHz DDR SDRAM. We used the RedHat 9.0 Linux distribution and the Linux-2.4.25smp kernel.org kernel. Each node was equipped with the 10GigE, IBA and Myrinet networks.

Cluster 3 has been used for the comparative evaluation of the three networks (10GigE, InfiniBand and Myrinet) as described in Section 6.

10GigE: The 10GigE network was based on Chelsio T110 10GigE adapters with TOEs connected to a 12-port Foundry XG800 switch. The driver version used on the network adapters is 1.2.0. For optimizing the performance of the 10GigE network, 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, (iii) increased window size to 10 MB and (iv) enabled hardware flow control to minimize packet drops on the switch. Detailed descriptions about these optimizations and their impact can be found in our previous work [26, 22, 9].

InfiniBand: The InfiniBand (IBA) network was based on Mellanox InfiniHost MT23108 dual-port 4x HCA adapters through an InfiniScale MT43132 twenty-four port completely non-blocking switch. The adapter firmware version is fw-23108-rel-3_2_0-rc4-build-001 and the software stack was based on the Voltaire IBHost-3.0.0-16 stack. Some research groups including Mellanox Technologies [24, 23] and the Ohio State University [6] have recently implemented research prototypes for different zero-copy implementations of SDP over InfiniBand. However, due to stability issues with these implementations, we do not present these results in this paper.

Myrinet: The Myrinet network was based on Myrinet-2000 'E' (dual-port) adapters connected by a Myrinet-2000 wormhole router crossbar switch. Each adapter is capable of a 4Gbps theoretical bandwidth in each direction. For SDP/Myrinet, we performed evaluations with two different implementations. The first implementation is using the GM/Myrinet drivers (SDP/GM v1.7.9 over GM v2.1.9). The second implementation is over the newly released MX/Myrinet drivers (SDP/MX v1.0.2 over MX v1.0.0). The SDP/MX implementation is a very recent release by Myricom (the vendor for Myrinet) and achieves a significantly better performance than the

older SDP/GM. However, as a part-and-parcel of being a bleeding-edge implementation, SDP/MX comes with its share of stability issues; due to this, we had to restrict the evaluation of some of the experiments to SDP/GM alone. Specifically, we present the ping-pong latency and uni-directional bandwidth results (in Section 6.1) for both SDP/MX as well as SDP/GM and the rest of the results for SDP/GM alone. With the current active effort from Myricom towards SDP/MX, we expect these stability issues to be resolved very soon and the numbers for Myrinet presented in this section to further improve.

5 Performance Comparison of host-based TCP/IP with the TOE

In this section, we evaluate the performance achieved by the Chelsio T110 10GigE adapter with TOE as compared to the host-based TCP/IP stack over 10GigE (referred to as non-TOE) using an extensive micro-benchmark suite.

For ease of understanding, we break down the evaluations into two 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 (in Section 5.1). Second, we perform evaluations based on multiple connections using the multi-stream, hot-spot, fan-in and fan-out tests (in Section 5.2).

5.1 Single Connection Micro-Benchmarks

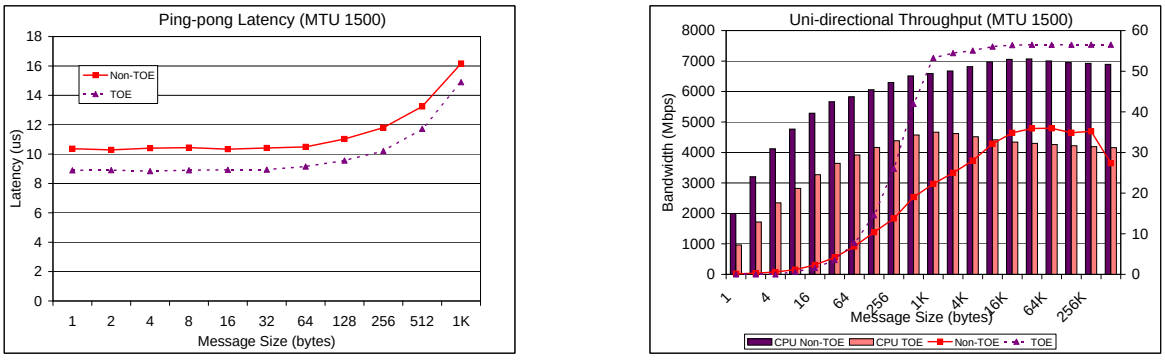

Figure 3. Sockets-level Micro-Benchmarks (MTU 1500): (a) Latency and (b) Throughput

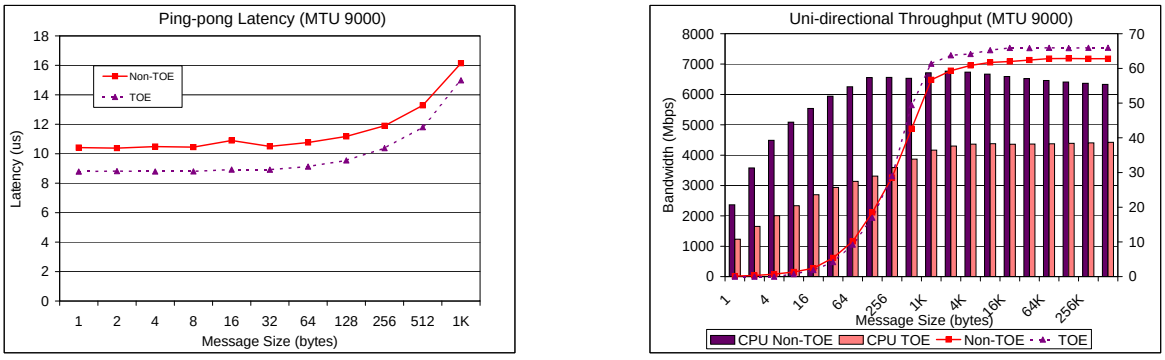

Figure 4. Sockets-level Micro-Benchmarks (MTU 9000): (a) Latency and (b) Throughput

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 Offbad 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 fi gure, 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 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.

5.2 Multiple Connection Micro-Benchmarks

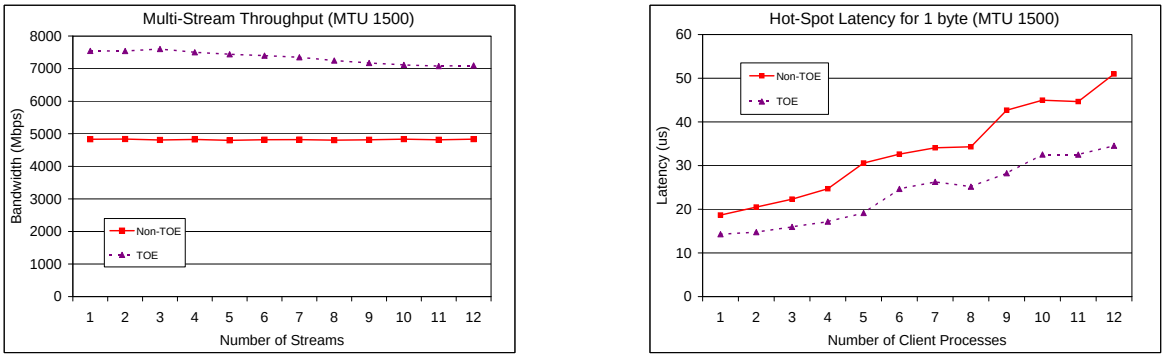

Figure 5. (a) Multi-stream Throughput and (b) Hot-Spot Latency

In this section, 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.

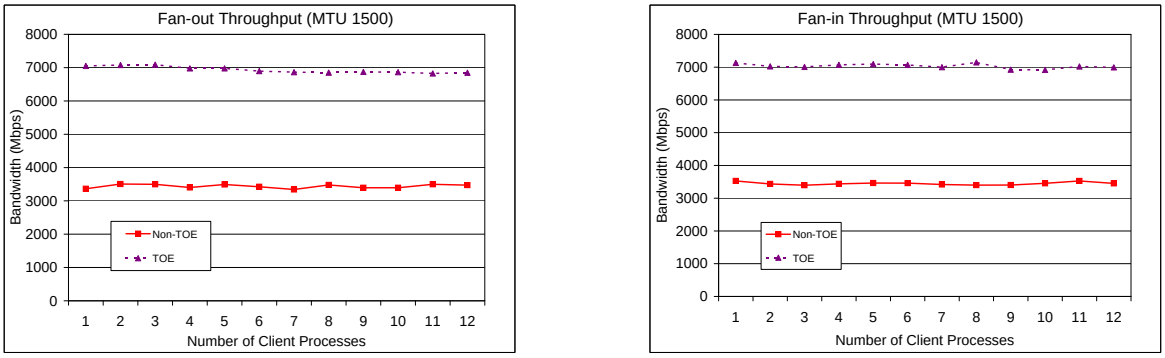

Figure 6. (a) Fan-out Test and (b) Fan-in Test

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 Throughput 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 throughput tests. In both these tests, one server process carries out uni-directional throughput tests simultaneously with a number 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.

6 Performance Comparison of 10GigE TOEs with IBA and Myrinet

In this section, we evaluate the performance achieved by the Chelsio T110 10GigE adapter with TOE as compared to the SDP implementations on top of InfiniBand and Myrinet. Specifically, in Section 6.1, we perform micro-benchmark evaluations and in Section 6.2 we evaluate sample applications from different domains over the three networks.

6.1 Micro-benchmark Comparison of 10GigE, IBA and Myrinet

In Section 5, we have shown the performance benefits of the 10GigE TOE as compared to the basic host-based TCP/IP stack over 10GigE. While this gives an indication of the capabilities of the TOE, the study is not complete without comparing this performance with that of the traditional Ethernot networks. In this section, we perform micro-benchmark evaluations of the 10GigE TOE and

compare with two other traditional SAN interconnects, viz., Infi niBand and Myrinet, over the sockets interface.

Figure 7 shows the basic micro-benchmark level performance of the 10GigE TOE as compared to SDP/IBA and SDP/Myrinet (both SDP/MX/Myrinet and SDP/GM/Myrinet).

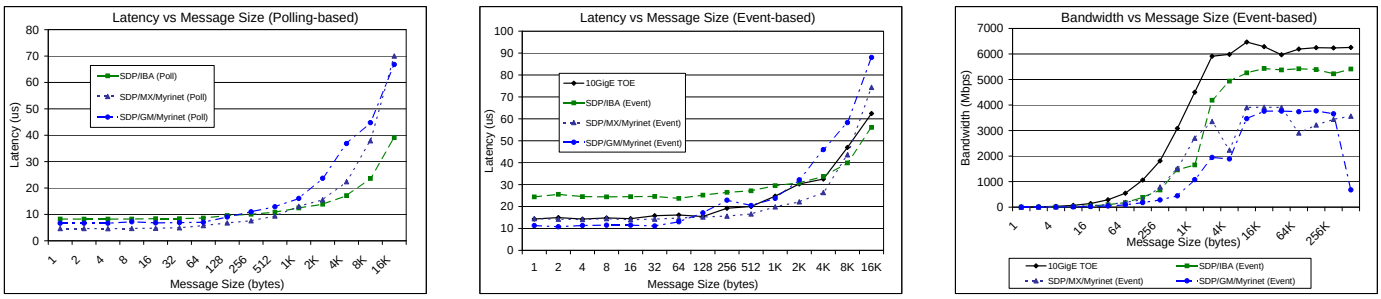

Figure 7. Single Connection Micro-Benchmarks: (a) Latency (polling-based), (b) Latency (event-based) and (c) Uni-directional Bandwidth (event-based)

Ping-Pong Latency Micro-Benchmark: Figures 7a and 7b show the comparison of the ping-pong latency for the different network stacks.

IBA and Myrinet provide two kinds of mechanisms to inform the user about the completion of data transmission or reception, namely polling and event-based. In the polling approach, the sockets implementation has to continuously poll on a predefi ned location to check whether the data transmission or reception has completed. This approach is good for performance but requires the sockets implementation to continuously monitor the data-transfer completions, thus requiring a huge amount of CPU resources. In the event-based approach, the sockets implementation requests the network adapter to inform it on a completion and sleeps. On a completion event, the network adapter wakes this process up through an interrupt. While this approach is more effi cient in terms of the CPU required since the application does not have to continuously monitor the data transfer completions, it incurs an additional cost of the interrupt. In general, for single-threaded applications the polling approach is the most effi cient while for most multi-threaded applications the event-based approach turns out to perform better. Based on this, we show two implementations of the SDP/IBA and SDP/Myrinet stacks, viz., event-based (Figure 7a) and polling-based (Figure 7b); the 10GigE TOE supports only the event-based approach.

As shown in the fi gures, SDP/Myrinet achieves the lowest small-message latency for both the polling as well as event-based models. For the polling-based models, SDP/MX/Myrinet and SDP/GM/Myrinet achieve latencies of 4.64μs and 6.68μs respectively, compared to a 8.25μs achieved by SDP/IBA. For the event-based models, SDP/MX/Myrinet and SDP/GM/Myrinet achieve latencies of 14.47μs and 11.33μs, compared to the 14.3μs and 24.4μs achieved by 10GigE and SDP/IBA, respectively. However, as shown in the fi gure, for medium-sized messages (larger than 2 kB for event-based and 4 kB for polling-based), the performance of SDP/Myrinet deteriorates. For messages in this range, SDP/IBA performs the best followed by the 10GigE TOE, and the two SDP/Myrinet implementations, respectively.

Unidirectional Bandwidth Micro-Benchmark: For the uni-directional bandwidth test, the 10GigE TOE achieves the highest bandwidth at close to 6.4 Gbps compared to the 5.4 Gbps achieved by SDP/IBA and the 3.9 Gbps achieved by the SDP/Myrinet implementations. The results for both event- and polling-based approaches are similar; thus, we only present the event-based numbers here.

The drop in the bandwidth for SDP/GM/Myrinet at 512-kB message size, is attributed to the high dependency of the implementation of SDP/GM/Myrinet on L2-cache activity. Even 10GigE TOE shows a slight drop in performance for very large messages, but not as drastically as SDP/GM/Myrinet. Our systems use a 512-KB L2-cache and a relatively slow memory (266-MHz DDR SDRAM) which causes the drop to be signifi cant. For systems with larger L2-caches, L3-caches, faster memory speeds or better memory architectures (e.g., NUMA), this drop can be expected to be smaller. Further, it is to be noted that the bandwidth for all networks is the same irrespective of whether a switch is used or not; thus the switches do not appear to be a bottleneck for this test.

6.2 Application-Level Comparison of 10GigE, IBA and Myrinet

In this section, we evaluate the performance of different applications across the three network technologies. Specifi cally, we evaluate a bio-medical image visualization tool known as the Virtual Microscope, an iso-surface oil reservoir simulator called Iso-Surface, a cluster fi le-system known as the Parallel Virtual File-System (PVFS), and a popular cluster management tool named Ganglia.

6.2.1 Data-Cutter Overview and Evaluation

Data-Cutter is a component-based framework [12, 19, 34, 36] that has been developed by the University of Maryland in order to provide a flexible and effi cient run-time environment for data-intensive applications on distributed platforms. The Data-Cutter framework implements a fi lter-stream programming model for developing data-intensive applications. In this model, the application processing structure is implemented as a set of components, referred to as *filters*, that exchange data through a *stream* abstraction. Filters are connected via *logical streams*. A *stream* denotes a unidirectional data flow from one fi lter (i.e., the producer) to another (i.e., the consumer). A fi lter is required to read data from its input streams and write data to its output streams only. The implementation of the logical stream uses the sockets interface for point-to-point stream communication. The overall processing structure of an application is realized by a *filter group*, which is a set of fi lters connected through logical streams. When a fi lter group is instantiated to process an application query, the run-time system establishes socket connections between fi lters placed on different hosts before starting the execution of the application query. Filters placed on the same host execute as separate threads. An application query is handled as a *unit of work* (UOW) by the fi lter group. An example is a visualization of a dataset from a viewing angle. The processing of a UOW can be done in a pipelined fashion; different fi lters can work on different data elements simultaneously, as shown in Figure 8.

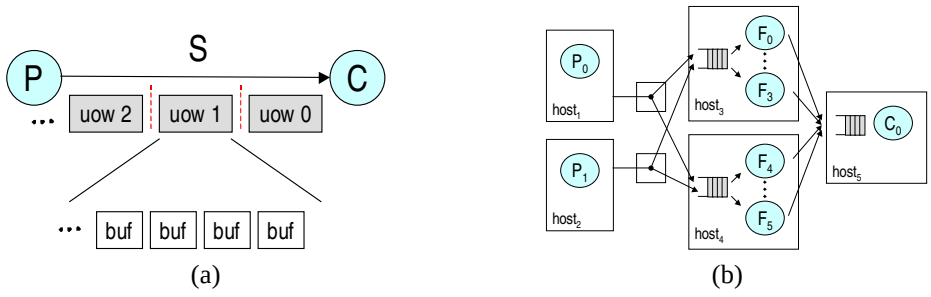

Figure 8. Data-Cutter stream abstraction and support for copies. (a) Data buffers and end-of-work markers on a stream. (b) P,F,C filter group instantiated using transparent copies.

Several data-intensive applications have been designed and developed using the data-cutter run-time framework. In this paper, we use two such applications, namely the Virtual Microscope (VM) and the Iso-Surface oil-reservoir simulation (ISO) application, for evaluation purposes.

Virtual Microscope (VM): VM is a data-intensive digitized microscopy application. The software support required to store, retrieve, and process digitized slides to provide interactive response times for the standard behavior of a physical microscope is a challenging issue [4, 18]. The main difficulty stems from the handling of large volumes of image data, which can range from a few hundreds of megabytes (MB) to several gigabytes (GB) per image. At a basic level, the software system should emulate the use of a physical microscope, including continuously moving the stage and changing magnification. The processing of client queries requires projecting high-resolution data onto a grid of suitable resolution and appropriately composing pixels mapping onto a single grid point.

Iso-Surface Oil-Reservoir Simulation (ISO): Computational models for seismic analysis of oil reservoirs simulate the seismic properties of a reservoir by using output from oil-reservoir simulations. The main objective of oil-reservoir modeling is to understand the reservoir properties and predict oil production to optimize return on investment from a given reservoir, while minimizing environmental effects. This application demonstrates a dynamic, data-driven approach to solve optimization problems in oil-reservoir management. Output from seismic simulations are analyzed to investigate the change in geological characteristics of reservoirs. The output is also processed to guide future oil-reservoir simulations. Seismic simulations produce output that represents the traces of sound waves generated by sound sources and recorded by receivers on a three-dimensional grid over many time steps. One analysis of seismic datasets involves mapping and aggregating traces onto a 3-dimensional volume through a process called seismic imaging. The resulting three-dimensional volume can be used for visualization or to generate input for reservoir simulations.

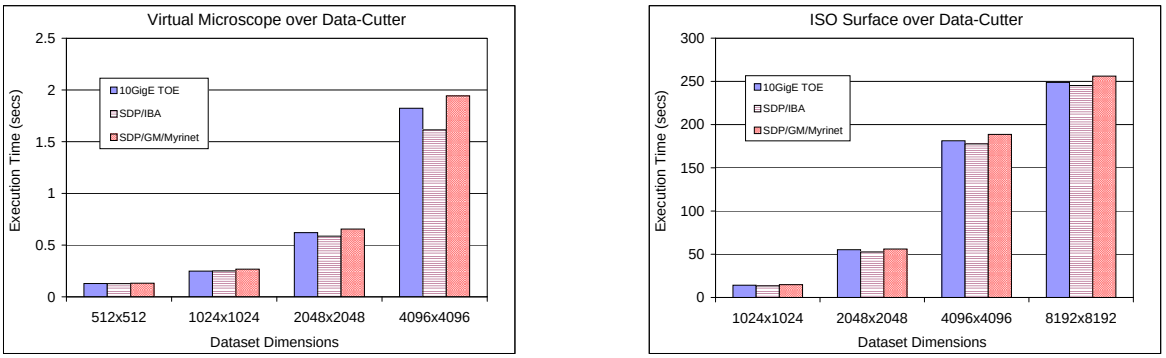

Figure 9. Data-Cutter Applications: (a) Virtual Microscope (VM) and (b) ISO-Surface (ISO)

Evaluating Data-Cutter: Figure 9a compares the performance of the VM application over each of the three networks (10GigE, IBA, Myrinet). As shown in the figure, SDP/IBA outperforms the other two networks. This is primarily attributed to the worse latency for medium-sized messages for 10GigE TOE and SDP/GM/Myrinet (shown in Figure 7a). Though the VM application deals with large datasets (each image was about 16MB), the dataset is broken down into small Unit of Work (UOW) segments that are processed in a pipelined manner. This makes the application sensitive to the latency of medium-sized messages resulting in better performance for SDP/IBA compared to 10GigE TOE and SDP/GM/Myrinet.

Figure 9b compares the performance of the ISO application for the three networks. The dataset used was about 64 MB in size. Again, though the performance of the three networks is much closer compared to the Virtual Microscope application, the trend with respect to the performance remains the same with SDP/IBA slightly outperforming the other two networks.

6.2.2 PVFS Overview and Evaluation

Parallel Virtual File System (PVFS) [17], is one of the leading parallel file systems for Linux cluster systems today, developed jointly by Clemson University and Argonne National Lab. It was designed to meet the increasing I/O demands of parallel applications in cluster systems. Typically, a number of nodes in the cluster system are configured as I/O servers and one of them (either an I/O server or a different node) as a metadata manager. Figure 10 illustrates a typical PVFS environment.

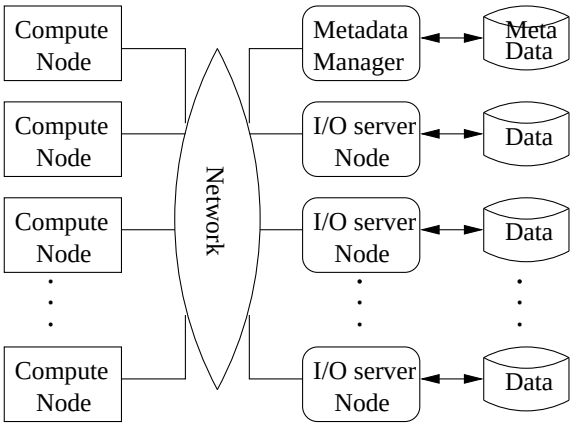

Figure 10. A Typical PVFS Setup

PVFS achieves high performance by striping files across a set of I/O server nodes, allowing parallel accesses to the data. It uses the native file system on the I/O servers to store individual file stripes. An I/O daemon runs on each I/O node and services requests from the compute nodes, in particular the read and write requests. Thus, data is transferred directly between the I/O servers and the compute nodes. A manager daemon runs on a metadata manager node. It handles metadata operations involving file permissions, truncation, file stripe characteristics, and so on. Metadata is also stored on the local file system. The metadata manager provides a cluster-wide consistent name space to applications. In PVFS, the metadata manager does not participate in read/write operations. PVFS supports a set of feature-rich interfaces, including support for both contiguous and noncontiguous accesses to both memory and files. PVFS can be used with multiple APIs: a native API, the UNIX/POSIX API, MPI-IO, and an array I/O interface called Multi- Dimensional Block Interface (MDBI). The presence of multiple popular interfaces contributes to the wide success of PVFS in the industry.

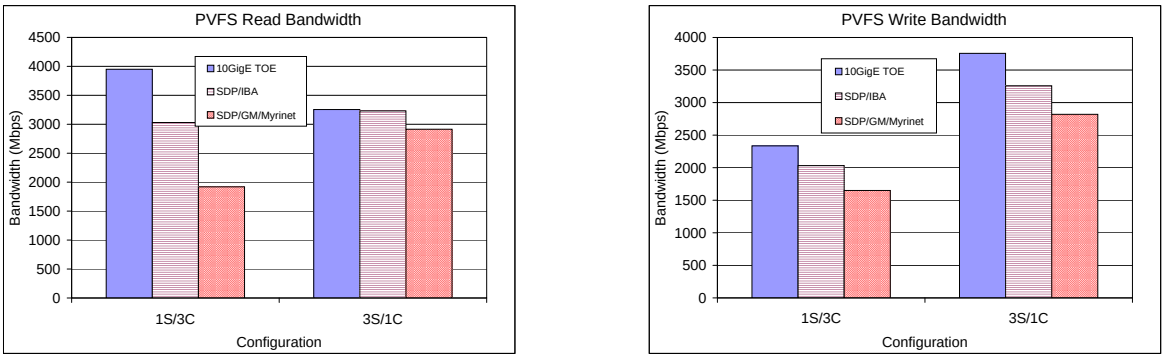

Figure 11. Concurrent PVFS Read/Write

Performance of Concurrent File I/O: In this test, we evaluate the performance of PVFS concurrent read/write operations using the *pvfs-test* program from the standard PVFS releases. For this test, an MPI program is used to parallelize file write/read access of contiguous 2-MB data buffers from each compute node. The native PVFS library interface is used in this test, more details of this

program can be found in [17].

Figure 11 shows PVFS file read and write performance on the different networks. We perform two kinds of tests for both read and write. In the first test, we use just one server; three clients simultaneously read or write a file from/to this server. In the second test, we use three servers and stripe the file across all three servers; a single client reads or writes the stripes from all three servers simultaneously. These two tests are represented as legends "1S/3C" (representing one server and three clients) and "3S/1C" (representing three servers and one client), respectively. As shown in the figure, the 10GigE TOE considerably outperforms the other two networks in both the tests for read as well as write. This follows the same trend as shown by the basic bandwidth results in Figure 7b. SDP/IBA, however, seems to achieve considerably lower performance as compared to even SDP/GM/Myrinet (which has a much lower theoretical bandwidth: 4 Gbps compared to the 10 Gbps of IBA).

Performance of MPI-Tile-IO: MPI-Tile-IO [37] is a tile-reading MPI-IO application. It tests the performance of tiled access to a two-dimensional dense dataset, simulating the type of workload that exists in some visualization applications and numerical applications. In our experiments, two nodes are used as server nodes and the other two as client nodes running MPI-tile-IO processes. Each process renders a 1×2 array of displays, each with 1024×768 pixels. The size of each element is 32 bytes, leading to a file size of 48 MB.

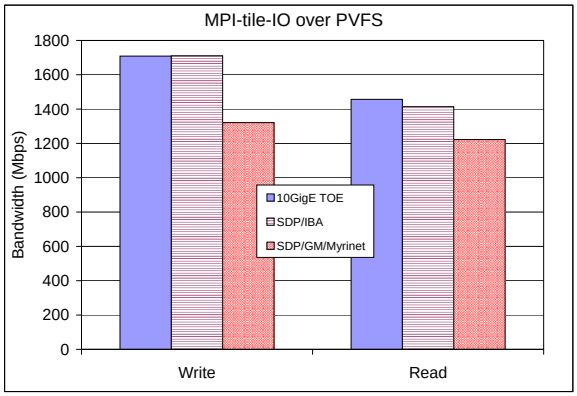

Figure 12. MPI-Tile-IO over PVFS

We evaluate both the read and write performance of MPI-Tile-IO over PVFS. As shown in Figure 12, the 10GigE TOE provides considerably better performance than the other two networks in terms of both read and write bandwidth. Another interesting point to be noted is that the performance of all the networks is considerably worse in this test versus the concurrent file I/O test; this is due to the non-contiguous data access pattern of the MPI-tile-IO benchmark which adds significant overhead.

6.2.3 Ganglia Overview and Evaluation

Ganglia [1] is an open-source project that grew out of the UC-Berkeley Millennium Project. It is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency.

The Ganglia system comprises of two portions. The first portion comprises of a server monitoring daemon which runs on each node of the cluster and occasionally monitors the various system parameters including CPU load, disk space, memory usage and several

others. The second portion of the Ganglia system is a client tool which contacts the servers in the clusters and collects the relevant information. Ganglia supports two forms of global data collection for the cluster. In the first method, the servers can communicate with each other to share their respective state information, and the client can communicate with any one server to collect the global information. In the second method, the servers just collect their local information without communication with other server nodes, while the client communicates with each of the server nodes to obtain the global cluster information. In our experiments, we used the second approach.

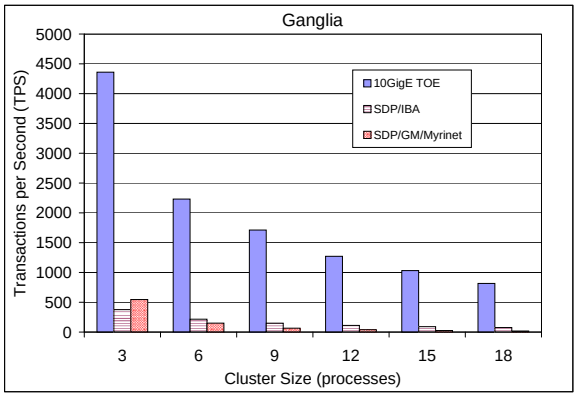

Figure 13. Ganglia: Cluster Management Tool

Evaluating Ganglia: Figure 13 shows the performance of Ganglia for the different networks. As shown in the figure, the 10GigE TOE considerably outperforms the other two networks by up to a factor of 11 in some cases. To understand this performance difference, we first describe the pattern in which Ganglia works. The client node is an end node which gathers all the information about all the servers in the cluster and displays it to the end user. In order to collect this information, the client opens a connection with each node in the cluster and obtains the relevant information (ranging from 2 KB to 10 KB) from the nodes. Thus, Ganglia is quite sensitive to the connection time and medium-message latency.

As we had seen in Figures 7a and 7b, 10GigE TOE and SDP/GM/Myrinet do not perform very well for medium-sized messages. However, the connection time for 10GigE is only about 60μs as compared to the *millisecond range* connection times for SDP/GM/Myrinet and SDP/IBA. During connection setup, SDP/GM/Myrinet and SDP/IBA pre-register a set of buffers in order to carry out the required communication; this operation is quite expensive for the Myrinet and IBA networks since it involves informing the network adapters about each of these buffers and the corresponding protection information. This coupled with other overheads, e.g., state transitions (INIT to RTR to RTS) that are required during connection setup for IBA, increase the connection time tremendously for SDP/IBA and SDP/GM/Myrinet. All in all, the connection setup time dominates the performance of Ganglia in our experiments, resulting in much better performance for the 10GigE TOE.

7 Related Work

Several researchers, including ourselves, have previously shown the benefits of high-performance sockets over protocol-offbad engines. Shah et. al. from Intel were one of the first to demonstrate such capabilities using Virtual Interface Architecture (VIA) based GigaNet cLAN networks [38]. This was soon followed by other implementations of high-performance sockets on VIA [27, 28, 11], Gigabit Ethernet [10], Myrinet [32] and InfiniBand [8]. Our work differs from these in two aspects. First, our work focuses on

protocol offbad for the upcoming 10-Gigabit Ethernet networking technology which has been previously unexplored. Second, while the existing implementations show the advantages of using protocol offbad engines compared to the host stack, there is no comparative study between the different networks, making it quite difficult for end users to gauge the pros and cons of the various networks. In our work, we fill this gap by having such a comparative study on a common testbed.

As mentioned earlier, recently, several research groups including Mellanox Technologies [24, 23] and the Ohio State University [6] have recently implemented research prototypes for zero-copy implementation of SDP over InfiniBand. Due to stability issues with these implementations, we do not present these results in this paper. However, as these implementations become more stable, these results need to be re-evaluated with such stacks. However, it is to be noted that the focus of this paper is not minor performance differences between the performance of different interconnects, but rather the broad trend in which Ethernet and Ethernot networks seem to be going in, i.e., a convergent Ethernet-Ethernot infrastructure.

We had previously done a similar study comparing MPI implementations over IBA, Myrinet and Quadrics [29]. Our current work, again, differs from this in two aspects. First, this work is intended to help place the position of 10GigE with respect to performance and capabilities as a SAN network (its capabilities as a WAN network are mostly undebated). Second, this work focuses on the sockets interface which is quickly gaining popularity with the upcoming high-performance sockets standards such as SDP.

8 Concluding Remarks

Traditional Ethernet-based network architectures such as Gigabit Ethernet (GigE) have delivered significantly worse performance than other system-area networks [e.g, InfiniBand (IBA), Myrinet]. Owing to the same, GigE was never considered a part of the system-area network family. With the advent of TCP Offbad Engines (TOEs) for 10GigE, we demonstrated that not only can the performance of basic 10GigE be significantly improved by utilizing the hardware offbaded protocol stack, but also the aforementioned performance gap can be largely bridged between 10GigE, IBA, and Myrinet via the sockets interface. Our evaluations show that in most experimental scenarios, 10GigE provides comparable performance with IBA and Myrinet. This demonstrates a successful first step on the part of Ethernet towards a convergent Ethernet-Ethernot network infrastructure, i.e., a network infrastructure which gives a high-performance in a system-area network and at the same time maintains compatibility with the wide-area network.

While the sockets interface is the most widely used interface for grids, file systems, storage, and other commercial applications, the Message Passing Interface (MPI) is considered the *de facto* standard for scientific applications. A feasibility study of 10GigE as a system-area network is definitely incomplete without a comparison of MPI over the various networks. However, in order to avoid diluting the paper and due to time and space restrictions, we defer this discussion to upcoming future work.

Acknowledgments: We would like to thank Dr. Saltz, Dr. Kurc and Dr. Catalyurek from the Department of Biomedical Informatics at the Ohio State University, for providing us access to the data-cutter runtime library and the application datasets; Felix Marti, Asgeir Eiriksson and Kianoosh Naghshineh from Chelsio Communications and Takeshi Horie and Vic Herric from Fujitsu for providing us with several useful insights into the architectural details about the 10GigE TOE adapters and the XG800 10GigE switch, respectively; Markus Fischer from Myricom Incorporation and Gali Zisman, Andy Hillaker, Erez Strauss, Yaron Haviv and Yaron Segev from Voltaire Corporation for providing us access to the SDP/Myrinet and SDP/IBA stacks, respectively.

References

[1] Ganglia Cluster Management System. http://ganglia.sourceforge.net/.

[2] SDP Specification. http://www.rdmaconsortium.org/home.

[3] Top500 Supercomputer List. http://www.top500.org.

[4] A. Afework, M. D. Beynon, F. Bustamante, A. Demarzo, R. Ferreira, R. Miller, M. Silberman, J. Saltz, A. Sussman, and H. Tsang. Digital Dynamic Telepathology - The Virtual Microscope. In *Proceedings of the 1998 AMIA Annual Fall Symposium*. American Medical Informatics Association, November 1998.

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

[6] P. Balaji, S. Bhagvat, H. W. Jin, and D. K. Panda. Asynchronous Zero-copy Communication for Synchronous Sockets in the Sockets Direct Protocol (SDP) over InfiniBand. In *workshop on Communication Architecture for Clusters (CAC)*, 2006.

[7] P. Balaji, W. Feng, Q. Gao, R. Noronha, W. Yu, and D. K. Panda. Head-to-TOE Evaluation of High Performance Sockets over Protocol Offbad Engines. In *Proceedings of the IEEE International Conference on Cluster Computing*, Boston, MA, Sep 27-30 2005.

[8] 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*.

[9] 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*.

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

[11] 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*.

[12] M. D. Beynon, T. Kurc, U. Catalyurek, C. Chang, A. Sussman, and J. Saltz. Distributed Processing of Very Large Datasets with DataCutter. *Parallel Computing*, October 2001.

[13] M. D. Beynon, T. Kurc, U. Catalyurek, and J. Saltz. A Component-based Implementation of Iso-surface Rendering for Visualizing Large Datasets. *Report CS-TR-4249 and UMIACS-TR-2001-34, University of Maryland, Department of Computer Science and UMIACS*, 2001.

[14] 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*.

[15] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, 1995.

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

[17] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur. PVFS: A Parallel File System For Linux Clusters. In *Proceedings of the 4th Annual Linux Showcase and Conference*, pages 317–327, Atlanta, GA, October 2000.

[18] U. Catalyurek, M. D. Beynon, C. Chang, T. Kurc, A. Sussman, and J. Saltz. The Virtual Microscope. *IEEE Transactions on Information Technology in Biomedicine*, 2002. To appear.

[19] Common Component Architecture Forum. *http://www.cca-forum.org*.

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

[21] W. Feng, P. Balaji, C. Baron, L. N. Bhuyan, and D. K. Panda. Performance Characterization of a 10-Gigabit Ethernet TOE. In *Proceedings of the IEEE International Symposium on High-Performance Interconnects (HotI)*, Palo Alto, CA, Aug 17-19 2005.

[22] 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*.

[23] D. Goldenberg, M. Kagan, R. Ravid, and M. Tsirkin. Transparently Achieving Superior Socket Performance using Zero Copy Socket Direct Protocol over 20 Gb/s InfiniBand Links. In *RAIT*, 2005.

[24] D. Goldenberg, M. Kagan, R. Ravid, and M. Tsirkin. Zero Copy Sockets Direct Protocol over Infi niBand - Preliminary Implementation and Performance Analysis. In *HotI*, 2005.

[25] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard. *Parallel Computing*.

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

[27] 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*.

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

[29] J. Liu, B. Chandrasekaran, J. Wu, W. Jiang, S. Kini, W. Yu, D. Buntinas, P. Wyckoff, and D. K. Panda. Performance Comparison of MPI Implementations over Infi niBand, Myrinet and Quadrics. In *SC '03*.

[30] Mellanox Technologies. Mellanox Infi niBand Infi niHost Adapters, July 2002.

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

[32] Myricom Inc. Sockets-GM Overview and Performance.

[33] Myricom Inc. The GM Message Passing Systems.

[34] R. Oldfi eld and D. Kotz. Armada: A Parallel File System for Computational Grids. In *Proceedings of CCGrid2001*, May 2001.

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

[36] B. Plale and K. Schwan. dQUOB: Managing Large Data Flows Using Dynamic Embedded Queries. In *HPDC*, August 2000.

[37] Rob B. Ross. Parallel I/O Benchmarking Consortium. http://www-unix.mcs.anl.gov/rross/pio-benchmark/html/.

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

[39] E. Yeh, H. Chao, V. Mannem, J. Gervais, and B. Booth. Introduction to TCP/IP Offbad Engine (TOE). http://www.10gea.org, May 2002.