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
Dept. of Computer Science,
Virginia Tech
feng@cs.vt.edu

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

Abstract— **A number of interconnect technologies, such as InfiniBand, Myrinet and Quadrics, have been introduced into the system-area network (SAN) environment, where the primary driving requirements are high performance and a feature-rich interface. On the other hand, Ethernet and other Ethernet-compatible technologies such as SONET/SDH already exist as ubiquitous technologies in wide-area network (WAN) environments.**

Traditionally, SAN technologies have been shut off from the WAN environment due to their incompatibility with the existing Ethernet-compatible infrastructure (e.g., Ethernet-based switches and routers). Conversely, Ethernet has traditionally not been considered a SAN interconnect due to its nearly order-of-magnitude performance lag when compared to other SAN interconnects such as InfiniBand, Myrinet and Quadrics (informally called Ethernot networks). But 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 workarounds 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) and providing hardware-offloaded protocol stacks (in particular, TCP/IP Offload Engines (TOEs)) similar to the other Ethernot networks. In this paper, we focus on the latter approach and study the potential of 10GigE TOEs in bridging the Ethernet-Ethernot performance gap, specifically compared to the InfiniBand and Myrinet networks.

Keywords: Sockets, 10-Gigabit Ethernet, InfiniBand, Myrinet

I. Introduction

In the past decade, a wide array of interconnect technologies have been introduced into the system-area network (SAN) environment. InfiniBand (IBA) (http://www.infinibandta.org), Myrinet [5], and Quadrics [9] are amongst the more widely used ones in this environment today. The primary drivers for the success of these networks are as follows: (1) their high performance due to hardware-offloaded protocol stacks and fast hardware routing and (2) their feature-rich interfaces that are exposed to end-host applications. However, these networks do not interoperate particularly well with the wide-area network (WAN) environment due to their routing incompatibility with the existing infrastructure.

On the other hand, Ethernet and other Ethernet-compatible technologies such as SONET/SDH already exist as ubiquitous technologies in the wide-area network (WAN). These technologies, coupled with the traditional IP/Ethernet infrastructure which dates back to the early 1970s, support today's Internet as well as the emerging overlay environments that run over it, e.g., computational grids and peer-to-peer networking. The ubiquity of Ethernet 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) [3], [7] backbones, as recently demonstrated by the longest continuous 10GigE connection between Tokyo, Japan and Geneva, Switzerland via Canada and the United States in late 2004 (http://www.gridtoday.com/04/1206/104373.html). This 18,500-km 10GigE connection used 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. While Gigabit Ethernet dominates the Top500 supercomputer list (http://www.top500.org) with nearly a 50% share of the list,[1] Gigabit Ethernet is still not seriously considered as a SAN interconnect because of its nearly order-of-magnitude performance lag relative to traditional SAN technologies such as InfiniBand, Myrinet, and Quadrics.

Broadly speaking, networks are informally broken down into two distinct classes, namely Ethernet and non-Ethernet (or Ethernot). Recently, however, there has been a significant push towards the convergence of these two types of networking technologies. Initiatives from both ends of the Ethernet-Ethernot spectrum are reaching a convergence point, one where both the high-performance and rich feature-set requirements of the Ethernot environment as well as the WAN-compatibility and interoperability requirements of the Ethernet environment are addressed.

From the Ethernot side, Myricom (the vendors for the Myrinet network) recently introduced network adapters that utilize Myrinet interconnect technology while using Ethernet as the underlying wire protocol (i.e., Ethernet as the MAC layer). Similarly, Quadrics (the vendors for the Quadrics QsNet network) introduced network switches which utilize Quadrics technology while talking Ethernet on the wire. Such initiatives from traditional Ethernot interconnects shows the convergence of Ethernot technologies towards the Ethernet market.

On the other end of the spectrum, the introduction of 10GigE TCP Offload Engines (TOEs) [11] indicates the convergence of Ethernet towards the performance capabilities of 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 802.3ae 10-Gb/s standard, supported by the 10GigE Alliance, already ensures interoperability with existing IP/Ethernet infrastructures, and the manufacturing volume of 10GigE is driving costs down exponentially, just as it did for Fast Ethernet and

[1] Myrinet runs a distant second with 14%

Gigabit Ethernet[2]. 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.

A. Characterizing the Performance Gap

With many networking technologies in the high-speed network 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 these 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) and the sockets interface have been two 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 traditional 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 increasing network speeds, IBA and other network technologies proposed a high-performance sockets interface, known as the Sockets Direct Protocol or SDP (http://www.rdmaconsortium.org). SDP is a mechanism to allow existing sockets-based applications to transparently take advantage of the hardware-offloaded protocol stack provided by these 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. Next, we compare the performance of 10GigE TOEs with that of other interconnects providing similar hardware offloaded protocol stacks such as IBA 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 biomedical image visualization tool known as the Virtual Microscope [1], an iso-surface oil reservoir simulator called Iso-Surface [4], a cluster file-system known as the Parallel Virtual File-System (PVFS) [6], and a popular cluster management tool named Ganglia (http://ganglia.sourceforge.net).

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

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

II. 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.

A. Overview of Protocol Offload Engines

Traditionally, the processing of protocols such as TCP/IP is accomplished via software running on the host CPU. But 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-Offload Engines (POEs) are emerging as a solution to limit the processing required by CPUs for networking.

The basic idea of a POE is to offload 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 offloaded onto the network-adapter hardware. Many 10GigE vendors, on the other hand, have chosen to offload the ubiquitous TCP/IP protocol stack in order to maintain compatibility with the traditional IP/Ethernet infrastructure, particularly over the WAN. Consequently, this offloading is popularly known as a TCP Offload Engine (TOE).

B. 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.

1) 10-Gigabit Ethernet: The 10GigE infrastructure that we used in this paper is built on two basic hardware blocks: (i) the Chelsio T110 TOE-based network adapter and (ii) 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 complete TCP/IP offloading from a host system at line speeds of 10 Gbps. The adapter consists of multiple components: (i) the terminator which provides the basis for offloading, (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.

Fujitsu XG800 Virtual Cut-through Switch: In our 10GigE network testbed, 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 flow-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.

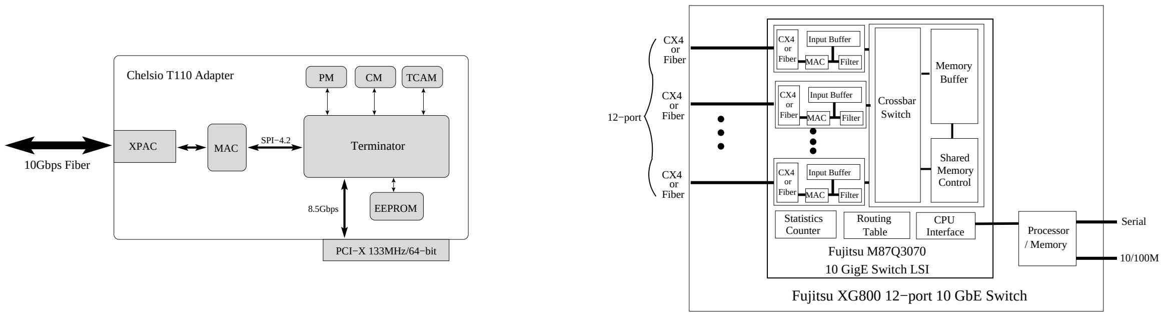


Fig. 1. 10-Gigabit Ethernet Network: (a) Chelsio T110 adapter architecture and (b) Fujitsu XG800 switch architecture

2) InfiniBand: The InfiniBand Architecture (IBA) 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 Technologies. InfiniScale is a full wire-speed switch with twenty-four 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 support for counters to allow 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.

3) Myrinet: Myrinet is a high-speed interconnect technology using wormhole-routed crossbar switches to connect all the network adapters. MX and GM are the low-level messaging layers for Myrinet clusters. They provide protected user-level access to the network adapters and ensure reliable and in-order message delivery. 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 adapter 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 adapter 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 network adapter can access host memory via the PCI-X bus through the DMA controller.

III. Interfacing with POEs

Because the Linux kernel does not currently support Protocol Offload 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.

A. 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 offloaded protocol stacks and (b) to allow applications to utilize most of the network performance by using the offloaded stack for protocol processing. These sockets layers override the existing kernel-based sockets and force the data to be transferred directly to the offloaded stack (Figure 2a). The Sockets Direct Protocol (SDP) is an industry-standard specification for high-performance sockets implementations.

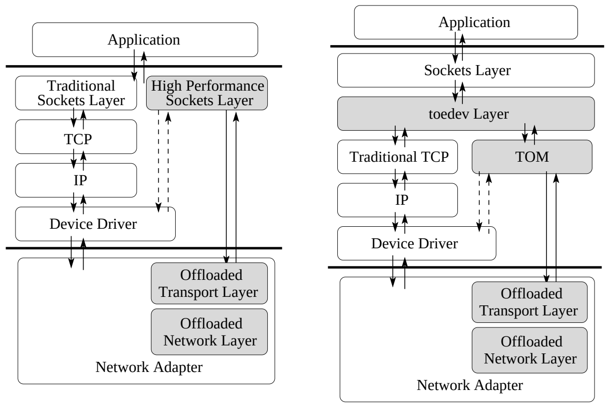


Fig. 2. Interfacing with POEs: (a) High-Performance Sockets and (b) TCP Stack Override

In the High-Performance Sockets 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 offloaded 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 offloaded protocol stacks.

B. 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 offloaded 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 offload module (TOM) 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 is responsible for implementing portions of TCP processing that cannot be done on the TOE. 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 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 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 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 has to be modified, forcing data directly from the sockets layer to the offloaded protocol stack, for an end application to be able to utilize this approach.

IV. Experimental Testbed

To experimentally evaluate the performance of the three networks, we performed experiments on three kinds of experimental testbeds.

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 via a 12-port Fujitsu XG800 10GigE switch with Chelsio T110 10GigE adapters at the end hosts.

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 Offload modules). These clusters were used for comparing the performance of the host-based TCP/IP stack on 10GigE with that of the 10GigE TOEs as described in Section V. 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 indicate the cluster used for each experiment throughout this paper.

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 is used for the comparative evaluation of the three networks (10GigE, InfiniBand and Myrinet), as described in Section VI.

10GigE: The 10GigE network is based on Chelsio T110 10GigE adapters with TOEs connected to a 12-port Fujitsu XG800 switch. The driver version used on the network adapters is 1.2.0.

IBA: The IBA network is based on Mellanox InfiniHost MT23108 dual-port 4x HCA adapters through an InfiniScale MT43132 24-port non-blocking switch. The adapter firmware version is fw-23108-rel-3_2_0-rc4-build-001, and the software stack is based on the Voltaire IBHost-3.0.0-16 stack. Some research groups including Mellanox Technologies [8] and the Ohio State University [2] have recently implemented research prototypes for different zero-copy implementations of SDP over IBA. However, due to stability issues with these implementations, we do not present these results in this paper and defer this to future work.

Myrinet: The Myrinet network is based on Myrinet-2000 'E' (dual-port) adapters connected by a Myrinet-2000 wormhole-routed crossbar switch. Each adapter is capable of a 4Gbps bandwidth in each direction. For SDP/Myrinet, we performed evaluations with two different implementations. The first implementation uses the GM/Myrinet drivers (SDP/GM v1.7.9 over GM v2.1.9). The second implementation runs over the newly released MX/Myrinet drivers (SDP/MX v1.0.2 over MX v1.0.0). The SDP/MX implementation is a recent release by Myricom and achieves significantly better performance than the older SDP/GM. However, 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 VI-A) for both SDP/MX as well as SDP/GM and the rest of the results for SDP/GM alone. With Myricom's current effort on SDP/MX, we expect these stability issues to be resolved very soon and the numbers for Myrinet presented in this section to further improve.

For all evaluations in Sections V and VI, each experiment was run 10 times; the highest and lowest values are dropped and the mean of the remaining 8 runs is reported. For micro-benchmark evaluations, each run consisted of 100,000 iterations.

V. Host-based TCP/IP Vs. TOE

In this section, we evaluate the performance achieved by 10GigE with TOE as compared to the host-based TCP/IP stack over 10GigE (referred to as non-TOE) using a suite of micro-benchmarks. All experiments in this section have been performed on Clusters 1 and 2 described in Section IV. Specifically, we used cluster 1 for all experiments that require just two nodes and Cluster 2 for the remaining.

For ease of understanding, we break down the evaluation into two categories. First, we perform evaluations based on a single connection measuring the point-to-point latency and unidirectional throughput together with the CPU utilization (in Section V-A). Second, we perform evaluations based on multiple connections using the multi-stream, hot-spot, fan-in and fan-out tests (in Section V-B).

A. Single-Connection Micro-Benchmarks

Figures 3 and 4 show the basic single-stream performance (point-to-point latency and unidirectional throughput) of the 10GigE TOE as compared to the host-based TCP/IP stack.

Point-to-point latency is defined as the time taken by a sockets application to transfer X bytes of data. In this experiment, the sender application process sends X bytes of data to the receiver process; the receiver process, upon receipt of this data, sends back X bytes of data to the sender process. This is repeated for N iterations and the average calculated. The one-way point-to-point latency is measured as one half of this value and reported. For the unidirectional throughput experiment, the sender process continuously sends N messages to the receiver process, each message containing X bytes of data. The receiver process, upon receipt of all the $N \times X$ bytes of data, sends back an acknowledgment message to the sender process. The sender process calculates the total time measured from just before it started sending the data until it got the acknowledgment; subtracting the one-way latency for the acknowledgment message from this value gives the total time taken to transfer the $N \times X$ bytes of data. We calculate the unidirectional throughput as the total amount of data transferred divided by the total time taken.

Figure 3a shows that the 10GigE 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 unidirectional throughput achieved by the TOE as compared to the non-TOE. As shown in the figure, the TOE achieves a maximum throughput of 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 that is 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 throughput 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 resulting in lower performance as compared to jumbo frames.

We also show the CPU utilization for the different stacks. For TOE, the CPU remains close to 38% for large messages. However, for the non-TOE, the CPU utilization increases slightly on using jumbo frames. To understand this behavior, we re-visit 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 sends 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 performance improves, we expect the host CPU to spend 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.

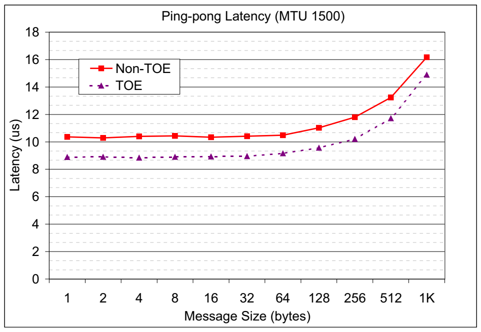
B. Multiple-Connection Micro-Benchmarks

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 abide by 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 unidirectional throughput tests. We see that the TOE achieves a throughput of 7.1 to 7.6 Gbps (equally divided between each thread). The non-TOE stack gets saturated at about 4.9 Gbps (again, equally divided between each thread). These results are similar to the single-stream results; thus, using multiple simultaneous streams to transfer data does not 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 quad-processor nodes hosting 12 client processes in total. The clients are allotted 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 designed two other tests, namely fan-out and fan-in throughput tests. In both of these tests, one server process carries out unidirectional throughput tests simultaneously with a number of client threads (performed on Cluster 2). The difference being that in a fan-out test the server sends data to the different clients (stressing the transmission path on the network adapter), and in a fan-in test, the clients send data to the server process (stressing the receive path on the network adapter). It is to be noted that these tests are different from a multi-stream test. In a



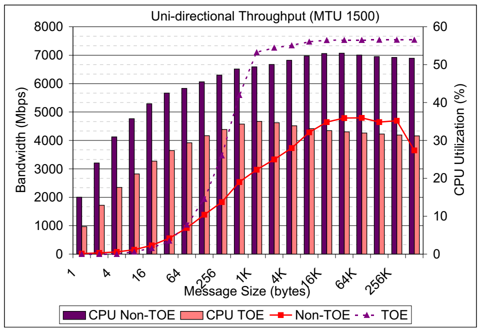
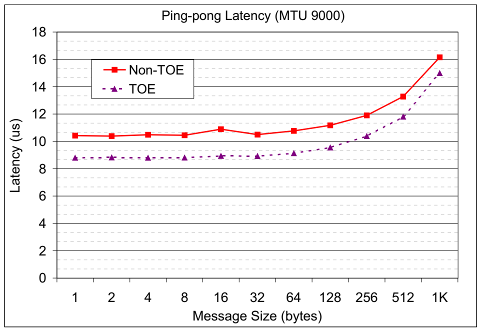


Fig. 3. Sockets-level Micro-Benchmarks (MTU 1500): (a) Latency and (b) Throughput



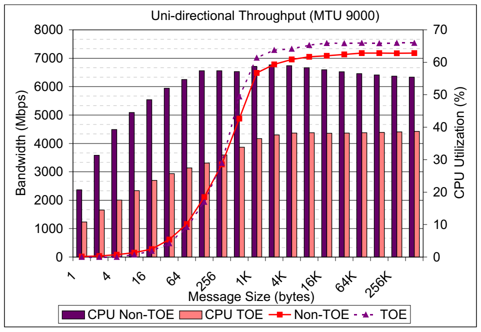
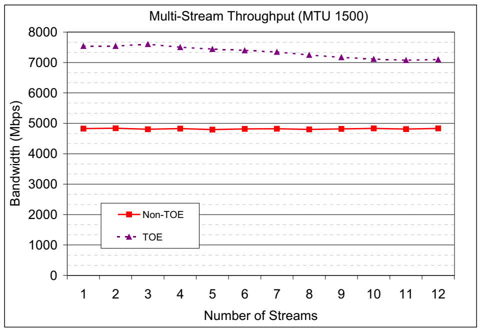


Fig. 4. Sockets-level Micro-Benchmarks (MTU 9000): (a) Latency and (b) Throughput



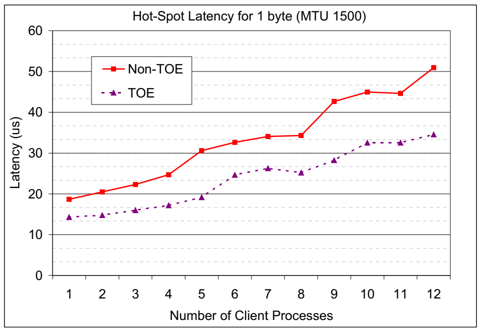
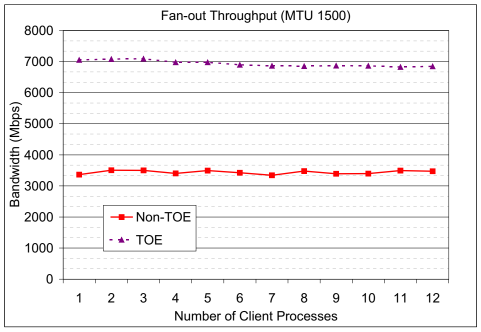


Fig. 5. (a) Multi-stream Throughput and (b) Hot-Spot Latency



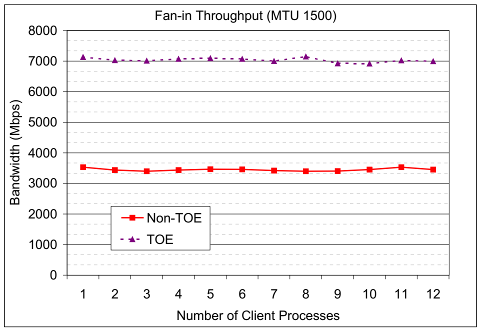


Fig. 6. (a) Fan-out Test and (b) Fan-in Test

multi-stream test, we perform throughput tests between multiple processes on the same two nodes while in the fan-in and fan-out tests, we perform throughput tests between one server process and multiple client processes residing on multiple physical nodes. Figure 6 shows the performance of the TOE stack as compared to the non-TOE stack for the fan-in and fan-out 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 transmit- and receive-path implementation.

VI. 10GigE TOEs Vs. 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 IBA and Myrinet. Specifically, in Section VI-A, we perform micro-benchmark evaluations, and in Section VI-B, we evaluate sample applications from different domains over the three networks. All experiments in this section have been performed on Cluster 3 described in Section IV.

A. Micro-benchmark Comparison

In Section V, we showed the performance benefits of 10GigE TOE over 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 it with two traditional SAN interconnects, IBA 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).

Ping-Pong Latency: 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 predefined 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 then sleeps. On a completion event, the network adapter wakes this process up through an interrupt. While this approach is more efficient in terms of the CPU required because the application does not have to continuously monitor the data-transfer completions, it incurs the additional cost of an interrupt. In general, for single-threaded applications, the polling approach is the most efficient, while for most multi-threaded applications, the event-based approach performs better. Based on this, we show two implementations of the SDP/IBA and SDP/Myrinet stacks, namely event-based (Figure 7a) and polling-based (Figure 7b); the 10GigE TOE supports only the event-based approach.

As shown in the figures, 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 the 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 figure, 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.

For each network above, we used their respective switches. So, for the above ping-pong latencies, the 10GigE (Fujitsu), IBA (Mellanox), and Myrinet (Myricom) switches contribute approximately 1000ns, 300ns, and 60ns, respectively.

Unidirectional Bandwidth: For the unidirectional 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 (Note: The theoretical peak for Myrinet in our testbed is 4 Gbps). 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 significant. 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. (Note that for each network, all the nodes are connected to a single switch corresponding to that network.)

B. Application-Level Comparison

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

1) Data-Cutter Overview and Evaluation: Data-Cutter is a component-based framework that has been developed by the University of Maryland in order to provide a flexible and efficient run-time environment for data-intensive applications on distributed platforms. The Data-Cutter framework implements a filter-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 filter (i.e., the producer) to another (i.e., the consumer). A filter is required to read data from its input streams and write data to its output streams only. The implementation

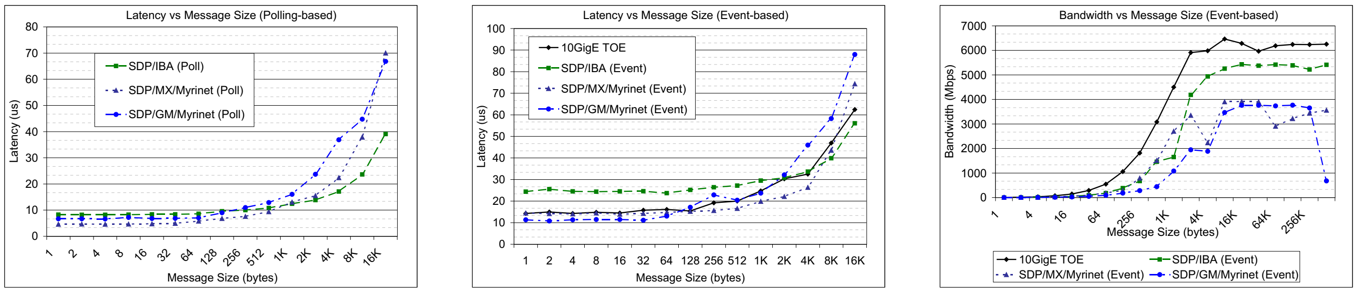


Fig. 7. Single Connection Micro-Benchmarks: (a) Latency (polling-based), (b) Latency (event-based) and (c) Unidirectional Bandwidth (event-based)

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 filters connected through logical streams. When a filter group is instantiated to process an application query, the run-time system establishes socket connections between filters 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 filter 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 filters can work on different data elements simultaneously, as shown in Figure 8.

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. The main difficulty stems from handling 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 three-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.

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 is 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 performance remains the same with SDP/IBA slightly outperforming the other two networks.

2) PVFS Overview and Evaluation: The Parallel Virtual File System (PVFS) [6] 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.

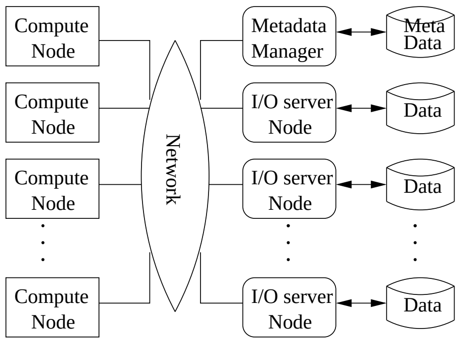
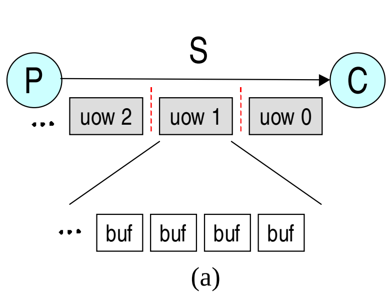


Fig. 10. A Typical PVFS Setup

PVFS achieves high performance by striping files across a



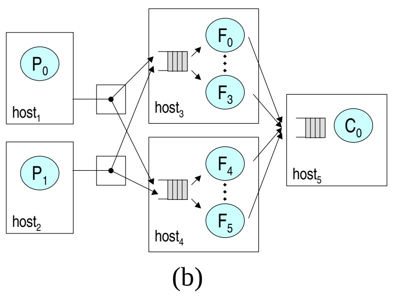
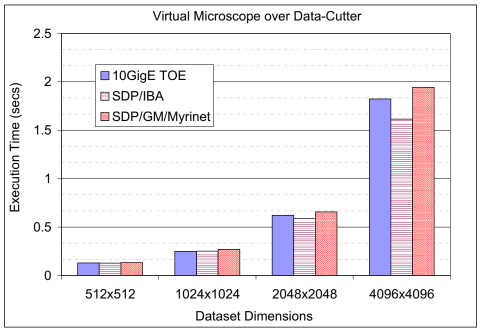


Fig. 8. Data-Cutter stream abstraction and support for copies. (a) Data buffers and end-of-work markers on a stream. (b) Producer (P), Filter (F) and Consumer (C) together form the filter group which is instantiated using transparent copies.



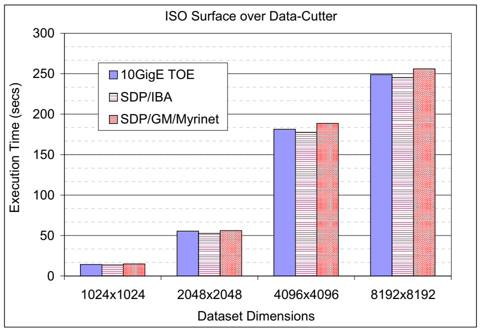


Fig. 9. Data-Cutter Applications: (a) Virtual Microscope (VM) and (b) ISO-Surface (ISO)

set of I/O server nodes, allowing parallel access 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 directly transferred 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 adoption of PVFS in the industry.

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 [6].

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 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 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.

Performance of MPI-Tile-IO: MPI-Tile-IO [10] 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.

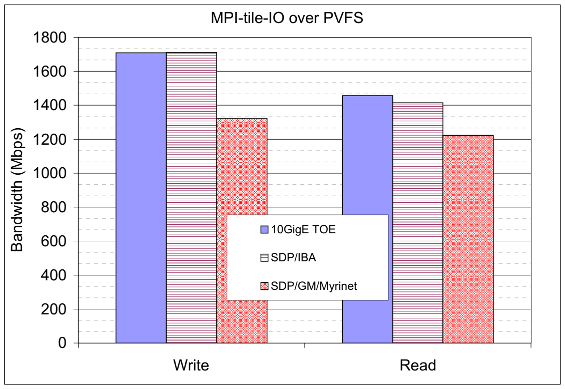
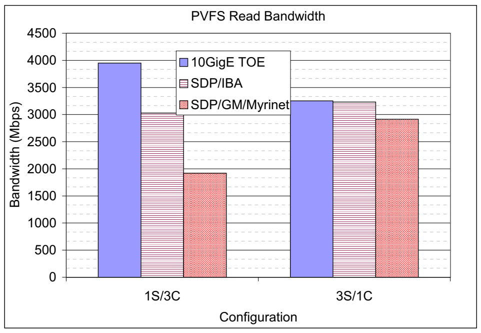


Fig. 12. MPI-Tile-IO over PVFS

We evaluate both the read and write performance of MPI-Tile-



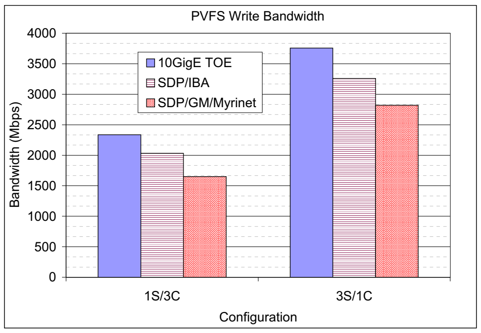


Fig. 11. Concurrent PVFS Read/Write

IO over PVFS. As shown in Figure 12, the 10GigE TOE provides 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 as compared to the 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.

3) Ganglia Overview and Evaluation: Ganglia 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 and 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 contains 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.

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 information about all 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-sized message latency.

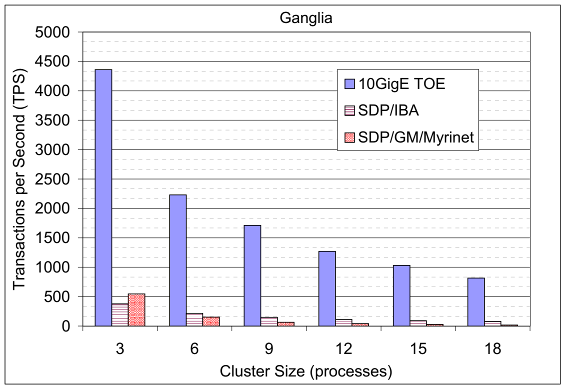


Fig. 13. Ganglia: Cluster Management Tool

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 connection times in the *millisecond range* 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 better performance for the 10GigE TOE.

VII. Discussion

There are several aspects in a network that contribute towards its acceptability in SAN or WAN environments. While performance is one of these (and the one discussed in this paper), aspects such as advanced network features, protocol layer limitations, cost, and so on, are contributors as well. In this section, we briefly provide an overview of some of these aspects.

Network-Supported One-Sided Communication: Most current Ethernot networks support advanced features such as one-sided communication (RDMA communication model). This allows applications to read from, write to, or even perform atomic operations on remote memory regions without any intervention from the target host. With the upcoming iWARP protocol stack over 10GigE, such features are soon to be realized by the Ethernet family as well.

Flow and Congestion Management: Ethernot networks such as Myrinet and IBA support specialized flow-control mechanisms (e.g., STOP-and-GO flow-control and end-to-end credit-based flow control) to avoid congestion in the network. Though the Ethernet specification does not support any such feature, several 10GigE adapters (e.g., Chelsio T110 adapters) support mechanisms to achieve this through rate-control based schemes, i.e., the sender can increase or decrease the rate at which it injects packets dynamically.

Switch-Support for Scalability: Several Ethernot networks (e.g., IBA and Quadrics) provide switch-support for allowing applications to communicate with a large number of processes in near-constant time (e.g., switch-based multicast). Though current Ethernet switches do not support multicast capabilities (i.e., selectively sending packets to a subset of the processes), they support broadcast capabilities (i.e., every packet is sent to all nodes in the subnet). Selectively dropping irrelevant packets is performed at the end-node network adapters.

Prioritization and QoS Features: Networks such as IBA support prioritization of packets belonging to one application over the other, providing service differentiation amongst applications, but it does not provide hard QoS such as bandwidth guarantees. 10GigE, on the other hand, provides bandwidth guarantees through rate-control mechanisms, but does not provide any prioritization.

Adaptive Routing Capabilities: Networks using destination-based routing (e.g., IBA and Ethernet) tend to use tree structures to prevent deadlocks while routing packets, i.e., every network is reduced to a tree by disabling some links. Myrinet, on the other hand, performs source-based routing, and thus, does not face such under-utilization of links.

VIII. 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 has never been considered a part of the system-area network family. With the advent of TCP Offload Engines (TOEs) for 10GigE, we demonstrated that not only can the performance of basic 10GigE be significantly improved by utilizing the hardware offloaded 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 performance comparable to 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 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. Thus, a feasibility study of 10GigE as a system-area network is incomplete without a comparison of MPI over the various networks. However, due to time and space restrictions, we defer this discussion as future work.

IX. 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 Herring 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.

This work was supported by Los Alamos National Laboratory contract W-7405-ENG-36, DOE Grant #DE-FC02-01ER25506, NSF grants #CNS-0403342 and #CNS-0509452, and technical and equipment support from Chelsio Communications and Fujitsu.

References

[1] 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.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. Goldenberg, M. Kagan, R. Ravid, and M. Tsirkin. Zero Copy Sockets Direct Protocol over InfiniBand - Preliminary Implementation and Performance Analysis. In *IEEE International Conference on High Performance Interconnects (HotI)*, 2005.

[9] F. Petrini, W. Feng, A. Hoisie, S. Coll, and E. Frachtenberg. The Quadrics Network (QsNet): High-Performance Clustering Technology. In *IEEE International Conference on High Performance Interconnects (HotI)*, 2001.

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

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