Head-to-TOE Evaluation of High-Performance Sockets over Protocol Offload Engines

P. Balaji, W. Feng, Q. Gao, R. Noronha, W. Yu and D. K. Panda

Technical Report
Los Alamos National Laboratory (LA-UR-05-4148)
Ohio State University (OSU-CISRC-5/05-TR35)

Head-to-TOE Evaluation of High-Performance Sockets over Protocol Offload Engines[*]

P. Balaji[†] W. Feng[‡] Q. Gao[†] R. Noronha[†] W. Yu[†] D. K. Panda[†]

[†]Computer Science and Engineering,
Ohio State University,
{balaji, gaoq, noronha, yuw, panda}@cse.ohio-state.edu

[‡]Computer and Computational Sciences,
Los Alamos National Laboratory,
feng@lanl.gov

Abstract

Despite the performance drawbacks of Ethernet, it still possesses a sizable footprint in cluster computing because of its low cost and backward compatibility to existing Ethernet infrastructure. In this paper, we demonstrate that these performance drawbacks can be reduced (and in some cases, arguably eliminated) by coupling TCP offload engines (TOEs) with 10-Gigabit Ethernet (10GigE).

Although there exists significant research on individual network technologies such as 10GigE, InfiniBand (IBA), and Myrinet; to the best of our knowledge, there has been no work that compares the capabilities and limitations of these technologies with the recently introduced 10GigE TOEs in a homogeneous experimental testbed. Therefore, we present performance evaluations across 10GigE, IBA, and Myrinet (with identical cluster-compute nodes) in order to enable a coherent comparison with respect to the sockets interface. Specifically, we evaluate the network technologies at two levels: (i) a detailed micro-benchmark evaluation and (ii) an application-level evaluation with sample applications from different domains, including a bio-medical image visualization tool known as the Virtual Microscope, an iso-surface oil reservoir simulator, a cluster file-system known as the Parallel Virtual File-System (PVFS), and a popular cluster management tool known as Ganglia. In addition to 10GigE's advantage with respect to compatibility to wide-area network infrastructures, e.g., in support of grids, our results show that 10GigE also delivers performance that is comparable to traditional high-speed network technologies such as IBA and Myrinet in a system-area network environment to support clusters and that 10GigE is particularly well-suited for sockets-based applications.

[*]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 Foundry Networks.

1 Introduction

Three years ago, virtually *none* of the supercomputers in the Top500 Supercomputer List [3] used Gigabit Ethernet (GigE) [18]. Today, GigE- and Myrinet-based [12] clusters dominate the Top500 with 35.2% and 38.6% shares, respectively.[1] Furthermore, Gigabit Ethernet is even more pervasive in the Top500 list than the list explicitly indicates as many of the Top500 supercomputers also have Ethernet-based control or management networks.

What are the drivers of the above Ethernet trend? Ease of deployment and cost over raw performance. Ethernet is already the ubiquitous interconnect technology for wide-area networks (WANs) in support of grids because it leverages the legacy Ethernet/IP 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) backbones, as recently demonstrated by the longest continuous 10GigE connection between Tokyo, Japan and Geneva, Switzerland via Canada and the United States [16] in late 2004. Researchers from Japan, Canada, the United States, and Europe completed an 18,500-km 10GigE connection between the Japanese Data Reservoir project in Tokyo and the CERN particle physical laboratory in Geneva; a connection that used 10GigE WAN PHY technology to set-up a *local-area network* at the University of Tokyo that appeared to include systems at CERN, which were 17 time zones away.

Although GigE is far behind the curve with respect to network performance, 10GigE can bridge the performance gap to exotic network technologies while achieving the ease of deployment and eventually the cost of GigE. The IEEE 802.3-ae 10-Gb/s standard already ensures interoperability with existing Ethernet/IP infrastructures, and the manufacturing volume of 10GigE is already driving costs down exponentially, just as it did for Fast Ethernet and Gigabit Ethernet.[2] What remains to be demonstrated is if 10GigE can bridge the performance gap to technologies such as InfiniBand (IBA) [5] and Myrinet.

Unfortunately, with several high-performance networks being introduced into the HPC market, 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

[1] With as rapidly as GigE-based clusters have grown, we expect GigE-based clusters to own a plurality of the next Top500 list in June 2005.

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

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) [26, 19, 13] 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 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 recently 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 (popularly known as TCP Offload Engines or TOEs) to provide high-performance support for existing sockets-based applications. In this paper, we concentrate on the sockets interface to characterize the *performance gap* between 10GigE and other exotic networks such as IBA and Myrinet.

Many researchers, including ourselves, have evaluated the benefits of sockets over offloaded protocol stacks on various networks including IBA, Myrinet, etc. However, to the best of our knowledge, there has been no work that compares and contrasts the capabilities and limitations of these technologies with the recently introduced 10GigE TOEs on a homogeneous experimental testbed. In this paper, we perform several evaluations to enable a coherent comparison between 10GigE, IBA and Myrinet with respect to the sockets interface. In particular, we evaluate the networks 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 [11], a cluster file-system known as the Parallel Virtual File-System (PVFS) [30], and a popular cluster management tool named Ganglia [1]. In addition to 10GigE's advantage with respect to compatibility to wide-area network infrastructures, e.g., in support of grids, our results show that 10GigE also delivers performance that is comparable to traditional high-speed network technologies such as IBA and Myrinet in a system-area network environment to support clusters and that 10GigE is particularly well-suited 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. Finally, we present experimental results in Section 4, discuss related work in Section 5, and conclude the paper in Section 6.

2 Background

In this section, we first provide a brief overview on 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 put a tremendous amount of 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 (e.g., see Figure 1).

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 legacy Ethernet/IP infrastructure, particularly over the wide-area network (WAN) [17]. Consequently, this offloading is more popularly known as a TCP Offload Engine (TOE).

As a precursor to complete protocol offloading, many operating systems incorporated support for features to offload specific compute-intensive features from the host to the underlying network adapter, e.g., TCP/UDP and IP checksum offload. But as network speeds increased beyond 100 Mbps, the need for further protocol processing offload became a clear requirement, e.g., offloading TCP/IP and UDP/IP segmentation

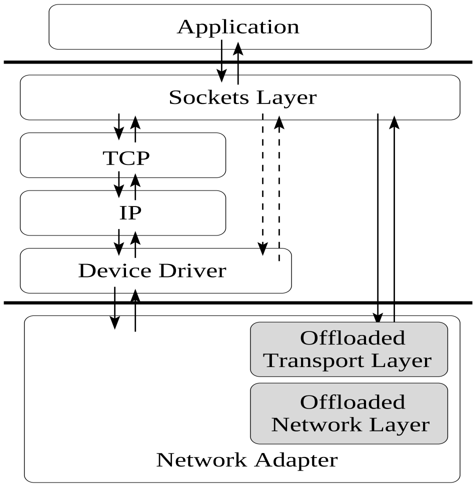


Figure 1. Protocol Offload Engines

onto the network adapter [21]. With the advent of multi-gigabit networks, the host-processing requirements became so burdensome that they ultimately led to adapter solutions with *complete* protocol offload.

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 Chelsio T110, as shown in Figure 2, is a PCI-X network adapter capable of supporting full TCP/IP offloading from a host system at line speeds of 10 Gbps. The adapter consists of multiple components: (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.

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

Memory Layout: Two types of on-board memory are available to the terminator. 256 MB of EFF FCRAM 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). For payload (packets), standard ECC SDRAM (PC2700) can be

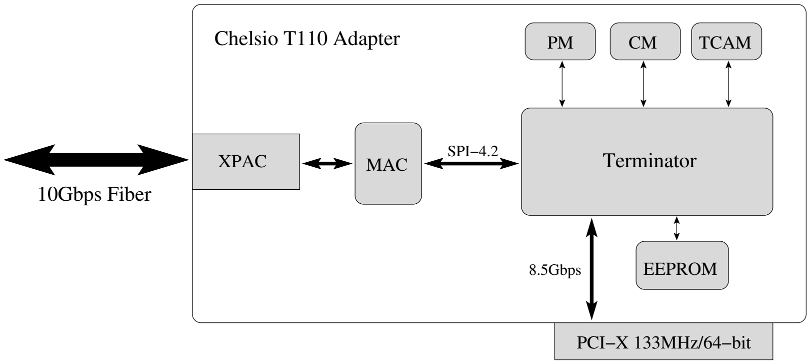


Figure 2. Chelsio T110 Adapter Architecture

used, ranging from 128 MB to 4 GB.

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

The above mentioned Chelsio T110 network adapters were interconnected using a Foundry FastIron SuperX 10GigE switch. The SuperX switch is built with high-performance ASICs to deliver high-density Gigabit Ethernet that can include Power over Ethernet and 10-Gigabit Ethernet. The SuperX switch comes with advanced layer 2 features and several layer 3 features. The 4.5μs flow-through latency offered by this switch is extraordinarily impressive given that it is a store-and-forward switch. On the other hand, the less general Fujitsu XG1200 switch uses virtual cut-through to achieve a 0.5μs flow-through latency.

2.2.2 InfiniBand

The InfiniBand Architecture (IBA) [5] defines a switched network fabric for interconnecting processing nodes and I/O nodes. It provides the communication and management infrastructure for inter-processor communication and I/O. In an IBA network, processing nodes and I/O 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 [25]. 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 [12] is a high-speed interconnect technology using wormhole-routed crossbar switches to connect all the NICs. GM [28] is the low-level messaging layer for Myrinet clusters. It provides protected user-level access to the network interface card and ensures reliable and in-order message delivery. GM provides a connectionless communication model to the upper layer. There is no connection setup phase between ports before the 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 in each direction. 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 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.

3.1 High-Performance Sockets

High-performance sockets are pseudo-sockets implementations that are built around two design 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 sustain most of the performance provided by the networks by utilizing the offloaded stack for protocol processing. These pseudo-sockets layers essentially try to override the existing kernel-based sockets layer and force the data to be transferred directly to the offloaded protocol stack (see Figure 3a). The Sockets Direct Protocol (SDP) is an industry-standard specification for high-performance sockets implementations.

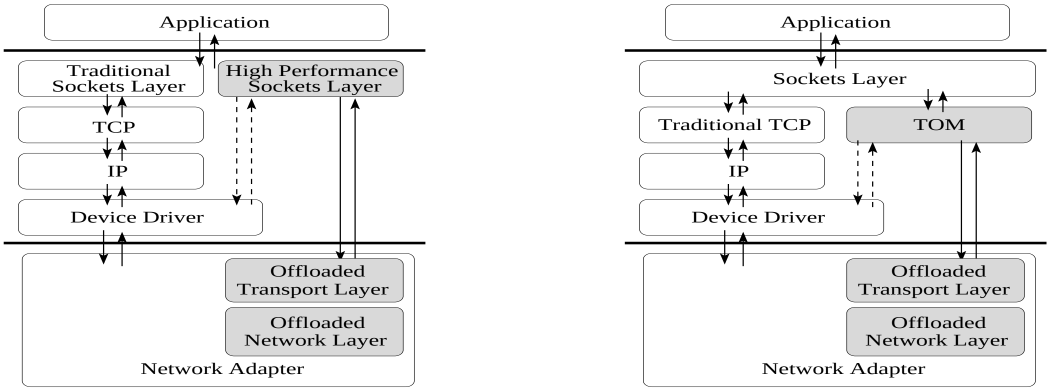


Figure 3. Interfacing with POEs: (a) High Performance Sockets and (b) TCP Stack Override

The advantage of this approach is that 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 as soon as the application calls them and directly mapped to the offloaded protocol stack. On the other hand, the disadvantage of this approach is that several aspects that are handled by the sockets layer (e.g., buffer management for data retransmission and pinning of buffers) now have to handled by the high-performance sockets layer resulting in a duplication of all these features in the SDP implementation. IBA and Myrinet use this approach to allow sockets-based applications to utilize their offloaded 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 offloaded protocol stack in order to bypass the host TCP/IP stack implementation (see Figure 3b). The Chelsio T110 adapter studied in this paper follows this approach. One of Chelsio's goals in constructing a TCP offload engine (TOE) was to keep it from being too invasive to the current structure of the system. By adding kernel hooks inside the TCP/IP stack and avoiding actual code changes, the current TCP/IP stack remains usable for all other network interfaces, including loopback.

The 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 can be thought of as the upper layer of the TOE stack. It is responsible for implementing portions of TCP processing that cannot be done on the TOE (e.g., TCP TIME_WAIT processing). The state of all offloaded connections is also maintained by the TOM. Not all of the Linux network API calls (e.g., tcp_sendmsg, tcp_recvmsg) are compatible with offloading to the TOE. Such a requirement would result in extensive changes in the TCP/IP stack. To avoid this, the TOM implements its own subset of the transport-layer API. TCP connections that are offloaded have certain function pointers redirected to the TOM's functions. Thus, non-offloaded connections can continue through the network stack normally.

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

4 Experimental Evaluation

In this section, we evaluate the performance achieved by the three networks, namely 10GigE, IBA and Myrinet, at two different levels: (i) a detailed micro-benchmark evaluation, as described in Section 4.2 and (ii) an application-level evaluation, as described in Section 4.3, with sample applications from multiple domains, including a bio-medical image visualization tool called the Virtual Microscope, an iso-surface oil reservoir simulator known as Iso-Surface, a cluster file-system named the Parallel Virtual File-System (PVFS) and a popular cluster management tool known as Ganglia.

4.1 Experimental Testbed

We used the following experimental testbed for the evaluations — a cluster system consisting 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. The 32-bit Xeon processors and the 2.4 kernel used in the testbed represent a large installation base; thus, the results described here would be most relevant for researchers using such testbeds to weigh the pros and cons of each network before adopting them.

10GigE: The 10GigE network was based on Chelsio T110 10GigE adapters with TOEs connected to a 16-port SuperX Foundry switch. The driver version used on the network adapters is 1.2.0, and the firmware on the switch is version 2.2.0. 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 [20, 17, 7].

InfiniBand: The InfiniBand (IBA) network was based on Mellanox InfiniHost MT23108 dual-port 4x HCA adapters through an InfiniScale MT43132 twenty-four 4x-port completely non-blocking InfiniBand 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-3.0.1-16 stack.

Myrinet: For SDP/Myrinet, we used the latest publicly available implementations (version 1.7.9). Myricom, the vendor for the Myrinet networks, is currently working on a next-generation SDP implementation over their new MX drivers. However, without access to this implementation, we had to stick with the publicly available 1.7.9 version. According to discussion with Myricom, they see better performance for their next-generation SDP implementation over their MX drivers as compared to the 1.7.9 implementation; thus, when these new drivers are out, we can expect the performance of SDP/Myrinet shown in this section to improve further.

4.2 Micro-Benchmark Evaluation

In this section, we perform micro-benchmark evaluations of the three networks over the sockets interface. We perform evaluations in two sub-categories. First, we perform evaluations based on a single connection measuring the point-to-point latency, uni-directional bandwidth, and the bi-directional bandwidth. Second, we perform evaluations based on multiple connections using the multi-stream bandwidth test, hot-spot test, and fan-in and fan-out tests.

4.2.1 Single Connection Micro-Benchmarks

Figures 4 and 5 show the basic single-connection performance of the 10GigE TOE as compared to SDP/IBA and SDP/Myrinet.

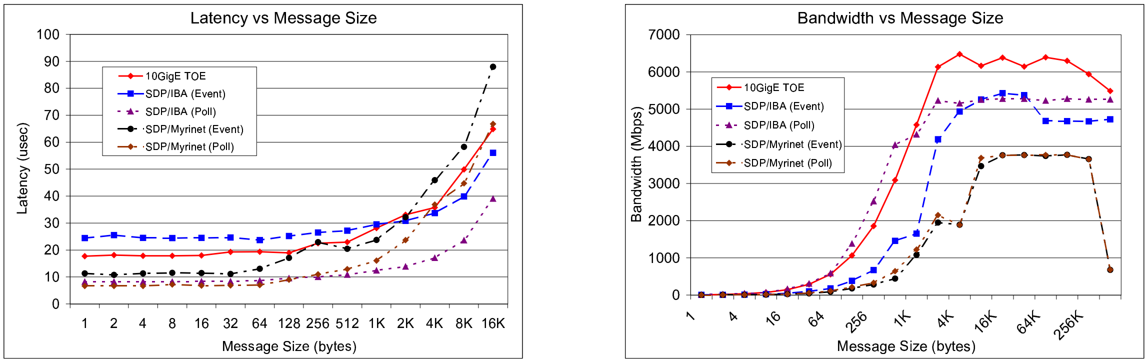


Figure 4. Single Connection Micro-Benchmarks: (a) Latency and (b) Uni-directional Bandwidth

Ping-Pong Latency Micro-Benchmark: Figure 4a shows the comparison of the ping-pong latency for the different networks.

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 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 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 efficient 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 (event-based and polling-based); the 10GigE TOE supports only the event-based approach.

As shown in the figure, SDP/Myrinet achieves the lowest small-message latency for both the polling as well as event-based models. For the polling-based models, SDP/Myrinet achieves a latency of 6.68μs compared to a 8.25μs achieved by SDP/IBA. For the event-based models, SDP/Myrinet achieves a latency of 11.33μs compared to the 17.7μs 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 SDP/Myrinet, respectively. We should note that the Foundry SuperX 10GigE switch that we used has approximately a 4.5-μs flow-through latency, which is amazing for a store-and-forward switch. By using the less general but virtual cut-through Fujitsu XG1200 switch, the flow-through latency is only 0.5 μs, resulting in a 10GigE end-to-end latency of only 13.7 μs.

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 (both polling- as well as event-based) and the 3.75 Gbps achieved by SDP/Myrinet (polling- as well as event-based)[3]. The drop in the bandwidth for SDP/Myrinet at 512-kB message size, is attributed to the high dependency of the implementation of SDP/Myrinet on L2-cache activity. Even 10GigE TOE shows a slight drop in performance for very large messages, but not as drastically as SDP/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 single-stream data transfers.

Bidirectional Bandwidth Micro-Benchmark: Similar to the unidirectional bandwidth test, the 10GigE TOE achieves the highest bandwidth (close to 7 Gbps) followed by SDP/IBA at 6.4 Gbps and SDP/Myrinet at 3.5 Gbps. 10GigE TOE and SDP/IBA seem to perform quite poorly with respect to the theoretical peak throughput achievable (20Gbps bidirectional). This is attributed to the PCI-X buses to which these network adapters are connected. The PCI-X bus (133 MHz/64 bit) is a shared network I/O bus that allows only a theoretical peak of 8.5 Gbps for traffic in both directions. Further, as mentioned earlier, the memory used in our systems is relatively slow (266-MHz DDR SDRAM). These, coupled with the header and other traffic

[3] On the Opteron platform, we see a better performance for 10GigE of up to 7.6Gbps. We also expect a better performance for the other networks on this platform, but unfortunately our current testbed does not allow this comparison on such a platform at this time. Further, with 32-bit Xeons being the largest installation base today, we feel that the numbers on our current platform might be more relevant to the community.

overheads, causes these networks to be saturated much below the theoretical bandwidth that the network can provide. For SDP/Myrinet, we noticed that the implementation is quite unstable and has not provided us with much success in getting performance numbers for message sizes larger than 64KB. Also, the peak bandwidth achievable is only 3.5 Gbps which is actually less than the unidirectional bandwidth that this implementation provides.

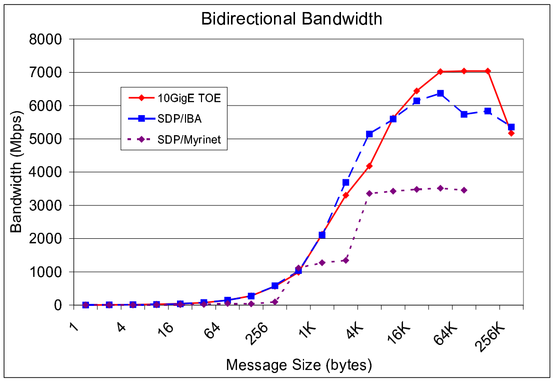


Figure 5. Bi-directional Bandwidth

4.2.2 Multiple Connection Micro-Benchmarks

Figures 6 and 7 show the multi-connection experiments performed with the three networks. These experiments demonstrate scenarios where either a single process or multiple processes on the same physical node open a number of TCP/IP connections. These tests are designed to understand the performance of the three networks in scenarios where the network has to handle several TCP/IP connections simultaneously.

It is to be noted that for multi-threaded applications the polling-based approach performs very badly due to its high CPU usage; therefore these results are not shown in this paper, and we stick to only the event-based approach for these applications.

Multi-Stream Bandwidth: Figure 6a illustrates the aggregate throughput achieved by two nodes performing multiple instances of uni-directional throughput tests. Because the performance of SDP/Myrinet seems to be a little inconsistent, it is difficult to characterize the performance of Myrinet with respect to the other networks, but we have observed that SDP/Myrinet generally achieves a throughput of about 3.15 to 3.75 Gbps. 10GigE TOE and SDP/IBA, on the other hand, quite consistently achieve throughputs around 5.9 to 6.2 Gbps with 10GigE performing slightly better most of the time.

Hot-Spot Latency: Figure 6b 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

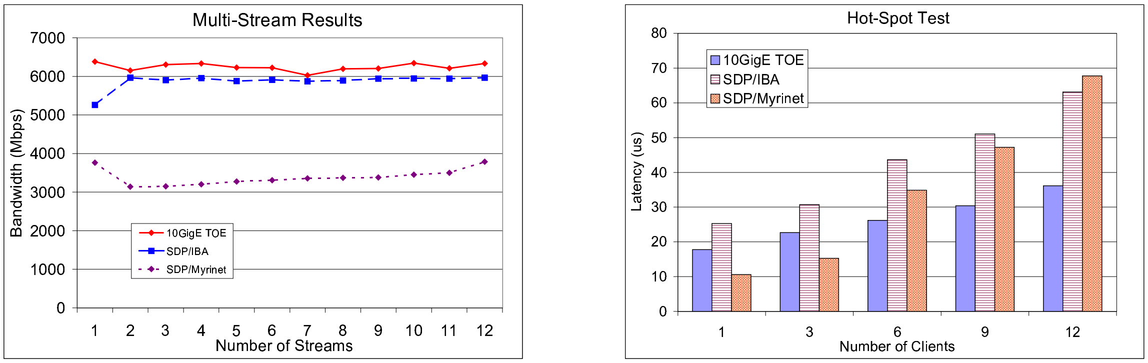


Figure 6. Multi-Connection Micro-Benchmarks: (a) Multi-Stream Bandwidth and (b) Hot-Spot Latency

forming a hot-spot on the server. We performed this experiment with one node acting as a server node and the other three dual-processor nodes hosting a total of 12 client processes. The clients are alloted in a cyclic manner, so three clients refers to having one client process on each of the three nodes, six clients refers to having two client processes on each of the three nodes, and so on. As shown in the figure, SDP/Myrinet performs the best when there is just one client followed by 10GigE TOE and SDP/IBA, respectively. However, as the number of clients increase, 10GigE TOE and SDP/IBA scale quite well while the performance of SDP/Myrinet deteriorates significantly; for 12 clients, for example, SDP/Myrinet provides the worst performance of the three while the 10GigE TOE performs significantly better than the other two. This shows that the lookup time for connection-related data structures is performed efficiently enough on the 10GigE TOE and SDP/IBA implementations and that they scale quite well with an increasing number of connections.

Fan-Out and Fan-In tests: With the hot-spot test, we have shown that the lookup time for connection-related data structures is quite efficient on the 10GigE TOE and SDP/IBA implementations. However, the hot-spot test does not stress the other resources on the network adapter such as management of memory regions for buffering data during transmission and reception. In order to stress such resources, we have designed two other tests, namely fan-out and fan-in. In both these tests, one server process carries out uni-directional throughput tests simultaneously with a number of client threads. The difference being that in a fan-out test, the server pushes data to the different clients (stressing the transmission path in the implementation), and in a fan-in test, the clients push data to the server process (stressing the receive path in the implementation). Figure 7 shows the performance of the three networks for both these tests. As shown in the figure, for both the tests, SDP/IBA and SDP/Myrinet scale quite well with increasing number of clients. 10GigE TOE, on the other hand, performs quite well for the fan-in test; however, we see a slight drop in its performance for the fan-out test with increasing clients.

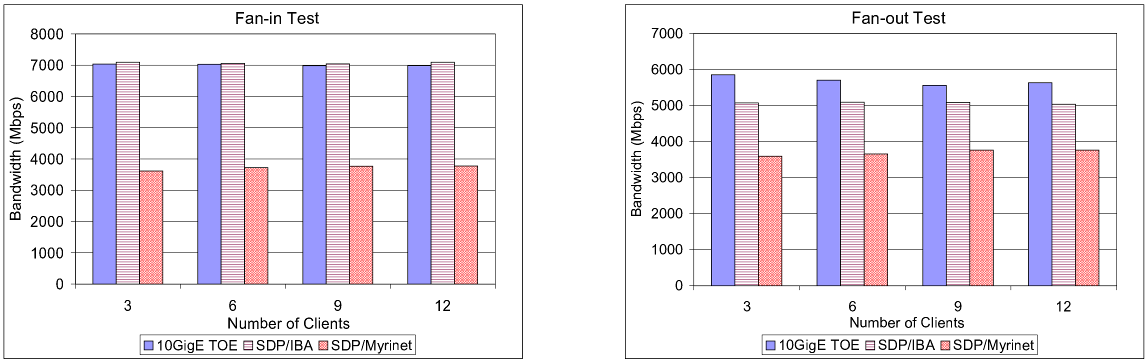


Figure 7. Multi-Connection Micro-Benchmarks: (a) Fan-in and (b) Fan-out

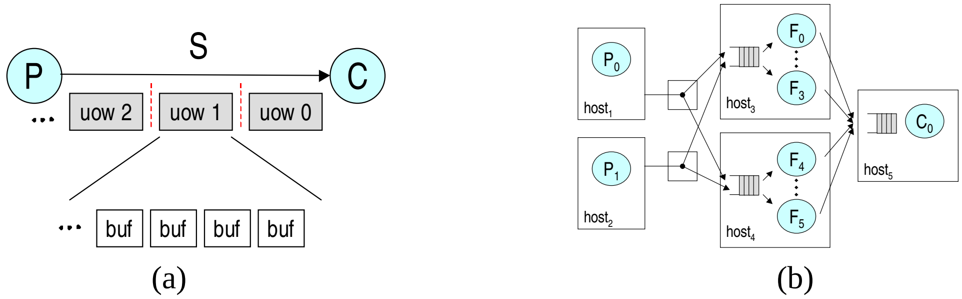
4.3 Application-Level Evaluation

In this section, we evaluate the performance of different applications across the three network technologies.

4.3.1 Data-Cutter Overview and Evaluation

In this section, we first provide a brief overview about Data-Cutter and then provide some experimental results evaluating the performance of the data-cutter library on each of the three networks (10GigE, IBA and Myrinet).

Overview of Data-Cutter: Data-Cutter is a component-based framework [10, 15, 29, 31] that has been developed by 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 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.



(a)

(b)

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, 14]. 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.

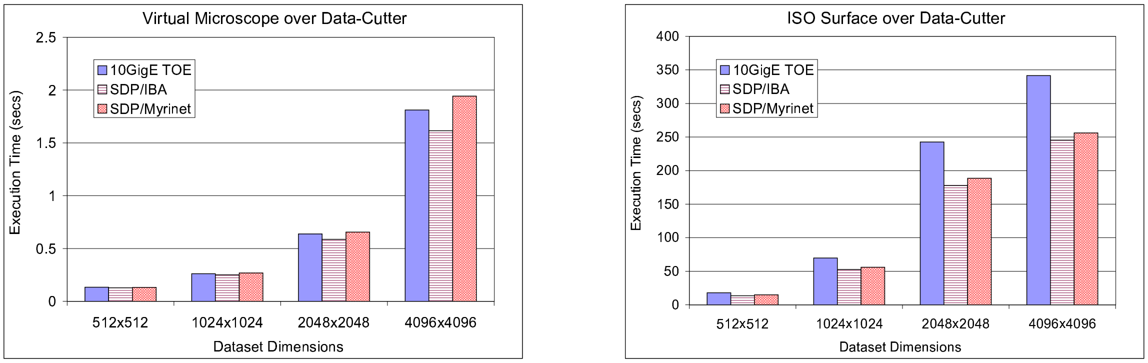


Figure 9. Data-Cutter Applications: (a) Virtual Microscope (VM) and (b) ISO-Surface (ISO)

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/Myrinet (shown in Figure 4a). 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/Myrinet.

Figure 9b compares the performance of the ISO application for the three networks. The dataset used was about 64 MB in size. Again, the trend with respect to SDP/IBA is similar to that of the VM application with SDP/IBA outperforming the other two networks. However, it is surprising that SDP/Myrinet outperforms the 10GigE TOE for nearly all dataset sizes, especially since the ISO application pipelines larger UOW chunks than the VM application, and we expect the 10GigE TOE to perform better than SDP/Myrinet from the basic latency results in Figure 4a. Preliminary evidence indicates that this might be due to the performance of the 10GigE TOE for non-cached data[4]; we are investigating this behavior further.

[4] It is to be noted that the latency test in Figure 4a always sends data from the same buffer maximizing cache effects, while

4.3.2 PVFS Overview and Evaluation

We also studied the performance benefits of the different networks on parallel file systems and their applications. A popular parallel file system, Parallel Virtual File System (PVFS), is used as a case study in this work. In this section, we provide an overview of PVFS, its basic file-I/O performance and the performance of a parallel-I/O application, *mpi-tile-io*.

PVFS Overview: The Parallel Virtual File System (PVFS) [30] 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.

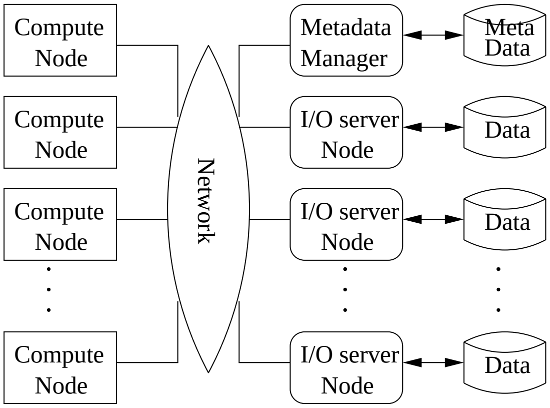


Figure 10. A Typical PVFS Setup

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

ISO always fetches files from the disk and cannot be expected to experience any cache effect whatsoever

Multi- Dimensional Block Interface (MDBI). The presence of multiple popular interfaces contributes to the wide success of PVFS in the industry.

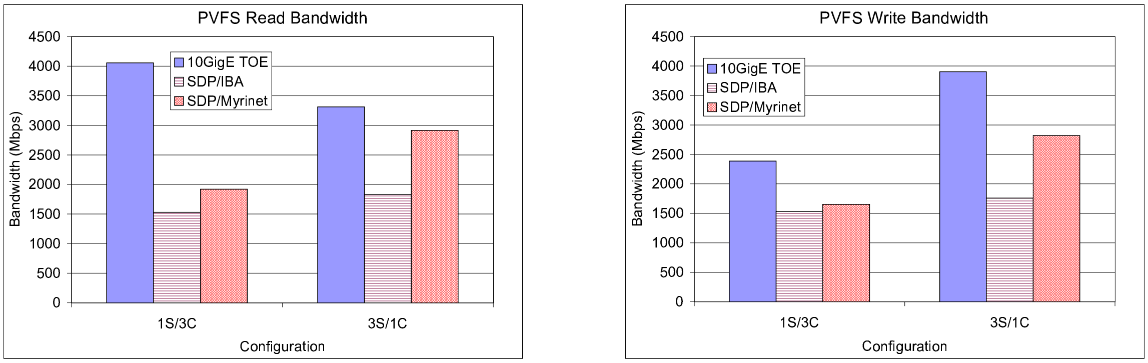


Figure 11. Concurrent PVFS Read/Write

Performance of Concurrent File I/O: In this test, we have evaluated 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 [30].

Figure 11 shows PVFS file read and write performance on the different networks. We have performed two kinds of tests for both read and write: (i) In the first test, we use just one server; three clients simultaneously read or write a file from/to this server, (ii) 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 and fan-in/fan-out micro-benchmark results in Figures 4b and 7. SDP/IBA, however, seems to achieve considerably lower performance as compared to even SDP/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 [32] 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.

We have evaluated 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.

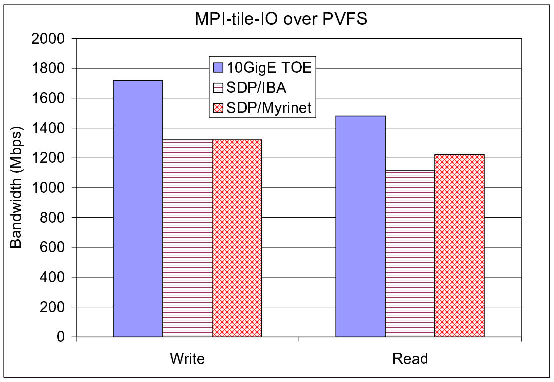


Figure 12. MPI-Tile-IO over PVFS

4.3.3 Ganglia Overview and Evaluation

The third application we use to evaluate the three networks is Ganglia [1]. 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, 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.

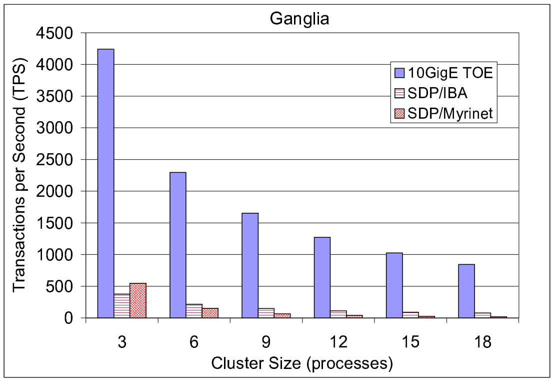


Figure 13. Ganglia: Cluster Management Tool

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 Figure 4a, 10GigE TOE and SDP/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/Myrinet and SDP/IBA. During connection setup, SDP/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/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.

5 Related Work

Several researchers, including ourselves, have previously shown the capabilities of high-performance sockets over protocol-offload engines. Shah et. al. from Intel were one of the first to demonstrate such capabilities using Virtual Interface Architecture (VIA) based GigaNet cLAN networks [33]. This was soon followed by other implementations of high-performance sockets on VIA [22, 23, 9], Gigabit Ethernet [8], Myrinet [27], InfiniBand [6], etc. However, while the literature based on these shows the advantages of using protocol offload engines compared to the host stack, there is no comparative study between the different networks making it quite difficult for end customers to gauge the advantages and disadvantages between the various networks. In our work, we fill this gap by having such a comparative study on a common testbed.

We have previously done a similar study comparing MPI implementations over IBA, Myrinet and Quadrics [24]. Our current work 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); this has not been studied as a part of the previous work on MPI. Second, this work focuses on the sockets interface which is quickly gaining popularity with the upcoming high-performance sockets standards such as SDP.

6 Concluding Remarks

Traditional Ethernet-based network architectures such as Gigabit Ethernet (GigE) have delivered significantly worse performance than other high-performance networks [e.g, InfiniBand (IBA), Myrinet]. In spite of this performance difference, the low cost of the network components and their backward compatibility with the existing Ethernet infrastructure have allowed GigE-based clusters to corner 35% of the Top500 Supercomputer List. With the advent of 10GigE and TCP Offload Engines (TOEs), we demonstrated that the aforementioned performance gap can largely be bridged between 10GigE, IBA, and Myrinet via the sockets interface. Our evaluations show that in most experimental scenarios, 10GigE provides comparable (or better) performance than IBA and Myrinet. Further, for grid environments, where legacy TCP/IP/Ethernet is dominant in the wide-area network, a comparison with IBA and Myrinet is practically a *no show* because of lack of compatibility of these networks with Ethernet.

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.

7 Acknowledgments

We would like to thank Dr. Saltz, Dr. Kurc and Dr. Catalyurek from the department of bio-medical informatics at the Ohio State University, for providing us access for the data-cutter runtime library as well as the relevant datasets for the Virtual Microscope and Iso-surface applications. We would also like to thank Felix Marti, Asgeir Eiriksson and Kianoosh Naghshineh from Chelsio Communications and Val Oliva, Todd Dewitt and Victor Stark from Foundry Networks for providing us with several useful insights into the architectural details about the 10GigE TOE network adapters and the SuperX 10GigE switch, respectively. We would also like to thank 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, November 2004.

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

[7] 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*.

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

[9] 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*.

[10] 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.

[11] 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.

[12] 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*.

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

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

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

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

[17] 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*.

[18] H. Frazier and H. Johnson. Gigabit Ethernet: From 100 to 1000Mbps. *IEEE Internet Computing*, Vol:3, Issue:1:24–31, January 1999.

[19] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*.

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

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

[22] 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*.

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

[24] 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 InfiniBand, Myrinet and Quadrics. In *Int'l Conference on Supercomputing*, 2003.

[25] Mellanox Technologies. Mellanox InfiniBand InfiniHost Adapters, July 2002.

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

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

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

[29] R. Oldfield and D. Kotz. Armada: A parallel file system for computational. In *Proceedings of CCGrid2001*, May 2001.

[30] P. H. Carns and W. B. Ligon III and 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.

[31] B. Plale and K. Schwan. dQUOB: Managing large data flows using dynamic embedded queries. In *HPDC*, August 2000.

[32] Rob B. Ross. Parallel i/o benchmarking consortium. http://www-unix.mcs.anl.gov/rross/pio-benchmark/html/.

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