RJ 10256 (A0209-023) September 12, 2002
Computer Science

IBM Research Report

Storage over IP: A Performance Study

Prasenjit Sarkar, Sandeep M. Uttamchandani, Kaladhar Voruganti
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Storage over IP: A Performance Study

Prasenjit Sarkar, Sandeep Uttamchandani, Kaladhar Voruganti
IBM Almaden Research Center
San Jose, CA 95120

1. Introduction

In the past, the storage model assumed the presence of storage attached to every host server. This type of host server-attached storage relied primarily on the *Small Computer System Interface* (SCSI) command protocol. The preferred transport for the SCSI command protocol in this server-attached storage model was Parallel SCSI where the storage devices were connected to the host server via a cable-based parallel bus. However, as the need for storage and servers grew, the limitations of this technology became obvious. The physical characteristics of the cable limit the number of storage devices as well as the distance of the storage devices from the host server. Also, the concept of attaching storage to every host server means that the storage had to be managed on a per-host server basis.

The lack of scalability and manageability of the host server-attached storage model led to the evolution of the concept of a *storage area network*. Storage devices are assumed to be independent machines that provide storage service via a network to a multitude of host servers. The advent of networking infrastructure capable of gigabit speeds as well as the development of transport protocols capable of sustaining such speeds further facilitates the service of storage over large-distance networks. Most storage area networks use Fibre Channel [3]; other storage area network technologies are Infiniband [6], VaxClusters [19], HIPPI [27] and IP [4,15,16].

This paper focuses on storage area networks based on the pervasive IP networking technology. The advantages of IP networks are obvious. First, using the same IP technology for both regular (non-storage) networks as well as storage networks removes the need to have two different types of networks in any infrastructure. Also, the use of a single popular networking infrastructure can leverage widely available network management skills. Second, the presence of well tested and established protocols allow IP networks both wide-area connectivity, scalable routing and addressing as well as proven bandwidth sharing capabilities. Third, the emergence of Gigabit Ethernet and the future arrival of 10 Gigabit Ethernet seems to indicate that the bandwidth requirements of serving storage over a network should not be an issue [1]. Finally, the commodity availability of IP networking infrastructure indicates the cost of building a storage area network will not be prohibitive.

The key contribution of this paper is to compare these three approaches for IP storage area networks with the help of both micro-benchmarks as well as macro-benchmarks. In the micro-benchmarks, the three approaches were compared with respect to latency and throughput by measuring their sensitivity to varying block sizes as well as CPU, I/O bus and memory speeds. The micro-benchmark analysis was projected onto the real world by running macro-benchmarks on each of the three approaches.

The results show that contrary to intuition, the hardware approaches are not inherently superior in terms of performance, which is surprising because of the cost of hardware offload. The results indicate that while the hardware support decreases the CPU utilization-to-throughput ratio for

large block sizes, the hardware support can itself be a performance bottleneck that hurts the rate of I/O operations. Consequently, not only is the software approach superior to the hardware approaches in terms of latency and throughput, but also outperforms the hardware approaches in macro-benchmarks. This projects for the need for an intelligent partitioning of storage transport functionality between hardware and software that can take advantage of the rate of increase in the computing power of general-purpose processors.

2. IP Storage

2.1. IP Storage Area Networks

A storage area network is the answer to the high total cost of ownership and the lack of scalability associated with server-attached storage. A storage area network consists of host servers connected to a set of storage subsystems via an interconnect fabric. The servers can access the storage as a service over the network, allowing these servers to share the storage in the subsystems. The emerging field of IP storage area networks can meet the requirements of storage area networks:

A high-bandwidth scalable network interconnect. A storage area network must provide high network bandwidths for storage to be delivered as a scalable service to applications residing in host servers. While IP networking can provide a scalable infrastructure as can be evidenced by the existence of the Internet, Gigabit Ethernet and 10 Gigabit Ethernet [10] can provide the necessary infrastructure for a high-bandwidth storage area network.

Reliable delivery. A storage area network needs a reliable transport protocol to exchange control and data between the host servers and the storage devices. Fortunately, the IP networking community has invested a lot of research into building a reliable in-order transport protocol called TCP [20]. With this in mind, the architects for IP storage area networks chose TCP as the primary transport protocol rather than pursue the time-consuming approach of inventing, deploying and fine-tuning a specialized protocol for storage transport.

Security and management. The IP networking infrastructure has support for security and management protocols that address the needs of a storage area network [7,9]. *SSL*, *Kerberos* and *IPSec* are some of the available IP security mechanisms. In terms of management, DNS allows for the unique worldwide naming, SLP for the discovery of resources on an IP network, and SNMP and SMI for the monitoring and diagnosis of IP network nodes.

2.2. Approaches to IP Storage

However, to properly analyze the merits of IP storage area network, it is important to note that there are several different approaches to building an IP storage area network.

First, the *software* approach envisages using a software TCP/IP stack for storage transport. Proponents of this approach claim that performance should scale with ever-increasing CPU speeds, obviating the need for any hardware assistance. However, preliminary results [13] using an IP storage area network based on a software TCP/IP stacks indicate that the main performance bottleneck for meeting the requirements of storage area networks is the high CPU utilization involved in large block transfers in software TCP/IP stacks. The two main components of the high CPU utilization in software TCP/IP stacks are:

- Interrupt overhead due to frame size transfers from the adapter to the host at high rates.
- The TCP copy-and-checksum overhead for large block transfers.

Second, the *jumbo frame* approach improves on the *software* approach by using 9KB Jumbo Ethernet frames to reduce the number of interrupts per bulk data transfer. However, the Jumbo Ethernet frames are not standardized and are not currently present in 10 Gigabit Ethernet. Consequently, this approach is not examined further in the paper.

Third, the *zero-copy* approach is also similar to the *software* approach but uses modified TCP/IP stacks with zero-copy transmit capability to reduce the TCP copy-and-checksum overhead; the responsibility of generating the checksum is also off-loaded to the network adapter. However, zero-copy receives are typically not possible on such stacks because the network adapters are unaware of the final destination of any frame. Re-mapping the network buffer onto an application buffer can remove the copy on the receive path but issues with page-alignment and virtual memory costs (particularly in SMP environments) has hindered adoption in production operating systems. Since the primary overheads on storage area networks occur on the receive path [13], this approach is also not examined further in the paper because of the lack of stable support for zero-copy receives.

Fourth, the *TOE* approach involved network adapters with TCP/IP offload engines [9] where the entire TCP/IP stack is offloaded onto the network adapter. This also reduces the TCP copy-and-checksum overhead. The interrupt overhead is also reduced because of the adapter generates at most one interrupt for every bulk data transfer. However, zero-copy receives are not possible on such stacks because the TCP/IP stack is also typically unaware of the final destination of any TCP/IP packet.

Finally, the *HBA* approach envisages the use of network adapters that have a specific storage transport interface (such as iSCSI) and is aware of the storage protocol semantics. This approach will also reduce the interrupt overhead, as the network adapter will ensure at most one interrupt per data transfer. More importantly, the protocol-specific data placement support in the adapter ensures that there are no copies on the receive path. As with the *TOE* approach, offloading the protocol processing to the adapter will eliminate the TCP/IP copy-and-checksum overhead. The HBA approach can be envisaged as a specialized version of the *RDMA* approach, which provides zero-copy receive support via direct data placement to any transport protocol; however, in terms of performance analysis, the HBA and RDMA approaches are functionally similar.

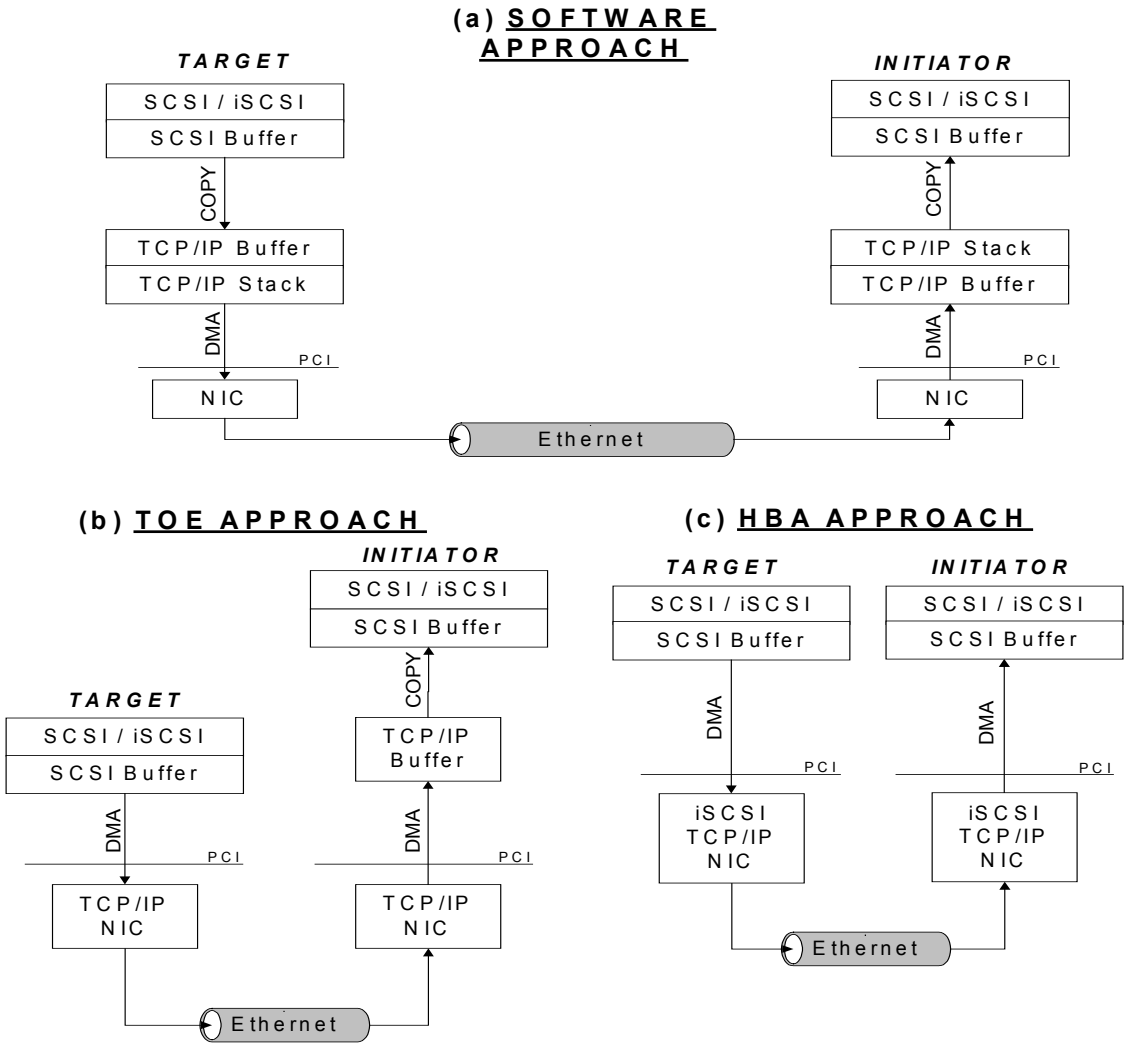

Figure 1. This figure shows the flow of a read response in each of the three approaches to IP storage for both the initiator and target (assuming that a read request has been made earlier). The storage transport protocol in this figure is assumed to be iSCSI. The software approach is shown in part (a) where the SCSI/iSCSI stack in the target copies the requested SCSI buffer into a TCP/IP buffer for transmission by the TCP/IP stack. The TCP/IP buffer is then DMA-ed onto the network adapter card where it is transmitted over the network. The receiving NIC then DMA-s this buffer into a TCP/IP buffer on the host for TCP/IP stack processing, after which the buffer is copied into the destination SCSI buffer for SCSI/iSCSI stack processing. The TOE approach is shown in part (b) where the SCSI/iSCSI stack in the target DMA-s the requested SCSI buffer directly onto a TCP/IP-capable NIC, where it is transmitted after the requisite TCP/IP protocol processing. On receiving the buffer, the TCP/IP-capable NIC on the receiving side DMA-s the buffer onto an anonymous TCP/IP buffer on the target. This anonymous buffer is then copied into the destination SCSI buffer for SCSI/iSCSI stack processing. The HBA approach is shown in part (c) where the SCSI stack in the target DMA's the requested buffer into the iSCSI-capable NIC for iSCSI/TCP/IP processing and transmission over the network. On receiving the buffer, the iSCSI-capable NIC in the receiving side performs iSCSI/TCP/IP protocol processing to learn the identity of the destination buffer and directly DMA-s to this destination buffer for SCSI protocol processing.

The software, TOE and HBA approaches are also currently the mainstream in IP storage area networks. Figure 1 shows the data flow of a read response in the software, TOE, and HBA approaches to better exemplify the differences between the approaches.

The rest of this paper compares the software, TOE and HBA approaches in terms of both micro-benchmarks and macro-benchmarks. The goal is to identify the incremental effect of TCP offload

as well as zero-copy receives on performance as we use the various approaches. The micro-benchmarks does a sensitivity analysis of each of these approaches with respect to block sizes as well as CPU, memory and I/O bus speeds to identify potential performance bottlenecks, while the macro-benchmarks project the various performance characteristics of each of these approaches onto application performance.

3. Micro-benchmarks

3.1. Experimental Setup

To evaluate the performance of IP storage, the protocol of choice was iSCSI [4]. The iSCSI protocol is an IETF proposed standard for transporting SCSI over TCP/IP and there are sufficient products available from many industry vendors to do an experimental analysis of all three approaches. For the micro-benchmark analysis, the experimental setup consisted of an iSCSI initiator workstation connected through an Alteon 180 Gigabit Ethernet switch to an iSCSI target server.

The iSCSI initiator workstation was powered by an AMD Athlon MP 1900 CPU (1.6 GHz CPU clock speed) with 4 GB of PC2100 DDR memory and support for both 64-bit 66 MHz and 32-bit 33 MHz PCI slots. The motherboard in the initiator workstation allowed the CPU front-side bus (FSB) speed to be varied from 90 MHz to 141 MHz with a fixed CPU clock multiplier; this enabled the measurement of the sensitivity of the results to CPU speed variations from 1.1 GHz to 1.7 GHz. The operating system running on the iSCSI initiator was Windows 2000 SP3.

The initiator workstation was configured with all the three approaches: software, TOE, and HBA. Evaluating a range of cards and choosing the card with the best performance decided the choice of network adapters for all three approaches. In addition, a performance analysis with the Smartbits tool [17] found no statistically significant difference between the performances of the Ethernet layer for the adapter cards chosen for each approach.

In the software approach, the initiator workstation ran the IBM Windows initiator v1.2.2 that implemented draft version 8 of the iSCSI standard. An Intel Pro/1000F Server NIC provided the connectivity. The checksum offloading feature of the NIC card was not utilized.

In the TOE approach, the NIC card was removed and replaced by an Alacritech SLIC TOE card that provides ASIC support to offload the fast-path TCP/IP functionality from the Windows TCP/IP stack. With the Alacritech TOE card, the iSCSI software used was still the IBM Windows initiator v1.2.2 except that the fast-path TCP functionality was taken over by a third-generation ASIC in the Alacritech TOE card. The partial RDMA feature of the Alacritech TOE card was not used as the complete RDMA functionality was evaluated using the HBA approach.

In the HBA approach, an Adaptec ASA-7211 iSCSI HBA replaced the Alacritech TOE card. The Adaptec TOE engine has an ASIC chip that provides TOE functionality. A 400 MHz StrongARM CPU on-board the Adaptec HBA provides an implementation of the draft version 8 of the iSCSI standard, though data transfers between the host system and the Adaptec HBA do not involve the StrongARM CPU. Consequently, there was no need to use the IBM Windows initiator in this approach.

Unless otherwise mentioned, the Intel, Alacritech and Adaptec cards were placed in a 64-bit 66 MHz PCI slot. In addition, the default CPU speed was always 1.6 GHz (133 MHz FSB speed).

The iSCSI target server was powered by a dual-800 MHz Pentium III CPU configuration with 1 GB of PC133 SDRAM memory and support for both 64-bit 66 MHz and 32-bit 33 MHz PCI slots. The target server was equipped with an IBM ServeRAID 4H SCSI PCI RAID controller card with 48 36-GB 10,000 RPM SCSI disks. An Intel Pro/1000F NIC provided the Gigabit Ethernet connectivity. Both cards were placed in 64-bit 66 MHz PCI slots in different PCI buses to avoid I/O bus contention. The operating system running on the target server was Linux 2.4.2-2.

The target server was only configured with the software approach while the approaches were varied in the initiator workstation to better identify the performance variation in the three approaches. Furthermore, the results of the performance analysis in Sections 3 and 4 also validate the fairness in the choice of the software approach in the target server. The target server ran an iSCSI server kernel daemon (IBM target v1.2.2) that implemented draft version 8 of the iSCSI standard. The kernel daemon also provided read caching functionality that allowed repeated read requests for blocks to be satisfied from the host memory of the iSCSI target server rather than from the RAID controller. The target server also provided support for write-back caching.

In the micro-benchmark experiments, the Iometer measurement application [12] on the initiator workstation would issue *read()* calls via the unbuffered block interface (ASPI) to the Microsoft SCSI layer. At this layer, the *read()* call would get translated to the corresponding SCSI commands and be sent to the low-level iSCSI driver. The use of the unbuffered block interface inhibits the use of caching in the Initiator workstation memory. This allows us to better measure the cost of transporting SCSI over TCP/IP without being polluted by cache effects in the initiator workstation. Experiments were also performed using *write()* calls but as the results did not reveal anything beyond the available conclusions, they are not reported due to space limitations.

In addition, all the read requests were directed to the same disk block address to take advantage of caching at the iSCSI target server. The reason for using cached read requests was to make sure that the results were focused on measuring storage transport efficiency and would not be contaminated by RAID and disk performance issues.

3.2. Performance Analysis

The metrics used to compare the three approaches to IP storage were throughput and latency. The throughput was measured using 16 worker threads in the Iometer measurement application described in Section 3.1. A larger number of worker threads were not used because the aggregate throughput did not increase beyond this number. Each thread in the measurement application issued 100,000 sequential read commands to the same disk block address and the aggregate throughput was measured on completion of the reads by all threads.

The latency measurements were performed using 1 worker thread in the Iometer measurement application described in Section 3.1. The worker thread in the measurement application issued 100,000 sequential read commands to the same disk block address and the latency was measured by dividing the elapsed time by the total number of commands.

Along with the measurement of the throughput, the CPU utilization at the initiator workstation and target server was measured using the PerfMon and *vmstat* tools respectively.

3.2.1. Block Size Sensitivity

The first experiment in the micro-benchmark performance analysis pertains to the sensitivity of throughput to variations in block size. The block size was varied from 0.5 KB to 64 KB and the

resultant throughput was measured for each of the three approaches and is shown in Figure 2. The corresponding initiator workstation CPU utilization is also shown in the right-hand side in the same figure. The target server CPU was not saturated in any of these experiments and moreover, the target server always used the software approach; consequently, the target server CPU utilization is not displayed.

The results show that the software approach achieves the best numbers in terms of throughput, though the initiator CPU is completely saturated for the lower block sizes of 0.5 KB to 8 KB. The question of whether a faster CPU can aid the performance is investigated in Section 3.2.2. In the larger block sizes of 16 KB to 64 KB, the performance of the software approach is limited by a resource bottleneck which can be attributed to either the PCI bus or the memory as the Intel adapter is capable of higher throughput. This resource bottleneck is investigated by using the I/O bus and memory speed sensitivity experiments in Section 3.2.3 and Section 3.2.4.

The initiator workstation CPU utilization to throughput ratio is of particular importance to applications that are sensitive to CPU cycle availability. These applications benefit only when the throughput is high and the CPU utilization to throughput ratio is low. Both the hardware approaches show lower ratios of initiator workstation CPU utilization to throughput, particularly when the block sizes are large. For example, at the 64 KB block size, the ratio for the TOE approach is just 52% of that of the software approach, though at the 4 KB and 0.5 KB block sizes, the ratio for the TOE approach is 77% and 96% of that of the software approach respectively. Similarly, the ratio of the HBA approach for the 64 KB, 4 KB and 0.5 KB block sizes is 17%, 73% and 113% of that of the software approach respectively. When the block size is large, the per-byte costs of the software approach due to TCP/IP copy-and-checksum and interrupt overhead (discussed in Section 2.2) increase the CPU utilization to throughput ratio. However, when the block size is small, the per-byte costs of the software approach are competitive with that of the hardware approaches and therefore, the CPU utilization reduction benefit of the two hardware approaches reduces significantly.

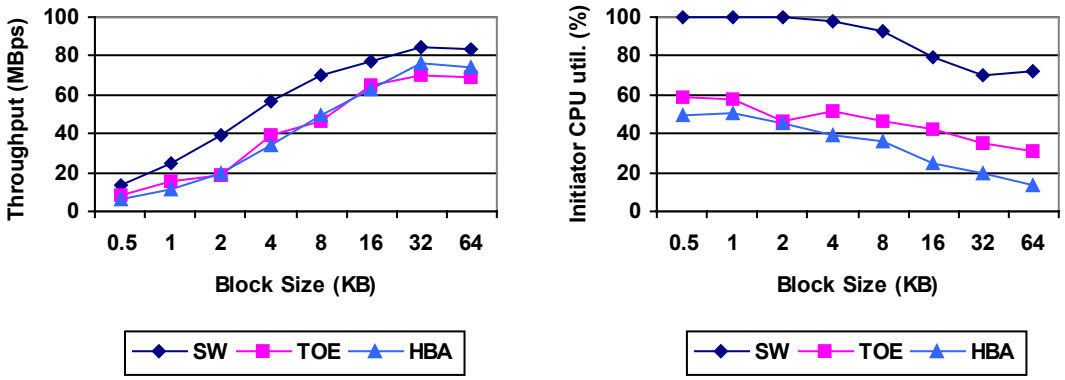

Figure 2. The figure shows the sensitivity of the throughput in each of the three approaches in relation to block size. The resultant initiator workstation CPU utilization is shown on the right.

The final thing to note is that the throughput of the hardware approaches does not match that of the software approach for any block size. It was observed that neither the initiator workstation CPU nor the target server CPU is completely utilized. Moreover, as the performance differential is also present when the block size is small (0.5 KB) and the throughput is not very high (~10 MBps), the PCI bus and memory speeds can be ruled out as a cause for the inferior performance of the hardware approaches. Consequently, the performance bottleneck can be pinpointed to the hardware offload in the TOE and HBA approaches.

The latency analysis shown in Table 1 also points to the performance bottleneck in the hardware offload in the TOE and HBA approaches. As was discussed in Section 2.2, the software approach has a high per-byte cost compared to the hardware approaches due to the TCP/IP copy-and-checksum in the receive path and the high interrupt overhead. In the smaller block sizes where the per-operation costs dominate per-byte costs, the software approach is clearly superior in terms of latency. When the block size is large, the per-byte costs dominate the per-operation costs and the superiority of the software approach is narrowed. The above points to increased per-operation costs in the hardware approaches compared to the software approach and indicates a performance bottleneck in terms of a mismatch in processing speeds between the host system and the hardware offload. The latency in the HBA approach is particularly high because of the fixed-cost involvement of the StrongARM CPU; however this is not considered inherent to the HBA approach as alternative HBA designs have merged iSCSI and TOE functions into a single ASIC.

	Latency (ms)			
Block Size (KB)	*0.5*	*4*	*8*	*64*
Software	0.12	0.17	0.22	0.97
TOE	0.17	0.26	0.28	1.01
HBA	0.41	0.47	0.51	1.52

Table 1. The table shows the sensitivity of the latency in each of the three approaches in relation to block size.

3.2.2. CPU speed sensitivity

The next experiment in the micro-benchmark performance analysis measured the sensitivity of throughput to the CPU speed in the initiator workstation. In this experiment, the CPU FSB speed was varied from 92 MHz to 141 MHz with a fixed clock multiplier, resulting in an effective CPU speed variation from 1.1 GHz to 1.7 GHz. The throughput was measured with the block sizes of 0.5 KB, 4 KB, 8 KB and 64 KB for each of the three approaches.

The results show that performance of the software approach scales with increasing CPU speeds, particularly for the smaller block sizes. However, when the block size is 64 KB, the resource bottleneck (investigated in the following sub-sections) prevents any scaling of throughput with respect to CPU speeds. The performance of the TOE approach is marginally sensitive to increasing CPU speeds while that of the HBA approach shows no sensitivity at all. This is to be expected due to the offload of protocol processing onto the adapter cards in the hardware approaches.

A similar effect was also observed in the sensitivity of the latency of the three approaches to increasing CPU speeds.

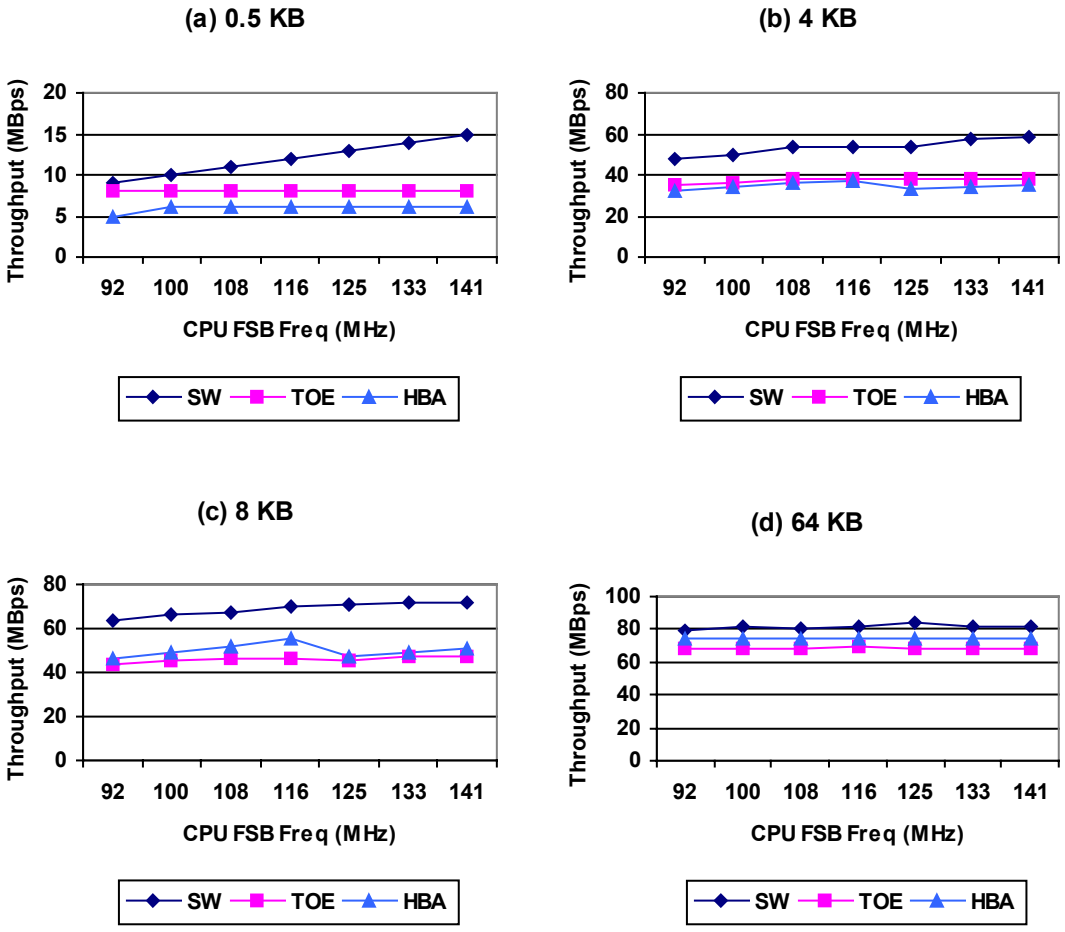

Figure 3. This figure shows the sensitivity of the throughput in relation to CPU speeds (denoted in terms of CPU FSB frequency) in each of the three approaches for the block sizes of (a) 0.5 KB, (b) 4 KB, (c) 8 KB, and (d) 64 KB.

3.2.3. I/O bus speed sensitivity

The third experiment in the micro-benchmark performance analysis measured the sensitivity of throughput to the PCI bus speed in the initiator workstation. The relevant adapter in each of the three approaches was placed alternately in a 32-bit 33 MHz PCI bus and a 64-bit 66 MHz PCI bus. The throughput was measured with the block sizes of 0.5 KB, 4 KB, 8 KB and 64 KB to observe the PCI bus effect.

The results are shown in Figure 4 and indicate that all the approaches are sensitive to the PCI bus speed of the slot holding the adapter corresponding to the approach, particularly at the 64 KB block size. However, the software approach is the most sensitive to the PCI bus speed as the throughput drops 28% for the 64 KB block size when the PCI bus speed is lowered from 64-bit 66 MHz to 32-bit 33 MHz. The corresponding numbers for the TOE and HBA approach are 18% and 10% respectively. This is due to the fact that the PCI overhead is amortized over 64 KB transfers in the hardware approaches, while the software approach incurs the same overhead over Ethernet MTU-sized transfers (1.5 KB). The impact of the difference in overhead is more visible in 32-bit 33 MHz PCI because of the slower PCI bus speed.

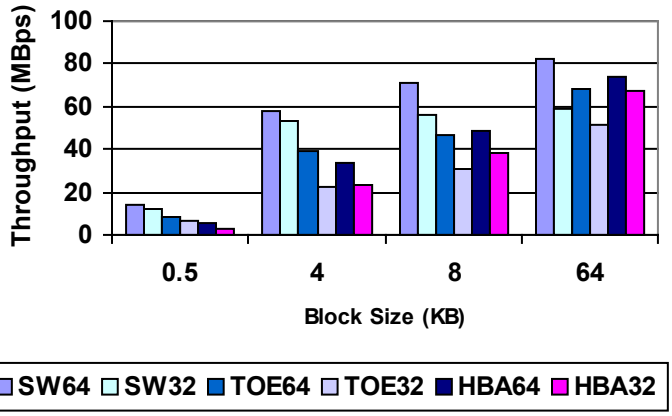

Figure 4. This figure shows the sensitivity of the throughput analysis with respect to PCI bus speeds. The legends SW64, TOE64 and HBA64 represent the throughput in relation to block size when the corresponding adapter is placed in a 64-bit 66 MHz PCI bus slot. Similarly, the legends SW32, TOE32 and HBA32 represent the throughput in relation to block size when the corresponding adapter is placed in a 32-bit 33 MHz PCI bus slot.

A CPU sensitivity analysis of the throughput of the software approach at the 64 KB block size when the adapter is placed in a 32-bit 33 MHz slot showed no improvement in throughput when the CPU speed was varied. A similar phenomenon was observed in Section 3.2.2 when the adapter was placed in a 64-bit 66 MHz PCI slot. This indicates that the PCI bus speed is an important factor in the resource bottleneck described in Section 3.2.1.

The latency analysis of the three approaches with regards to the sensitivity to the PCI bus was also measured. The analysis showed that the differential between the latency using a 32-bit 33 MHz PCI slot and that using a 64-bit 66 MHz slot was marginal (< 5%) for all approaches when the block size used was small (0.5 KB, 4KB) but increased to as much as 50% when the block size was large (64 KB).

3.2.4. Memory speed sensitivity

The final experiment in the micro-benchmark performance analysis measured the sensitivity of throughput and latency to the memory speed in the initiator workstation. The latency and throughput measurements were taken with the default DDR2100 memory (266 MHz) as well as the alternate DDR1600 memory (200 MHz) in the initiator workstation. The throughput and latency was measured with the block sizes of 0.5 KB, 4 KB, 8 KB and 64 KB to observe the memory speed effect. No statistically significant differences were found between the results of the throughput and latency analysis for both the memory speeds, indicating that the current memory speeds are not a resource bottleneck at these rates of throughput. This further indicates that the principal resource bottleneck discussed in Section 3.2.1 is the PCI bus speed.

4. Macro-benchmarks

4.1. Experimental Setup

The macro-benchmark analysis attempts to project the results obtained from Section 3 onto application performance. For the macro-benchmark analysis, the experimental setup was identical to that of the micro-benchmark analysis. As in Section 3, the target server was only configured with the software approach while the approaches were varied in the initiator workstation.

4.2. Performance Analysis

The TPC-C and Postmark benchmarks were used to evaluate the software, TOE and HBA approaches so as to capture the principal application categories of databases and email. We are currently evaluating the three approaches using the TPC-H and SPC-1 benchmarks.

4.2.1. TPC-C

The TPC Benchmark C or TPC-C [8] is an on-line transaction processing (OLTP) benchmark. Multiple transaction types, database complexity, and overall execution structure distinguish the TPC-C benchmark. The metric for measuring performance is number of transactions completed per minute (tpmC). The benchmark specifies a method for scaling the database based on an assumed business expansion path of the supplier.

The database used was DB2 v7.2 Enterprise Edition. The database settings were fine-tuned for performance on the initiator workstation configuration based on input from IBM Toronto Laboratory. For this analysis, we varied the number of database warehouses from 200 to 800 and observed the tpmC. The database resides in a single file-system spanning 42 drives on the target server so that the benchmark is not limited by disk drive performance in the experimental configuration. The TPC-C benchmark is characterized by random page-sized I/Os with reads slightly dominating writes. The benchmark is highly multi-threaded.

The results (Figure 5(a)) show that the software approach has better results compared to the hardware approaches even for this CPU-intensive benchmark. Even though the software approach has a higher CPU utilization to throughput ratio for the paged sized transfers used in the benchmark, the software approach is able to compensate for this by a higher rate of I/O operations at this block size as seen in Section 3.2.1. Figure 5(b) shows that the software approach has a higher average CPU utilization during the benchmark.

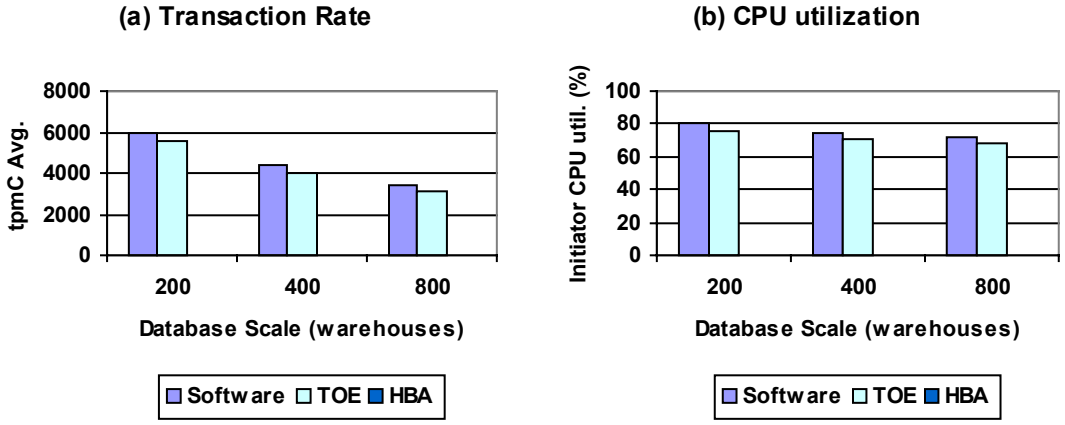

Figure 5. (a) This figure shows the average transactions completed per minute for the three approaches: Software, TOE and HBA when running the TPC-C benchmark for a database scale that spans 200 to 800 warehouses. (b) This figure shows the average CPU utilization in the initiator workstation for the experiments described in (a).

4.2.2 PostMark

The PostMark benchmark [10] simulates workloads that measure the performance of a file system when dealing with small sized files. This benchmark captures the behavior of electronic mail,

news and web commerce applications. The benchmark consists of creating files, performing read and write operations (called transactions), and then deleting the files. The benchmark allows for the specification of the number of files to be created, file size, read and write probability, number of transactions, and unit of data transfer between the client and the server. In contrast to the previous benchmark, this benchmark is single-threaded.

The PostMark benchmark was run with the default parameters except that the number of files was reduced to 150 so as to remove the effect of file system lookup efficiency from storage transport protocol analysis. As the file sizes were varied, there was not much difference between the performances of the three approaches for the small file sizes (<= 64 KB). This was not unexpected as the overhead of setting up a PostMark transaction was comparable to the actual PostMark transaction cost. However, for larger file sizes (> 64 KB), the software approach starts showing superiority in terms of time to run the benchmark by as much as 40% over the hardware approaches (shown in Figure 6) because of the superior latency in the block sizes used by the benchmark.

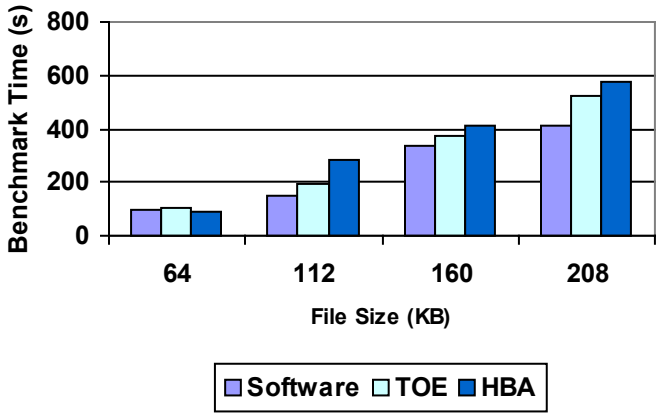

Figure 6. This figure shows the time to complete the PostMark benchmark for the three approaches: Software, TOE and HBA when the size for 150 files is varied from 64 KB to 208 KB in increments of 48 KB. The remaining PostMark parameters are the default.

5. Discussion

Summarizing the results of Sections 3 and 4, the software approach achieves the best numbers in terms of throughput and latency compared to the hardware approaches. The software approach has a high CPU utilization-to-throughput ratio for large block sizes as of the result of the high per-byte overhead inherent in this approach. However, the performance of the software approach scales with CPU speeds. While current CPU speeds are high enough to absorb the overheads for Gigabit Ethernet networks, it is in the near future that they will be high enough for 10 Gigabit Ethernet. In this particular experimental setup, the maximum throughput of the software approach is limited by PCI bus speeds. The software approach is also more sensitive to PCI bus speed because of the use of smaller transfer sizes on the PCI bus. Overall, the software approach is very competitive with the hardware approaches in the block sizes typically used by database and email applications and does not suffer from hardware performance bottlenecks. Consequently, the software approach shows superiority in such application benchmarks.

The designers of the hardware approaches do achieve lower CPU utilization-to-throughput ratios for large block sizes (64 KB), but the benefit of this reduction reduces significantly when the block size is not high (0.5 KB to 4 KB). Moreover, the latency and throughput analysis points to

a hardware bottleneck in protocol processing primarily because of the disparity in the processing speeds of the host system and the hardware offload. Also, the hardware approaches are not sensitive to increases in CPU speeds, a phenomenon with twin implications. First, all improvements in the performance of the hardware approaches must come from increasing the processing speeds of the hardware offload through superior ASIC technology. Second, the increase in processing speed of the hardware offload must keep up with the corresponding increases in general-purpose processors so as to be competitive with the software approach. An alternative to the current trend in the hardware approaches would be to examine the division of protocol processing between software and hardware so as to better take advantage of Moore's Law in general-purpose computing.

6. Related Work

Rod Van Meter et al [2,14] was one of the early proponents of the concept of storage over IP. The VISA (Virtual Internet SCSI Adapter) infrastructure implemented a SCSI transport layer and aimed to demonstrate that Internet protocols could serve as a communication base for SCSI devices. The initial performance analysis identified CPU overhead as well as protocol de-multiplexing as potential bottlenecks. Around the same time, Garth Gibson et al [24] proposed two innovative architectures for exposing storage devices to the network for scalability and performance. The NetSCSI architecture envisaged exposing SCSI devices to the generic network while the NASD architecture involved exporting secure object storage services over the network.

Numerous performance studies have been conducted in the past examining TCP/IP stack overheads for gigabit networks. Keng et al [5] and Gallatin et al [1] evaluated TCP/IP at near-Gigabit speeds to provide a breakdown of the various TCP/IP costs. The studies point out that lack of zero-copy and checksum offloading impact TCP/IP performance at high speeds because these operations increase the host CPU utilization. The zero-copy implementation used in these papers assumes page-aligned sends and receives, and uses page re-mapping techniques to implement zero-copy functionality. However, the page re-mapping zero-copy functionality differs from traditional implementations in that it has page alignment issues, and incurs virtual memory mapping overheads, particularly in SMP environments. Furthermore, this particular approach does not also mitigate the interrupt overhead at high data rates.

The effect of large MTU sizes on TCP/IP performance has been also well studied before by Chase et al [25]. The authors show that a large MTU size reduces the number of interrupts and improves TCP/IP performance. As has been pointed out in the previous section, large MTU sizes are not a viable alternative in Ethernet environments due to lack of standardization. In contrast, the TOE approach builds upon the interrupt coalescing analysis by studying the impact of interrupt coalescing even for standard Ethernet frames by using generic TCP/IP offload cards that provide interrupt coalescing regardless of the MTU size.

Magoutis et al [26] performed a thorough analysis of the key architectural elements of DAFS and reported results comparing the DAFS and NFS-nocopy protocols that utilize different zero-copy receive mechanisms. The analysis shows that while both DAFS and NFS-nocopy can achieve high throughput, the direct data placement architecture of DAFS results in lower CPU utilization. In contrast, the performance analysis in this paper complements the DAFS study by focusing on block storage protocols and incrementally examining the effect of TCP/IP offload and zero-copy support on the same protocol using the TOE and HBA approaches. The incremental analysis is useful because of the emergence of TOE adapters that do not have support for zero-copy receives.

Wee Teck Ng et al [23] performed an exhaustive performance analysis of iSCSI over wide-area network. This analysis proposes techniques for reducing data access times to combat the high latency in such long-haul networks. The focus of our paper is on the high-bandwidth low-latency local area network environment, and we assess the impact of the various approaches to IP storage to maximize throughput and minimize latency. Thus, the results of the wide-area network analysis are also complementary to our study.

Boon Ang et al [11] did a performance investigation on the impact of TCP/IP offload. Their study indicates that if the TCP/IP offload technology uses inexpensive network processors instead of ASICs, then the resulting performance may not be able to reach wire speeds.

References

[1]A. Gallatin, J. Chase, and K. Yocum, "Trapeze/IP: TCP/IP at Near-Gigabit Speeds", Proceedings of USENIX Technical Conference (FreeNix Track), June 1999.
[2]S. Hotz, R. Van Meter, and G. Finn, "Internet Protocols for Network Attached Peripherals", 6th IEEE/NASA Conference on Mass Storage Systems and Technologies, March 1998.
[3]A. Benner, "Fibre Channel: Gigabit Communications and I/O For Computer Networks", McGraw-Hill, 1996.
[4]J. Satran et al., "iSCSI", Internet Draft, IETF, 2002.
[5]Hsiao Keng, and J. Chu, "Zero-copy TCP in Solaris", USENIX 1996 Annual Technical Conference, January 1996.
[6]http://www.infinibandta.org
[7]E. Riedel et al., "A Framework for Evaluating Storage System Security", FAST, 2002.
[8]TPC Benchmark C Standard Revision 3.3.2, Transaction Processing Performance Council. 1997.
[9]E. Miller and D. Long, "Strong Security for Network-Attached Storage", FAST, 2002.
[10]J. Katcher, "PostMark: A New File System Benchmark", Technical Report TR3022, Network Appliance Inc, 1997.
[11]Boon S. Ang, "An Evaluation of an Attempt at Offloading TCP/IP Protocol Processing on to an i960RN-based iNIC", Hewlett-Packard Technical Report HPL-2001-8, January 2001.
[12] http://www.intel.com/design/servers/devtools/iometer
[13]P. Sarkar and K. Voruganti, "IP Storage: The Challenge Ahead", 19th IEEE Symposium on Mass Storage Systems, April 2002.
[14] R. Van Meter, G. Finn, and S. Hotz, "VISA: Netstation's Virtual Internet SCSI Adapter", ASPLOS 8, 1998.
[15]J. Tseng et al., "iFCP - A Protocol for Internet Fibre Channel Storage Networking", Internet Draft, IETF, 2002.
[16]M. Rajagopal et al., "Fibre Channel Over TCP/IP", Internet Draft, IETF, 2002.
[17]http://www.spirentcom.com
[18]S. Kent et al., "Security Architecture for the Internet Protocol", RFC 2401, 1998.
[19 N. Kronenberg et al. "VaxClusters: A Loosely Coupled Distributed System", ACM TOCS, 1986.
[20]Information Sciences Institute, "Transmission Control Protocol", RFC 760, 1980.
[21]http://www.nwfusion.com/newsletters/stor/2001/00655082.html
[22]http://www.iscsistorage.com/iscsidevices.htm
[23]Wee Teck Ng, H. Sun, B. Hillyer, E. Shriver, E. Gabber, B. Ozden, "Obtaining High Performance for Storage Outsourcing", FAST, 2002.
[24]G. Gibson et al., "A Case for Network-attached Secure Disks", CMU SCS TR CMU-CS-96-142, 1996.
[25]J. Chase et al., "End-System Optimizations for High-Speed TCP", IEEE Communications, 39:4, 2001.
[26]K. Magoutis, S. Addetia, A. Federova, M. Seltzer, J. Chase, A. Gallatin, R. Kisley, R. Wickremsinghe, E. Gabber, "Structure and Performance of the Direct Access File System", USENIX Technical Conference, June 2002.
[27]HIPPI, ANSI Standard X3T9.3/90-043, 1990.