USENIX Association

Proceedings of FAST '03: 2nd USENIX Conference on File and Storage Technologies

San Francisco, CA, USA
March 31–April 2, 2003

Storage over IP: When Does Hardware Support help?

Prasenjit Sarkar, Sandeep Uttamchandani, Kaladhar Voruganti
IBM Almaden Research Center
San Jose, California, USA
{prsarkar, sandeepu, kaladhar}@us.ibm.com

Abstract

This paper explores the effect of the current generation of hardware support for IP storage area networks on application performance. In this regard, this paper presents a comprehensive analysis of three competing approaches to build an IP storage area network that differ in their level of hardware support: software, TOE (TCP Offload Engine) and HBA (Host Bus Adapter). The software approach is based on the unmodified TCP/IP stacks that are part of a standard operating system distribution. For the two hardware-based approaches (TOE, HBA), we experimented with a range of adapters and chose a representative adapter for the current generation of each of the hardware approaches.

The micro-benchmark analysis reveals that while hardware support does reduce the CPU utilization for large block sizes, the hardware support can itself be a performance bottleneck that hurts throughput and latency with small block sizes. Furthermore, the macro-benchmark analysis demonstrates that while the current generation of the hardware approaches may have the potential to provide performance improvements in CPU-intensive applications, overall the analysis does not demonstrate any performance benefits in database, scientific and email benchmarks. The analysis in this paper points out that a disparity in the processing power between the host and the adapter is the primary cause of the performance bottleneck in the current generation of the hardware approaches. The paper aims to guide the designers of the next generation of hardware-assisted adapters to better leverage the increasing processing power in the host. In particular, future adapters should be capable of handling small operations at wire speed.

1. Introduction

In the past, a typical server-class installation assumed the presence of storage attached to every host system. This type of host system-attached storage relied primarily on the block-based SCSI protocol. The preferred transport for the SCSI protocol in this model was Parallel SCSI where the storage devices were connected to the host system via a cable-based parallel bus. However, as the need for storage grew, the limitations of this technology became obvious. The physical characteristics of the cable limited the number of storage devices as well as the distance of the storage devices from the host system. Also, storage had to be managed on a per-host system basis.

The lack of scalability and manageability of the host system-attached storage model led to the evolution of the concept of a storage area network. In this new model, storage devices are independent entities that provide block storage service via a network to a multitude of host systems. The advent of gigabit networking coupled with the development of high-speed transport protocols further facilitates the service of storage over networks. Most storage area networks use Fibre Channel [Benner96]; other storage area network technologies are Infiniband [Shankley02], VaxClusters [Kronenberg86], HIPPI [Ansi90] and IP [Satran02, Rajagopal02]. A comparison of the principal storage area networking technologies can be found in [Voruganti01].

This paper focuses on storage area networks based on IP networking technology. The advantages of using IP networks are many. 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 as well as proven bandwidth sharing capabilities. Third, the emergence of Gigabit Ethernet seems to indicate that the bandwidth requirements of serving storage over a network should not be an issue. Finally, the availability of commodity IP networking infrastructure indicates the cost of building a storage area network will not be prohibitive.

The aim of this paper is to explore the effect of the current generation of hardware support for IP storage area networks on application performance. The paper begins

by describing the various approaches possible in IP storage area networks, and focuses on the three prevalent approaches that differ in the level of hardware support. In the software approach, all TCP/IP and storage transport protocol processing is done on the host system. In addition, the software approach relies on unmodified TCP/IP stacks that are part of the standard operating system distribution. In the TOE (TCP Offload Engine) approach, the TCP/IP protocol processing is offloaded to the network adapter while the storage transport protocol processing is done in the host system. Finally, in the HBA (Host Bus Adapter) approach, the entire storage transport protocol processing is offloaded to the network adapter along with TCP/IP protocol processing.

The key contribution of this paper is to compare these three approaches for IP storage area networks with the help of micro-benchmarks and macro-benchmarks. In the micro-benchmark analysis, the three approaches were compared with respect to latency and throughput by measuring their sensitivity to block sizes as well as CPU, I/O bus and memory speeds. The micro-benchmark analysis was projected onto the real world by running database, scientific and email macro-benchmarks on each of the three approaches. To obtain a representative adapter for each of the hardware approaches, we experimented with a range of adapters and then chose the one with the best performance profile for the micro-benchmark and macro-benchmark analysis.

The results show that contrary to intuition, the representative adapters of the current generation of the hardware approaches are not inherently superior in terms of performance, which is surprising given 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 in comparison to the software approach for small block sizes. This performance bottleneck can be isolated to the disparity in computing power between the host and the current generation of the hardware-assisted adapters. Consequently, the current generation of the hardware approaches is not superior in terms of latency and throughput. This phenomenon is also observed in database, email and scientific benchmarks. This calls for the need for intelligent hardware support that can take advantage of the increased computing power of general-purpose processors.

2. IP Storage

The emerging field of IP storage area networks has the necessary technical infrastructure that makes it possible to transport block storage traffic:

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 systems. In the context of IP networks, Gigabit Ethernet 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 systems and the storage devices. Fortunately, the IP networking community has invested a lot of research into building a widely-deployed, reliable and in-order transport protocol called TCP. 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. SSL, Kerberos and IPSec are some of the available security mechanisms. In terms of management, DNS allows for unique worldwide naming, SLP for discovery of resources on an IP network, and SNMP and SMI for monitoring and diagnosis of IP network nodes.

It should be noted here that IP storage area networks focus on block service in contrast to network file systems that provide remote file access over IP networks.

2.1. Approaches to IP Storage Area Networks

There are three main approaches to building an IP storage area network, each with its distinct performance characteristics.

2.1.1. Software

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 [Sarkar02] using this approach indicate that the main

performance bottleneck is the high CPU utilization involved in large block transfers. The two major components of this high CPU utilization are:

- Interrupt overhead due to the high rate of Ethernet frame-sized transfers from the adapter to the host.
- The TCP copy-and-checksum overhead for large block transfers.

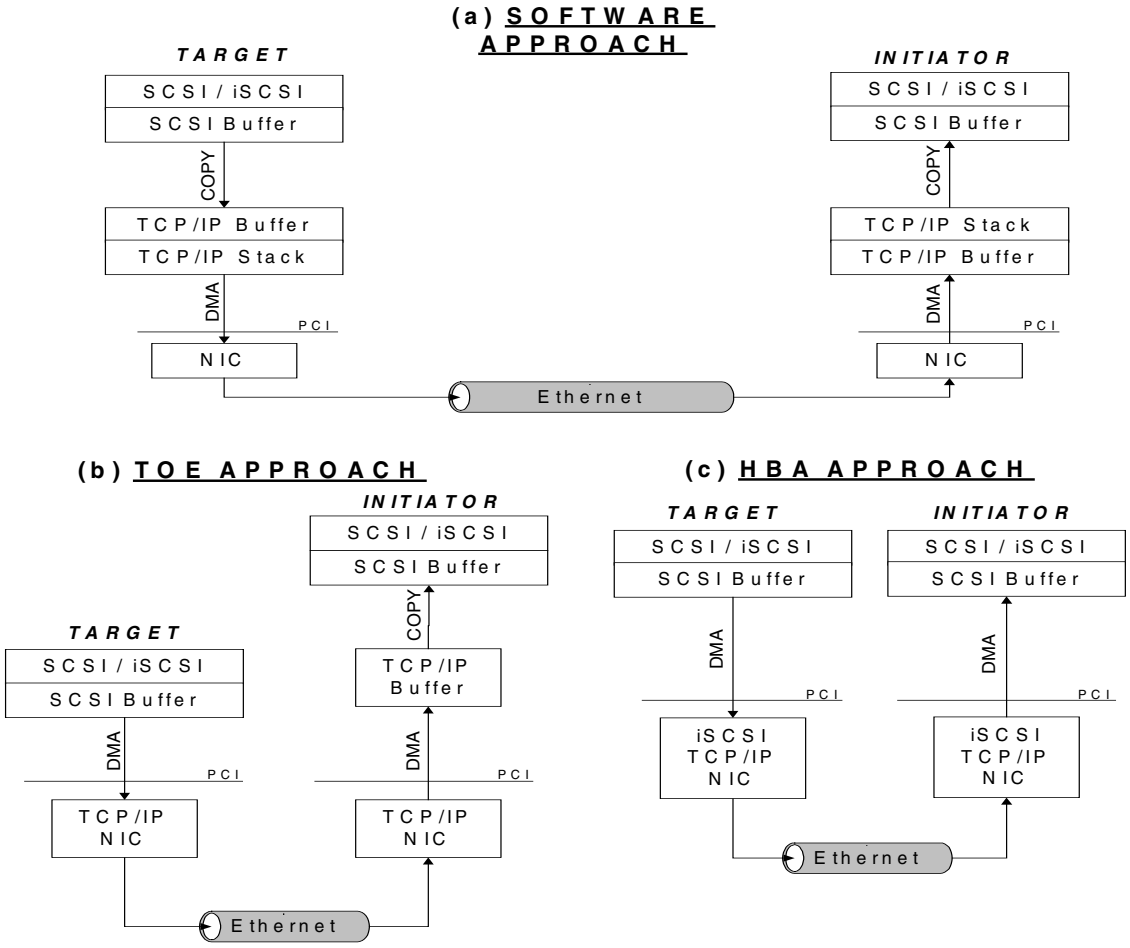


Figure 1. This figure shows the flow of a read response in each of the three approaches to IP storage for both the initiator and target (assuming that a read request has been made earlier). The storage transport protocol in this figure is assumed to be iSCSI. The software approach is shown in part (a) 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 *jumbo frame* approach is a variant of the software approach and improves on the software approach by using 9KB Jumbo Ethernet frames to reduce the per-packet overhead. However, Jumbo Ethernet frames are controversial as detractors claim that large frame sizes are detrimental to efficient routing and quality of service. Due to a lack of consensus, Jumbo Ethernet frames are not standardized and may not be present in the future. Consequently, this approach is not examined further in the paper.

Yet another variant of the software approach is the *zero-copy approach* which uses modified TCP/IP stacks with zero-copy transmit capability. This approach reduces the TCP copy-and-checksum overhead as the responsibility of generating the checksum is offloaded 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. However, issues with page-alignment and virtual memory costs (particularly in SMP environments) have hindered adoption in production operating systems. Since the primary overheads on storage area networks occur on the receive path [Sarkar02], this approach is also not examined further in the paper because of the lack of stable support for zero-copy receives.

2.1.2. TOE

The *TOE approach* involves network adapters with TCP/IP offload engines 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 large block transfer. However, zero-copy receives are usually not possible on such stacks because the TCP/IP stack is also typically unaware of the final destination of any TCP/IP packet, though the discussion of re-mapping the network buffer in Section 2.1.1 is also relevant here. Another complication for zero-copy receives is the presence of higher-level protocol headers in the data stream that complicates buffer alignment.

2.1.3. HBA

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 direct 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 [Bailey02, Compaq97], 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 both provide zero-copy receives and TCP/IP offload and are functionally similar.

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 whether the incremental hardware support necessarily improves performance. The micro-benchmarks do a sensitivity analysis of each of these approaches with respect to block sizes, CPU, memory and I/O bus speeds so as to identify potential performance bottlenecks. The macro-benchmarks project the various performance characteristics of each of these approaches onto real-world applications.

3. Micro-benchmarks

3.1. Experimental Setup

To evaluate the performance of IP storage, the protocol of choice was iSCSI [Satran02]. The iSCSI protocol is an IETF proposed standard for transporting SCSI over TCP/IP. There are sufficient iSCSI 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.

3.1.1. Initiator Workstation Setup

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 supported both 64-bit 66 MHz and 32-bit 33 MHz PCI slots. The Athlon 1900 CPU was in the 48-th percentile

of all the available CPUs in terms of SpecInt performance at the time of testing. 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 CPU speed variations from 1.1 GHz to 1.7 GHz. The operating system running on the iSCSI initiator workstation was Windows 2000 SP3. There were no modifications made to the TCP/IP stack or any other kernel component.

The initiator workstation was configured with all the three approaches: software, TOE, and HBA.

The network adapter for each approach was chosen after evaluating five HBA and TOE cards from different manufacturer. All of these products were released on 2002 and represent the state of the art. We performed read-cache hit benchmarks with the block sizes of 512 bytes and 64 KB from the initiator workstation to the target server to get an initial performance profile of each adapter. We used this benchmarks to select the best adapter for each of the hardware approaches. The percentage difference between the best performing adapter and the worst performing adapter on the 512 byte and 64 KB read-cache hit test was 12% and 15% respectively.

In the software approach, the initiator workstation ran the IBM Windows initiator v1.2.2 kernel-mode driver that implemented draft version 8 of the iSCSI standard. The kernel-mode driver did not have support for zero-copy receives because of buffer alignment issues. 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 selected TOE adapter replaced the Intel NIC card. This TOE card provided ASIC support to offload the fast-path TCP/IP functionality from the Windows TCP/IP stack. With the 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 TOE card.

In the HBA approach, the selected iSCSI HBA card replaced the TOE card. The TOE engine on the HBA card has an ASIC chip that provides TOE functionality. Firmware running on a CPU on-board the HBA card provides an implementation of the draft version 8 of the iSCSI standard, though data transfers between the host system and the HBA do not involve the CPU.

Unless otherwise mentioned, the adapter cards for the three approaches 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 maximum number of outstanding I/Os in the iSCSI protocol was set to 60 in each of the three approaches. Since jumbo frames were not universally supported, we used the default Ethernet frame size of 1.5K.

3.1.2. Target Server Setup

The iSCSI target server was powered by a dual-800 MHz Pentium III CPU configuration with 1 GB of PC133 SDRAM memory, and supported 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 with no modification to the TCP/IP stack or any other kernel component.

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.

3.1.3. Measurement Tool

In the micro-benchmark experiments, the Iometer measurement application [Intel02] on the initiator workstation issued read() calls via the unbuffered block interface (ASPI) to the SCSI layer. At this layer, the read() call got translated to the corresponding SCSI commands and was 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.

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 Iometer. 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 (Section 3.1). The worker thread in Iometer 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. The CPU utilization at the initiator workstation and target server was also measured.

3.2.1. Block Size Sensitivity

The first experiment in the micro-benchmark performance analysis pertains to the sensitivity of throughput to variations in the block size used by the Iometer measurement tool. The block size was varied from 0.5 KB to 64 KB and the resultant throughput is shown in Figure 2. The corresponding initiator workstation CPU utilization is shown on the right-hand side in the same figure. The target CPU was not saturated in any experiment.

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 threshold which can be attributed to either the PCI bus or the memory as the Intel adapter is capable of higher throughput. This resource threshold is investigated by using the I/O bus and memory speed sensitivity experiments in Sections 3.2.3 and 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 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 (Section 2.1) 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 resulting in comparable CPU utilization-to-throughput ratios.

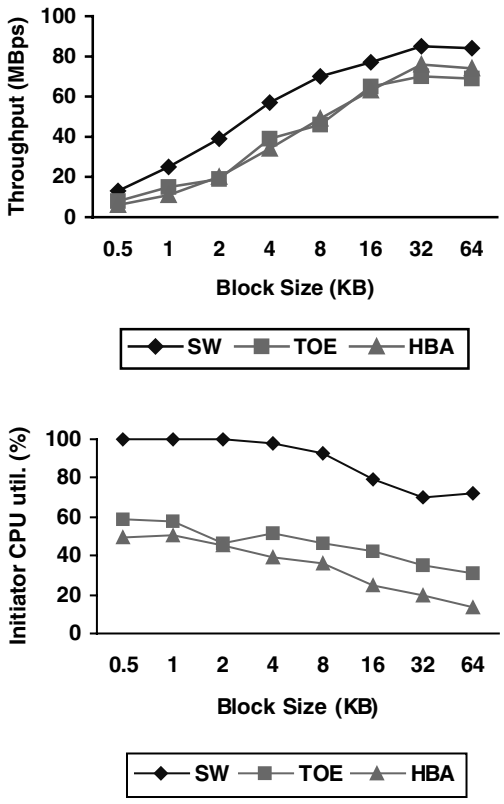


Figure 2. The figure shows the sensitivity of the throughput in each of the three approaches in relation to block size. The resultant initiator workstation CPU utilization is shown on the right.

The 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 reconfirms the performance bottleneck in the hardware offload for the TOE and HBA approaches. As was discussed in Section 2.1, the software approach has a high per-byte cost compared to the hardware approaches due to the TCP/IP copy-and-checksum and the interrupt overhead in the receive path. 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 latency in the HBA approach is particularly high because of the involvement of the slow 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

The performance bottleneck in the hardware approaches could be either in the software drivers or in the offloaded protocol processing engines. To further analyze the bottleneck, we measured the per-operation cost in the host systems for each of these approaches. We conducted an experiment on the default setup with a block size of 512 bytes to reduce the per-byte costs to a minimum. We then measured the CPU utilization and the rate of operations for reads on the target server for this particular block size. The number of threads in this read experiment was limited to one so as to remove the effect of interrupt coalescing and get a clearer picture of per-operation costs.

Table 2 shows that the initiator workstation CPU overhead of a single operation for the hardware approaches is less than that of the software approaches. This indicates that the software driver overhead does not contribute to the performance bottleneck in the hardware approaches. Consequently, the high per-operation costs in the hardware approaches can be attributed to a mismatch in processing speeds between the host system and the hardware offload.

	Ops per second	**Initiator CPU util. (%)**	**Initiator CPU util. per op (%)**
Software	8267	38	0.0046
TOE	5959	22	0.0036
HBA	2580	7	0.0027

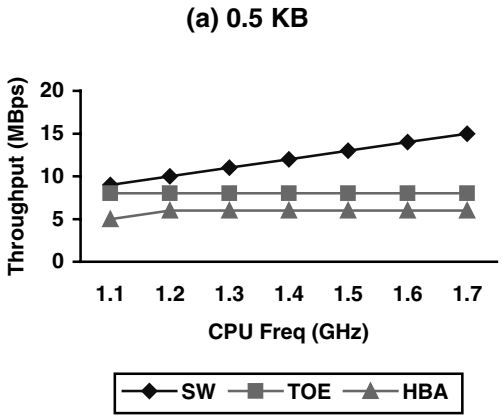
Table 2. The table shows the sensitivity of the rate of operations, the initiator workstation CPU utilization and the per-operation initiator workstation CPU utilization for a single-threaded 0.5 KB read test for each of the three approaches to IP storage.

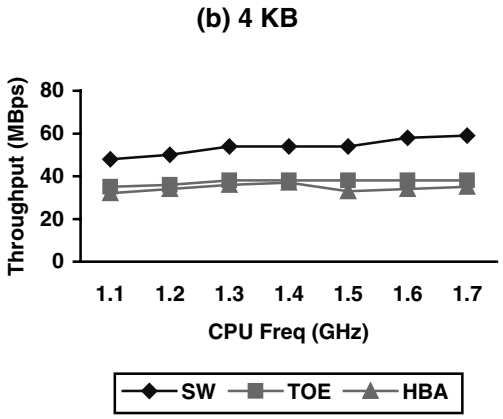
3.2.2. CPU speed sensitivity

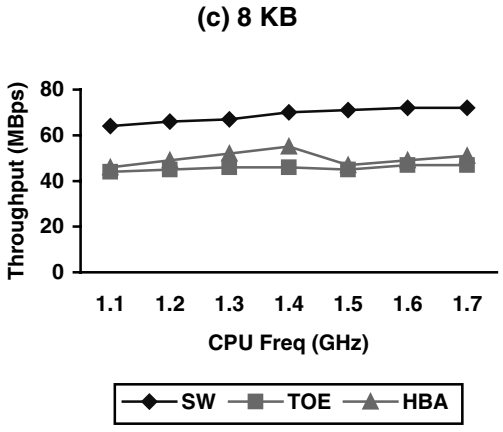
The second experiment in the micro-benchmark performance analysis measured the sensitivity of throughput to the CPU speed in the initiator workstation. In this experiment, the CPU FSB speed was varied from 92 MHz to 141 MHz with a fixed clock multiplier, resulting in an effective CPU speed variation 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.







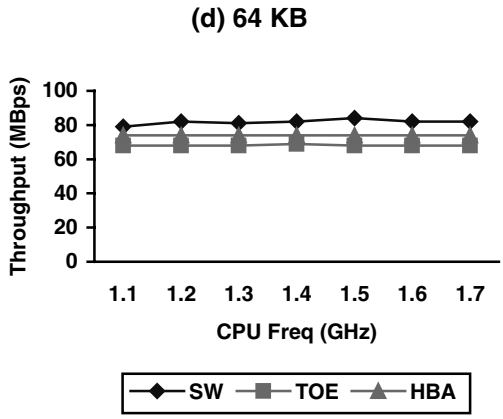


Figure 3. This figure shows the sensitivity of the throughput in relation to CPU speeds (denoted in terms of CPU frequency) in each of the three approaches for the block sizes of (a) 0.5 KB, (b) 4 KB, (c) 8 KB, and (d) 64 KB.

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, 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 frame-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.

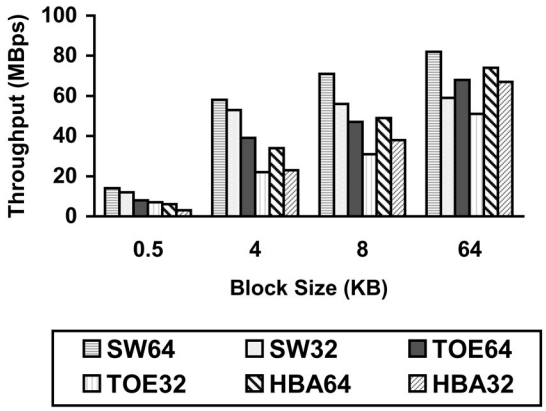


Figure 4. This figure shows the sensitivity of the throughput analysis with respect to PCI bus speeds. The legends SW64, TOE64 and HBA64 represent the throughput in relation to block size when the corresponding adapter is placed in a 64-bit 66 MHz PCI bus slot. Similarly, the legends SW32, TOE32 and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

The software approach shows no improvement in throughput when the CPU speed is varied with the adapter at the 32-bit 33 MHz PCI bus slot. 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 threshold 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 latency differential between the two PCI buses is marginal (< 5%) for all approaches when the block size used is small (0.5 KB, 4KB) but increases to as much as 50% when the block size is large (64 KB).

This experiment shows that the PCI bus speed is an important performance factor in overall performance, particularly the software approach.

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 threshold 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, Postmark and TPIE-Merge benchmarks were used to evaluate the software, TOE and HBA approaches so as to capture the primary application categories of databases and email.

4.2.1. TPC-C

The TPC Benchmark C or TPC-C [TPC97] is an online 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 suggestions from IBM Toronto Laboratory staff. For this analysis, we varied the number of database warehouses from 200 to 800. The database resides in a single file-system spanning 42 drives on the target server. An experimental run of the benchmark with the 42 drives attached as local disks completely saturated the initiator workstation CPU. This indicates that the benchmark is well tuned and not limited by disk drive performance in this particular configuration. The TPC-C benchmark is characterized by highly multi-threaded random page-sized (4 KB) I/Os with reads dominating writes.

The results (Figure 5(a)) show that the software approach has better results compared to the hardware ap-

proaches even for this CPU-intensive benchmark. Even though the software approach has a higher CPU utilization-to-throughput ratio for the page-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. The effective bandwidth of the three approaches is directly proportional to the transaction rate in the benchmark (tpmC).

Figure 5(b) shows that the software approach has a higher average CPU utilization during the benchmark. The average CPU utilization of all the approaches in this particular benchmark does not saturate the initiator CPU because of a limit on the maximum concurrency in the iSCSI protocol (60). Since there was no means to increase the limit, it was not possible to measure the relative performance with a saturated initiator CPU.

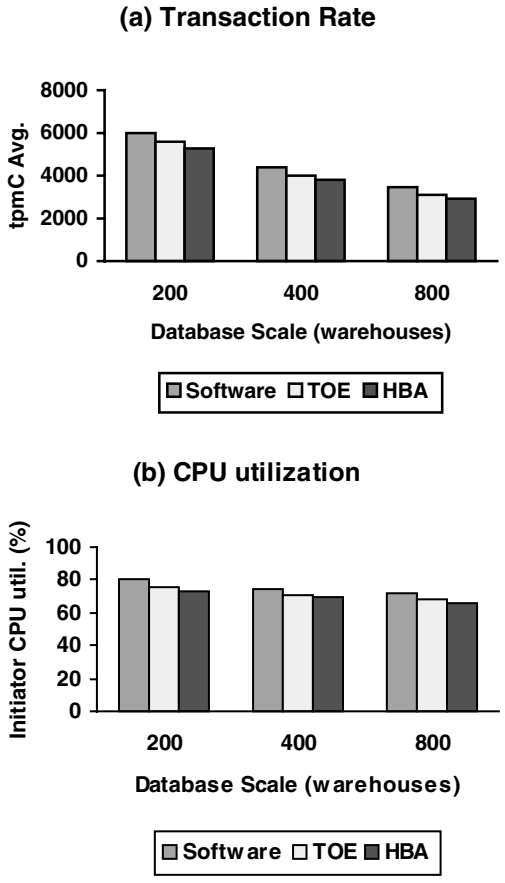


Figure 5. (a) This figure shows the average transactions completed per minute for the three approaches: Software, TOE and HBA when running the TPC-C benchmark for a database scale that spans 200 to 800 warehouses. (b) This figure shows the average CPU utilization in the initiator workstation for the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

4.2.2 PostMark

The PostMark benchmark [Katcher97] simulates workloads that capture the storage 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. However as the benchmark does not have any application processing, it cannot be considered a true email, news or web commerce application. 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.

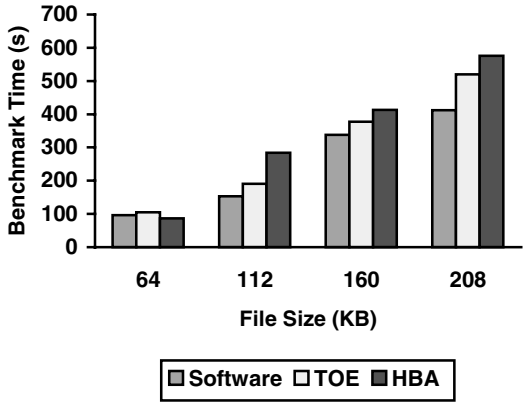


Figure 6. This figure shows the time to complete the PostMark benchmark for the three approaches: Software, TOE and HBA when the size for 150 files is varied from 64 KB to 208 KB in increments of 48 KB. The remaining PostMark parameters are the default.

4.2.3 TPIE-Merge

The TPIE benchmark [Vengroff96] combines raw sequential read/write I/O operations with application processing. The objective behind using this benchmark is to assess the I/O performance of the different approaches when the application processing has high CPU utilization. This benchmark will penalize any approach that has a high CPU utilization-to-throughput ratio and favors efficient storage transport protocols.

The TPIE toolkit provides scan, merge and sort routines that can be used to generate I/O and application processing scenarios. In this experiment, unsorted files were read, merged and a single sorted output file was generated. The underlying file system was NTFS. In the setup, we used 16 threads, each with 16 input files. There were 500,000 (four byte) records per input file.

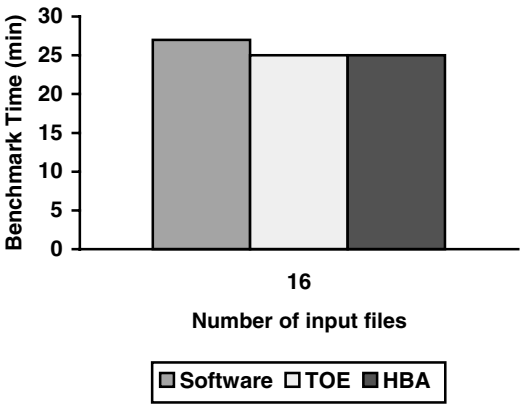


Figure 7. This figure shows the time to complete the TPIE-Merge benchmark for the three approaches: Software, TOE and HBA when the number of input merge files (each with 500,000 records) is varied from 8 to 16.

The results in Figure 7 show a small degradation (7%) in performance of the software approach with respect to the hardware approaches. The average initiator CPU utilization was close to 100% during the benchmark for all of the approaches. This particular benchmark uses the NTFS block size of 4 KB. Though the difference in the CPU utilization-to-throughput ratio for this particular block size for the hardware approaches is 33% better than that of the software approach, this difference applies to the non-application processing time. The measurements from the Perfmon tool indicate that the non-application processing time is 10-15% of the total CPU utilization and thus minimizes the impact of the reduced CPU utilization-to-throughput. The throughput requirements of this benchmark (< 10 MBps) did not approach the networking bandwidth available or the adapter processing limits. This shows that scientific applications that run on typical file systems with smaller block sizes may not get the expected performance improvements with the hardware approaches. However, a recent study of the Direct Access File System presents results from this TPIE merge benchmark that shows hardware support can improve performance by as much as a factor of two when the communication overhead is a large component of overall processing [Magoutis02].

5. Discussion

This section summarizes the performance characteristics of the software and hardware approaches based on the results obtained in Sections 3 and 4.

5.1. Software Approach

The software approach achieves the best numbers in terms of throughput and latency compared to the hardware approaches. The software approach has the disadvantage of high CPU utilization-to-throughput ratio for large block sizes as a result of high per-byte overheads. At the same time, the performance of the software approach scales with CPU speed and current CPU speeds are high enough to absorb the overheads for Gigabit Ethernet networks. In summary, the software approach is very competitive with the current generation of 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.

5.2. Hardware Approaches

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, because the critical processing has been offloaded to the adapter.

The above phenomenon has two 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. However, cost, heating, power and area constraints make this challenge harder than that in general purpose CPUs. Thus, it may be possible to decrease the parity between the processors on the host and the adapter by using superior technology, but then the resultant product may not have the right cost-performance ratio in a highly competitive storage market.

However, the hardware approaches may be well suited for high-end environments. Proponents of the hardware approaches argue that the hardware acceleration is more effective at 10 Gbps networks. The impact of communication overhead is so pronounced at such speeds that current CPUs will not be able to take advantage of the capacity of the network in the software approach. Another advantage for the hardware approaches is in high-end storage subsystems that have to support large numbers of initiators (32-1024). In the multi-initiator scenario, any reduction in the CPU utilization-to-throughput ratio can allow the storage subsystem to support a larger number of initiators. Furthermore, these high-end storage subsystems also have multiple adapters (32-64) that allow the adapters to have processing power comparable to that of the host subsystem.

The hardware approaches are also well suited for those applications where the communication overhead is a significant component of total CPU utilization. In such cases, using hardware approaches with their better CPU utilization-to-throughput ratio can significantly reduce the overhead. The benchmarks in Section 4 do not cover this category of applications.

Even so, this study is useful in designing the next generation of hardware adapters. 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 the superior processing power in general-purpose computing.

6. Related Work

Rod Van Meter et al [Hotz98, VanMeter98] 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 [Gibson97] 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 [Keng96] and Chase et al [Chase01] evaluated TCP/IP at near-Gigabit speeds to provide a breakdown of the various TCP/IP costs. Their 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 implementation used in these papers uses page re-mapping techniques to implement zero-copy functionality. However, page re-mapping has alignment issues and incurs virtual memory mapping overheads, particularly in SMP environments. Also, page re-mapping does not generalize to upper-layer protocols like iSCSI.

The effect of large frame sizes on TCP/IP performance has been also well studied before by Chase et al [Chase01]. The authors show that a large frame size reduces the number of interrupts and per-packet overheads to improve TCP/IP performance. As has been pointed out in Section 2.1, large frame sizes are currently not a viable alternative in Ethernet environments due to lack of standardization. Many network adapters provide interrupt suppression features to reduce interrupt overhead even for standard Ethernet frames. In particular, the TOE approach allows for interrupt coalescing even for standard Ethernet frames by offloading the TCP/IP stack.

Magoutis et al [Magoutis02] performed a thorough analysis of the key architectural elements of DAFS [DeBergalis02]. The study 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 for iSCSI.

Wee Teck Ng et al [Ng02] performed an exhaustive performance analysis of iSCSI over a 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 [Ang01] 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.

There is also interest in low-overhead IP storage networking for network file systems as well [Magoutis02]. This approach uses network adapters following the RDMA approach specified in Section 2.1. This study is based on a single-vendor interconnect technology whereas the conclusions in this study are applicable to network file systems using RDMA over IP networks [DeBergalis03].

7. Conclusions

This paper presents an analysis of the software, TOE and HBA approaches to build an IP storage area network. The software approach is based on the unmodified TCP/IP stacks that are part of a standard operating system distribution. For the two hardware-based approaches (TOE, HBA), we experimented with a range of adapters and chose a representative adapter for the current generation of each of the hardware approaches. The goal of the analysis is to answer the question whether current generation of hardware support necessarily helps performance.

The micro-benchmark analysis reveals that hardware support reduces the CPU utilization-to-throughput ratio for large block sizes. However, at the same time the current generation of hardware support can itself be a performance bottleneck that can hurt throughput and latency. This result is also supported by the macro-benchmark analysis that shows that the hardware approaches do not provide performance benefits in database, scientific and email benchmarks compared to the software approach, even though the hardware approaches have the potential to provide benefits in CPU-intensive applications. The analysis in this paper points out that a disparity in the processing power between the host and the adapter is the primary cause of the performance bottleneck in the current generation of the hardware approaches. This points to the need for an introspection of the current trend in hardware support so as to take advantage of the increased computing power in general-purpose processors.

Acknowledgments

We would like to thank our shepherd Jeff Chase as well as the anonymous reviewers for their valuable feedback regarding the quality of the draft submission. We would also like to thank John Hufferd, Nirmala Venkatramani and Matthew Lietzke for their invaluable assistance in setting up the experimental framework. We would finally like to thank Moidin Mohiuddin and Honesty Young for their active encouragement of our work.

References

[Ang01] 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, 2001.

[Ansi90] HIPPI, ANSI Standard X3T9.3/90-043, 1990.

[Bailey02] S. Bailey, "The Architecture of Direct Data Placement (DDP) and Remote Direct Memory Access (RDMA) On Internet Protocols", Internet Draft, IETF 2002, Work in Progress.

[Benner96] A. Benner, "Fibre Channel: Gigabit Communications and I/O For Computer Networks", McGraw-Hill, 1996.

[Chase01] J. Chase, A. Gallatin and K. Yocum, "End-System Optimizations for High-Speed TCP", IEEE Communications, 39:4, pp 68-74, 2001.

[Compaq97] Compaq, Intel, Microsoft, "Virtual Interface Architecture Specification", v1.0, December 1997.

[DeBergalis03] M. DeBergalis, P. Corbett, S. Kleiman, A. Lent, D. Noveck, T. Talpey, M. Wittle., "The Direct Access File System", FAST 2003.

[Gibson97] G. Gibson, D. Nagle, K. Amiri, J. Butler, F. Chang, E. Feinberg, H. Gobyoff, C. Lee, B. Ozceri, E. Riedel, D. Rochberg, J. Zelenka., "File Server Scaling with Network-attached Secure Disks", Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems, June 1997.

[Hotz98] S. Hotz, R. Van Meter, and G. Finn, "Internet Protocols for Network Attached Peripherals", 6th IEEE/NASA Conference on Mass Storage Systems and Technologies, 1998.

[Intel02] Intel Corp., "Iometer Performance Analysis Tool", http://www.intel.com/design/servers/devtools/iometer

[Katcher97] J. Katcher, "PostMark: A New File System Benchmark", Technical Report TR3022, Network Appliance Inc, 1997.

[Keng96] Hsiao Keng, and J. Chu, "Zero-copy TCP in Solaris", USENIX 1996 Annual Technical Conference, pp 253-264, 1996.

[Kent98] S. Kent and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, 1998.

[Kronenberg86] N. Kronenberg, H. Levy and W. Stecker. "Vax-Clusters: A Loosely Coupled Distributed System", ACM Transactions on Computer Systems, 4:2, pp 130-146, 1986.

[Magoutis02] K. Magoutis, S. Addetia, A. Federova, M. Seltzer, J. Chase, A. Gallatin, R. Kisley, R. Wickremsinghe, E. Gab-

ber, "Structure and Performance of the Direct Access File System", USENIX Technical Conference, pp 1-14, 2002.

[Magoutis03] K. Magoutis, S. Addetia, A. Federova, M. Seltzer, “Making the Most out of Direct-Access Network Attached Storage”, FAST 2003.

[Ng02] Wee Teck Ng, H. Sun, B. Hillyer, E. Shriver, E. Gabber, B. Ozden, "Obtaining High Performance for Storage Outsourcing", FAST, pp 145-158, 2002.

[Rajagopal02] M. Rajagopal, E.Rodriquez, R. Weber., “Fibre Channel Over TCP/IP”, Internet Draft, IETF, 2002, Work in Progress.

[Sarkar02] P. Sarkar and K. Voruganti, “IP Storage: The Challenge Ahead”, 19th IEEE Symposium on Mass Storage Systems, pp 35-42, 2002.

[Satran02] J. Satran, K. Meth, C. Mallikarjun, C. Sapuntzakis, E. Zeidner, "iSCSI", Internet Draft, IETF, 2002, Work in Progress.

[Shanley02] T. Shanley and J. Winkley, “Infiniband Network Architecture”, Addison-Wesley, 2002.

[TPC97] TPC Benchmark C Standard Revision 3.3.2, Transaction Processing Performance Council. 1997.

[VanMeter98] R. Van Meter, G. Finn, and S. Hotz, "VISA: Netstation's Virtual Internet SCSI Adapter", ASPLOS 8, pp 71-80, 1998.

[Vengroff96] D. E. Vengroff and J. S. Vitter, “I/O-Efficient Scientific Computation using TPIE”, IEEE Conference on Mass Storage Systems and Technologies, pp 553-570, 1996.

[Voruganti01] K. Voruganti and P. Sarkar, “An Analysis of Three Gigabit Storage Networking Protocols”, pp 259-265, IPCCC 2001.