appeared in ASPLOS 8, 10/98

VISA: Netstation's Virtual Internet SCSI Adapter *

Rodney Van Meter‡, Gregory G. Finn and Steve Hotz
Information Sciences Institute
University of Southern California
Marina del Rey, CA 90292
{rdv,finn,hotz}@ISI.Edu

Abstract

In this paper we describe the implementation of VISA, our Virtual Internet SCSI Adapter. VISA was built to evaluate the performance impact on the host operating system of using IP to communicate with peripherals, especially storage devices. We have built and benchmarked file systems on VISA-attached emulated disk drives using UDP/IP. By using IP, we expect to take advantage of its scaling characteristics and support for heterogeneous media to build large, long-lived systems. Detailed file system and network CPU utilization and performance data indicate that it is possible for UDP/IP to reach more than 80% of SCSI's maximum throughput without the use of network coprocessors. We conclude that IP is a viable alternative to special-purpose storage network protocols, and presents numerous advantages.

1 Introduction

Storage system architectures are increasingly network-oriented, exploiting the ubiquity of networks to replace the direct host channel. Peripherals attached directly to networks are called network-attached peripherals (NAPs), or more specifically, network-attached storage devices (NASDs).

We have proposed that NAPs use the Internet protocol suite; in this paper we provide data on a sample implementation which supports the claim that the Internet Protocol (IP) can perform acceptably for NAPs.

In the experiments presented in this paper, we have used the User Datagram Protocol (UDP) as a transport protocol to send and receive SCSI commands and data to emulated network-attached disk drives. We have achieved data rates of 70+ megabits per second (Mbps) for read and write through the file system over Myrinet before becoming CPU limited, and known optimizations could be expected to raise that to approximately 95 for write and 110 for read. TCP is predicted to be 10-25% slower than UDP. Fast ethernet is only 10% slower than Myrinet, with the difference caused primarily by ethernet's smaller MTU raising the amount of per-packet processing which must be done. Our analysis predicts that SCSI performance on the same hardware would become CPU-bound at 110 Mbps write, 133 Mbps read. Our conclusion is that more than 80% of maximum SCSI performance can be achieved using IP.

By demonstrating comparable performance, we open the door to the adoption of the Internet protocol suite by operating system vendors and disk drive manufacturers as an alternative to the numerous storage networks now being developed. This adoption would improve the sharability and scalability of storage systems with respect to numbers of network-attached devices and client systems, while substantially reducing legacy problems as technology advances, shortening development time and leveraging networking technology. In addition, TCP/IP enables such new wide-area uses as remote backup and mirroring of devices across the Internet.

This work was done in the course of the Netstation project, which concentrates on operating systems, network protocols, hardware mechanisms, and security and sharing models for network-attached peripherals. Some of the goals of the project are to demonstrate (1) that IP can provide acceptable performance in a host operating system when used to access peripherals, (2) that IP can be implemented efficiently inside network-attached peripherals, and (3) that our derived virtual device model enables efficient, secure use of NAPs. This paper addresses only the first point; the other two are presented in separate papers [HVF98, VHF96].

The paper begins with a brief description of Netstation, the network-as-backplane system architecture of which VISA forms a part. Section 3 describes the principles of networking as applied to network-attached peripherals, with special emphasis on the problems of scalability and host OS adaptation. We then describe related work, followed by the VISA architecture, performance, and possible performance improvements. Finally, we present our conclusions.

2 Netstation

Netstation is a heterogeneous distributed system composed of processor nodes and network-attached peripherals [Fin91, FM94]. The peripherals are attached to a shared 640 Mbps Myrinet network or to a 100 Mbps ethernet, as in Figure 1.

*This research was sponsored by the Defense Advanced Research Projects Agency under Contract No. DABT63-93-C-0062. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policies, either expressed or implied, of ARPA, the U.S. Government, or any person or agency connected with them.

[0]Author's current address: Quantum Corp., Milpitas, CA, Rodney.VanMeter@qntm.com.

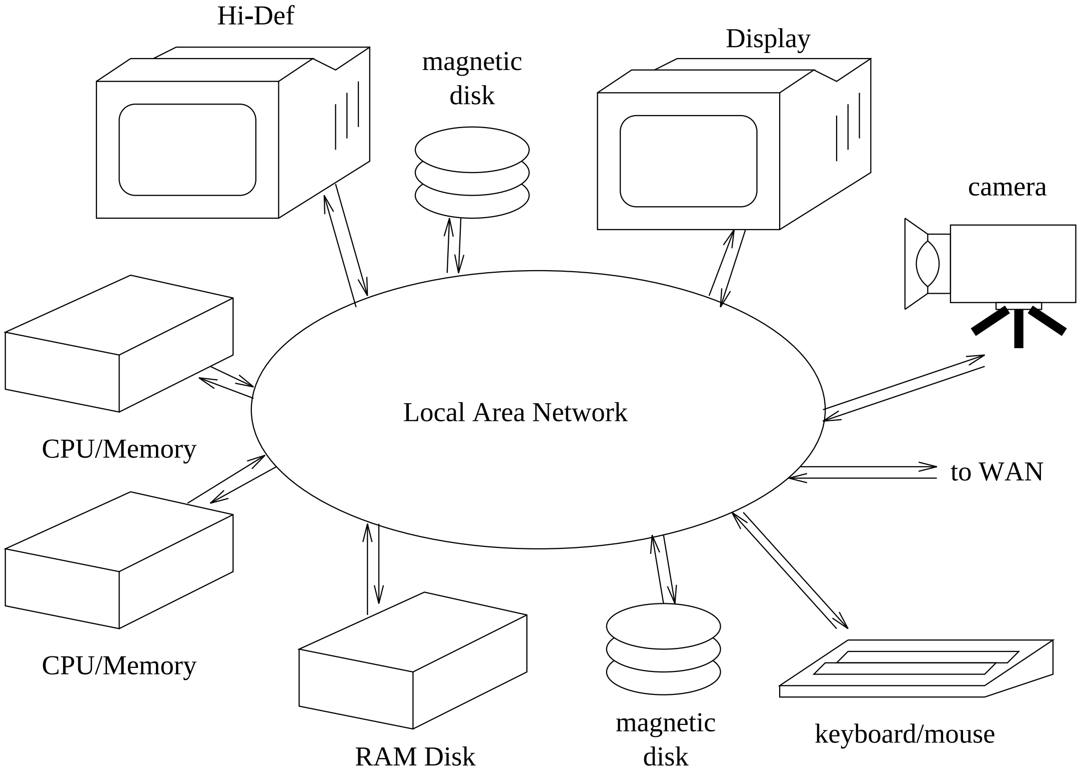

Figure 1: A Netstation network

The display and camera NAPs have been built; the other NAPs are currently emulated. The CPU nodes are Sparc 20/71 workstations running SunOS 4.1.3; the ideal Netstation CPU node would have only a CPU, memory and a network interface.

Connecting peripherals directly to the network allows sharing of resources and improves system configuration flexibility. Network clients can access peripherals without the intervention of a server.

Because the devices are attached to an open network with both trusted and untrusted nodes on the net, security at the NAPs is critical. We have developed a model we refer to as the *derived virtual device*, or DVD [VHF96]. DVDs provide a protected execution context at the device, allowing direct use of the devices by untrusted clients, such as user applications. The owner of a device *defines* the security policy, downloads a description to the NAP, and the NAP *enforces* the policy. This allows the owner to define a set of resources and operations allowed. Thus, a camera can be granted write access to only a specific region of a frame buffer, or a user application can be given read-only access to a DVD which represents a disk-based file or disk partition.

The only network-attached peripheral used during the experiments described in this paper is *IPdisk*, our emulated network-attached disk drive. IPdisk will be described in more detail in section 5.1.

VISA, our Virtual Internet SCSI Adapter, is the primary topic of this paper. It is the OS mechanism for supporting access to storage peripherals via the network.

3 Networking for NAPs

Netstation's shift to network-attached peripherals has a profound impact on the overall system architecture. In this section, we explore the technological and architectural motivations to make the change and its effect on host operating systems. We discuss the problems that must be solved to build large systems on heterogeneous networks, and show how choosing the TCP/IP suite solves these problems. Finally, we briefly discuss the appropriate high-level interface NAPs should present.

Developers of storage network technologies have started with different goals and assumptions about important issues such as number and type of devices and hosts to be interconnected, physical distance, cost and bandwidth. The result has been a proliferation of technologies developed primarily for NAPs, including 1394 (Firewire), Fibre Channel fabrics and Arbitrated Loop, HiPPI, and Serial Storage Architecture (SSA), as well as vendor-specific networks. Most of these have included development of complementary new physical, link, network and transport layers [Van96].

3.1 Motivation

Netstation uses network-attached peripherals to better share peripherals and take advantage of the relative technological trends of buses, networks, and peripheral processors.

The architectural reasons to shift from host adapter-attached devices to network-attached are better sharing of devices and reduction of the server's workload [RG96]. By allowing clients to directly access the devices, the server is no longer in the data path, reducing latency and demands on its buses, memory and processors. Devices can also communicate directly with each other without sending data across a single shared system bus.

Buses do not scale well. They do not scale in distance; shortening a bus can raise data rates, forcing a direct trade-off in design. They do not scale in the number of devices interconnected, generally having a firm upper bound below twenty. They do not scale in aggregate bandwidth with the number of devices, as bandwidth is shared among the connected devices, and, due to increased capacitance, available bandwidth may actually decrease as devices are added.

Networks, especially serial optical networks, are improving rapidly in speed, typically scale well to large numbers of nodes, and can stretch over significant distances. Such networks are pushing into the gigabit per second range, a speed comparable to popular low-end buses such as PCI. Multicomputers and clustered systems have interconnected processors via networks for many years; I/O systems are now adopting networks to receive the same benefits.

3.2 Host I/O System and Networking Interaction

There are several characteristics which differentiate I/O for file systems from interactive application-level network I/O such as HTTP or telnet. Netstation uses some of these to improve the operating system efficiency, but further improvements are possible.

File transfers are generally large, page-aligned multiples of the system's page size (in our case, 4KB). On write, they are already pinned into memory and mapped into the kernel address space by the file system before they are handed to the bus or network subsystem. This may reduce the mapping and copying operations typically used to move data from user buffers to kernel buffers. On read, pages have already been selected, so the memory destination of incoming data is known in advance.

The file system cares only about the completion of the entire I/O operation. Data buffers will not be released to the application or virtual memory (VM) system until the read or write is finished. Thus, it is unnecessary to send partial completion status up the protocol stack until the entire transfer has either completed or failed. Typically, modern SCSI host bus adapters post only one interrupt to the host on completion or error, with requests that may be megabytes long. The host OS maintains a simple timer for the completion of the entire I/O. Sophisticated cards maintain one context per target device on the bus, and are capable of multiplexing among them. Current Fibre Channel interfaces are approaching a similarly autonomous level of operation, implementing the FC transport layer in the network interface card.

3.3 Scaling Problems Faced by I/O Networks

Netstation adopted the TCP/IP suite to leverage off the Internet community's experience with scale and heterogeneity. We believe that this experience makes the TCP/IP suite an increasingly appropriate choice and LAN-specific solutions increasingly inappropriate as more clients and servers are attached to more and larger heterogeneous storage networks.

The I/O network technologies deployed to date appear to offer only limited scalability, due to weakness in one or more areas. Media bridging, heterogeneity along many axes, security, latency tolerance, and congestion and flow control are several important areas that must be addressed.

Media bridging or routing becomes important as more hosts spread over more hops, and more legacy systems (both hosts and nets) must be accommodated. In many environments, support for multiple networks and complex topologies, of the same or different types, is likely to be critical. This brings up issues of formatting, addressing, and especially underlying protocol interoperability.

3.4 Networking Protocols

In the Netstation project, we have chosen to work with UDP/IP and TCP/IP, reasoning that the more general architecture provides features that will be critical to network-attached peripherals in the long run. We expect that the performance shortcomings will be resolved by the networking research and development community.

Media-specific development of network and transport layers leaves systems with potential interoperability and legacy problems. While it is possible, for example, to route Fibre Channel frames across HiPPI networks, creating such exchange protocols for every possible pair of network technologies results in $O(N^2)$ protocols. If other networks cannot provide in-order delivery and support the flow control mechanisms Fibre Channel expects (either for class 2 or class 3 service), interoperability will be difficult. Additions of new clients or devices to the network are limited to network technologies which interoperate, regardless of external forces such as economic constraints, availability of new network technologies, etc.

Most developers and users of such networks (and devices for them) have discarded the possibility of adapting the TCP/IP protocol suite for communicating with NAPs [1]. Reasons cited include inappropriateness for the type of traffic, TCP's wide-area tuning, the complexity of TCP, and especially performance of the entire TCP/IP suite in both bandwidth and latency [G$^+$97]. We argue that these concerns are misplaced for several reasons.

Complexity is inherent as systems become larger. TCP/IP's complexity was not created arbitrarily, but developed in response to particular external stimuli, solving the problems presented above. Fibre Channel and other network transport protocols will have to face similar decisions as they attempt to address the same problems.

Earlier analyses of TCP/IP performance may have been based on poorly tuned or now outdated TCP/IP implementations. The Fibre Channel standardization effort, for example, was begun in 1988; much of the research on efficient networking implementations cited here has been conducted in the last decade. Older implementations often impose penalties such as extra data copies, separate checksumming passes, and inefficient demultiplexing of incoming data. Some of the improvements will be discussed in section 7.

3.5 Device Command Model

Once the architectural decision has been made to connect the peripheral to a network, the most important choice is the command request interface. Disk drives traditionally use a block-level interface, while network file servers use a file model appropriate to the needs of a particular set of clients. In the course of rearchitecting distributed storage systems, other choices are possible, such as an "object" model in which the disk is responsible for layout decisions but not file naming and security [G$^+$96].

We have chosen a block interface, enhanced for security with derived virtual devices, rather than a file or file-like

[1] Most of these net technologies can carry IP traffic for host-to-host communication, however.

model. This allows the simplest reuse of existing file system and operating system technology (data layout, partitioning, inode manipulation, the `sd` SCSI disk driver, `fsck` and `format`, virtual memory interaction, etc.). This also allows the broadest range of uses of the device, providing clients with the choice to build a Fast File System (FFS) or log-structured file systems, non-Unix file systems, network RAID, and other uses of raw partitions such as swap space, hierarchical storage management cache and databases.

4 Related Work

The projects most similar to Netstation are MIT's ViewStation [HAI$^+$95] and Cambridge's Desk Area Network (DAN) [BHMP95]. Both use ATM networks as their device interconnect, and establish a physical boundary to the system for security purposes, while Netstation uses protocol-based security. They have defined a useful taxonomy of dumb, supervised and smart devices.

Network-attached storage is an area of much current research and development. Fibre Channel disk drives, new distributed file server architectures, and custom development of storage networks all play a part.

A TCP/IP RAID controller was developed at Lawrence Livermore National Labs; it is the first TCP disk device of which we are aware [WM95].

Mainframe channels have been extended to run over phone lines and even WANs for remote device mirroring; products from CNT, EMC and others perform such functions. However, they typically use media-specific protocols.

Fibre Channel-attached disk drives utilize a simple SCSI block interface with no security on a moderately complex network. As described above, this raises concerns about security, scalability, legacy systems and interoperability with other types of networks.

The CMU Parallel Data Lab's Network Attached Secure Disk (NASD) project divides NFS-like functionality between a file manager and the disk drives themselves, so that not only read and write commands but also attribute set and get are executed at the drive [G$^+$96, RG96, G$^+$97]. The file manager is responsible primarily for verifying credentials and establishing access tokens.

Soltis' Global File System (GFS) uses Fibre Channel disk drives modified to support a lock primitive [SRO96, Sol97]. This provides simple, efficient distributed locking. The drive itself attaches no meaning to the locks; by convention among the clients, they are used to lock inodes and other data structures.

The Petal distributed disk system provides a virtual disk model on which the Frangipani distributed file system is built [LT96, TML97]. Due to the virtualization of storage space, their model moves the actual backing store allocation to the disk, though the interface between Petal and Frangipani is a block-level one.

Various system vendors have developed their own networks on which distributed device sharing takes place. VAXclusters [KLS86] and ServerNet [HG97] are two examples which use message passing between devices and hosts; the new SGI Origin series uses custom hardware to implement a shared address space on a switched network which includes many processors and I/O nodes [LL97].

5 VISA Architecture

VISA, Netstation's Virtual Internet SCSI Adapter, is an operating system module which makes Internet-attached pe-

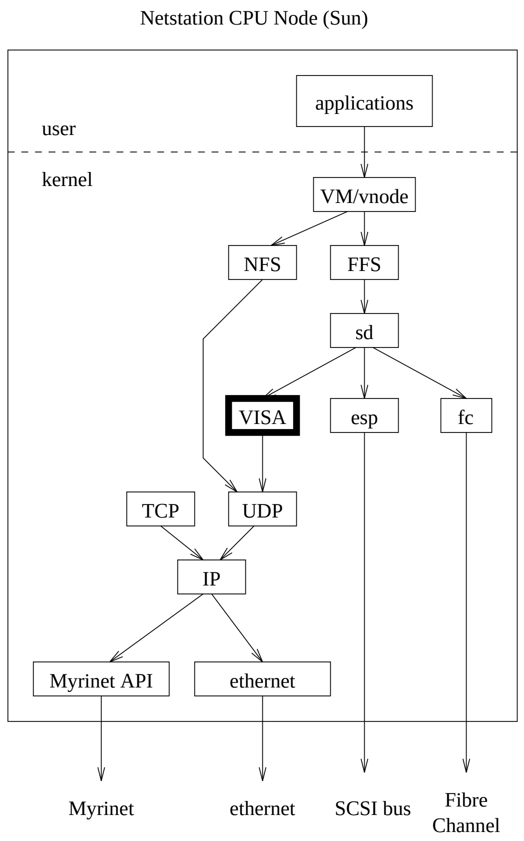

Figure 2: Netstation MPU node OS components. sd = SCSI disk device driver, esp = SCSI bus adapter driver, VISA = Virtual Internet SCSI Adapter, VM = virtual memory, FFS = Fast File System

ripherals appear as if they were attached to a local SCSI bus.

VISA implements an instantiation of the `scsi_transport` structure. SunOS uses a layered device driver model for SCSI devices. Device drivers which present the standard block or raw interfaces found in `/dev` are specific to a type of peripheral, such as `sd` for SCSI disks and `st` for SCSI tapes. For SCSI devices, these drivers in turn depend on lower-level services to communicate with the specific type of host adapter present. The `scsi_transport` structure consists primarily of pointers to nine high-level functions that implement a well-defined interface for sending commands to a SCSI device and managing the memory for returned data.

In figure 2, the `sd` SCSI disk driver is shown transmitting requests to VISA and to `esp`. Esp is the standard SCSI adapter type present on Sun SPARC workstations. Additional implementations exist for third-party SCSI adapters. It is at this layer that support for networked SCSI, such as Fibre Channel, is installed. Because we are using standard SCSI commands, no additional packaging or formatting, such as XDR, is required.

VISA transmits packets by calling UDP, which uses IP to send packets over any supported network medium. We have used both 100bT ethernet and Myrinet in these experiments.

We use UDP as the transport-layer protocol because we expected UDP to be faster than TCP as well as easier to work with inside the kernel. On top of UDP we found it necessary to build a simple reliability layer. While the network itself is highly reliable, SunOS provides only limited buffering in the sockets, and packets are discarded when the buffer is full. Our reliability layer works with a fixed-size window and assumes in-order delivery; on every 48KB transmitted, the sender pauses for an ACK. On receipt of out-of-order data, a NAK is sent and the sender rolls back. A timeout of one second is currently implemented only at the device (IPdisk). The host depends on either IPdisk to discover network glitches, or the higher-level I/O request to timeout and be restarted within the sd device driver. This is purely a convenience; a production-quality implementation would have timers at both ends.

We use 8 kilobyte data payloads, plus a small header including sequence numbers, on top of the UDP packet. Over Myrinet, this is a single packet. Over ethernet, this forces IP fragmentation in order to work within the 1500 byte MTU.

As is common with UDP, checksumming is turned off. The data integrity is still protected by the link layer checksum. Both hardware and software mechanisms for doing the TCP/UDP checksum with zero CPU cost are known. It can be done during data copy, DMA or transmission [PP93, FHV96].

We use one kernel pseudo-process per device attached to the VISA virtual bus. Much like NFS `biods`, these are responsible for the communication with the device, and are necessary because the SunOS kernel is not multithreaded.

5.1 IPdisk

IPdisk is our Netstation emulated disk drive. It runs as a user process on another Sun, emulating the SCSI block device command set running over UDP or TCP. It can be configured to use RAM to emulate disk storage, or to use a regular file, or access an actual raw SCSI disk. For the experiments described in this paper, IPdisk was used with a 32 MB RAM buffer, the fastest form of backing store, in order to stress the host operating system. In this mode, the physical characteristics (rotational and seek latency, zone-variable transfer rates, etc.) of a disk are not emulated, so the host CPU becomes the bottleneck.

IPdisk supports our derived virtual device model. This allows the owner of the device to download small programs which act as filters on the SCSI RPCs before those RPCs are actually executed. This feature is primarily expected to be used for sharing devices among multiple hosts and executing third-party (direct device-to-device) copies. Because the purpose of these experiments is to saturate the host CPU, DVDs, which only affect execution time at the device, are not enabled for the experiments presented here.

6 Performance

We have run experiments to measure the performance of regular Berkeley fast file systems built on top of VISA-attached disks. The read and write throughput for sequential accesses are detailed in the following subsections, then in the next section we discuss potential improvements to this performance.

We measured the CPU utilization for file systems on VISA-attached disks and on a directly-attached fast, narrow (10 MB/sec.) SCSI bus. This provides a very direct comparison of the actual impact of different data transport mechanisms in a complete file system environment.

Our test configurations utilize a 75 MHz Sparc 20/71 with 64 MB of RAM and an 800 Mbps Sbus as the client. The CPU has 20KB/16KB I/D on-chip caches and 1MB of external cache. The STREAM benchmark reports memory copy bandwidth of 61.7 MB/s [McC98]. Identical 20/71s running IPdisk emulate the network-attached disk drives. Our absolute performance numbers are low by today's standards due to the age of the systems used, but the relative numbers and conclusions remain valid.

The biggest performance problem we encountered is excess calls to `bcopy`. On send, the Myrinet device driver copies the data from the `mbuf` chain to a special send buffer allocated and maintained by the device driver itself in main memory. From that buffer, the data is then DMAed to the buffer RAM on the network interface card, from where it is transmitted onto the network. The nominal reason for the copy is the expense and complexity of establishing DMA mappings for arbitrary memory addresses, as well as concerns about data alignment. However, when we are sending data from complete VM pages, as in VISA or NFS, the pages are already pinned in memory, properly aligned, and large enough to more than amortize the cost of creating the mapping, making it the proper choice.

The benchmark we are using is a modified version of Bonnie which reports additional CPU utilization and can loop on individual subtests to reduce the overhead of process creation. Bonnie, written by Tim Bray and available on the Internet, performs several tests to measure I/O operation overhead and throughput; we have concentrated here on throughput. 25MB files were written or read repeatedly until the total amount of data was one gigabyte.

6.1 Write Performance

Table 1 lists the measured and predicted performance of file writes on VISA-attached disk drives. Our write throughput is CPU-limited at 72 Mbps when using Myrinet, with kernel profiling disabled. Table 2 lists the measured write throughputs of other configurations and subsystems for comparison. Writing just to the file system buffer cache (tech-

VISA configuration	thru
Myrinet UDP	72
100bT UDP	64
without extra `bcopy` (est.)	89
with faster UDP (est.)	95
TCP (est.)	58
with faster TCP (est.)	82

Table 1: VISA Write Rates. Throughput in Mbps.

nically, virtual memory pages, under SunOS) achieves approximately 115 Mbps (direct measurement of this number is difficult). We measured the write throughput of a file system on a physical SCSI disk (a Seagate ST31200W) as only 22 Mbps, but extrapolation from the CPU consumption to CPU saturation suggests a throughput of 110 Mbps can be achieved (assuming the presence of adequately fast disks and SCSI buses). Our current VISA throughput, therefore, is only 65% of the estimated SCSI throughput.

The CPU utilization figures in tables 3 and 4 were measured using a kernel compiled with `gprof` profiling enabled. The kernel subsystem figures are calculated by assigning each kernel function profiled to one of several groups. The raw data and tables of which functions were assigned to which groups are available for verification. These numbers are used to estimate the potential performance gains for different optimizations.

It can be seen from these CPU utilization numbers that, except for bcopy, the networking CPU cost is only about half the file system CPU cost and comparable to the "miscellaneous" kernel functions.

As described above, when using Myrinet under SunOS, write carries a penalty of an unnecessary data copy, and the 100bT ethernet driver appears to have a similar problem. We estimate the performance possible if this copy can be eliminated at approximately 89 Mbps on Myrinet, or approximately 81% of the maximum estimated SCSI throughput, as follows: $147.5 - 16.4 = 131.1$ CPU seconds for measured VISA UDP after removing the `gprof` overhead; $131.1 - 24.7 = 106.4$ CPU seconds after removing the `bcopy`. Multiplying by the measured bandwidth, $72 * 131/106 = 89$ Mbps for our estimate of the `bcopy`-less CPU saturation point. The other numbers are estimated similarly.

For reasons explained in section 7.1, we have also predicted the performance using a UDP which consumes 40% fewer CPU cycles in conjunction with a bcopy-less write.

Application benchmarks show that sending 1GB using TCP consumed approximately 30 seconds more CPU time on send than UDP. We added this 30 seconds to the 131 seconds (after removing `gprof` overhead) required to write 1GB on a file system built on a VISA-attached disk. From this we estimate the TCP throughput to be 58 Mbps. Approximately one-third of the increase in CPU time is in calculating the TCP checksum, the rest is sequence management with acknowledgements and timer processing. The "faster TCP" estimate assumes elimination of checksum and bcopy, but no reduction in the other overhead.

We recorded the distribution of command sizes on one experimental run of Bonnie writing one gigabyte. A total of 9,409 write commands were sent to the device for a total of 2,064,821 blocks (1.008GB), counting `format`, `newfs`, `mount`, and 1GB of file write with metadata updates.

Although the application always writes in chunks of 8KB,

tuned NFS	60
app. to FS buffer cache (VM pages)	~ 115
host memory to NIC memory via Sbus	~ 300
SCSI through FS @19% CPU	22
SCSI @100% CPU (est.)	110
UDP blast	172
TCP blast	104

Table 2: Write Rate Comparison (Mbps)

Module	time (secs)	% CPU
user:		
benchmark application	31.7	21.2%
kernel:		
file system code	37.9	25.4%
Myricom driver `bcopy`	24.7	16.6%
other networking code	18.6	12.5%
gprof profiling code	16.4	11%
miscellaneous kernel	18.2	12.2%
total	147.5	

Table 3: CPU Utilization to Write 1GB on UDP VISA

Module	time (secs)	% CPU
user:		
benchmark application	35.0	35.0%
kernel:		
file system code	31.9	31.9%
`esp` SCSI adapter code	2.6	2.6%
gprof profiling code	13.8	13.8%
miscellaneous kernel	16.9	16.9%
total	100.2	

Table 4: CPU Utilization to Write 1GB on an `esp` SCSI Bus

VISA configuration	thru
Myrinet UDP @85% CPU	60
Myrinet UDP @100%CPU (est.)	71
100bT UDP @82% CPU	53
100bT UDP @100% CPU (est.)	64
without `bcopy` (est.)	99
with faster UDP (est.)	109
TCP (est.)	59
faster TCP (est.)	82

Table 5: VISA Read Rates. Throughput in Mbps.

the operating system's write-behind mechanism, called `kluster`, coalesces the smaller individual writes into larger ones when it can. As a result, more than 85% of the total data written is in commands larger than 100KB. This reduces the collective I/O operation overhead and wastes fewer disk revolutions than smaller operations.

6.2 Read Performance

For the read tests, a 25MB file is read sequentially, the VM/file buffer cache is flushed using a SunOS `ioctl`, and it is reread. This process is repeated until 1GB of data has been read. The file system is clearly effectively detecting sequential activity and reading ahead of the process, with the result that almost all operations are 56KB long. This slightly exceeds our protocol window, creating some idle time and reducing our throughput.

Table 5 lists the measured and estimated throughput for several VISA configurations, and table 6 lists the measured and estimated throughput for related subsystems. Our VISA read throughput is 60 Mbps at 85% CPU utilization, from which we infer the CPU will saturate at 71 Mbps, essentially the same as our write rate. This is a lower percentage, 53%, of the SCSI potential throughput than for write.

Tables 7 and 8 show the measured CPU utilization of different kernel subsystems during read. The network overhead is higher, while the file system overhead is lower for read than for write.

Again, however, the primary culprit is `bcopy`; if it can be eliminated VISA UDP throughput is estimated to reach 74% of SCSI bus potential. Adopting the optimizations from section 7.1, we expect to reach 82% of SCSI bus potential, roughly equivalent to our predicted best write performance.

For TCP, the estimates are derived by adding the measured 24 second overhead for an application to receive 1GB over TCP instead of UDP. As with write, the checksum is the largest addition; we have estimated the performance possible by eliminating checksum, bcopy, and various `select` related functions which a kernel VISA TCP implementation would not use and listed this in table 5 as "faster TCP".

The per-packet costs are a significant concern, but we see only a 10% throughput drop from 8KB Myrinet packets to 1500 byte ethernet packets.

7 Potential Performance Improvements

Basic functionality at a reasonable data rate has been demonstrated. Nevertheless, it is clear from these performance numbers that host operating systems need substantial tuning to be efficient for the types of transfers common in file systems. Many of the necessary optimizations are understood, if not yet widely deployed; one research project

tuned NFS	60
from FS buffer cache (VM pages, no device access)	200
NIC memory to host memory via Sbus	$\sim$ 400
SCSI through FS @19% CPU	25
est. SCSI @100% CPU	133
UDP blast	150
TCP blast	104

Table 6: Read Rate Comparison (Mbps)

Module	time (secs)	% CPU
user:		
benchmark application	15.0	12%
kernel:		
file system code	34.4	27.5%
network `bcopys`	37.6	30.0%
other networking code	20.6	14.6%
gprof profiling code	16.1	11%
miscellaneous kernel	21.2	17.0%
total	144.9	

Table 7: CPU Utilization to Read 1GB on UDP VISA

Module	time (secs)	% CPU
user:		
benchmark application	26.6	33%
kernel:		
file system code	26.1	32.3%
`esp` SCSI adapter code	5.1	6.3%
gprof profiling code	11.1	13.7%
miscellaneous kernel	11.7	14.5%
total	80.6	

Table 8: CPU Utilization to Read 1GB on an `esp` SCSI Bus

demonstrated more than twice our TCP performance on similar hardware with an experimental version of Solaris and hardware support for checksumming [Chu96].

7.1 Software

VISA is implemented in SunOS 4.1.3 for Sun4m architectures. When the Netstation work was begun, Myrinet was the network of choice, and SunOS was the primary platform on which Myrinet was supported. Although SunOS is now considered an older operating system, it is well-studied and mature. It is representative of many more modern BSD-derived operating systems. However, many recent innovations in TCP/IP implementations are not present.

Several research projects have implemented fast demultiplexing packet filters which simplify the process of determining the data destination address on packet reception [DWB+93, YBMM94, EK96, Kay95]. Adoption of such an approach should eliminate the associated `bcopy` and complement other per-packet processing improvements.

Partridge demonstrated that the per-packet CPU cost for UDP processing can be reduced by 26-40% relative to the stock 4.3BSD/SunOS 4.1.1 implementation on a SPARC [PP93]. This should translate to 10-25% reduction in the total CPU use in file systems, giving corresponding throughput improvements. Similar improvements have been adopted into the 4.4BSD code.

7.2 Hardware

In this paper, we have concentrated on evaluating the cost of doing IP in the host operating system software. It is also possible to move some or all of the network protocol stack into the network interface card itself. Based on our data, this has the potential to eliminate the 14% (UDP write) to 38% (TCP read) performance overhead of doing CPU-based network protocol processing. Adding hardware to achieve such modest performance gains may be the correct architectural approach only under certain conditions.

Minimally, the NIC should support TCP checksumming on both transmit and receive and fast demultiplexing on receive. This can significantly reduce the data touching workload of the CPU for a modest increase in hardware complexity.

A complete NIC-side transport implementation would accept for transmission an undifferentiated chunk of memory and perform all packetization as well as managing transmission windows and processing and generating ACKs. This would leave the host as unaware of the details of TCP/IP as it currently is of the handshakes on a SCSI bus. Matters has shown that this is possible in an I2O-compliant NIC [Mat98].

Myer and Sutherland recognized thirty years ago that the boundary between work done by the CPU and by peripheral processors moves in cycles over time; they referred to this as the "wheel of reincarnation" [MS68]. Over the last decade, numerous hardware experiments were conducted and suggestions were made for putting more functionality into the network adapter [Kan88, CSSZ90, Che87, Sid91, DWB+93, Dav91, TS91, BJM+96, SWR91, BPP91, DPD91, TP96]. Most of the more radical features have not been accepted into mainstream adapters. Some of these approaches have not been adopted into the mainstream because they are specific to Open Systems Interconnection (OSI) or other protocols. Others have failed simply because the improvement in general-purpose CPUs has outstripped the improvement in the processors on outboard NICs when the researchers have fallen behind the performance development curve.

8 Conclusions

Network-based storage architectures share peripherals among many clients and remove the server from the data path by attaching the peripherals directly to the network. Peripherals can be accessed over some distance and can take advantage of the scaling properties of networks.

We have conducted the most extensive analysis to date of the feasibility of using the TCP/IP protocol suite instead of media-specific protocols for accessing shared, networked storage devices. The experiments presented in this paper concentrate primarily on the host operating system mechanisms necessary for such a change, examining the achievable throughput and the CPU performance required. Our data indicate that performance more than 80% of direct-attached SCSI device performance is achievable using UDP without the addition of network coprocessors. TCP performance would be somewhat lower, estimated at 60-75%. Our conclusion is that, while current performance is somewhat lacking, with appropriate hardware and software engineering effort taking advantage of the unique characteristics of I/O for file systems, TCP/IP is a viable transport protocol alternative to Fibre Channel and other protocols.

The architectural implications of the adoption of TCP/IP would be significant. Storage systems which easily span heterogeneous local networks as well as wide area networks can be constructed. Legacy systems, both today and in the future, will be less problematic because TCP/IP runs over many types of network media. Systems which use TCP/IP are already known to scale to large numbers of hosts. Storage I/O will be able to easily share both hardware and software with host-to-host network traffic, reducing development effort and deployed hardware costs.

Availability and Acknowledgments

The source code for VISA itself can be made available only to those sites with SunOS 4.1.3 source code licenses. Code for the disk emulator and Bonnie and the measured performance data will be made available on the Netstation project web page and the author's home page.

This paper has been substantially improved by feedback from John Heidemann, Ted Faber, Bill Manning and Joe Bannister of ISI, Paul Borrill and Daniel Stodolsky of Quantum, and the anonymous referees.

References

[ACM91] ACM. *Proc. SIGCOMM '91.* ACM, September 1991.

[BHMP95] P. Barham, M. Hayter, D. McAuley, and I. Pratt. Devices on the desk area network. *J. Selected Areas in Communications*, 13(4):722–732, May 1995.

[BJM+96] Greg Buzzard, David Jacobson, Milon Mackey, Scott Marovich, and John Wilkes. An implementation of the Hamlyn sender-managed interface architecture. In *Proc. Second USENIX Symp. on Operating Systems Design and Implementation*. USENIX, October 1996.

[BPP91] Mary L. Bailey, Michael A. Pagels, and Larry L. Peterson. The x-chip: An experiment in hardware demultiplexing. In *Proceedings of the*

IEEE Workshop on High Performance Communications Subsystems, February 1991.

[Che87] Greg Chesson. Protocol engine design. In *Proc. 1987 Summer USENIX Conference*, pages 209–215. USENIX, 1987.

[Chu96] Hsiao-keng Jerry Chu. Zero-copy TCP in Solaris. In *Proc. 1996 USENIX Technical Conference*, pages 253–264. USENIX, January 1996.

[CSSZ90] Eric C. Cooper, Peter A. Steenkiste, Robert D. Sansom, and Brian D. Zill. Protocol implementation on the Nectar communication processor. In *Proc. SIGCOMM '90*, pages 188–199. ACM, 1990.

[Dav91] Bruce S. Davie. A host-network interface architecture for ATM. In *Proc. SIGCOMM '91* [ACM91], pages 307–315.

[DPD91] Peter Druschel, Larry L. Peterson, and Bruce S. Davie. Experience with a high-speed network adaptor: A software perspective. In *Proc. SIGCOMM '94*. ACM, ACM, 1991.

[DWB+93] Chris Dalton, Greg Watson, David Banks, Costas Calamvokis, Aled Edwards, and John Lumley. Afterburner. *IEEE Network*, 7(4):35–43, July 1993.

[EK96] Dawson R. Engler and M. Frans Kaashoek. DPF: Fast, flexible message demultiplexing using dynamic code generation. In *Proc. SIGCOMM '96*, pages 53–59. ACM, ACM, October 1996.

[FHV96] Gregory Finn, Steve Hotz, and Rodney Van Meter. The impact of a zero-scan internet checksumming mechanism. *ACM Computer Communication Review*, 26(5):27–39, October 1996.

[Fin91] Greg Finn. An integration of network communication with workstation architecture. *ACM Computer Communication Review*, October 1991. Available online at http://www.isi.edu/netstation/.

[FM94] Gregory G. Finn and Paul Mockapetris. Netstation architecture: Multi-gigabit workstation network fabric. In *Proc. NetWorld+InterOp Engineer Conference*, 1994.

[G+96] Garth Gibson et al. A case for network-attached secure disks. Technical Report CMU-CS-96-142, CMU, June 1996.

[G+97] Garth A. Gibson et al. File server scaling with network-attached secure disks. In *Proc. ACM International Conference on Measurement and Modeling of Computer Systems*. ACM, June 1997.

[HAI+95] Henry H. Houh, Joel F. Adam, Michael Ismert, Christopher J. Lindblad, and David L. Tennenhouse. The VuNet desk area network: Architecture, implementation and experience. *J. Selected Areas in Communications*, 13:710–721, May 1995.

[HG97] Robert W. Horst and David Garcia. ServerNet SAN I/O architecture. In Randy Rettberg and William Dally, editors, *Hot Interconnects Symposium V*. IEEE Computer Society, 1997.

[HVF98] Steve Hotz, Rodney Van Meter, and Greg Finn. Internet protocols for network-attached peripherals. In Ben Kobler, editor, *Proc. Sixth NASA Goddard Conference on Mass Storage Systems and Technologies in Cooperation with Fifteenth IEEE Symposium on Mass Storage Systems*, March 1998.

[Kan88] Hemant Kanakia. The VMP network adapter board (NAB): High-performance network communication for multiprocessors. In *Proc. SIGCOMM '88*, pages 175–187. ACM, 1988.

[Kay95] Jonathan Simon Kay. *Path IDs: A Mechanism for Reducing Network Software Latency*. PhD thesis, UCSD, 1995.

[KLS86] Nancy P. Kronenberg, Henry M. Levy, and William D. Strecker. Vaxclusters: A closely-coupled distributed system. *ACM Transactions on Computer Systems*, 4(2):130–146, May 1986.

[Kob96] Ben Kobler, editor. *Fifth NASA Goddard Conference on Mass Storage Systems and Technologies*, September 1996.

[LL97] James Laudon and Daniel Lenoski. The SGI Origin: A ccNUMA highly scalable server. In *Computer Architecture News, Proc. 24th Annual International Symposium on Computer Architecture*, pages 241–251. ACM, June 1997.

[LT96] Edward K. Lee and Chandramohan A. Thekkath. Petal: Distributed virtual disks. In *Proc. ACM Seventh International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, October 1996.

[Mat98] Todd Matters. Offloading TCP/IP to intelligent adapters. In *Server I/O '98*. Strategic Research Corp., January 1998. Slides from a presentation.

[McC98] John D. McCalpin. STREAM: Measuring sustainable memory bandwidth in high performance computers. web page, July 1998. http://www.cs.virginia.edu/stream/.

[MS68] T. H. Myer and I. E. Sutherland. On the design of display processors. *Communications of the ACM*, 17(6):410–414, June 1968.

[PP93] Craig Partridge and Stephen Pink. A faster UDP. *IEEE/ACM Trans. on Networking*, 1(4):429–440, August 1993.

[RG96] Erik Riedel and Garth Gibson. Understanding customer dissatisfaction with underutilized distributed file servers. In Kobler [Kob96], pages 371–388. also known as CMU-CS-96-158.

[Sid91] Michael A. Sidenius. Hardware support for implementation of transport layer protocols. *Protocols for High Speed Networks*, pages 251–267, 1991.

[Sol97] Steven R. Soltis. *The Design and Implementation of a Distributed File System base on Shared Network Storage.* PhD thesis, U. Minnesota, 1997.

[SRO96] Steven R. Soltis, Thomas M. Ruwart, and Matthew T. O'Keefe. The global file system. In Kobler [Kob96], pages 319–342.

[SWR91] Martin Siegel, Mark Williams, and Georg Rosler. Overcoming bottlenecks in high-speed transport systems. In *Proc. 16th IEEE Conference on Local Computer Networks*, pages 399–407. IEEE, 1991.

[TML97] Chandramohan A. Thekkath, Timothy Mann, and Edward K. Lee. Frangipani: A scalable distributed file system. In *Proc. 16th ACM Symposium on Operating Systems Principles*, pages 224–237. ACM, October 1997.

[TP96] J. Touch and B. Parham. Implementing the internet checksum in hardware. Technical Report Internet RFC 1936, ISI, April 1996.

[TS91] C. Brendan Traw and Jonathan M. Smith. A high-performance host interface for ATM networks. In *Proc. SIGCOMM '91* [ACM91], pages 317–325.

[Van96] Rodney Van Meter. A brief survey of current work on network attached peripherals (extended abstract). *ACM Operating Systems Review*, pages 63–70, January 1996. Full version available on the web at http://www.isi.edu/˜rdv/nap-research/.

[VHF96] Rodney Van Meter, Steven Hotz, and Gregory Finn. Derived virtual devices: A secure distributed file system mechanism. In Kobler [Kob96].

[WM95] Dave Wiltzius and Kim Minuzzo. Network-attached peripherals (NAP) for HPSS/SIOF. web page, October 1995. http://www.llnl.gov/liv_comp/siof/siof_nap.html.

[YBMM94] Masanobu Yuhara, Brian N. Bershad, Chris Maeda, and J. Eliot B. Moss. Efficient packet demultiplexing for multiple endpoints and large messages. In *Proc. USENIX Winter 1994 Technical Conference*, January 1994.