High-Performance Network and Channel Based Storage

RANDY H. KATZ, SENIOR MEMBER, IEEE

Invited Paper

In the traditional mainframe-centered view of a computer system, storage devices are coupled to the system through complex hardware subsystems called I/O channels. With the dramatic shift toward workstation-based computing, and its associated client/server model of computation, storage facilities are now found attached to file servers and distributed throughout the network. In this paper, we discuss the underlying technology trends that are leading to high-performance network-based storage, namely advances in networks, storage devices, and I/O controller and server architectures. We review several commercial systems and research prototypes that are leading to a new approach to high-performance computing based on network-attached storage.

I. INTRODUCTION

The traditional mainframe-centered model of computing can be characterized by small numbers of large-scale mainframe computers, with shared storage devices attached via I/O channel hardware. Today, we are experiencing a major paradigm shift away from centralized mainframes to a distributed model of computation based on workstations and file servers connected via high-performance networks.

What makes this new paradigm possible is the rapid development and acceptance of the client/server model of computation. The client/server model is a message-based protocol in which clients make requests of service providers, which are called servers. Perhaps the most successful application of this concept is the widespread use of file servers in networks of computer workstations and personal computers. Even a high-end workstation has rather limited capabilities for data storage. A distinguished machine on the network, customized either by hardware, software, or both, provides a file service. It accepts network messages from client machines containing open/close/read/write file requests and processes these, transmitting the requested data back and forth across the network.

This is in contrast to the pure distributed storage model, in which the files are dispersed among the storage on workstations rather than centralized in a server. The advantages of a distributed organization are that resources are placed near where they are needed, leading to better performance, and that the environment can be more autonomous because individual machines continue to perform useful work even in the face of network failures. While this has been the more popular approach over the last few years, there has emerged a growing awareness of the advantages of the centralized view. That is, every user sees the same file system, independent of the machine they are currently using. The view of storage is pervasive and transparent. Further, it is much easier to administer a centralized system, to provide software updates and archival backups. The resulting organization combines distributed processing power with a centralized view of storage.

Admittedly, centralized storage also has its weaknesses. A server or network failure renders the client workstations unusable and the network represents the critical performance bottleneck. A highly tuned remote file system on a 10 megabit (Mbit) per second Ethernet can provide perhaps 500K bytes per second to remote client applications. Sixty 8K byte I/O's per second would fully utilize this bandwidth. Obtaining the right balance of workstations to servers depends on their relative processing power, the amount of memory dedicated to file caches on workstations and servers, the available network bandwidth, and the I/O bandwidth of the server. It is interesting to note that today's servers are not I/O limited: the Ethernet bandwidth can be fully utilized by the I/O bandwidth of only two magnetic disks!

Meanwhile, other technology developments in processors, networks, and storage systems are affecting the relationship between clients and servers. It is well known that processor performance, as measured in MIPS ratings,

---

Manuscript received October 1, 1991; revised March 24, 1992. This work was supported by the Defense Advanced Research Projects Agency and the National Aeronautics and Space Administration under contract NAG2-591 (Diskless Supercomputers: High Performance I/O for the TeraOp Technology Base). Additional support was provided by the State of California MICRO Program in conjunction with industrial matching support provided by DEC, Emulex, Exabyte, IBM, NCR, and Storage Technology corporations.

The author is with the Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 9203670.

0018–9219/92$03.00 © 19

CROSSROADS EXHIBIT 2038

is increasing at an astonishing rate, doubling on the order of once every 18 months to two years. The newest generation of RISC processors has performance in the 50 to 60 MIPS range. For example, a recent workstation announced by the Hewlett-Packard Corporation, the HP 9000/730, has been rated at 72 SPECMarks (1 SPECMark is roughly the processing power of a single Digital Equipment Corporation VAX 11/780 on a particular benchmark set). Powerful shared memory multiprocessor systems, now available from companies such as Silicon Graphics and Solborne, provide well over 100 MIPS performance. One of Amdahl's famous laws equated one MIPS of processing power with one megabit of I/O per second. Obviously such processing rates far exceed anything that can be delivered by existing server, network, or storage architectures.

Unlike processor power, network technology evolves at a slower rate, but when it advances, it does so in order of magnitude steps. In the last decade we have advanced from 3 Mbit/second Ethernet to 10 Mbit/second Ethernet. We are now on the verge of a new generation of network technology, based on fiber-optic interconnect, called FDDI. This technology promises 100 Mbits per second, and at least initially, it will move the server bottleneck from the network to the server CPU or its storage system. With more powerful processors available on the horizon, the performance challenge is very likely to be in the storage system, where a typical magnetic disk can service 30 8K byte I/O's per second and can sustain a data rate in the range of 1 to 3 Mbytes per second. And even faster networks and interconnects, in the gigabit range, are now commercially available and will become more widespread as their costs begin to drop [1].

To keep up with the advances in processors and networks, storage systems are also experiencing rapid improvements. Magnetic disks have been doubling in storage capacity once every three years. As disk form factors shrink from 14 inch to 3.5 inch and below, the disks can be made to spin faster, thus increasing the sequential transfer rate. Unfortunately, the random I/O rate is improving only very slowly, owing to mechanically limited positioning delays. Since I/O and data rates are primarily disk actuator limited, a new storage system approach called disk arrays addresses this problem by replacing a small number of large-format disks by a very large number of small-format disks. Disk arrays maintain the high capacity of the storage system, while enormously increasing the system's disk actuators and thus the aggregate I/O and data rate.

The confluence of developments in processors, networks, and storage offers the possibility of extending the client/server model so effectively used in workstation environments to higher performance environments, which integrate supercomputer, near supercomputers, workstations, and storage services on a very high performance network. The technology is rapidly reaching the point where it is possible to think in terms of diskless supercomputers in much the same way as we think about diskless workstations. Thus, the network is emerging as the future "backplane" of high-performance systems. The challenge is to develop the new hardware and software architectures that will be suitable for this world of network-based storage.

The emphasis of this paper is on the integration of storage and network services, and the challenges of managing the complex storage hierarchy of the future: file caches, on-line disk storage, near-line data libraries, and off-line archives. We specifically ignore existing mainframe I/O architectures, as these are well described elsewhere (for example, in [2]). The rest of this paper is organized as follows. In the next three sections, we will review the recent advances in interconnect, storage devices, and distributed software, to better understand the underlying changes in network, storage, and software technologies. Section V contains detailed case studies of commercially available high-performance networks, storage servers, and file servers, as well as a prototype high-performance network-attached I/O controller being developed at the University of California, Berkeley. Our summary, conclusions, and suggestions for future research are found in Section VI.

II. INTERCONNECT TRENDS

A. *Networks, Channels, and Backplanes*

Interconnect is a generic term for the "glue" that interfaces the components of a computer system. Interconnect consists of high-speed hardware interfaces and the associated logical protocols. The former consists of physical wires or control registers. The latter may be interpreted by either hardware or software. From the viewpoint of the storage system, interconnect can be classified as high-speed networks, processor-to-storage channels, or system backplanes that provide ports to a memory system through direct memory access techniques.

Networks, channels, and backplanes differ in terms of the interconnection distances they can support, the bandwidth and latencies they can achieve, and the fundamental assumptions about the inherent unreliability of data transmission. While no statement we can make is universally true, in general, backplanes can be characterized by parallel wide data paths and centralized arbitration, and are oriented toward read/write "memory mapped" operations. That is, access to control registers is treated identically to memory word access. Networks, on the other hand, provide serial data, distributed arbitration, and support more message-oriented protocols. The latter require a more complex handshake, usually involving the exchange of high-level request and acknowledgment messages. Channels fall between the two extremes, consisting of wide data paths of medium distance and often incorporating simplified versions of networklike protocols.

These considerations are summarized in Table 1. Networks typically span more than 1 km, sustain 10 Mbit/second (Ethernet) to 100 Mbit/second (FDDI) and beyond, experience latencies measured in several milliseconds (ms), and the network medium itself is considered to be inherently unreliable. Networks include extensive data integrity features within their protocols,

Table 1 Comparison of Network, Channel, and Backplane Attributes

	Network	Channel	Backplane
Distance	>1000 m	10–100 m	1 m
Bandwidth	10–100 Mb/s	40–1000 Mb/s	320–1000+ Mb/s
Latency	high (>ms)	medium	low (<μs)
Reliability	low Extensive CRC	medium Byte Parity	high Byte Parity

The comparison is based upon the interconnection distance, transmission bandwidth, transmission latency, inherent reliability, and typical techniques for improving data integrity.

including CRC checksums at the packet and message levels, and the explicit acknowledgment of received packets.

Channels span small 10's of meters, transmit at anywhere from 4.5 Mbytes/second (IBM channel interfaces) to 100 Mbytes/second (HiPPI channels), incur latencies of under 100 μs per transfer, and have medium reliability. Byte parity at the individual transfer word is usually supported, although packet-level check-summing might also be supported.

Backplanes are about 1 m in length, transfer from 40 (VME) to over 100 (FutureBus) MBytes/second, incur sub μs latencies, and the interconnect is considered to be highly reliable. Backplanes typically support byte parity, although some backplanes (unfortunately) dispense with parity altogether.

In the remainder of this section, we will look at each of the three kinds of interconnect, network, channel, and backplane, in more detail.

B. Communications Networks and Network Controllers

An excellent overview of networking technology can be found in [3]. For a futuristic view, see [4] and [5]. The decade of the 1980's has seen a slow maturation of network technology, but the 1990's promise much more rapid developments. Today, 10 Mbit/second Ethernets are pervasive, with many environments advancing to the next generation of 100 Mbit/second networks based on the FDDI (Fiber Distributed Data Interface) standard [6]. FDDI provides higher bandwidth, longer distances, and reduced error rates, largely because of the introduction of fiber optics for data transmission. Unfortunately cost, especially for replacing the existing copper wire network with fiber, coupled with disappointing transmission latencies, has slowed the acceptance of these higher speed networks. The latency problems have more to do with FDDI's protocols, which are based on a token passing arbitration scheme, than anything intrinsic in fiber-optic technology.

Application	Detailed information about the data being exchanged
Presentation	Data representation
Session	Management of connections between programs
Transport	Delivery of packet sequences
Network	Format of individual packets
Link	Access to and control of transmission medium
Physical	Medium of transmission

Fig. 1. Seven-layer ISO protocol model. The physical layer describes the actual transmission medium, be it coax cable, fiber optics, or a parallel backplane. The link layer describes how stations gain access to the medium. This layer deals with the protocols for arbitrating for and obtaining grant permission to the media. The network layer defines the format of data packets to be transmitted over the media, including destination and sender information as well as any check sums. The transport layer is responsible for the reliable delivery of packets. The session layer establishes communication between the sending program and the receiving program. The presentation layer determines the detailed formats of the data embedded within packets. The application layer has the responsibility of understanding how these data should be interpreted within an applications context.

A network system is decomposed into multiple protocol layers, from the application interface down to the method of physical communication of bits on the network. Figure 1 summarizes the popular seven-layer ISO protocol model. The physical and link levels are closely tied to the underlying transport medium, and deal with the physical attachment to the network and the method of acquiring access to it. The network, transport, and session levels focus on the detailed formats of communications packets and the methods for transmitting them from one program to another. The presentation and applications layers define the formats of the data embedded within the packets and the application-specific semantics of that data.

A number of performance measurements of network transmission services point out that the significant overhead is not protocol interpretation (approximately 10% of instructions are spent in interpreting the network headers). The culprits are memory system overheads arising from data movement and operating system overheads related to context switches and data copying [7]–[10]. We will see this again and again in the sections to follow.

The network controller is the collection of hardware and firmware that implements the interface between the network and the host processor. It is typically implemented on a small printed circuit board, and contains its own processor, memory mapped control registers, interface to the network, and small memory to hold messages being transmitted and received. The on-board processor, usually in conjunction with VLSI components within the network interface, implements the physical and link-level protocols of the network.

The interaction between the network controller and the host's memory is depicted in Fig. 2. Lists of blocks containing packets to be sent and packets that have been received are maintained in the host processor's memory. The locations of buffers for these blocks are made known to the network controller, and it will copy packets to and from the request/receive block areas using direct memory access (DMA) techniques. This means that the copy of data across the peripheral bus is under the control of the network controller, and does not require the intervention of the host processor. The controller will interrupt the host whenever a message has been received or sent.

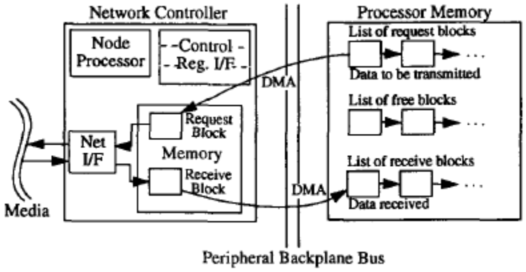

Fig. 2. Network controller/processor memory interaction. The figure describes the interaction between the network controller and the memory of the network node. The controller contains an on-board microprocessor, various memory-mapped control registers through which service requests can be made and status checked, a physical interface to the network media, and a buffer memory to hold request and receive blocks. These contain network messages to be transmitted or which have been received respectively. A list of pending requests and messages already received resides in the host processor's memory. Direct memory operations (DMA's), under the control of the node processor, copy these blocks to and from this memory.

While this presents a particularly clean interface between the network controller and the operating system, it points out some of the intrinsic memory system latencies that reduce network performance. Consider a message that will be transmitted to the network. First the contents of the message are created within a user application. A call to the operating system results in a process switch and a data copy from the user's address space to the operating system's area. A protocol-specific network header is then appended to the data to form a packaged network message. This must be copied one more time, to place the message into a request block that can be accessed by the network controller. The final copy is the DMA operation that moves the message within the request block to memory within the network controller.

Data integrity is the aspect of system reliability concerned with the transmission of correct data and the explicit flagging of incorrect data. An overriding consideration of network protocols is their concern with reliable transmission. Because of the distances involved and the complexity of the transmission path, network transmission is inherently lossy. The solution is to append check-sum protection bits to all network packets and to include explicit acknowledgment as part of the network protocols. For example, if the check sum computed at the receiving end does not match the transmitted check sum, the receiver sends a negative acknowledgment to the sender.

C. Channel Architectures

Channels provide the logical and physical pathways between I/O controllers and storage devices. They are medium-distance interconnect that carry signals in parallel, usually with some parity technique to provide data integrity. In this subsection, we will describe three alternative

interface), HIPPI (high-performance parallel interface), and FCS (fibre channel standard).

1) Small Computer System Interface SCSI is the channel interface most frequently encountered in small form factor (5.25 in diameter and smaller) disk drives, as well as a wide variety of peripherals such as tape drives, optical disk readers, and image scanners. SCSI treats peripheral devices in a largely device-independent fashion. For example, a disk drive is viewed as a linear byte stream; its detailed structure in terms of sectors, tracks, and cylinders is not visible through the SCSI interface. A SCSI channel can support up to eight devices sharing a common bus with an 8-bit-wide data path. In SCSI terminology, the I/O controller counts as one of these devices, and is called the host bus adapter (HBA). Burst transfers at 4 to 5 Mbytes/s are widely available today. In SCSI terminology, a device that requests service from another device is called the master or the initiator. The device that is providing the service is called the slave or the target.

SCSI provides a high-level message-based protocol for communications between initiators and targets. While this makes it possible to mix widely different kinds on devices on the same channel, it does lead to relatively high overheads. The protocol has been designed to allow initiators to manage multiple simultaneous operations. Targets are intelligent in the sense that they explicitly notify the initiator when they are ready to transmit data or when they need to throttle a transfer.

It is worthwhile to examine the SCSI protocol in some detail, to clearly distinguish what it does from the kinds of messages exchanged on a computer network. The SCSI protocol proceeds in a series of phases, which we summarize below:

- Bus Free: No device currently has the bus allocated.
- Arbitration: Initiators arbitrate for access to the bus. A device's physical address determines its priority.
- Selection: The initiator informs the target that it will participate in an I/O operation.
- Reselection: The target informs the initiator that an outstanding operation is to be resumed. For example, an operation could have been previously suspended because the I/O device had to obtain more data.
- Command: Command bytes are written to the target by the initiator. The target begins executing the operation.
- Data Transfer: The protocol supports two forms of the data transfer phase, *Data In* and *Data Out*. The former refers to the movement of data from the target to the initiator. In the latter, data move from the initiator to the target.
- Message: The message phase also comes in two forms, *Message In* and *Message Out*. *Message In* consists of several alternatives. *Identify* identifies the reselected target. *Save Data Pointer* saves the place in the current data transfer if the target is about to disconnect. *Restore Data Pointer* restores this pointer. *Disconnect* notifies the initiator that the targetis about to give up the data

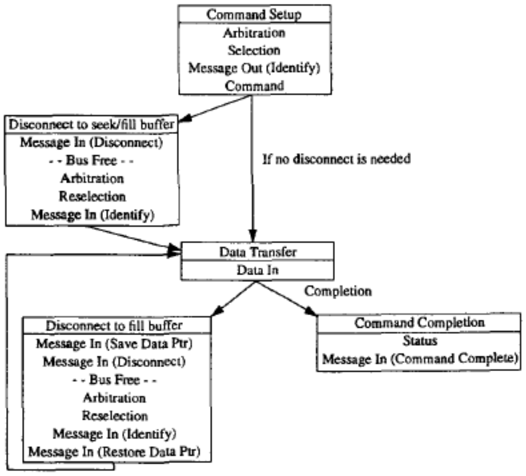

Fig. 3. SCSI phase transitions on a read. The basic phase sequencing for a read (from disk) operation is shown. First the initiator sets up the read command and sends it to the I/O device. The target device disconnects from the SCSI bus to perform a seek and to begin to fill its internal buffer. It then transfers the data to the initiator. This may be interspersed with additional disconnects, as the transfer gets ahead of the internal buffering. A command complete message terminates the operation. This figure is adapted from [40].

Out has just one form: *Identify*. This is used to identify the requesting initiator and its intended target.

- Status: Just before command completion, the target sends a status message to the initiator.

To better understand the sequencing among the phases, see Fig. 3. This illustrates the phase transitions for a typical SCSI read operation. The sequencing of an I/O operation actually begins when the host's operating system establishes data and status blocks within its memory. Next, it issues an I/O command to the HBA, passing it pointers to command, status, and data blocks, as well as the SCSI address of the target device. These are staged from host memory to device-specific queues within the HBA's memory using direct memory access techniques.

Now the I/O operation can begin in earnest. The HBA arbitrates for and wins control of the SCSI bus. It then indicates the target device it wishes to communicate with during the selection phase. The target responds by identifying itself during a following message out phase. Now the actual command, such as "read a sequence of bytes," is transmitted to the device.

We assume that the target device is a disk. If the disk must first seek before it can obtain the requested data, it will disconnect from the bus. It sends a disconnect message to the initiator, which in turn gives up the bus. Note that the HBA can communicate with other devices on the SCSI channel, initiating additional I/O operations. Now the device will seek to the appropriate track and will begin to fill its internal buffer with data. At this point, it needs to reestablish communications with the HBA. The device

the reselection phase, and identifies itself to the initiator to reestablish communications.

The data transfer phase can now begin. Data are transferred one byte at a time using a simple request/acknowledgment protocol between the target and the initiator. This continues until the need for a disconnect arises again, such as when the target's buffer is emptied, or perhaps the command has completed. If it is the first case, the data pointer must first be saved within the HBA, so we can restart the transfer at a later time. Once the data transfer pointer has been saved, the target sequences through a disconnect, as described above.

When the disk is once again ready to transfer, it rearbitrates for the bus and identifies the initiator with which to reconnect. This is followed by a restore data pointer message to reestablish the current position within the data transfer. The data transfer phase can now continue where it left off.

The command completion phase is entered once the data transfer is finished. The target device sends a status message to the initiator, describing any errors that may have been encountered during the operation. The final command completion message completes the I/O operation.

The SCSI protocol specification is currently undergoing a major revision for higher performance. In the so-called SCSI-1, the basic clock rate on the channel is 10 MHz. In the new SCSI-2, "fast SCSI" increases the clock rate to 20 MHz, doubling the channel's bandwidth from 5 Mbyte/s to 10 Mbyte/s. Recently announced high-performance disk drives support fast SCSI. The revised specification also supports an alternative method of doubling the channel bandwidth, called wide SCSI. This provides a 16-bit data path on the channel rather than SCSI-1's 8-bit width. By combining wide and fast SCSI-2, the channel bandwidth quadruples to 20 Mbyte/s. Some manufacturers of high-performance disk controllers have begun to use SCSI-2 to interface their controllers to a computer host.

2) High-Performance Parallel Interface The high performance parallel interface, HIPPI, was originally developed at the Los Alamos National Laboratory in the mid 1980's as a high-speed unidirectional (simplex) point-to-point interface between supercomputers [11]. Thus, two-way communications requires two HIPPI channels, one for commands and write data (the *write channel*) and one for status and read data (the *read channel*). Data are transmitted at a nominal rate of 800 Mbit/s (32-bit-wide data path) or 1600 Mbit/s (64-bit-wide data path) in each direction.

The physical interface of the HIPPI channel was standardized in the late 1980's. Its data transfer protocol was designed to be extremely simple and fast. The source of the transfer must first assert a request signal to gain access to the channel. A connection signal grants the channel to the source. However, the source cannot send until the destination asserts ready. This provides a simple flow control mechanism.

The minimum unit of data transfer is the *burst*. A burst consists of 1 to 256 words (the width is determined by

DOCKET
ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.