ERIK RIEDEL, SEAGATE RESEARCH

Storage Systems

Not Just a Bunch of Disks Anymore

The sheer size and scope of data available today puts tremendous pressure on storage systems to perform in ways never imagined.

The concept of a storage device has changed dramatically from the first magnetic disk drive introduced by the IBM RAMAC in 1956 to today's server rooms with detached and fully networked storage servers. Storage has expanded in both large and small directions—up to multi-terabyte server appliances and down to multi-gigabyte MP3 players that fit in a pocket. All use the same underlying technology—the rotating magnetic disk drive—but they quickly diverge from there.

Here we will focus on the larger storage systems that are typically detached from the server hosts—the specialized appliances that form the core of data centers everywhere. We will introduce the layers of protocols and translations that occur as bits make their way from the magnetic domains on the disk drives and interfaces—around the corner or around the world—to your desktop.

BANDWIDTH

Let's start by looking at the internals of a modern desktop or server computer with direct-attached storage (DAS) and illustrate what happens to data after it leaves the disk drive on its way to you.

First, data passes through a set of buffers on the disk drive through the SCSI bus over to the SCSI host bus adapter, then across the peripheral component interconnect (PCI) bus into the system memory buffers, and finally into the application memory for further processing, or onto your screen. SCSI may be replaced with ATA (also known as

Storage Systems

Not Just a Bunch of Disks Anymore

EIDE), and PCI may be replaced with HyperTransport, InfiniBand, or a similar architecture. The key is that the disks are directly attached to the server where the data is likely used and secured. Figure 1 illustrates this and gives the maximum (peak) advertised performance of each of these components.

These peak specifications are rarely met in practice however, as shown in Table 1. The first two columns of data are from a 1997 study ["A Performance Study of Sequential I/O on Windows NT," by Erik Riedel, Catharine van Ingen, and Jim Gray, *Proceedings of the Second Usenix Windows NT Symposium*, Seattle, WA, August 1998], and the second set of data is from a study three years later

["Windows 2000 Disk IO Performance," by Leonard Chung, Bruce Worthington, Robert Horst, and Jim Gray, Microsoft Research MS-TR-2000-55, June 2000].

Each layer in the system takes its "cut" off the performance of the layer below. Each layer has some amount of overhead that reduces the total bandwidth available to the next higher part of the system. For example, the SCSI protocol has several communication stages before any data can be transferred. To transmit data, a device must arbitrate for the bus, select a device to talk to, send its command, and wait for the data. This introduces a fixed amount of overhead on each request. The less data being moved, the higher this overhead is relative to the data-transfer (useful work) portion of the request. A similar handshake protocol is performed on the PCI bus between the SCSI host adapter and memory, and then possibly again between the memory controllers as the data is transferred to the application's memory space. This is true not just in storage protocols, but throughout computer systems—a 1-gigabyte-per-second (GBps) Ethernet connection almost never reaches its peak "advertised" performance in real-world use. In most cases, out-of-the-box performance of a set of components is below the maximum achievable performance, and tuning is required to

even come close to the theoretical maximums.

A common technique used to fill the upstream high-performance pipes is to introduce multiple drives that are combined for higher aggregate bandwidth, as shown in Figure 2. Sixteen individual disks are combined four at a time onto four independent SCSI buses, which connect to two independent PCI buses, and finally to the host memory controller. If there were no overhead along the way, this should produce $16 \times 75 \text{ MBps} = 1.2 \text{ GBps}$ of bandwidth into the application. In reality this is not the case; most systems will produce only one-half to two-thirds of this bandwidth.

TABLE 1

Advertised vs. Measured Performance

Performance	1997 Advertised	1997 Measured	2000 Advertised	2000 Measured
Disk drive	10-15 MBps	9 MBps	26 MBps	24 MBps
SCSI bus	40 MBps	31 MBps	160 MBps	98 MBps*
PCI bus	133 MBps	72 MBps	133 MBps	98 MBps
Memory bus	422 MBps	142 MBps	1,600 MBps	975 MBps*

*read performance; write throughput is slower

TABLE 2 Upper-layer Protocol Overheads

Protocol	SCSI	Fibre Channel (FC)	TCP/IP
Startup	ARBITRATE	LOGIN	SYN
	SELECT/RESELECT	R_RDY	ACK
Send command	COMMAND	COMMAND	DATA (request)
Transfer data	DATA	DATA IN/OUT	DATA (response)
Cleanup	STATUS	STATUS	FIN

This technique of adding multiple drives is typically called “just a bunch of disks,” or JBOD. That is because it is the host server operating system (OS) that must then manage these disks to create the needed storage for the system. Introducing multiple drives results in more than just bandwidth improvement. Multi-drive volume management techniques and buffering help separate the storage from the server, as we will see later.

LATENCY

The story for latency over this hierarchy of buses is similar to that for bandwidth, but the impact is more difficult to hide. Overhead and buffering occur at each layer of protocol translation, and the time to service a request includes all the costs to get through all the layers

for each request. When a large amount of data is transferred, the requests can usually be pipelined. That is, the startup overheads for the next request can be overlapped with the data-transfer portion of the current request. The overlap is not possible when only a single request is issued or the data payload transfer time is small compared with the request setup time. The user wait time, or latency, for an individual request is the total time to overcome all the layers of requests and get the result back.

Table 2 shows the command requirements of the most popular storage protocols. Each requires multiple rounds of messaging. Different protocols will have varying amounts of overhead, depending on their assumptions for the underlying network medium.

This is not quite an apples-to-apples comparison. To use TCP/IP for storage, an application

layer must go on top of it—e.g., Internet SCSI (iSCSI) to implement the SCSI command set or Network File System (NFS) to serve files. A physical and data link layer must go below it, as is provided with Gigabit Ethernet. Although not depicted here, the typical protocol stack for a storage system contains four or more layers, matching the Open System Interconnection (OSI) model. Fibre Channel (FC) defines all these layers, with the SCSI command set being the highest. But, in fact, TCP/IP and then Ethernet can be used to implement the lower three layers of the stack. As a result, SCSI, FC, TCP/IP, and Ethernet could compose the protocol stack, and all these startup and other command overheads have to be negotiated at each layer. Latency can therefore build up if efficiency is not considered across the system as a whole.

Storage Systems

Not Just a Bunch of Disks Anymore

SCSI and FC assume that data packets will arrive at the remote node intact and in order. The underlying physical interconnects are closer to circuit-switched channels than packet-switched, multi-hop routed networks. IP and the underlying Ethernet, however, do not guarantee delivery in order. It is not until the TCP driver delivers the results up the protocol stack that the “bitstream” is considered in order. Therefore, while TCP/IP and Ethernet are considered possible replacements in low- to medium-performance storage networks, extra work is necessary to ensure the basic principals assumed by the SCSI command set are followed. Luckily, storage networks today tend to be dedicated and local, meaning packet-switched protocols should handle the common case efficiently and push only the occasional exception to the higher-level protocol.

The SCSI physical layer is intended for short hops without many intervening switches, whereas FC is often deployed in settings with multiple switches, but local to the server room or campus. Ethernet can have a larger switch fabric but does not have the built-in provisioning and circuit-switched nature of FC. Routed TCP/IP is intended for much longer, more complex hops with multiple paths between the source and destination, passing through numerous intervening switches, routers, and links of varying speed and reliability. Latency is introduced not only by the particular protocol stack and the

various command setups, but also simply by the distance of the communication.

Table 3 shows the latency of a fiber-optic storage network over various distances and numbers of switches (hops). For local traffic, the disk-access time dominates the overall latency, but once the signal must cover a regional or national distance, the time in transit becomes the limiting factor. The assumptions are within rough orders of magnitude for many FC and Ethernet switches. Routing and wide-area networks (WANs) would add still more delay.

The table uses a per-hop latency of 2 microseconds, a wire time-of-flight delay of 5 microseconds per kilometer, a disk-access time of 4 milliseconds, a data block transfer size of 2 KB, and a wire switching speed of 1 Gbps. The numbers in the table are for non-cut-through switching, but this does not matter for the longer runs. The apparent switch latency decreases from 22 microseconds to 2 microseconds for a 2-KB block if cut-through is possible, because the whole packet does not have to be received before it can be forwarded to its next destination. Short distances are dominated by the disk-access time and not the number of hops; long distances are dominated by the time-of-flight delay through the fiber. Cut-through switching becomes more important in virtualized networks with distributed caches that dramatically reduce the average disk-access time by avoiding the disk altogether.

MANAGEMENT OF MULTIPLE DISKS

The architecture—in hardware or software—used to combine the capacity and bandwidth of multiple disk drives is usually one of several configurations of redundant arrays of independent disks (RAID), which provide both increased performance and improved reliability. The details of RAID are thoroughly discussed elsewhere [see, for example, “The Parallel Data Lab RAID Tutorial,” presented at ISCA ’95, <http://www.pdl.cmu.edu/RAIDtutorial/>], but

it is worth briefly touching on some of them here to provide insight.

The basic options are mirroring (the same data replicated on two different drives) and striping (data spread evenly over a larger number of drives). Mirroring can offer higher performance than striping when accessing a block of data

TABLE 3

Switch Latency

	Distance	Hops	One-Way Trip (one bit)	2-KB read
NYC (campus)	1 km	1	7 μ s	4 ms
NYC (campus)	1 km	20	45 μ s	5 ms
NYC to Stamford, CT (metro)	70 km	3	350 μ s	6 ms
NYC to Hartford, CT (regional)	200 km	3	1 ms	10 ms
NYC to Los Angeles, CA (national)	4,000 km	50	20 ms	125 ms

across multiple drives. The biggest component determining performance is the time to access data on a disk—the average seek and positioning time. In mirroring, the same request is broadcast to two different drives. Because each has the same average access time, but likely different arm positions, average performance is roughly twice that of a single disk. For striping, on the other hand, performance is determined by the wait for the last of the disk arms to get into the proper position. Throughput in striping is higher if many requests are outstanding, but in the worst case—with only a single isolated request—performance is worse as you wait for all the disks to respond. A combination of the two techniques may well achieve the best overall performance (the detailed trade-offs are best left for another article).

If a RAID technique is not being used, then combining multiple disks into a single logical volume is JBOD. Note that this is the opposite concept of what typically occurs in desktop systems, where a disk is partitioned into smaller logical volumes. The idea of combining multiple disks is to make the management and distribution of storage independent of the physical devices. This is helpful also as smaller, faster drives provide better performance than fewer, larger but slower drives.

NETWORKED STORAGE

If the computer system is dedicated as a server for networked storage, then the entire data flow is repeated in reverse as data is copied out of the local memory and onto a network adapter [see “Network Attached Storage Architecture,” by Garth A. Gibson and Rodney Van Meter, *Communications of the ACM*, 43 (11)]. This forms the basic picture of a networked storage server, as illustrated in Figure 3. It consists of a storage controller with direct-attached disks or disk arrays. The controller aggregates the disks and hides the traditional storage functions such as RAID, buffering, and volume management. For easy setup, any server or PC can usually be

configured to perform this controller function.

Shown in Figure 3 is a network-attached storage (NAS) device that serves files using the Windows Common Internet File System (CIFS) file share or NFS over Ethernet and IP. Replace the Ethernet with switched Fibre Channel and provide a block interface, and you have a typical storage area network (SAN) architecture. For higher density, the SCSI bus connecting the disks may be replaced with an FC arbitrated loop that allows as many as 120 disks and supports dual ports and dual channels per disk for improved reliability.

NAS and SAN are very different architectures from the server perspective, although they are similar from a hardware perspective. A NAS system provides shared file-system functions, including individual file security, on a central controller. A traditional NAS system is a way to provide a centralized, multi-disk shared resource that can be easily used by desktops or servers on the LAN.

SAN relies on each server connecting to it to provide security and management of the portion of the storage pool assigned to it. Only recently are techniques being introduced that allow storage pools to be shared among servers, but this is usually only when the host servers are tightly clustered. The key is that the traditional SAN system is trying to present a view of direct-attached, dedicated disks to each server while pulling out the reliability, backup, and disk management issues from single-server control. SAN is geared for raw performance and generally

FIG 3

Storage Systems

Not Just a Bunch of Disks Anymore

achieves better performance than direct-attached disks as a result of larger caches and buffering. In fact, early SANs had SCSI connections to the hosts, as well as to the back-end disks.

The storage controller can present either a file-server interface (NFS or CIFS) to the network as traditionally done in NAS or a dedicated raw-block interface to logical disk volumes as is typical of SAN. Typically NAS storage controllers connect to the LAN and provide file services to many desktop clients. SAN storage controllers, on the other hand, use a dedicated storage network such as Fibre Channel or even SCSI to connect directly to a smaller number of servers. For reliability and management reasons, Fibre Channel may use a switched network, similar to Ethernet switches, between multiple storage controllers and servers.

The larger, special-purpose storage systems that allow hundreds of disks to be attached and managed include more specialized hardware. They tend to integrate some of the protocol layers into hardware accelerators or bulk up on buffer space to improve total throughput, reduce latency, or improve the number of volumes that can be served to multiple servers simultaneously. Data movement through the server is optimized by adding special-purpose hardware to perform a data-mover function (similar to direct memory access techniques in computer architectures of old) and ensure a steady flow of data between the disks and switched storage network fabric. In this scenario, the

storage controller or server processor does not participate in the common-case data-block transfer, but is invoked only for “management” functions such as request scheduling or cache maintenance. In the largest systems, such as those from EMC and Hitachi Data Systems, this data mover consists of a large set of components—front-end and back-end communication modules all connected with multi-Gbps crossbar switches and a large amount of cache memory. Many disk arrays, however, use the same commodity hardware as the general-purpose servers already described. SAN with a data mover is illustrated in Figure 4.

One additional significant feature of network storage devices using storage controllers—both the NAS and SAN varieties—is the use of caching to enhance performance. The use of large memory caches with prefetching helps avoid the mechanical delay and can actually improve the performance of the storage device over traditional direct-attached disks. But this introduces reliability concerns because data stored in memory is not persistent across power failures or other faults. This is not a problem for reads—when data is moving from the disks to the network—because any data dropped in transit still exists on the drives themselves. For writes, however, the issue quickly becomes a thorny one.

If data is placed into the memory cache before being written permanently to the disk drives, then this memory must be protected to keep the data safe and recoverable after a power failure or other catastrophic error. Doing so allows the storage device to acknowledge a user write

request as soon as the data enters the cache, rather than waiting for it to be permanently written. This greatly reduces latency, which is often the critical performance point in a system trying to improve fault tolerance. Implementing immediate acknowledge caches requires carefully designed hardware, software, and protocols for redundancy and error recovery in the various system components, such as multiple server processors, multiple paths to the cache, and potentially multiple batteries. This is further complicated if caching is performed at multiple layers in the storage system with each point immediately acknowledging the write. It quickly becomes difficult to prove strong statements about the overall reliability of the stored data.

How many of you have been working on your desktop late at night when the network suddenly seems to go down? Network performance slows, and disks become virtually unavailable with very long access times. This is likely caused by your IT department using the LAN to back up desktop or server storage to tape systems. Using a dedicated storage network with storage controllers can begin to isolate this problem from the LAN. In many large IT environments today, large automated, robotic tape backup systems are connected via SCSI or possibly Fibre Channel directly into the storage network fabric. The storage controller can take on the added task of moving the data from the disks to the tapes and back, independently of the service requests to hosts. The key is simply to provision enough communication bandwidth for this requirement. This allows backups to be occurring continually or at will whenever it is convenient. Software techniques such as journaling and file-system snapshots help provide this concurrent use of the disks and backup. In fact, in many systems today, snapshots are replacing the daily backup function, handling simple file restores without resorting to tape.

With the reliability and fault tolerance through mirroring already built into disk storage systems, the need to archive to tape at all is quickly disappearing. Network storage protocols allow geographically dispersed mirroring, sometimes even in realtime. The cost, density, and reliability of a magnetic disk drive is now comparable to magnetic tape, certainly when you consider the robotic systems needed to operate the tape. It is even becoming possible today to have RAID available in the operating system and provide higher reliability for desktop computer disk drives. This avoids the need for critical backups to guard against hardware failure, leaving only geographic dispersion and user “oops” restores to deal with. Most importantly, almost all computer systems today are

networked; thus, critical storage can be centralized and managed independently of desktops and servers.

LAYERS AND LAYERS— VIRTUALIZATION EVERYWHERE

There has been much use—and misuse—of the term *virtualization* when applied to storage systems [see *Virtual Storage Redefined—Technologies and Applications for Storage Virtualization*, by Paul Massiglia with Frank Bunn, Veritas Software Corp., 2003]. Any computer science student knows that virtualization—introducing a layer of indirection—is a favorite trick of system designers (most prominently in describing the memory space for processes). In the context of storage systems, virtualization means an additional layer of software or hardware between the physical storage device and the user.

In its simplest form, virtualization is just another form of aggregation of available block storage space that is then partitioned into logical volumes independently of the physical drives behind it—in this case, often the large storage controllers themselves. Historically in SANs, one storage controller would have multiple host interfaces that then would feed directly into multiple server hosts or switched fabric to increase the number of hosts supported. Logical volumes are mutually exclusive and dedicated to one host. Virtualization allows multiple storage controllers—possibly heterogeneous and from multiple vendors—to be connected and their storage aggregated. Figure 5 shows the hardware variant—in this case, another device—once again with the same structure as the devices described previously.

The important point about virtualization is that it is not an end in itself. It is a layer of indirection that enables additional functions, including remote mirroring (making a copy of written data to a distant site, usually a disaster recovery site), caching (with additional memory for read or write caching), interoperability (allowing devices from two different vendors to be seen as a single device), global namespace (providing a single system view for either end users or system administrators), and scaling (allowing multiple devices to produce higher aggregate performance, as illustrated for a single system in Figure 2). This additional hardware device will again take a “cut” of the bandwidth available to the higher level and increase the latency for short requests. In the case of write caching, where data is stored temporarily on these intermediaries, the reliability concerns mentioned previously again make the design more challenging, introducing opportunities for error and, in some cases, leading to unpredictable user-visible behavior.

Storage Systems

Not Just a Bunch of Disks Anymore

As shown in Figure 6, most of the functions performed by a hardware virtualization device can also be done by an additional layer of software and possibly hardware executed directly on or at each server host. This software-based virtualization is usually done by coordination among the hosts (often referred to as *clustering*) and maybe a specialized asymmetric meta-data server that sits off to the side of the common-case data transfers—just like the component labeled “Server” in Figure 6. This server may be deployed at various points in the network,

perhaps close to the hosts, close to the storage servers, or even as part of the switch fabric. This component server is optional, as the same functionality can be achieved in a peer-to-peer configuration among the individual hosts acting as a group. Such an architecture presents additional problems in reliability and deployment, but is an appropriate architecture for a range of uses. Note also that clustering and virtualization can be implemented with striping on the hosts. As with a single server with direct-attached disks, however, this does not separate the functionality, reliability, and management of the storage from the server.

This type of virtualization is not limited to SAN devices as illustrated, but is also done at the file layer with NAS devices [Shared Storage Model, Storage Networking Industry Association, http://www.snia.org/tech_activities/shared_storage_model/], where virtualization supports the same set of functions. For many in the storage research community, this is nothing new, as most of these functions have been the subject of research in file systems and distributed file systems for years [“Scale and Performance

in a Distributed File System,” J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, and M. West, *ACM Transactions on Computer Systems*, vol. 6, February 1988].

THE NEAR FUTURE

The ever-growing appetite for storage and the increasing importance that corporations (and individuals) place on their data makes this an exciting time in the storage industry. Data is becoming increasingly distributed, and access to data is becoming increasingly widespread. Imagine both the distance and scale made possible by the Internet and corporate intranets and the huge increase in the number of people with access to shared data. Inexpensive storage devices, and inexpensive networks

same as network security. Although some of the same types of solutions apply, viewing stored data as simply a “message” with a very long time between sending and delivery is an analogy that quickly breaks down—the eventual message “destination” is often unknown when the message is “sent,” and the “sender” may no longer be active when the message is finally “delivered” [refer to “Network Security and Storage Security: Symmetries and Symmetry-Breaking,” by Don Beaver, 1st International IEEE Security in Storage Workshop, December 2002].

The trend has been to centralize data stor-

to connect data with users, make it possible for a much wider audience to share data, collaborate, and benefit from collective knowledge.

This opportunity brings many challenges. The sheer size and scope put pressure on our ability to make far-away data seem close. Users expect corporate databases to perform as well—to be as accessible and as easy to use—as their MP3 players or PDAs, and this illusion is difficult to maintain.

Users must be able to find the data they are looking for within a vast sea of information. I don’t know about you, but I am barely able to find anything in the files and e-mail messages and bookmarks I’ve collected over the past six months, much less all of those that my employer has collected over the past 20 years. The problem is how to organize all the data in our lives so that it is accessible and usable when needed. How many people do you know ignore the file system altogether and use their e-mail inbox with file attachments as their only organizational tools for accessing various versions of files or messages?

Because data is more critical to day-to-day operations in large companies, it is also more sensitive, so security is a growing concern. This goes beyond traditional reliability from component failures. Data today must be protected from malicious adversaries. This is a considerably more difficult problem. Storage security is not the

age, but the required scale may call for a solution that is much closer to a peer-to-peer system. The eventual solution may well be a hybrid approach that uses judicious “offloading” of common-case functions, as illustrated by the fast-paths described earlier, both inside devices and between devices and hosts across a network, which are already appearing in systems today. Such a model devolves more responsibility to the many, many individual devices that will have a significant amount of autonomy and will be able to function together in ways that will surpass the sum of their parts. Q

ERIK RIEDEL leads the Interfaces and Architecture Department at Seagate Research in Pittsburgh, Pennsylvania. The group focuses on storage systems with increased intelligence for optimized performance, automated management, and content-specific optimizations. Before joining Seagate, he was a researcher in the storage program at Hewlett-Packard Laboratories in Palo Alto, California. He received a doctorate in computer engineering from Carnegie Mellon University. Over the years he has spent time looking at I/O in a number of areas, including parallel apps, data mining, databases, file systems, and scientific data processing.

© 2003 ACM 1542-7730/03/0600 \$5.00

BIO