Disk System Architectures for High Performance Computing

Randy H. Katz, Garth A. Gibson, David A. Patterson

Computer Science Division
Electrical Engineering and Computer Science Department
University of California, Berkeley
Berkeley, CA 94720

ABSTRACT: Because of the mismatch between I/O and CPU speeds, high performance computers have long been forced to confront the fundamental I/O bottleneck. As processing power and memory size continue to grow rapidly for micro and mini computers, they too will become I/O limited. A number of hardware and software approaches, such as parallel read-out disks, expanded storage (e.g., solid state disks), and disk striping, have been used to increase I/O bandwidth and thus narrow the CPU-I/O performance gap. In addition, new developments driven by advances in small diameter (i.e., 5.25" and 3.5") disk drives, promise very high I/O bandwidth if large numbers of devices can be organized into arrays of disks. In this paper, we shall review the state of the art in disk devices and I/O controllers, and will describe new approaches for very high performance I/O based on redundant arrays of inexpensive disks (RAIDs).

KEY WORDS AND PHRASES: High performance I/O architectures, technology trends in I/O devices, redundant arrays of inexpensive disks

1. Introduction

Architects of high performance computers have long been forced to acknowledge the existence of a large gap between the speed of the CPU and the speed of its attached I/O devices. A number of techniques have been developed in an attempt to narrow this gap, and we shall review them in this paper. However, the computational environment for high performance computing is undergoing radical changes. The distinction between supercomputers, minisupercomputers, superminicomputers, and high performance workstations is rapidly becoming blurred. For example, the recently announced Intel i860 processor chip possesses the same processing power as the original CRAY-1 (approximately 33 scalar MIPS), yet sells in OEM quantities at $750. The concept of the *diskless supercomputer*, i.e., a high performance computaticnal machine connected via high speed network to a file server with large numbers of disk devices, is becoming a possibility. This presents new challenges for system architects since the I/O system is likely to be attached to a machine with less computational resources than the supercomputer, though not much less, and it will need to provide acceptable I/O performance for many kinds of high performance applications, including image processing and conventional timesharing [1].

To better understand technological forces driving the performance of CPU, memory, and I/O, and the resulting CPU-I/O gap, let us examine the rapid development in speed and memory sizes of VLSI (Very Large Scale Integration) processors. Over the last de-

cade, these "micro" processors have experienced more rapid growth in processing power than other classes of uniprocessors. Bell has observed that they have improved in performance by 40% per year between 1974 and 1984, about twice the rate of minicomputers [2]. Using the Intel microprocessor family, Myers makes a similar statement, observing that processor power doubled every 2.25 years since 1978 [3]. Joy has predicted an even faster rate of growth that has so far been born out by the introduction of commercial RISC (Reduced Instruction Set Computer) processor chips [4]:

$$MIPS = 2^{\text{Year-1984}}$$

Figure 1.1 shows the rapid growth of single chip processor performance as observed and predicted by Myers and Joy.

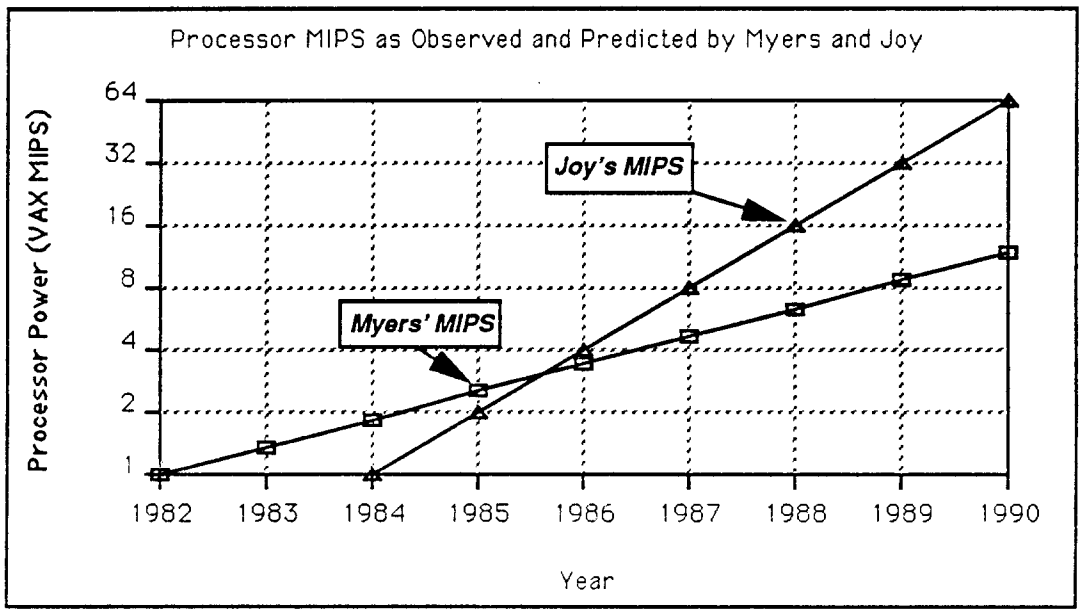


FIGURE 1.1: Single Chip Processor MIP Rates.
Microprocessor MIP rates (measured relative to the DEC VAX 11/780) have been doubling approximately every two years. An even faster growth rate has been predicted for RISC processors.

The memory system must also become faster and larger to match the increase in the processor's demand for instructions and data. Amdahl related CPU speed to main memory size using the following rule [5]:

Each CPU instruction per second requires one byte of main memory;

If computer system costs are not dominated by the cost of memory, then this "rule of thumb" suggests that memory chip capacity should grow at the same rate as CPU speed. Moore observed that growth rate over 20 years ago:

$$transistors/chip = 2^{Year-1964}$$

As predicted by Moore's Law, RAMs have quadrupled in density every two [6] to three years [3], as Figure 1.2 demonstrates.

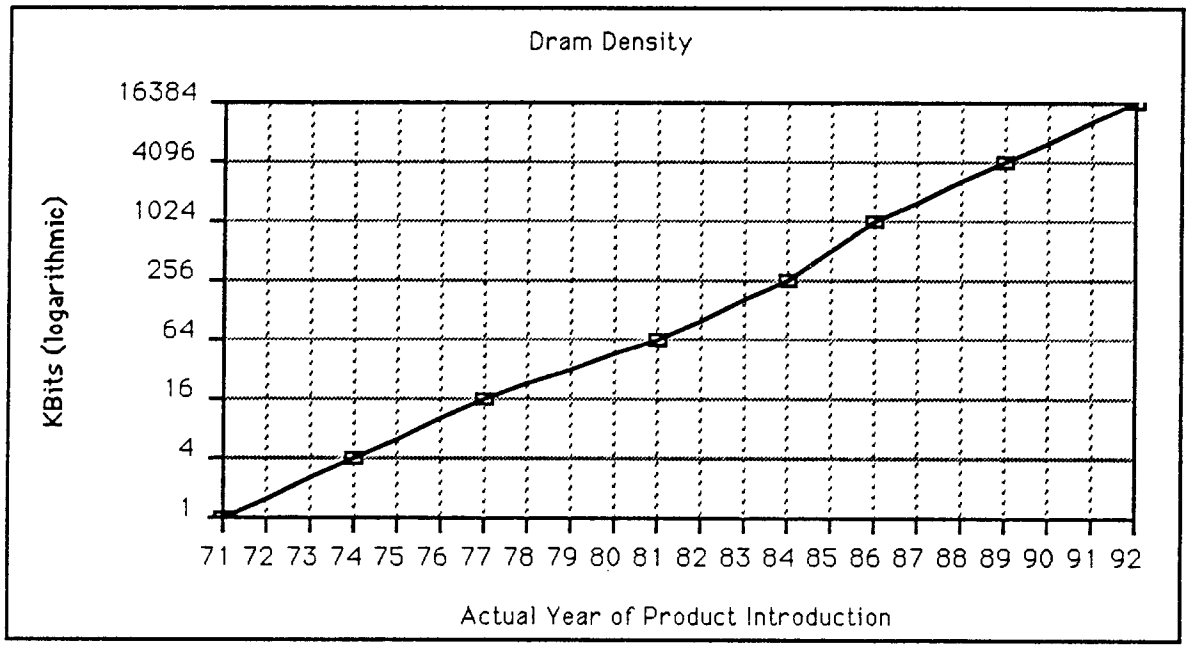


FIGURE 1.2: DRAM Chip Densities by Year.
MOS memory chips have been doubling in capacity every two to three years.

Recently the ratio of megabytes of main memory to MIPS has been defined as *alpha* [7], with alpha = 1 corresponding to Amdahl's rule. In part because of the rapid drop of memory prices (at least for memories matched in speed to today's microcomputer and minicomputer CPUs), main memory sizes have grown even faster than CPU speeds, and many machines are shipped today with alphas of 3 or higher.

To maintain the balance of costs, secondary storage must also match the improvements in other parts of the system. A key measure of magnetic disk technology is the growth in the maximum number of bits that can be stored per square inch, i.e., the bits per inch in a disk track times the number of tracks per inch of media. Called M.A.D., for maximal areal density, the "First Law in Disk Density" predicts [8]:

$$MAD = 10^{(Year-1971)/10}$$

This is plotted against several real disk products in Figure 1.3. Magnetic disk technology has doubled capacity and halved price every three years, in line with the growth rate of semiconductor memory. Between 1967 and 1979 the growth in disk capacity of the average IBM data processing system more than kept up with its growth in main memo-

ry, maintaining a ratio of 1000:1 between disk capacity and physical memory size [9].

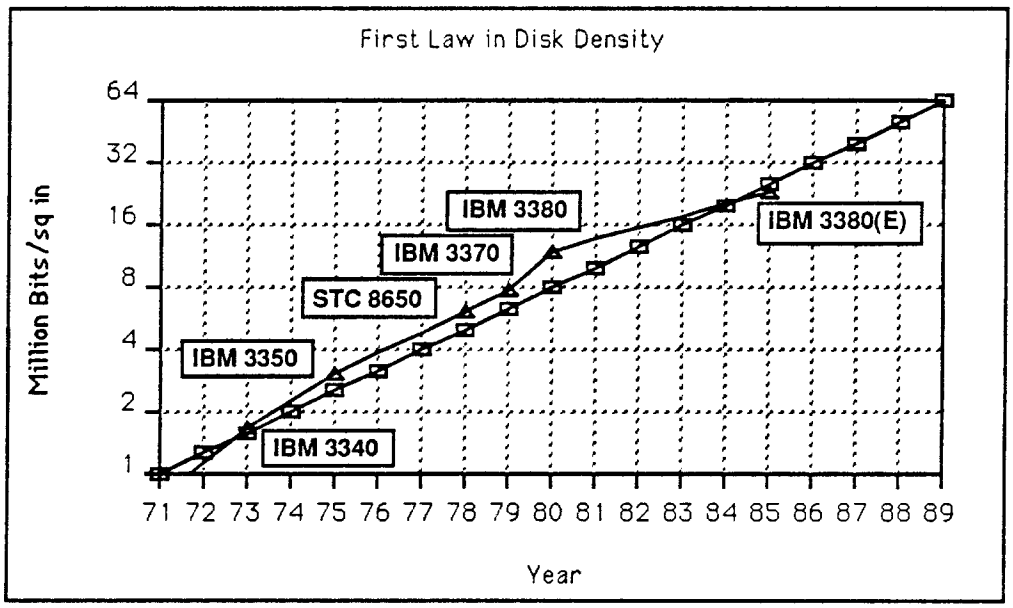


FIGURE 1.3: Maximal Areal Density Law.
Squares represented predicted density; Triangles arethe M.A.D. reported for the indicated products.

Capacity is not the only memory characteristic that must grow rapidly to maintain system balance, since the speed with which data and instructions are delivered to the CPU also determines its ultimate performance. Main memory speed has managed to keep pace for two reasons:

(1) *caches*, i.e., a technique for organizing memory by providing a small high speed buffer in front of slower, larger memory that can contain a substantial fraction of memory references [10];

(2) *static RAMs*, the technology used to build caches, whose speed has been improving at the rate of 40% or more per year.

In contrast to primary memory technologies, the performance of conventional magnetic disks has improved only modestly. These *mechanical* devices, the elements of which are described in more detail in the next section, are dominated by seek and rotation delays: from 1971 to 1981, the raw seek time for a high-end IBM disk improved by only a factor of two while the rotation time did not change [11]. Greater recording density translates into a higher transfer rate once the information is located, and extra positioning actuators for the read/write heads can reduce the average seek time, but the raw seek time only improved at a rate of 7% per year. This is to be compared to a doubling in processor power every year, a doubling in memory density every two years, and a doubling in disk density every three years, the implication of which is summarized in Figure 1.4. The gap between processor performance and disk speeds continue to

widen, and there is no reason to expect a radical improvement in raw disk performance in the near future.

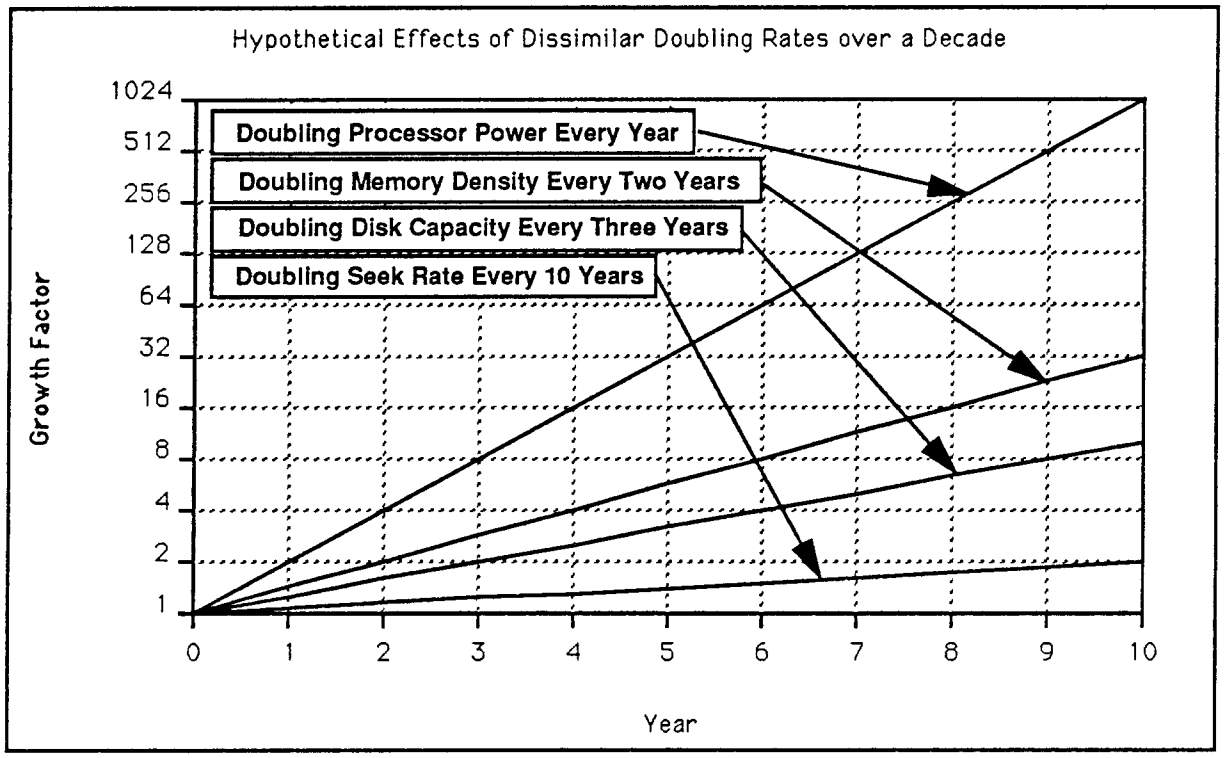


FIGURE 1.4: Implications for the CPU-IO Gap.
The dissimilar rates of doubling for processor power (every year), memory size (every two years), disk capacity (every three years), and seek rate (every 10 years) implies an ever widening gap in performance between the CPU and its attached I/O system.

To maintain balance, computer systems have been using even larger main memories or solid state disks to buffer some of the I/O activity. This may be an acceptable solution for applications whose I/O activity has locality of reference and for which volatility is not an issue, but applications dominated by a high rate of random requests for small pieces of data (e.g., transaction processing) or by a small number of sequential requests for massive amounts of data (e.g., supercomputer applications) face a serious performance limitation.

The rest of the paper is organized as follows. In the next section, we will briefly review the fundamentals of disk system architecture. Section 3 describes the characteristics of the applications that demand high I/O system performance. Conventional ways to improve disk performance are discussed in Section 4. Section 5 introduces the new developments in disk array systems, while Section 6 describes controller architectures. Our summary and conclusions are given in Section 7.

2. Basic Magnetic Disk System Architecture

In this section we quickly review the basic terminology of magnetic disk devices and controllers, and then examine the disk subsystems of three manufacturers (IBM, Cray, and DEC). Throughout this section we are concerned with technologies that support random access, rather than sequential access (e.g., magnetic tape). A more detailed discussion, focusing on the structure of small dimension disk drives, can be found in [12]. The basic concepts are illustrated in Figure 2.1. A *spindle* consists of a collection of *platters*. Platters are metal disks covered with a magnetic material for recording information. Each platter contains a number of circular recording *tracks*. A *sector* is a unit of a track that is physically read or written at the same time. In traditional magnetic disks the constant angular rotation of the platters dictate that sectors on inner tracks are recorded more densely than sectors on the outer tracks. Thus, the platter can spin at a constant rate and the same amount of data can be recorded on the inner and outer tracks[1]. Some modern disks use zone recording techniques to more densely record data on the outer tracks, but this requires more sophisticated read/write electronics.

The read/write *head* is an electromagnet that produces switchable magnetic fields to read and record bit streams on a platter's track. It is associated with a disk arm, attached to an actuator. The head "flies" close to, but never touches, the rotating platter (except perhaps when powered down). This is the classical definition of a *Winchester* disk. The actuator is a mechanical assembly that positions the head electronics over the appropriate track. It is possible to have multiple read/write mechanisms per surface, e.g., multiple heads per arm - at one extreme, one could have a head per track position,

1. Some optical disks use a technique called Constant Linear Velocity, *CLV*, where the platter rotates at different speeds depending on the relative position of the track. This allows more data to be stored on the outer tracks than the inner tracks, but because it takes more delay to vary the speed of rotation, the technique is better suited to sequential rather than random access.

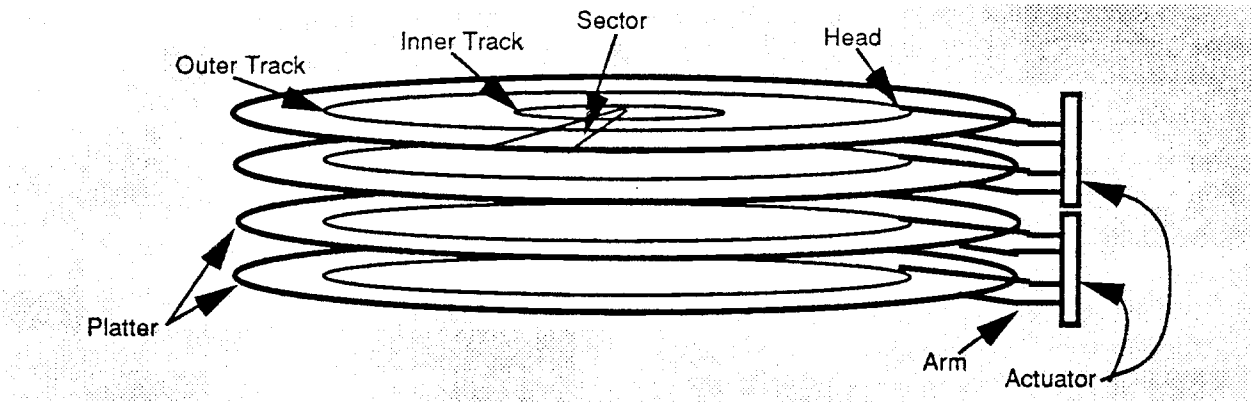


Figure 2.1: Disk Terminology.
Heads reside on arms which are positioned by actuators. Tracks are concentric rings on platters. A sector is the basic unit of read/write. A cylinder is a stack of tracks at one actuator position. An HDA is everything in the figure plus the airtight casing. In some devices it is possible to transfer from multiple surfaces simultaneously. The collection of heads that participate in a single logical transfer that is spread over multiple surfaces is called a head group.

that is, the disk equivalent of a magnetic drum - or multiple arms per surface through multiple actuators. Due to costs and technical limitations, it is usually uneconomical to build a device with a large number of actuators and heads (see Section 4).

A *cylinder* is a stack of tracks at one actuator position. A *Head Disk Assembly* (HDA) is the collection of platters, heads, arms, and actuators, plus the air-tight casing. A *drive* is an HDA plus all associated electronics. A *disk* might be a platter, an actuator, or a drive depending the context.

We can illustrate these concepts by describing two first generation supercomputer disks, the Cray DD-19 and the CDC 819 [13]. These were state-of-the-art disks around 1980. Each disk has 40 recording surfaces (20 platters), 411 cylinders, and 18 (DD-19) or 20 (CDC 819) 512 byte sectors per track. Both disks possess a limited "parallel read-out" capability. A given data word is actually byte interleaved over four surfaces. Rather than a single set of read/write electronics for the actuator, these disks have four sets, so it is possible to read or write with four heads at a time. Four heads on adjacent arms are called a *head group*. A disk track is thus composed of the stacked recording tracks of four adjacent surfaces, and there are 10 tracks per cylinder, spread over forty surfaces. The advances over the last decade can be illustrated by the Cray DD-49, which is a typical high end supercomputer disk of today. It consists of 16 recording surfaces (9 platters), 886 cylinders, 42 4096 byte sectors per track, with 32 read/write heads organized into eight head groups, four groups on each of two independent actuators. Each actuator can sweep the entire range of tracks, and by "scheduling" the arms to position the actuator closest to the target track of the pending request, the average seek time can be reduced. The DD-49 has a capacity of 1.2 Gigabytes of storage, and can transfer at a sustained rate of 9.6 Megabytes per second.

A variety of standard and proprietary interfaces are defined for transferring the data recorded on the disk to or from the host. We concentrate on industry standards here. On the disk surface, information is represented as alternating polarities of magnetic fields. These signals need to be sensed, amplified, and decoded into synchronized pulses by the read electronics. For example, the pulse-level protocol ST506/412 standard describes the way pulses can be extracted from the alternating flux fields. The bit-level ESDI, SMD, and IPI-2 standards describe the bit encoding of signals. At the packet-level, these bits must be aligned into bytes, error correcting codes need to be applied, and the extracted data must be delivered to the host. These "intelligent" standards include SCSI (Small Computer Standard Interface) and IPI-3.

The ST506 is a low cost but primitive interface, most appropriate for interfacing floppy disks to personal computers and low end workstations. For example, the controller must perform data separation on its own; this is not done for it by the disk device. As a result, its transfer rate is limited to .625 Megabytes/second. The SMD interface is higher performance, and is used extensively in connecting disks to mainframe disk controllers. ESDI is similar, but geared more towards smaller disk systems. One of its innovations over the ST506 is its ability to specify a seek to a particular track number rather than requiring track positioning via step-by-step pulses. Its performance is in the range

of 1.25 - 1.875 Megabytes/second. SCSI has so far been used primarily with workstations and minicomputers, but offers the highest degree of integration and intelligence. Implementations with performance at the level of 1.5 - 4 Megabytes/second are common. The newer IPI-3 standard has the advantages of SCSI, but provides even higher performance at a higher cost. It is beginning to make inroads into mainframe systems. However, because of the very wide spread use of SCSI, many believe that SCSI-2, an extension of SCSI to wider signal paths, will become the de facto standard for high performance small disks.

The connection pathway between the host and the disk device varies widely depending on the desired level of performance. A low-end workstation or personal computer would use a SCSI interface to directly connect the device to the host. A higher end file server or minicomputer would typically use a separate disk controller to manage several devices at the same time. These devices attach to the controller through SMD interfaces. It is the controller's responsibility to implement error checking and corrections and direct memory transfer to the host.

Mainframes tend to have more devices and more complex interconnection schemes to access them. In IBM terminology [14], the *channel path*, i.e., the set of cables and associated electronics that transfer data and control information between an I/O device and main memory, consists of a *channel*, a *storage director*, and a *head of string* (see Figure 2.2). The collection of disks that share the same pathway to the head of string is called a *string*.

In earlier IBM systems, a channel path and channel are essentially the same thing. The channel processor is the hardware that executes channel programs, which are fetched from the host's memory. A *subchannel* is the execution environment of a channel program, similar to a process on a conventional CPU. Formerly, a subchannel was statically assigned for execution to a particular channel, but a major innovation in high-end IBM systems (308X and 3090) allows subchannels to be dynamically switched among channel paths. This is like allocating a process to a new processor within a multiprocessor system every time it is rescheduled for execution.

I/O program control statements, e.g., *transfer in channel*, are interpreted by the channel, while the storage director (also known as the *device controller* or *control unit*) handles seek and data transfer requests. Besides these control functions, it may also perform certain datapath functions, such as error detection/correction and mapping between serial and parallel data. In response to requests from the storage director, the device will position the access mechanism, select the appropriate head, and perform the read or write. If the storage director is simply a control unit, then the datapath functions will be handled by the head of string (also known as a *string controller*).

To minimize the latency caused by copying into and out of buffers, the IBM I/O system uses little buffering between the device and memory[2]. In a high performance envi-

2. Only the most recent generation of storage directors (e.g., IBM 3880, 3990) incorporate disk caches, but care must be taken to avoid cache management related delays [15].

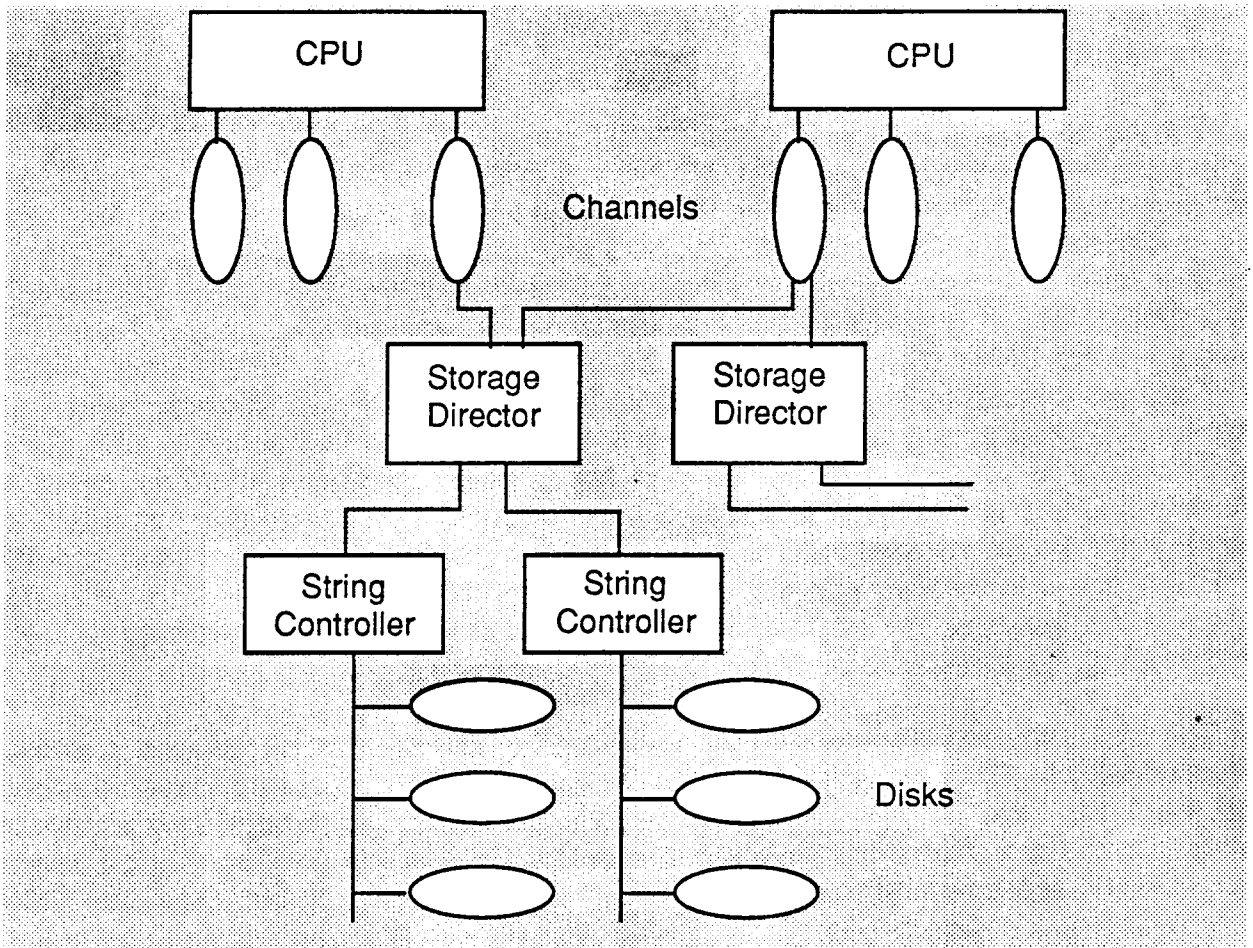


Figure 2.2: Host-to-Device Pathways.
For large IBM mainframes, the connection between host and device must pass thru a channel, storage director, and string controller. Note that multiple storage directors can be attached to a channel, multiple string controllers per storage director, and multiple devices per string controller. This multipathing approach makes it possible to share devices among hosts and to provide alternative pathways to better utilize the drives and controllers. While logically correct, the figure does not reflect the true physical components of high-end (308X, 3090) IBM systems. The concept of channel has disappeared from these systems and has been replaced by a channel path.

ronment, devices spend a good deal of time waiting for the pathway's resources to become free. These resources are used for time periods related to disk transfer speeds, measured in milliseconds. One possible method for improving utilization is to support disconnect/reconnect. A subchannel can connect to a device, issue a seek, disconnect to free the channel path for other requests, and reconnect later to perform the transfer when the seek is completed. Unfortunately, not all reconnects can be serviced immediately, because the control units are busy servicing other devices. These *RPS misses* (to be described in more detail in Section 3) are a major source of delay in heavily utilized IBM storage subsystems [16]. Performance can be further improved by providing multiple paths between memory and devices. To this purpose, IBM's high-end systems support *dynamic path reconnect*, a mechanism that allows a subchannel to change its channel path each time it cycles through a disconnect/reconnect with a given device.

Rather than wait for its currently allocated path to become free, it can be assigned to another available path.

Turning to supercomputer I/O systems, we now examine the I/O architecture of the Cray machines. Because the CRAY I/O System (IOS) varies from model to model, the following discussion concentrates on the IOS found on the CRAY X-MP and Y-MP [17]. In general, the IOS consists of two to four I/O processors, each with its own local memory and sharing a common buffer memory with the other IOPs. The IOP is designed to be a simple, fast machine for controlling data transfers between devices and the central memory of the CRAY main processors. Since it executes the control statements of an I/O program, it is not unlike the IBM Channel Processor in terms of its functionality, except that IO programs reside in its local memory rather than in the host's. An IOP's local memory is connected through a high speed communications interface, called a *channel* in CRAY terminology, to a Disk Control Unit (DCU). A given port into the local memory can be time multiplexed among multiple channels. Data is transferred back and forth between devices and the main processors through the IOP's local memory, which is interfaced to central memory through a 100 Megabyte/second channel pair (one pathway for each direction of transfer).

The DCU provides the interface between the IOP and the disk drives, and is similar in functionality to IBM's storage director. It oversees the data transfers between devices and the IOP's local memory, provides speed matching buffer storage, and transmits control signals and status information between the IOP and the devices. Disk Storage Units (DSUs) are attached to the DCU through point-to-point connections. The DSU contains the disk device and is responsible for dealing with its own defect management, by using a technique called sector slipping. Figure 2.3 summarizes the elements of the CRAY I/O System.

Digital Equipment Corporation's high-end I/O strategy is described in terms of the Digital Storage Architecture (DSA), and is embodied in system configurations such as the VAXcluster shared disk system (see Figure 2.4). The architecture provides a rigorous definition of how storage subsystems and host computers interact. It achieves this by defining a client/server message-based model for I/O interaction based on device independent interfaces [18; 19]. A *Mass Storage Subsystem* is viewed at the architectural level as consisting of logical block machines capable of storing and retrieving fixed blocks of data, i.e., the I/O system supports the transfer of logical blocks between CPUs and devices given a logical block number. From the viewpoint of physical components, a subsystem consists of *controllers* which connect computers to *drives*.

The software architecture is divided into four levels: the *Operating System Client* (also called the Class Driver), the *Class Server* (Controller), the *Device Client* (Data Controller), and the *Device Server* (Device). The Disk Class Driver, resident on a host CPU, accepts requests for disk I/O service from applications, packages these requests into messages, and transmits them via a communications interface (such as the *Computer Interconnect* port driver) to the Disk Class Server resident within a controller in the I/O subsystem. The command set supported by the Class Server includes such

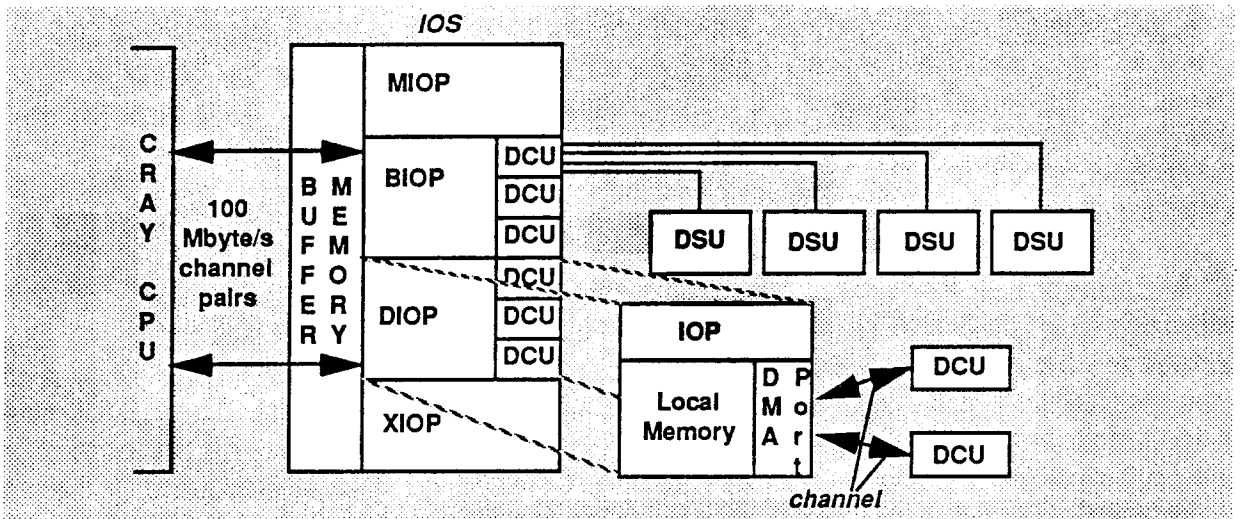


FIGURE 2.3: Elements of the CRAY I/O System for the Y-MP

An IOS contains up to four IOPs. The MIOP connects to the operator workstation and performs mainly maintenance functions. The XIOP supports block multiplexing, and is most appropriate for controlling relatively slow speed devices, such as tapes. The BIOP and DIOP are designed for controlling high speed devices like disks. Up to four disk storage units (DSUs) can be attached through the disk control unit (DCU) to the IOP. Three DCUs can be connected to each of the BIOP and DIOP, leading to a total of 24 disks per IOS. The Y-MP can be configured with two IOSs, for a system total of 48 devices.

relatively device independent operations as: read logical block, write logical block, bring on-line, and request status. The Disk Class Server[3] interprets the transmitted commands, handles the scheduling of command execution, tracks their progress, and reports status back to the Class Driver. Note the absence of seek or select head commands. This interface can be used equally well for solid state disks as for conventional

3. Other kinds of class servers are also supported, such as for tape drives.

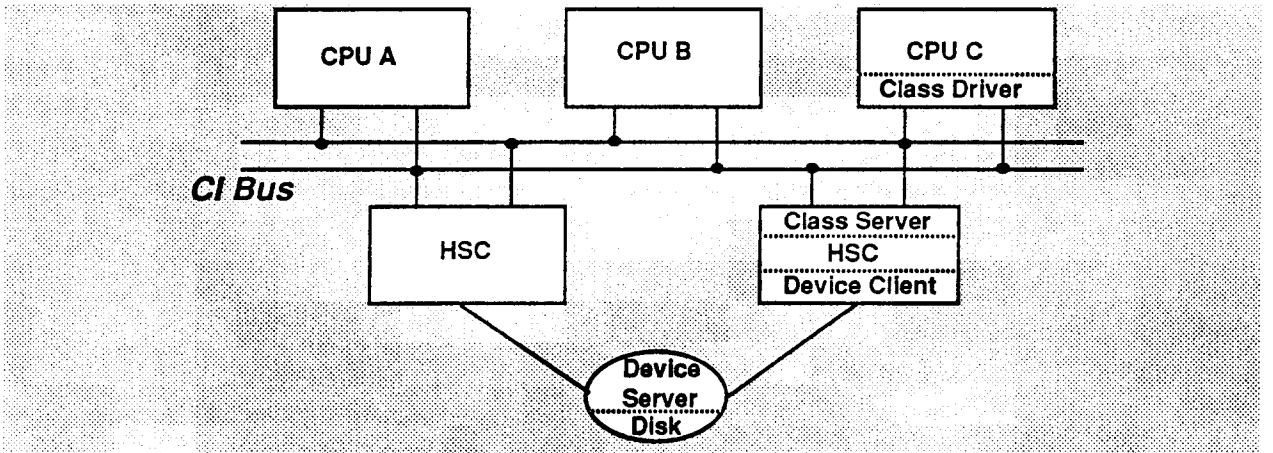


FIGURE 2.4: VAXCluster Architecture.

CPUs are connected to HSCs (hierarchical storage controllers) through a dual CI (Computer Interconnect) bus. Thirty-one hosts and thirty-one HSCs can be connected to a CI. Up to 32 disks can be connected to an HSC-70.

magnetic disks. Device specific commands are issued at a lower level of the architecture, i.e., between the Device Client (i.e., disk controller) and Device Server (i.e., disk device). The former provides the path for moving commands and data between hosts and drives, and is usually realized physically by a piece of hardware that corresponds to the device controller. The latter coincides with the physical drives used for storing and retrieving data.

It is interesting to contrast these proprietary approaches with an industry standard approach like SCSI, admittedly targeted for the low to mid range of performance. SCSI defines the logical and physical interface between a host bus adapter (HBA) and a disk controller, usually embedded within the assembly of the disk device. The HBA accepts I/O requests from the host, initiates I/O actions by communicating with the controllers, and performs direct memory access transfers between its own buffers and the memory of the host. Requesters of service are called initiators, while providers of service are called targets. Up to eight nodes can reside on a single SCSI string, sharing a common pathway to the HBA. The embedded controller performs device handling and error recovery. Physically, the interface is implemented with a single daisy chained cable, and the 8 bit datapath is used to communicate control and status information, as well as data. SCSI defines a layered communications protocol, including a message layer for protocol and status, and a command/status layer for target operation execution. The HBA roughly corresponds to the function of the IBM Channel Processor or Cray IOP, while the embedded controller is similar to the IBM storage director/string controller or the Cray DCU. Despite the differences in terminology, the systems we have surveyed exhibut significant commonality of function and similar approaches for partitioning these functions among hardware components.

3. Characterization of I/O Workloads

Before characterizing the I/O behavior of different workloads, it is necessary to first understand the elements of disk performance. Disk performance is a function of the service time, which consists of three main components: *seek time*, *rotational latency*, and *data transfer time*.[4] *Seek time* is the time needed to position the heads to the appropriate track position containing the desired data. It is a function of a substantial initial start-up cost to accelerate the disk head (on the order of 6 ms) as well as the number of tracks that must be traversed. Typical average seek times, i.e., the time to traverse between two randomly selected tracks (approximately 28% of the data band), are in the range of 10 to 20 ms. The track-to-track seek time is usually below 10 ms, and as low as 2 ms.

The second component of service time is *rotational latency*. It takes some time for the desired sector to rotate under the head position before it can be read or written. Today's devices spin at a rate of approximately 3600 rpm, or 60 revolutions per second (we expect to see rotation speeds increase to 5400 rpm in the near future). For today's disks, a full revolution is 16 ms, and the average latency is 8 ms. Note that the worse case latencies are comparable to average seeks.

The last component is the *transfer time*, i.e., the time to physically transfer the bytes from disk to the host. While the transfer time is a strong function of the number of bytes to be transferred, seek and rotational latencies times are independent of the transfer blocksize. If data is to be read or written in large chunks, it makes sense to choose a large blocksize, since the "fixed cost" of seek and latency are better amortized across a large data transfer.

A low performance I/O system might dedicate the pathway between the host and the disk for the entire duration of the seek, rotate, and transfer times. Assuming small blocksizes, transfer time is a small component of the overall service time, and these pathways can be better utilized if they are shared among multiple devices. Thus, higher performance systems support independent seeks, in which a device can be directed to detach itself from the pathway while seeking to the desired track (recall the discussion of dynamic path reconnect in the previous section). The advantage is that multiple seeks can be overlapped, reducing overall I/O latency and better utilizing the available I/O bandwidth.

However, to make it possible for devices to reattach to the pathway, the I/O system must support a mechanism called *rotational position sensing*, i.e., the device interrupts the I/O controller when the desired sector is under the heads . If the pathway is currently in use, the device must pay a full rotational delay before it can again attempt to transfer. These rotational positional reconnect miss delays (RPS delays) represent a major

4. In a heavily utilized system, delays waiting for a device can match actual disk service times, which in reality is composed of device queuing, controller overhead, seek, rotational latency, reconnect misses, error retries, and data transfer.

source of degradation in many existing I/O systems [16]. This arises from the lack of device buffering and the real-time service requirements of magnetic disks. At the time that these architectures were established, buffer memories were expensive and the demands for high I/O performance were less pressing with slower speed CPUs. An alternative, made more attractive by today's relative costs of electronic and mechanical components, is to associate a *track buffer* with the device that can be filled immediately. This can then be used as the source of the transfer when the pathway becomes available [20].

I/O intensive applications vary widely in the demand they place on the I/O system. They run the gamut from processing small numbers of bulk I/Os that must be handled with minimum delay (supercomputer I/O) to large numbers of simple tasks that touch small amounts of data (transaction processing). An important design challenge is to develop an I/O system that can handle the performance needs of these diverse workloads.

A given workload's demand for I/O service can be specified in terms of three metrics: *throughput*, *latency*, and *bandwidth*. Throughput refers to the number of requests for service made per unit time. Latency measures with how long it takes to service an individual request. Bandwidth gauges the amount of data flowing between service requesters (i.e., applications) and service providers (i.e., devices).

As observed by Bucher and Hayes [13], supercomputer I/O can be characterized almost entirely by sequential I/O. Typically, computation parameters are moved in bulk from disk to in-memory data structures, and results are periodically written back to disk. These workloads demand large bandwidth and minimum latency, but are characterized by low throughput. Contrast this with transaction processing, which is characterized by enormous numbers of random accesses, relatively small units of work, and a demand for moderate latency with very high throughput.

Figure 3.1 shows another way of thinking about the varying demands of I/O intensive applications. It shows the percent of time different applications spend in the three components of I/O service time. Transaction processing systems spend the majority of their service time in seek and rotational latency, thus technological advances which reduce the transfer time will not affect their performance very much. On the other hand, scientific applications spend a more equal amount of time in seek and data transfer, and their performance is sensitive to any improvement in disk technology. As we shall see in Section 5, it is possible to organize an array of disks in such a fashion as to provide good I/O performance for these widely varying workloads.

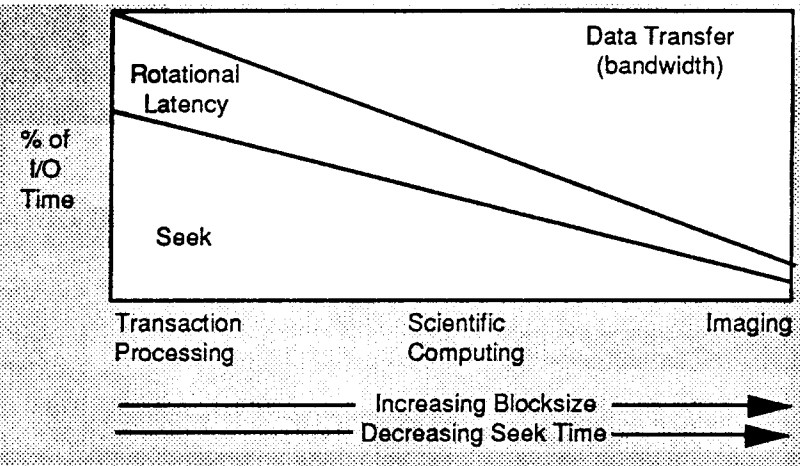


Figure 3.1: I/O System Parameters as a Function of Application

Transaction processing applications are seek and rotational latency limited, since only small blocks are usually transferred from disk. Image processing applications, on the other hand, transfer huge-blocks, and thus spend most of their I/O time in data transfer. Scientific computing applications tend to fall in between. This figure is from [21].

4. Architectures for Attaining Higher Performance I/O

4.1. Extensions to Conventional Disk Architectures

In this subsection, we will focus on techniques for improving the performance of conventional disk systems, i.e., methods which allow us to reduce the seek time, rotational latency, or transfer time of conventional disks. By reducing disk service times, we also decrease device queuing delays. These techniques include fixed head disks, parallel transfer disks, increased disk density, solid state disks, disk caches, and disk scheduling.

4.1.1. Fixed Head Disk

The concept of a fixed head disk is to place a read/write head at every track position. The need for positioning the heads is eliminated, thus eliminating the seek time altogether. The approach does not assist in reducing rotational latencies, nor does it lessen the transfer time.

Fixed head disks were often used in the early days of computing systems as a back-end store for virtual memory. However, since modern disks have hundreds of tracks per surface, placing a head at every position is no longer viewed as an economical solution.

4.1.2. Parallel Transfer Disks

Some high performance disk drives make it possible to read or write from multiple disk surfaces at the same time. For example, the Cray DD-19 and DD-49 disks described in Section 2 have a parallel transfer capability. The advantage is that much higher transfer rates can be achieved, but no assistance is provided for seek or rotational latency. Thus transfer units are correspondingly larger in these systems.

A number of economic and technological issues limit the usefulness of parallel transfer disks. From the economic perspective, providing more than one set of read/write electronics *per actuator* is expensive. Further, current disks use sophisticated control systems to lock on to an individual track, and it is difficult to do this simultaneously across tracks within the same cylinder. Hence, the CRAY strategy of limiting head groups to only four surfaces. There appears to be a fundamental tradeoff between track density and the number of platters: as the track density increases, it becomes ever more difficult to lock on to tracks across many platters, and the number of surfaces that can participate in a parallel transfer is reduced. For example, current CRAY track densities are around 980 tracks/inch, and require a rather sophisticated closed-loop track following servo system to position the heads accurately with finely controlled voice coil actuators. A lower cost ($/MB) high performance disk system can be constructed from several standard drives than from a single parallel transfer device

(see Section 4.2), in part because of the relatively small sales volume of parallel transfer devices compared to standard drives.

4.1.3. Increasing Disk Density

As described in Section 1, the improvements in disk recording density are likely to continue. Higher bit densities are achieved through a combination of the use of thinner films on the disk platters (e.g., densities improve from 16000 bpi to 21000 bpi when thick iron oxide is replaced with thin film materials), smaller gaps between the poles of the read/write head's electromagnet, and heads which fly closer to the disk surface.

While vertical recording techniques have long been touted as the technology of the future, advances in head technology make it possible to continue using conventional horizontal methods, but still keep disks on the M.A.D. curve. These *magneto-resistive heads* employ non-inductive methods for reading, which work well with dense horizontal recording fields. However, a more conventional head is needed for writing, but this dual head organization permits separate optimizations for read and write.

Also, the choice of coding technique can have a significant effect on density. Standard modified frequency modulation techniques require approximately one flux change per bit, while more advanced run-length limited codes can increase density by additional factor of 50%. Densities as high as 31429 bpi can be attained with these techniques. As the recording densities increase, the transfer times decrease, as more bits transit beneath the heads per unit time. Of course, this approach provides no improvement in seek and latency times. And most of the increase in density comes from increases in the number of tracks per inch, which does not improve (and may actually reduce) performance.

Although increased densities are inevitable, the problem is primarily economic. Increasing the tracks per inch may make seeks slower as it becomes more time consuming for the heads to correctly "lock" onto the appropriate track. The sensing electronics get more complex and thus more expensive. Once again, it can be argued that higher capacity can be achieved at lower cost by using several smaller disks rather than one expensive "high density" disk.

4.1.4. Solid State Disks

Solid state disks (SSD), constructed from relatively slow memory chips, can be viewed either as a kind of large, slow main memory or as a small, high speed disk. When viewed as large main memory, the SSD is often called Expanded Storage (ES). The expanded storage found in the IBM 3090 class machines [14] supports operations for paging data blocks from and to main memory. Usually, the expanded storage looks to the system more like memory than an I/O device: it is directly attached to main memory through a high speed bus rather than an I/O controller. The maximum transfer bandwidth on the IBM 3090 between expanded store and memory is two orders of magnitude faster than conventional devices: approximately 216 megabytes/second - one

word each 18.5 ns!

Further, unlike conventional devices, a transfer between memory and expanded storage is performed synchronously with the CPU. This is viewed as acceptable, because the transfer requires so little time and does not involve the usual operating system overheads of I/O set-up and interrupts. Note that to transfer data from ES to disk requires the data to be first staged into main memory.

The CRAY X-MP and Y-MP also support solid state disks, which can come in configurations of up to 4096 Megabytes, approximately four times the capacity of the DD-49. The SSD has the potential for enormous bandwidth. It can be attached to the Cray IO System or directly to the CPU through up to two 1000 Megabyte/second channels. Access can be arranged in one of three ways [22]. The first alternative is to treat the SSD as a logical disk, with users responsible for staging heavily accessed files to it. Unfortunately, this leads to the inevitable contention for SSD space. Further, the operating system's disk device drivers are not tuned for the special capabilities of SSDs, and some performance is lost. The second alternative is use the SSD as an extended memory, in much the same manner as IBM's extended storage. Special system calls for accessing the SSD bypass the usual disk handling code, and a 4096 byte sector can be accessed in 25 microseconds. The last alternative is to use the SSD as a logical device cache, i.e., as a second-level cache for multi-track chunks of files that resides between the system's in-main memory file cache and the physical disk devices. Cray engineers have observed workload speedups for their UNIX-like operating system of a factor of 4 over conventional disk when the cache is enabled. These results indicate that solid state disks are most appropriate for containing "hot spot" data. Conventional wisdom has it that 20% of the data receives 80% of the accesses, and this has been widely observed in transaction processing systems [23].

If solid state disks are to be used to replace magnetic disks, then they must be made non-volatile, and herein lies their greatest weakness. This can be achieved through battery back-up, but the technique is controversial. First, it is difficult to verify that the batteries will be fully charged when needed, i.e., when conventional power fails. Second, it is difficult to determine how long is long enough when powering the SSD with batteries. This should probably be long enough to off-load the disk's contents to magnetic media. Fortunately, low power DRAM and wafer scale integration technology are making feasible longer battery hold times.

Another weakness is their cost. At the present time, there is more than a ten to twenty times difference in price between the cost of a megabyte of magnetic disk memory and a megabyte of DRAM. While wafer scale integration may bring this price down in the future, for the near term solid state disk will be limited to a staging or caching function.

4.1.5. Disk Caches

Disk caches place buffer memories between the host and the device. If disk data is

likely to be re-referenced, caches can be effective in eliminating the seek and rotational latencies. Unfortunately, this effectiveness depends critically on the access behavior of the applications. Truly random access with little re-referencing cannot make effective use of disk caches. However, applications that exhibit a large degree of sequential access can use a cache to good purpose, because data can be staged into the cache before it is actually requested.

Disk caches can become even more useful if they are made non-volatile using the battery back-up techniques described in the previous subsection (and with the same potential problems). A non-volatile cache will allow "fast writes": the application need not wait for the write I/O to actually complete before it is notified that it has completed. For some applications environments, disk caches have the beneficial effect of reducing the number of reads, and thus the number of I/O requests seen by the disks. This has the interesting side effect of increasing the percentage of writes found in the I/O mix, and some observers believe that writes may dominate I/O performance in future systems.

As we have already mentioned, a disk cache can also lead to better utilization of the host to device pathways. A device can transfer data into a cache even if the pathway is in use by another device on the same string. Thus caches are effective in avoiding rotational position sensing misses.

4.1.6. Disk Scheduling

The mechanical delays as seen by a set of simultaneous I/O requests can be reduced through effective disk scheduling. For example, seek times can be reduced if a *shortest seek time first* scheduling algorithm is used [24]. That is, among the queue of pending I/O requests, the one next selected for service is the one that requires the shortest seek time from the current location of the read/write heads.

The literature on disk scheduling algorithms is vast, and the effectiveness of a particular scheduling approach depends critically on the workload. It has been observed that scheduling algorithms work best when there are long queues of pending requests, unfortunately, this situation seems to occur rarely in existing systems [24].

4.2. Disk Arrays

An alternative to the approaches just described is to exploit parallelism by grouping together a number of physical disks and making these appear to applications as a single logical disk. This has the advantage that the bandwidth of several disks can be harnessed to service a single logical I/O request or can support multiple independent I/Os in parallel. Further, arrays can be constructed using existing, widely available disk technology, rather than the more specialized and more expensive approaches described in the previous subsection. For example, Cray offers a device called the DS-40 which appears as a single logical disk device, but which is actually implemented internally as four drives. A logical track is constructed from sectors across the four disks. The DS-40 can transfer at a peak rate of 20 Megabytes per second, with a sustained transfer rate of 9.6

Megabytes/second, and thus is strictly faster than the DD-49. We will elaborate on ways of organizing disk arrays in the subsections below.

4.2.1. Disk Array Taxonomy

Disk arrays are a relatively new concept, and the field still suffers from some confusing terminology. For example, Kim describes disk array organizations in terms of *synchronous and asynchronous interleaving* [25; 26], while Salem and Garcia-Molina describe their approach as *disk striping* [27]. Livny writes about *declustering* [28] while Bitton and Gray talk about *disk shadowing* [29]. Copeland and Keller introduce the additional terms of *mirrored declustering* and *interleaved declustering* [30]. In interleaving, each data block is divided into portions, and succeeding portions of a block are stored on successive disks. This is a well-known technique, implemented by several supercomputer operating systems and frequently called striping. The goal of interleaving is to reduce the latency for accessing a single block, by partitioning it into pieces and taking advantage of the parallel transfer capabilities of the disk array. Declustering, on the other hand, spreads the blocks of a file across multiple disks. Its goal is to support simultaneous block I/Os to the same file, including an ability to perform multiblock I/Os in parallel. The objective here is to improve I/O throughput rather than latency. Shadowing has the same goal, but achieves it by spreading file copies around the disks rather than interleaving at the granularity of transfer blocks. Mirrored declustering is shadowing with exactly one additional copy per file allocated to a single back-up "mirror" disk. Interleaved declustering is a generalization that still creates a single back-up copy, but partitions it and interleaves the pieces across multiple disks. Note that shadowing, mirrored declustering, and interleaved declustering are as much concerned with keeping data available in the event of disk crashes as they are concerned with improving read performance, at some additional costs to writes.

A general way to understand the proposed disk array organizations is to consider the following three orthogonal issues:

1. *Degree of Interleaving:* data can be interleaved among the elements of the array by bit, byte, block, track, or cylinder.

2. *Arm Independence:* the actuators of the elements of the array can seek as unit or can be positioned to tracks independently.

3. *Rotation Independence:* either the disk spindles rotate together, so that the same sector number is moving beneath the heads for all disks, or the spindles rotate independently.

Figure 4.1 classifies the proposed organizations in terms of arm and rotation independence. The degree of interleaving is the third dimension. For example, it is possible to conceive of an array with synchronized rotation and arms that move as a unit where the degree of interleaving is either the bit, the byte, or the block. In the following subsections, we will examine the alternative disk array organizations more closely, based on

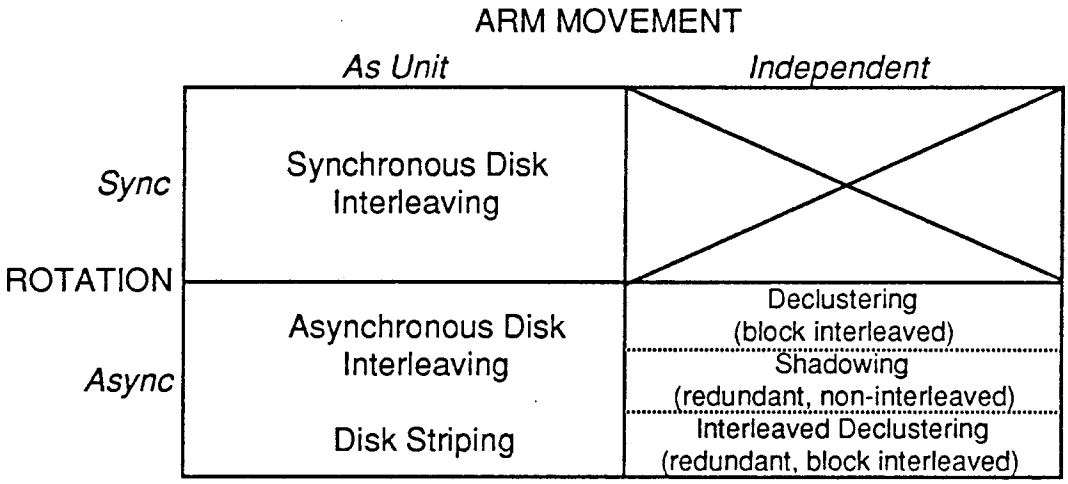


Figure 4.1: Disk Array Taxonomy
The figure shows the proposed disk array organizations characterized by arm movement and rotation synchronization. The arms can move as unit, to service a single logical I/O to the entire array, or can be positioned independently to service multiple logical I/Os in-parallel. The platter rotation can be synchronized, so that the same sector number passes under all heads within the array at the same time, or the platters can rotate independently. To our knowledge, an organization that combines synchronized rotation and independent arm movement has not been proposed.

the taxonomy of Figure 4.1.

4.2.2. Synchronous Rotation/Unit Arms

The basic idea behind the synchronous rotation/unit arms organization is to make the array appear to the host as a single logical disk, but of greater capacity and greater transfer bandwidth than a single physical disk. A logical disk block is "interleaved" among multiple physical disks [25; 27] in a manner similar to memory system interleaving, where logically adjacent memory blocks are placed in physically different memory banks. For example, consider an **M** byte transfer block that will be spread across **N** physical disks. The logical block will be decomposed into **N** X **M/N** byte physical blocks that will be placed on each of the underlying physical disks.

Since the array appears as a single logical disk, a logical seek is replaced by **N** physical seeks. Disk scheduling is simplified since only a single I/O queue needs to be maintained. Thus the transfer time for a logical I/O is reduced by a factor of **N**, assuming that the controllers can keep pace with the transferring disks, but the seek and rotational delays are unaffected. This approach has the additional advantage of achieving more uniform disk utilization [25]. The interleaved approach can spread the 20% of the data that receives 80% of the accesses more uniformly over the devices, thereby improving latency and reducing bottlenecks.

A potential problem is that a single failed disk may render the data in the array inaccessible. We will address these fault tolerance issues in more depth in Section 5, but the problem is intrinsic to any organization with many components that takes a file and

spreads it across multiple devices. Synchronized organizations have the additional reliability problem of having a single point of failure in terms of the synchronization clock.

One more problem, intrinsic to the synchronized approach, is how to handle bad sectors. Logical sectors (or tracks) sometimes go "bad", i.e., after a while it is not possible to read their contents without error (fortunately, the ability to reread the bad data coupled with error correcting codes makes this problem non-fatal). The usual approach is to remap these bad areas onto another portion of the disk's surface that is reserved for this purpose. Thus, a remapped area could cause havoc with the movement of the disk arms, as they move to one track position to access data from most of the disks and then reposition themselves to access the remapped data. Hence the CRAY disks use a technique called *sector slipping* to maintain the physical sequence of the sectors on the track. Rather than remap to a spare sector at the end of the track, the the track contents after the bad sector, including the reconstructed contents of the bad sector, are shifted one sector position along the track. Fortunately, most bad areas are detected during the "burn-in" period where the disks are manufactured. For arrayed disks, it may be necessary to mark as bad the union of those areas detected to be bad on any individual disk.

4.2.3. Asynchronous Rotation/Unit Arms

This organization is similar to the above case, but does not require that the rotation of the physical disks be synchronized. It retains all of the advantages of the previous organization, i.e., the simplicity of the single I/O request queue and an effective increase in I/O bandwidth per operation, but eliminates the single point of failure represented by the synchronization clock and the bad sector problem. Such arrays are simpler to construct from readily available devices, since special "synchronized" disk spindles are not necessary.

Once again, the actual achievable parallelism is limited by the amount of bandwidth the I/O controller can handle. Since the disk rotations are unsynchronized, Kim and Tantawi have also observed that the performance of the array will tend toward the performance of the slowest disk [26]. In other words, rotational latency for the array will tend towards the worst case rather than the average case.

4.2.4. Asynchronous Rotation/Independent Arms/Block Interleaved

In the previous approaches, a logical block was partitioned and spread among multiple underlying physical disks. Declustering [28] is a form of this kind of organization that assigns a logical block to a single physical device. However, logically adjacent blocks are placed on separate physical disks. In essence, declustering assumes block level interleaving. It must be performed in conjunction with the file system, to spread blocks of a single logical file across multiple physical drives.

Unlike the previous interleaved approaches, declustering can support the concurrent operation of multiblock I/Os as well as independent I/Os. Thus, a large sequential

access can be broken up into multiple parallel I/Os that are scheduled independently. The problem is the additional complexity of multiple request queues, and the fault tolerance issues of spreading files among multiple devices.

4.2.5. Asynchronous Rotation/Independent Arms/Redundant, Non-interleaved

Disk shadowing is an alternative form of asynchronous rotation with independent arm positioning. Its primary motivation is fault tolerance: a disk image is simultaneously maintained on more than one physical disk. If a disk crashes, its contents can be obtained from one of its shadows. In addition, shadowing can supporting multiple in-parallel I/Os to the same file by spreading copies of it to disks within the array. In effect, this increases the read bandwidth if the controller can support multiple independent reads to the shadow set, and results in shorter expected seek times for reads [29]. On the other hand, write bandwidth is sacrificed, since every logical write is really a set of physical writes, and full data redundancy represents a significant penalty in capacity.

4.2.6. Asynchronous Rotation/Independent Arms/Redundant, Block Interleaved

Interleaved declustering [30] combines aspects of declustering with disk shadowing. As in disk shadowing, the primary goal is availability: a single copy is made and interleaved across multiple disks. And just as in shadowing, parallel I/Os to the same file can be supported. If the partitioned copy is broken up and interleaved by blocks, then it can provide comparable advantages for large sequential reads as the declustered case. While avoiding the fault tolerance problems of declustering, it suffers the same degraded write performance as disk shadowing.

5. Redundant Arrays of Inexpensive Disks

In this section, we will review progress in harnessing the bandwidth of arrays of disks. While these techniques can be attempted with large format disks, a new opportunity is provided by constructing such arrays from very large numbers of the smaller format hard disks. It is now possible to construct I/O systems from hundreds or thousands of drives, at reasonable costs for capacity, volume, and power [31; 21].

5.1. Why Inexpensive Disks?

Over the course of the evolution of hard disk technology, from the RAMAC drive of 1956 to the Winchester drives of today, there has been a steady decrease in platter size. Mainframe disks appear in 14" and 10.5" formats, while minicomputer disks are available in 8" formats, and workstation and personal computer disks are 5.25" and 3.5". This has been driven primarily by the desire to use hard disk technology in a wider range of lower cost applications. The introduction of each succeedingly smaller format has brought with it a lower cost per megabyte, a higher number of units shipped, and a steeper curve of increasing storage capacity over time. As Hoagland has stated:

> "For many years IBM paced the evolution of both magnetic data storage technology and products as well as creating the market. Now ... the industry can be characterized as serving a commodity market with high volume and great price sensitivity, intense competition, higher product differentiation from increasing innovation, and finally the wide introduction of advanced technology where initially there was only the application of one common technology level, in this case 'Winchester' technology.
>
> Magnetic recording data storage thus not only has a level of R & D characteristic of a high technology, but also responds to a commodity market, more normally associated with a mature technology." [32]

Thus the economies of scale are such that some of the most exciting innovations in disk technology are happening in the smaller format disks. For example, consider Table 5.1, which compares a number of critical disk parameters for the IBM 3380, Fujitsu M2361A, and Conners CP3100 devices. Notice that the number of I/Os per second per actuator in the smaller format Conners device is within a factor of two of the the larger disks. In several of the remaining metrics, including price per megabyte, the Conners disk is superior or equal to the large disks. These trends are even more strongly indicated in Figures 5.1 and 5.2. The small size and low power are even more impressive given that devices like the Conners disk contain full track buffers and most of the functions of the traditional mainframe controller. These devices contain single-chip *embedded* SCSI controllers, made economically feasible by the very large volume of disks sold for the PC market.

The key to very high I/O performance is a large aggregate bandwidth from the devices. It is technologically and economically easiest to achieve these high bandwidths by using large numbers of small format disks. As with any approach that spreads data across multiple physical devices, the main drawback is fault tolerance. That is, the sys-

Characteristics	*IBM 3380*	*Fujitsu M2361A*	*Conners CP3100*	*3380 v. 3100*	*2361 v. 3100*
				(>1 means 3100 is better)	
Disk diameter (inches)	14	10.5	3.5	4	3
Formatted Data Capacity (MB)	7500	600	100	.01	.2
Price/MB (controller incl.)	$18-$10	$20-$17	$10-$7	1-2.5	1.7-3
MTTF Guaranteed (hours)	30,000	20,000	30,000	1	1.5
MTTF in practice (hours)	100,000	?	?	?	?
No. Actuators	4	1	1	.2	1
Max I/Os/Sec/Actuator	50	40	30	.6	.8
Typ I/Os/Sec/Actuator	30	24	20	.7	.8
Max I/Os/Sec/Box	200	40	30	.2	.8
Typ I/Os/Sec/Box	120	24	20	.2	.8
Transfer Rate (MB/sec)	3	2.5	1	.3	.4
Power/Box (W)	1650	640	10	165	64
Volume (cu. ft.)	24	3.4	.03	800	110

TABLE 5.1. *Comparison of IBM 3380 disk model AK4 for mainframe computers, the Fujitsu M2361A "Super Eagle" disk for minicomputers, and the Conners Peripherals CP 3100 disk for personal computers. By "Maximum I/Os per second" we mean the maximum number of average seeks and average rotates for a single sector access. Cost and reliability information on the 3380 come from widespread experience* [33] [23] *and the information on the Fujitsu from the manual* [34]*, while some numbers on the new CP 3100 are based on speculation. The price per megabyte is given as a range to allow for different prices for volume discount and different mark-up practices of the vendors.*

tem failure rate will be high with large numbers of disks. Throughout the rest of this section we will be dealing with array organizations that attempt to increase fault tolerance while maintaining both high bandwidth for reads and writes, and high utilization of the available disk capacity.

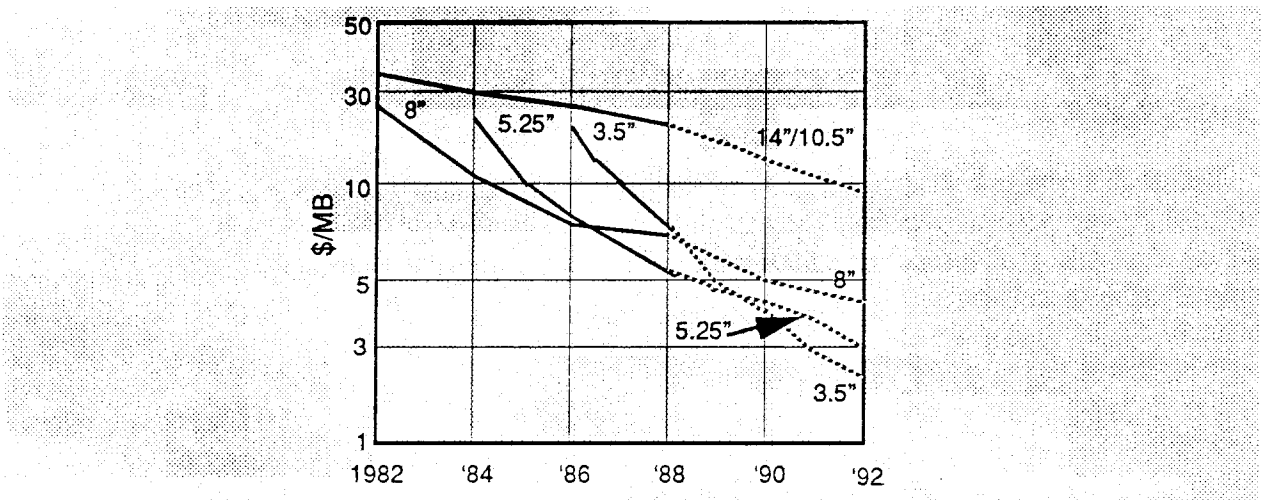
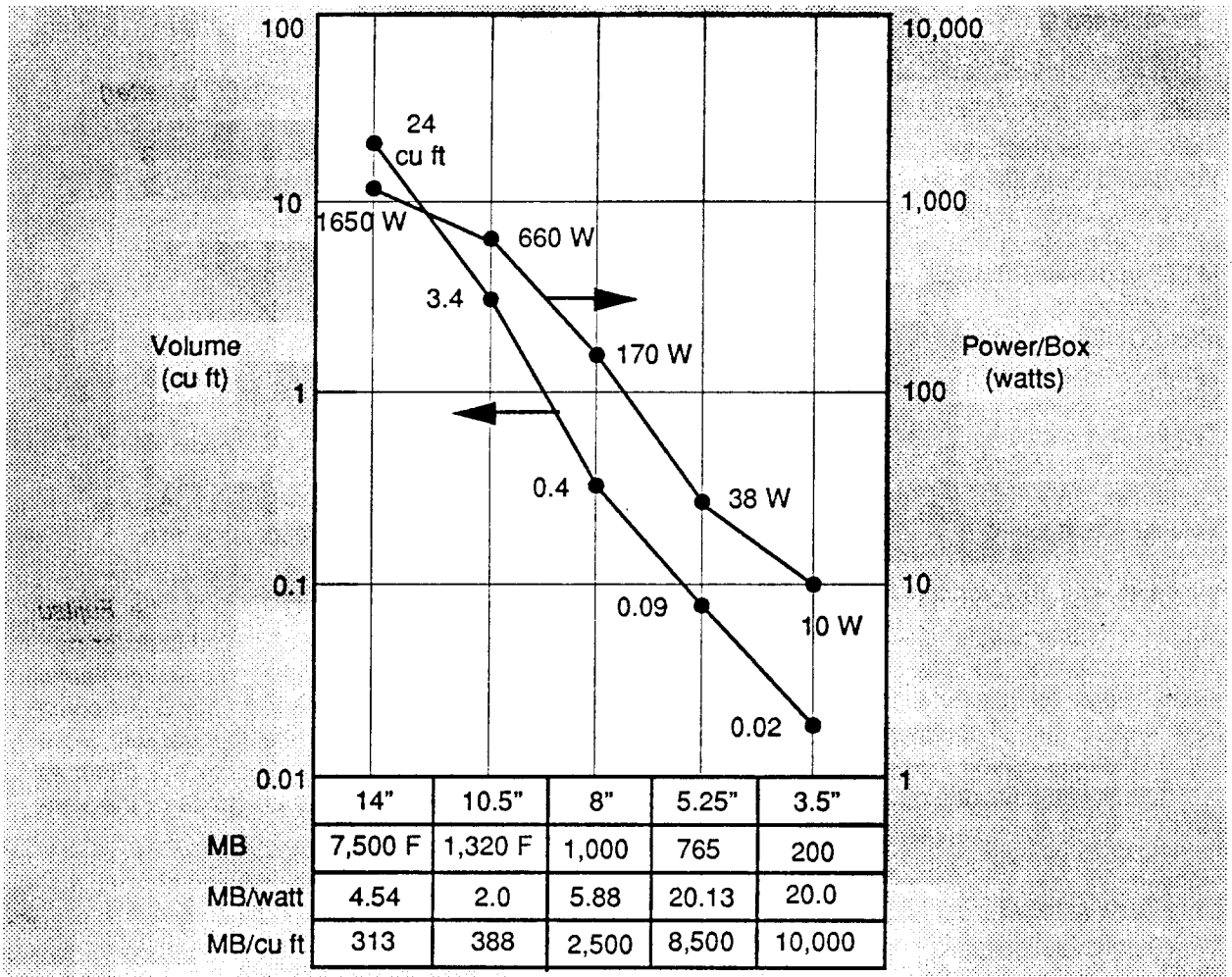


Figure 5.1: Disk Storage Prices
Small format disks are dropping more rapidly in price than larger format disks. In 1988, 5.25" are the price leader, but it is expected that by 1992, 3.5" devices will do even better [21].



	14"	10.5"	8"	5.25"	3.5"
MB	7,500 F	1,320 F	1,000	765	200
MB/watt	4.54	2.0	5.88	20.13	20.0
MB/cu ft	313	388	2,500	8,500	10,000

Figure 5.2: Environmental Efficiency of Small Disk Drives
Smaller drives provide a significantly higher capacity per watt and volumetric efficiency. This figure is reproduced with corrections from [21].

5.2. RAID Levels

Table 5.1 makes it dramatically clear that I/O bandwidth can be significantly increased by replacing conventional disks by large numbers of inexpensive disks, with no loss of capacity and at comparable cost. Further, each of the smaller format devices appear to have comparable reliability to the larger disks. Unfortunately, the reliability of an array of devices decreases as additional devices are added to the array. Assuming a constant failure rate, i.e., an exponentially distributed time to failure, and that failures are independent (both assumptions are frequently made by disk manufacturers when calculating their Mean Times To Failure - MTTF), the MTTF of a Disk Array =

$$\frac{\text{MTTF of a Single Disk}}{\text{Number of Disks in the Array}}$$

Using the the information in Table 5.1, the MTTF of 100 CP 3100 disks is 30,000/100 = 300 hours, or less than two weeks. This is considerably worse than the 30,000 hour (> 3 years) MTTF of the single IBM 3380 the array is meant to replace. If we consider scaling the array to 1000 disks, then the MTTF is even worse, i.e., 30 hours or about one day.

It is imperative that some I/O bandwidth and storage capacity be sacrificed to achieve a higher level of fault tolerance in disk arrays. We must make use of extra disks containing redundant information to recover the original information when a disk fails. Hence the acronym ***RAID*** for Redundant Arrays of Inexpensive Disks. There are several different ways of organizing disks into a RAID configurations, and we will present a taxonomy of these, orthogonal to the taxonomy of Section 4, in the subsections to follow. Interestingly enough, manufacturers are beginning to deliver products that represent each of these RAID "levels". Each represents a different tradeoff between I/O bandwidth and available capacity dedicated to redundant information.

Our basic approach will be to break the arrays into reliability groups, with each group having extra "check disks" containing redundant information. When a disk fails, we assume that within a short time the failed disk will be replaced and the information will be reconstructed on the new disk using the redundant information. This is time is called the mean time to repair (MTTR). The MTTR can be reduced if the system includes extra disks to act as "hot" standby spares; when a disk fails, a replacement disk is switched in electronically. Periodically, a human operator replaces all failed disks. The basic four step process is:

1. Detect a failed disk;
2. Error correct while processing continues;
3. Reconstruct lost data on a hot spare;
4. Periodically replace broken disks with new ones.

We will use the following terminology throughout the subsequent subsections:

D = total number of disks with data (not including extra check disks);
G = number of data disks in a group (not including extra check disks);
C = number of check disks in a group;
n_G = D/G = number of groups.

From [31], the MTTF for a single error repairing RAID, ignoring failures in support components, such as power supplies, cables, etc., can be expressed as $MTTF_{RAID}$:

$$\frac{(MTTF_{Disk})^2}{(D + C * n_G) * (G + C - 1) * MTTR}$$

As already mentioned in Section 3, classes of I/O intensive applications tend to

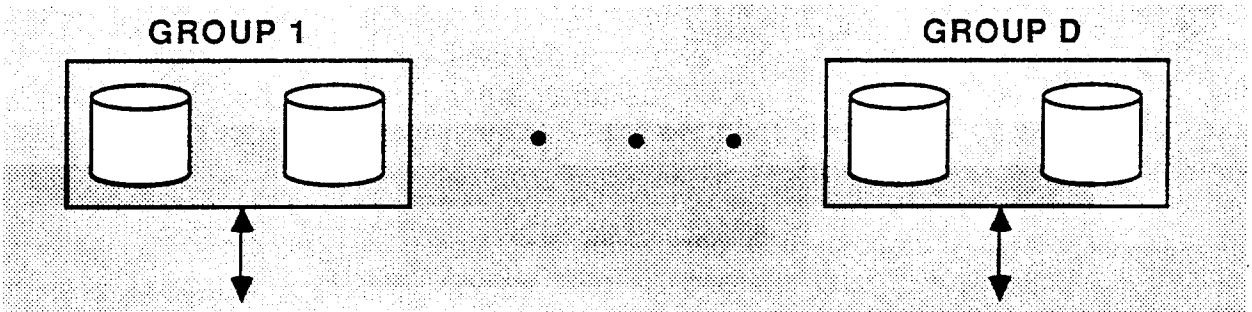


Figure 5.3. Raid Level 1 -- Mirrorred Disks
An array constructed from 2D disks yields D groups of 2 disks each. 50% of the available disks contain redundant data. Since each logical write is actually two physical writes, the array can only support half as many writes as 2D independent disks. However, assuming redundant controllers and I/O pathways, the full read bandwidth of the 2D disks can be achieved.

have different access patterns and rates, hence we need different metrics to evaluate them. For supercomputers, we count the number of reads and writes per second for large blocks of data, with *large* being defined as at least one sector from each data disk in a group. During large transfers all the disks in a group act as a single unit, each reading or writing a portion of the large data block in parallel. A better metric for transaction processing applications would be individual (i.e., single device) reads or writes per second. Thus supercomputer applications are concerned with high transfer bandwidth, while transaction processing applications are concerned with high I/O rates.

5.2.1. RAID I: Mirrored Disks

Mirrored disks are a traditional approach for improving the reliability of magnetic disks (see Figure 5.3). This is the most expensive option we consider, since all disks are duplicated ($G = 1$ and $C = 1$), and every write to a data disk is also a write to a check disk. This supports our contention that some I/O bandwidth and storage capacity must be sacrificed to obtain higher reliability. Mirroring is extensively employed by Tandem Computers, who also duplicate the controllers and I/O pathways for reasons of fault tolerance (more on this in Section 6, especially Figure 6.3). This version of mirrored disks supports parallel reads.

Digital Equipment Corporation's HSC-70 disk controller supports a generalization of mirroring, which they call *shadowing* [18]. The contents of a disk are replicated among the members of its *shadow set.* A shadow set of two is identical to mirrored disks. A given sector can be read from any device within the shadow set. When a sector is written, all devices within the set must be updated. The controller has the ability to predict that a device will soon fail, and can allocate a hot spare to create a back up and keep it consistent through the shadow set mechanism. The failing disk can then be powered down.

Large group accesses are defined as touching a sector from each of the D data disks in the group. To provide a fair comparison with the other RAID organizations introduced below, we shall assume that a group access must encompass as many mir-

rored pairs as there are data disks in the other organization, or D such mirrored pair groups. When many arms seek to the same track, then rotate to the desired sector, the average seek and rotational delay will be larger than the average for a single disk, tending towards the worse case. This effect should not more than double the average access time to a given sector, while still making it possible to access a sector per pair in parallel. To capture this effect, we introduce a slowdown factor *S*, when there are more than two disks that participate in a logical access. In general, *S* will be greater than 1 and less than 2 whenever groups of disks work in parallel.

Rather than give absolute numbers for group transfers, we calculate efficiency, i.e., number of events per second for a RAID vs. number of events per second for a single disk. Consider the case of a group read, i.e., a logical read operation that reads D physical sectors in parallel. A RAID could do many more such operations than a single disk. Assuming redundant controllers and pathways, and taking account of the slowdown factor described above, a RAID Level 1 can perform *2D/S* sector reads per second compared to a single disk. The write bandwidth is half of this, or *D/S* writes per second compared to a single disk.

Duplicating all disks can mean doubling the costs of the system or alternatively, utilizing only 50% of the available disk capacity for data storage. The capacity overhead for fault tolerance is 100%. Such lack of economies inspire the next levels of RAID.

5.2.2. RAID II: Bit Interleaved Array

Main memory organization suggests a way to reduce the capacity cost to achieve reliability. 4K and 16K DRAM chips were the first to fail extensively because of alpha particles. Since there were many single bit wide DRAMs in a system, and usually accessed in groups of 16 to 64 bits at a time, system designers added redundant chips to correct single bit errors and to detect double bit errors in each group. This increased the number of memory chips by 12% to 38%, depending on the size of the group, but it significantly improved reliability.

As long as all the data bits in a group are read or written together, the addition of correction codes has no impact on performance. However, reads of less than the group size require reading the whole group to be sure that the information is correct, by recomputing the check bits and comparing them to the stored check bits. Writes to a portion of the group require three steps:

1. *read all bits across the group, including the bits not being updated;*
2. *merge the updated bits with those not being updated, recomputing the check bits;*
3. *rewrite the full group, including check information.*

We can mimic the DRAM solution by bit-interleaving the data across the disks of a group, extended with enough check disks to detect and correct a single bit error (see Figure 5.4). A single parity disk can detect a single error, but more disks are needed to correct an error. For a group of 10 data disks (*G*) we need 4 check disks (*C*) in total,

and if G = 25 then C = 5 [35]. To limit the cost of redundancy, we assume the group size will vary from 10 to 25.

Since our individual data transfer unit is just a sector, bit-interleaved disks imply that a large transfer for this RAID must be at least G sectors. Like DRAMs, reads to a smaller amount implies reading a full sector from each of the bit-interleaved disks in a group. For G = 10, the total number of disks is 1.4D (40% overhead; 71% usable storage capacity). For G = 25, the check disks represent a smaller fraction of the data disks. In this case, the total number of disks is 1.2D, with 20% overhead and 83% effective capacity. This organization only supports a single I/O per group, whether large or small, because of the action of the correcting codes. This presents an interesting issue for organizing the array: larger group sizes yield a reduced capacity overhead for fault tolerance, but fewer groups mean fewer logical I/Os can be performed in parallel within the array. Using reasoning similar to the previous subsection, the RAID Level 2 events/second compared to a single disk for large reads and writes are *D/S*.

For large writes, the Level 2 system has similar performance to Level 1 even though it uses fewer check disks, and so on a per disk basis it outperforms Level 1. For small transfers, the performance is dismal, since the whole group must be read or written as a unit anyway. Thus, Level 2 RAIDs are desirable for supercomputers, but inappropriate for transaction processing. Thinking Machines Inc. supports a Level 2 RAID for its Connection Machine called the "Data Vault," with G = 32 and C = 10, including three hot standby spares [36].

5.2.3. RAID III: Hard Failure Detection and Parity

Most of the check disks used in RAID Level 2 are needed to identify the incorrect bit position. These disks are truly redundant since most disk controllers can already detect when a disk has failed: either through special signals supported by the disk interface or by additional coding information stored on the disk information and used to correct tran-

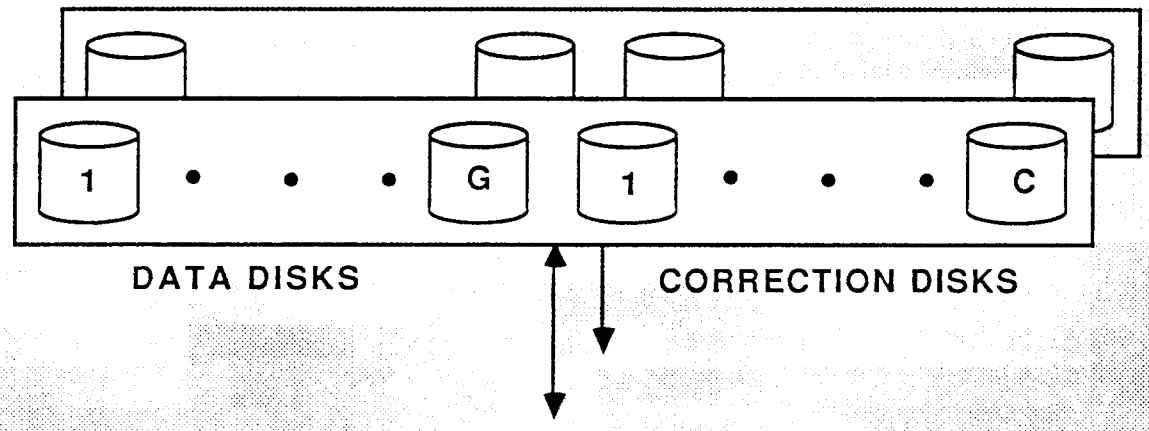


Figure 5.4: RAID Level 2 -- Error Correcting Codes.
A group consists of G data disks and C correction disks. The bits stored on any single lost disk can be reconstructed using conventional ECC techniques. Single error correction/double error detection costs O(log(G)) check disks.

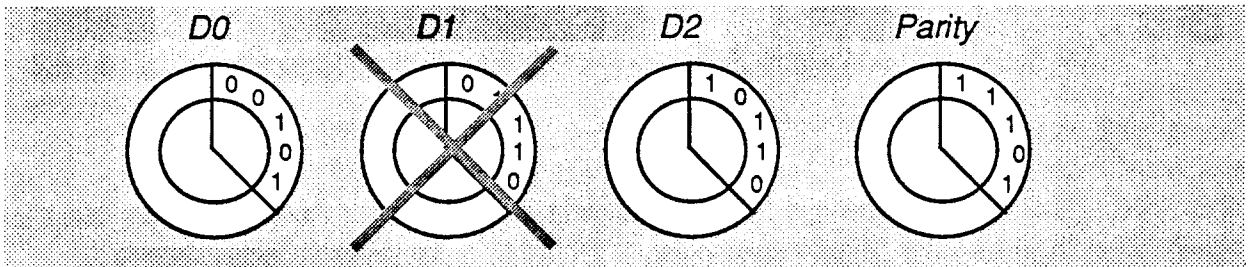


Figure 5.5: RAID Level 3 -- Parity Disks
Data is bitwise interleaved across three data disks. A fourth parity disk contains the bitwise even parity of the data disks. It is possible to recover the data on any single lost disk by reading the contents of the surviving disks, and recomputing the parity.

sient errors. In fact, if the controller can detect the failed bit position, then only a single parity bit is needed to reconstruct the lost data. Figure 5.5 illustrates how a single parity disk can be used to reconstruct any single hard disk error.

Reducing the check disks to one per group (C = 1) reduces the capacity overhead to between 4% and 10% for the group sizes we have been considering. The events/second for the third level RAID system is the same as the Level 2 RAID, but the effective performance per disk increases since fewer check disks are required.

Park and Balasubramanian proposed a third level RAID system without suggesting a particular application [37]. Our calculations suggest it is a much better match to supercomputer applications than to transaction processing systems. At least two manufacturers have announced level 3 RAIDs using synchronized 5.25" SCSI disks with G = 4 and C = 1: one from Maxtor and one from Micropolis [38], and other manufacturers are reported to be working on 3.5" device-based Level 3 arrays. Maximum Strategies has announced a Level 3 product based on SMD and ESDI disk drives [39]. Each of these products appear to the host as a single logical disk with four times the transfer bandwidth and four times the capacity and much higher reliability.

5.2.4. RAID IV: Intragroup Parallelism

Spreading a transfer across all disks within the group has the advantage that large or grouped transfer time is reduced because the bandwidth of the entire array can be exploited. On the other hand, reading/writing to a disk in a group requires accessing every disk in the group, i.e., Level 2 and 3 RAIDs can perform only one I/O at a time per group. Further, these group accesses are slowed down by some factor, i.e., it is not possible to obtain the average seek and rotational latencies unless the disks are synchronized.

Level 4 RAID improves the performance of small transfers through parallelism, by providing the ability to perform more than one small I/O per group at a time. We no longer spread the logical transfer block across several disks, but keep each individual block on a single disk.

The virtue of bit-interleaving is the easy calculation of the Hamming code needed to detect or correct errors in Level 2. In Level 3 RAIDs, we rely on the disk controller to detect disk errors within a single sector. Hence, if we store an individual transfer block in a single sector, we can detect errors on an individual read without accessing any other disk. The primary change between Levels 3 and 4 is that we interleave data between disks at the sector level rather than at the bit or byte level.

It may appear that writes are just as complex as in Level 3, requiring a read of the group and a merge of the written data to recompute check (parity) information. Actually, updating the parity information in Level 4 is more simple. To compute the new parity, all we need is the old data block, old parity block, and new data block:

new parity = (old data **xor** *new data)* **xor** *old parity*

In Level 4, a small write uses two disks to perform four access (data plus parity read, data plus parity write). The performance for group reads and writes is as before, but small (single disk) reads and writes are greatly improved. Unfortunately, the improvement is not sufficiently dramatic to displace Level 1 as the organization of choice for transaction processing environments. Recently, Salem and Garcia-Molina proposed a Level 4 system [27].

5.2.5. RAID V: Rotated Parity to Parallelize Writes

RAID Level 4 achieved parallelism for single device reads, but writes are still limited to one per group since every write must read and write the parity disk. Level 5 improves on Level 3 by distributing the check information across all disks within the group. This is shown in Figure 5.6.

This small change has an enormous performance impact for small writes. If writes can be scheduled to touch different disks for their data and corresponding parity blocks, then up to (G + 1)/2 writes can be processed in parallel. This organization has good performance for both large reads and writes as well as small reads and writes, making it the most appropriate organization for mixed environments.

5.2.6. RAID VI: Two-Dimensional Parity for Even More Reliability

It is possible to envision a further point in the fault tolerance/bandwidth tradeoff. RAID Level 5 is really only one dimension of disks, with the second dimension being the sectors. Now consider the arrangement of disks into a two dimensional array, with a third dimension being sectors. We can have both row parities, as in Level 5, as well as column parities, which in turn can be interleaved to permit parallel writes. Such an organization can survive any loss of two disks as well as many losses of three disks. However, every logical write is really six physical accesses: old data, row parity, column parity reads and new data, new row parity, new column parity writes. This overhead may be acceptable for some very highly fault tolerant applications, but is probably unac-

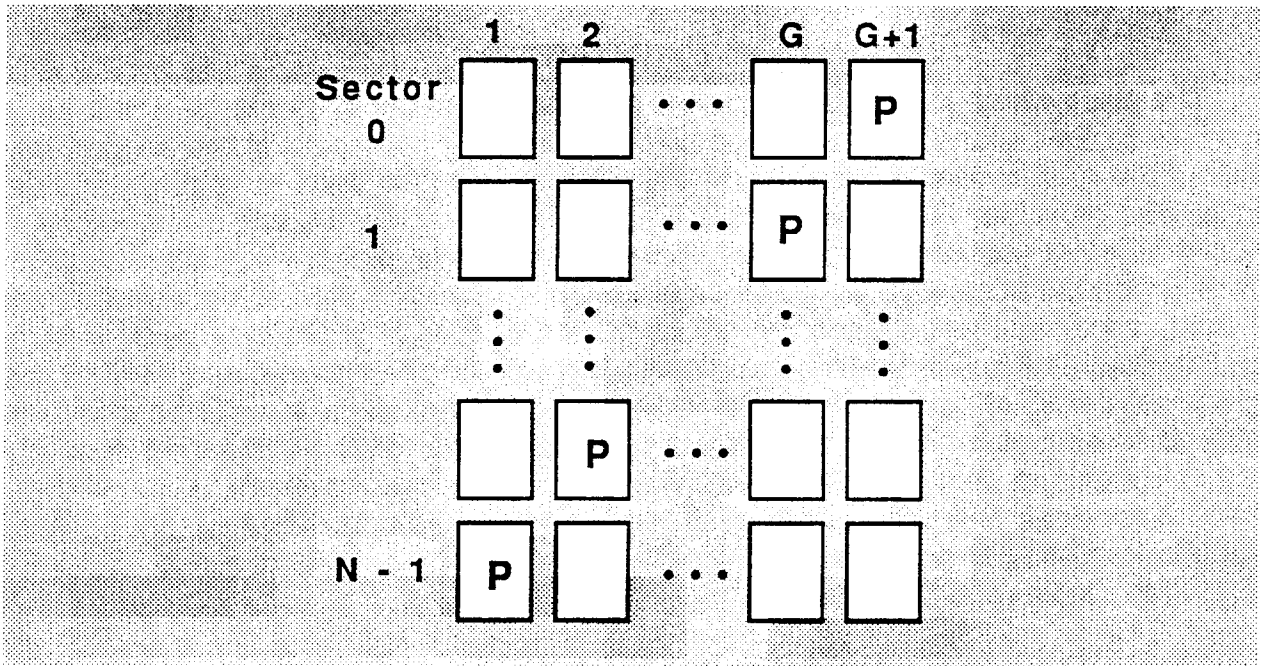


Figure 5.6. RAID Level 5 -- Rotated Parity Blocks
The disks within the group are represented by the columns, while the rows represent sectors within each disk. Parity blocks, rather than being placed exclusively on the "G+1"st disk, are interleaved among the disksof the group in such a manner as to guarantee that there is only one parity block per sector row. Since there are many more sectors per disk than disks, the pattern shown above will be repeated many times. Two writes can take place in parallel as long as the data and parity blocks for their rows fall into different columns. Thus, at most (G+1)/2 writes can occur at the same time.

ceptable for conventional supercomputer or transaction processing applications. The topic of efficient multi-dimensional codes for disk failure correction is discussed in [40].

5.3. Comparison Among The RAID Levels

From the viewpoint of providing a general purpose subsystem that can support both small block and large block accesses, the only serious candidates are RAID Level I, *Mirrored Disks*, and RAID Level 5, *Rotated Parity*. For a fair comparison, we will hold the number of data disks fixed, and normalize performance by the total number of disks in the candidate configuration. Thus, for a group with G data disks, RAID Level 1 would contain 2 G disks organized as G pairs, while the RAID Level 5 would contain G+1 disks. The normalization by total number of disks is necessary, because Level 1 requires more disks than Level 5, and naturally can perform a larger absolute number of I/Os.

Because of the ability to simultaneously read from either disk in the mirrored pair, the full read bandwidth of all 2 G disks within a mirrored disk array can be effectively harnessed for both small or large reads. Normalizing by the number of disks yields 100% of the available read bandwidth being used. The same is true of a collection of disks organized as a RAID 5: G+1 small reads can be performed independently by all G+1 disks, yielding 100% usable bandwidth. A similar argument applies to large reads.

The place where they differ is in terms of write performance. For large writes, i.e., a block write that spans every mirrored pair, the effective bandwidth of the array is only half of the total available write bandwidth. This is because each write to a data disk must also be written to its mirror. The same degradation in write bandwidth is experienced whether the writes are large or small. RAID Level 5, however, does better on large writes but somewhat worse on small writes. For large writes, G data sectors and 1 parity sector are written, and the resulting write bandwidth is G/(G+1) percent of the total available write bandwidth. Thus, a ten data disk array organized as a RAID 5 (actually consisting of eleven disks) will experience a degradation in write bandwidth to 91% of maximum, while the same disks organized as a RAID I (consisting of twenty disks) will endure degraded write performance to 50% of the maximum. Unfortunately, RAID 5 provides only 25% of the available write bandwidth for small writes, because each logical write is actually four physical I/Os, while RAID 1 will continue to provide 50% of available write bandwidth.

These effects have been observed empirically in a performance experiment on a high-end Amdahl system reported by Chen in [41]. RAID Level 1 small read performance was actually better than predicted because of the effectiveness of seek scheduling: the actuator within the pair closest to the requested seek position is selected to satisfy the request. This effectively decreases the average seek time, thus increasing the number of small reads that can be serviced per unit time. RAID Level 5 large write performance was slightly worse than expected. A *stripe* is the row of sectors in the same position across the disks of the group. Not all large writes can be scheduled on a stripe boundary, because files cannot be allocated to stripe boundaries without significant loss of capacity. Thus, writes to the beginning and end of files may stride stripes, and thus behave more like small writes in terms of the physical I/Os they generate.

In general, if small reads and writes predominate, and capacity cost is no issue, then RAID Level 1 will provide the best performance. However, if the application is capacity cost sensitive or the occurrence of small writes can be reduced (e.g., a high read to write ratio, effective buffering of read-modify-write sequences, or small writes turned into large writes through file caching strategies), then RAID Level 5 can provide superior performance, especially in terms of price/performance.

6. Controller Architectures

In this section, we contrast the structure of an I/O controller for a disk array with the more conventional host/channel/director/head-of-string/device architecture presented in Section 2 and particularly, Figure 2.2. The key technological difference is the wide spread use of embedded controllers implemented as VLSI chips as well as a more extensive use of speed matching buffers throughout the host to device pathway.

Figure 6.1 shows the elements of a simple I/O disk array subsystem. The *host adapter* manages the interface between the host and the array, and usually employs a direct memory access technique to transfer data between the host memory and the I/O devices. The *array controller* is responsible for interfacing to the the individual disk controllers, and includes such elements as speed matching memory buffers and the parity logic to reconstruct the data contents of failed disks. Perhaps the most important function of the array controller is its management of the mapping between logical and physical disk sectors, for example, to handle the placement of interleaved parity sectors in a RAID V and to remap reconstructed sectors onto a hot spare disk within the array. The *single board controller* (SBC), usually physically piggy-backed with the disk drive in small format disks, handles the physical control of the device. It provides a logical interface to the array controller, is responsible for media error detection and correction, and usually includes a track buffer, which can be used either for speed matching or to eliminate RPS misses. The SBC may also support spindle synchronization, if a synchro-

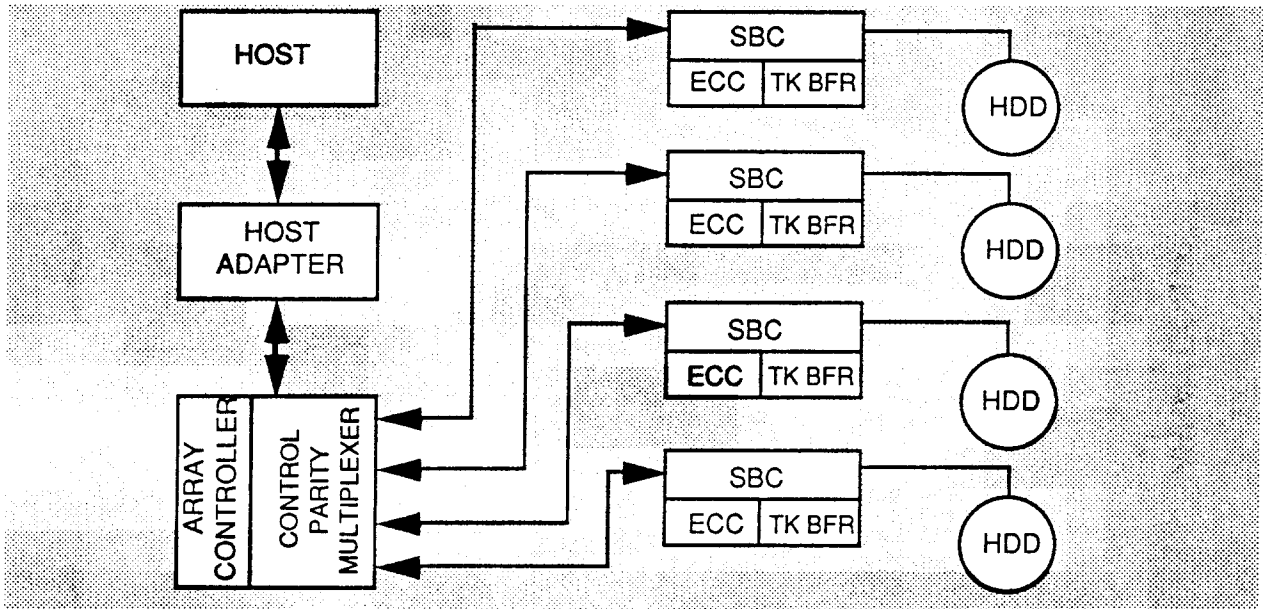


Figure 6.1: Components of a Disk Array Controller
The pathway between the host and the device consists of a host bus adapter, an array controller, and a single board controller (SBC). The interface between the host and host adapter is usually a standard backplane bus; the interface between the array controller and the SBCs is a high level disk protocol, such as SCSI. HDD is a hard disk device, ECC is error correction circuitry, and TR BFR is a track buffer. The latter two are resident within the SBC.

nized array organization is desired. The pathway between the array controller and the single board controllers might be a single multiplexed high speed bus, or single pathway per SBC.

Figure 6.1 shows a single "column" of hard disk drives, which can be treated as individual drives for small I/Os, and a single group for large I/Os. One (or more) of the drives in the column can be held in reserve as a "hot spare" to replace another drive in the column that may fail. Combining the concept of a column transfer group with the notion of a disk string already introduced in Section 2 yields the two dimensional array structure of Figure 6.2. Here the single board controller has been replaced by a string controller, and the single drive has been replaced by a string of drives. In reality, a high level disk protocol such as SCSI can allow this string controller function to be distributed among individual SBCs, one per device. The key feature that must be supported in the string configuration is the ability to have overlapped seeks. Because the potential I/O bandwidth is dramatically improved in this configuration over Figure 6.1, it may become necessary to provide a higher bandwidth or multi-port interface to the host to realize the benefits of this organization.

The organizations examined so far concentrate on high I/O performance, and through the use of redundant parity information on disk, very high availability of the magnetic medium. It should be clear, however, from Figures 6.1 and 6.2 that there is much more to a disk array than many disks. Figure 6.3 shows a more truly fault tolerant array configuration. All controller hardware and pathways are duplexed to guarantee that no single point of failure anywhere in the subsystem will render the data on disk un-

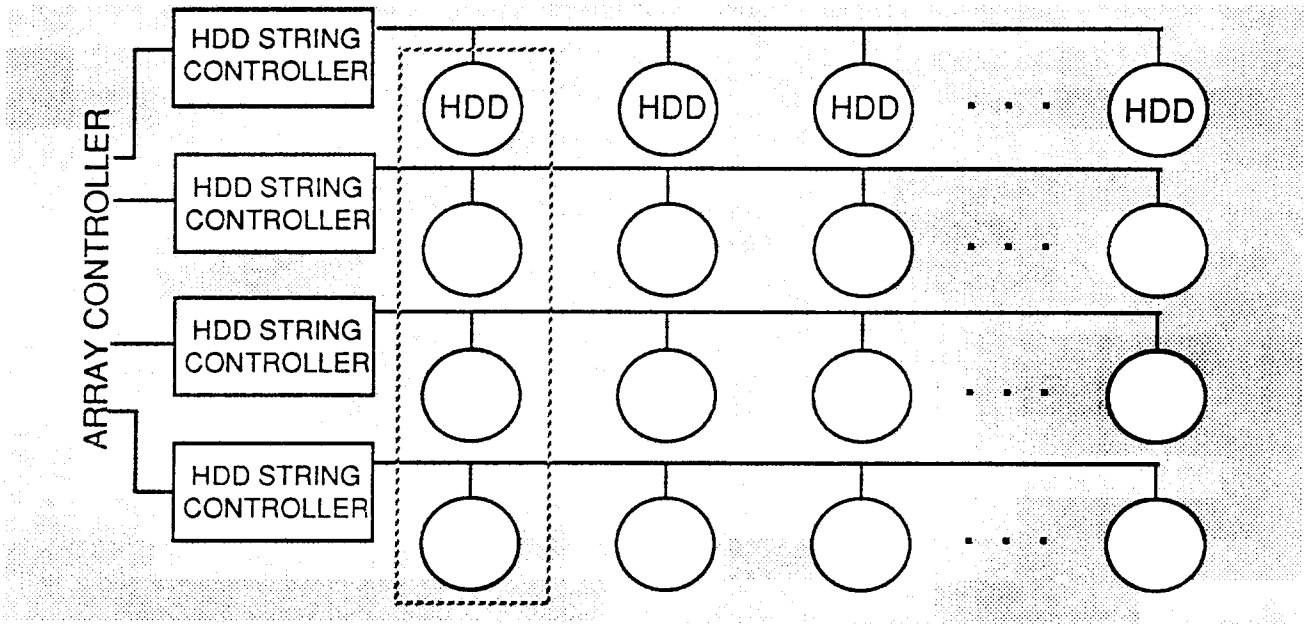


FIGURE 6.2: Two Dimensional Array Organization
The string concept can be combined with the basic array organization to construct a two dimensional array. Disks within a column form a transfer group. Any disk can be accessed individually or as part of a large I/O transfer group.

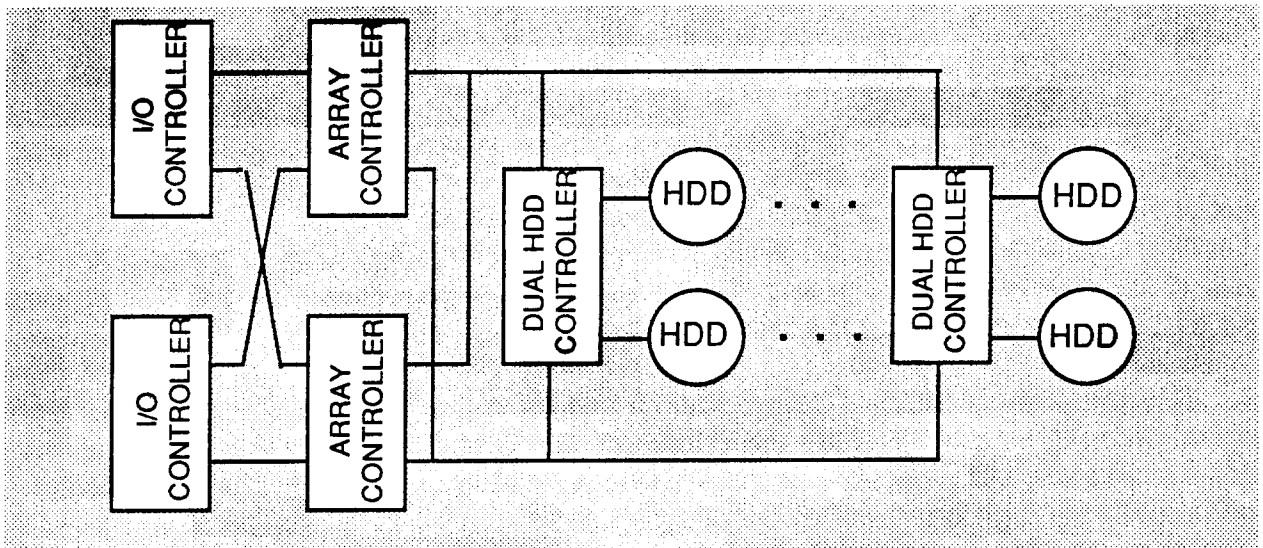


FIGURE 6.3: Elements of a True Fault Tolerant Disk Array

A true fault tolerant array requires duplexed controller hardware and I/O pathways. Because of the high cost of the duplexed approach, true fault tolerance architectures are more of interest in transacyion processing environments than in supercomputer I/O. However, a bonus of the duplexed controllers and pathways is that higher transfer bandwidth can be sustained when all components of the system are operational.

available. This kind of organization is better suited to transaction processing applications, where high availability is a key issue, than supercomputer I/O. However, the duplexed paths and controllers can support higher I/O rates than the conventional single path and controller organization. A more detailed examination of high reliability disk arrays can be found in [42].

7. Summary and Conclusions

In this paper, we have examined the fundamental gap between CPU and I/O system performance, and have reviewed basic I/O system architecture and the standard approaches for improving I/O performance, such as parallel transfer disks, increased disk density, and solid state disks. While they appear promising, each of these techniques still must overcome certain economic and technical limitations before seeing widespread use in supercomputer environments.

One of the most exciting developments has been the rapid progress in small format disks. We have observed their trend towards significantly higher efficiency (in terms of $/MB,cubic feet/MB, watts/MB) than their larger cousins. We are now in a position to construct very high bandwidth I/O subsystems from large arrays of small format disks. The approach exploits parallelism in much the same way that a very high performance computer can be more easily (and cheaply) constructed from arrays of smaller machines.

However, by dramatically increasing the number of devices in the I/O system, we adversely affect its reliability. Thus any feasible array architecture must provide some solution to deal with the reliability problem. There is an intrinsic trade-off between capacity, fault tolerance, and I/O bandwidth. In the organizations we have described, a variety of techniques based on redundant information (complete mirroring, error correcting codes, parity) were introduced to make possible the recovery of lost data. This redundant information not only steals disk capacity, but also steals some bandwidth, especially on writes, since the redundant information must be maintained. We have analyzed the organizations in Section 5 in terms of their capacity overhead and ability to support reads and writes per second.

The Level 1 RAID (Disk Mirroring) provides the best support for small (single sector/single device) reads and writes, while Level 3 RAID (Byte Interleaved/Single Parity) provides the best support for group I/Os. The Level 5 RAID (Sector Interleaved/Rotated Parity) appears to be the best compromise organization for I/O environments that must support a mix of small and large I/Os. As computing environments continue to evolve in the direction of diskless supercomputers networked to visualization workstations, we expect the latter environment to predominate.

High raw I/O bandwidth from the disks still cannot guarantee a very high performance system. In Section 6, we examined some of the organizations for constructing I/O controllers for disks arrays. The key issue here is multipathing, i.e., providing alternative pathways between memory and devices, both for fault tolerance as well as greater I/O bandwidth. The trend is towards more intelligence and more buffer memory placed near the device, exploiting advances in VLSI technology that make possible embedded disk controllers.

I/O architecture presents one of the greatest impediments to achieving high perfor-

mance in future computing system. Disk arrays are one promising approach for narrowing the CPU-I/O gap, but much further analysis is necessary to understand the best way to proceed. I/O system design, long a "black art", is beginning to emerge into the limelight, and will share the center stage with CPU design as we reach for ever faster machines.

Acknowledgements

We wish to thank our colleagues on the RAID project for their contributions to the work reported in this paper. These include Peter Chen, Ann Chevernak, Rich Drewes, Ed Lee, Ken Lutz, John Ousterhout, Martin Schulze, Steve Sprouse, and Mike Stonebraker. Jim Gray of Tandem Computers and Dieter Gawlick of Digital Equipment Corporation provided useful insights in reliability issues and transaction processing system performance. Steve Luzmoor of Cray Research answered many questions about the CRAY I/O System. The referees provided perceptive comments for improving and expanding the presentation. This work has been supported by the National Science Foundation under Grant # MIP-8715235 and the California MICRO Program, with matching industrial participation through Digital Equipment Corporation, Intel Scientific Computers, Inc., International Business Machines, Inc., and SUN Microsystems, Inc.

8. Bibliography

[1] Klietz, A., J. Turner, T. C. Jacobson, "Turbo NFS: Fast Shared Access for Cray Disk Storage," *Proceedings Cray Users Group Conference*, 1988.

[2] C. G. Bell, "The Mini and Micro Industries," *IEEE Computer*, Vol. 17, No. 10, (October 1984), pp. 14-30.

[3] G. J. Myers, A. Y. C. Yu, D. L. House, "Microprocessor Technology Trends," *Proc. IEEE*, Vol. 74, No. 12, (December 1986), pp. 1605-1622.

[4] W. Joy, Presentation at ISSCC '85 Panel Session, (February 1985).

[5] D. P. Sierwiorek, C. G. Bell, A. Newell, Computer Structures: Principles and Examples, McGraw Hill, 1982, p. 46.

[6] G. E. Moore, "Progress in Digital Integrated Electronics," *Proc. IEEE Digital Integrated Electronic Device Meeting*, (1975), p. 11.

[7] H. Garcia-Molina, R. Cullingford, P. Honeyman, R. Lipton, "The Case for Massive Memory," Technical Report 326, Dept. of EE and CS, Princeton University, (May 1984).

[8] P. D. Frank, "Advances in Head Technology," Presentation at Challenges in Disk Technology Short Course, Institute for Information Storage Technology, University of Santa Clara, Santa Clara, CA, (December 12-15, 1987).

[9] L. D. Stevens, "The Evolution of Magnetic Storage," *IBM Journal of Research and Development*, Vol. 25, No. 5, (Sept. 1981), pp. 663-675.

[10] Smith, A. J., "Cache Memories," ACM Computing Surveys, V 14, N 3, (September 1982), pp. 473-530.

[11] J. M. Harker, et. al., "A Quarter Century of Disk File Innovation," *IBM Journal of Research and Development*, Vol. 25, No. 5, (September 1981), pp. 677-689.

[12] J. Voelcker, "Winchester Disks Reach for a Gigabyte," *IEEE Spectrum*, (February 1987), pp. 64-67.

[13] Bucher, I. Y., A. H. Hayes, "I/O Performance Measurement on Cray-1 and CDC 7600 Computers," *Proceedings of the Cray Users Group Conference*, (October 1980).

[14] Buzen, J. P., A. Shum, "I/O Architecture in MVS/370 and MVS/XA," *CMG Transactions*, Vol. 54, (Fall 1986), pp. 19-26.

[15] Buzen, J., "BEST/1 Analysis of the IBM 3880-13 Cached Storage Controller," *Proc. CMG XIII Conference*, (1982).

[16] Buzen, J. P., A. Shum, "A Unified Operational Treatment of RPS Reconnect Delays," *Proc. 1987 Sigmetrics Conference*, Performance Evaluation Review, V. 15, N. 1, (May 1987).

[17] Cray Research, Inc., "CRAY Y-MP Computer Systems Functional Description

Manual," HR-4001, (January 1988).

[18] Massiglia, P., *Digital Large System Mass Storage Handbook*, Digital Equipment Corporation, Colorado Springs, CO, 1986.

[19] Kronenberg, N. P., H. Levy, W. D. Strecker, "VAXClusters: A Closely-Coupled Distributed System," *ACM Trans. on Comp. Systems*, V 4, N 2, (May 1986), pp. 130-146.

[20] Houtekamer, G., "The Local Disk Controller," *Proc. 1985 Sigmetrics Conference*, (August 1985).

[21] A. Vasudeva, "A Case for Disk Array Storage System," *Proc. Reliability Conference*, Santa Clara, CA, (1988).

[22] Reinhardt, S., "A Blueprint for the UNICOS Operating System," *Cray Channels*, V. 10, N. 3, (Fall 1988), pp. 20-24.

[23] Gawlick, D., Private Communication, (November 1987).

[24] Smith, A. J., "Input/Output Optimization and Disk Architectures: A Survey," *Performance and Evaluation 1*, North-Holland Publishing Company, (1981), pp. 104-117.

[25] Kim, M. Y., "Synchronized Disk Interleaving," *IEEE Transactions on Computers*, V. C-35, N 11, (November 1986), pp. 978-988.

[26] M. Y. Kim and A. N. Tantawi, "Asynchronous Disk Interleaving," IBM Research Report RC12497, (January 30, 1987).

[27] Salem, K., H. Garcia-Molina, "Disk Striping," *Proc. IEEE Data Engineering Conference*, Los Angeles, CA, (February 1986).

[28] Livny, M., S. Khoshafian, H. Boral, "Multi-Disk Management Algorithms," *Proc. SIGMETRICS*, (May 1987).

[29] D. Bitton, J. Gray, "Disk Shadowing," *Proc. Very Large Database Conference*, Long Beach, CA, (August 1988).

[30] Copeland, G., T. Keller, "A Comparison of High-Availability Media Recovery Techniques," *Proc. ACM SIGMOD Conference*, Portland, OR, (May 1989).

[31] Patterson, D. A., G. A. Gibson, R. H. Katz, "A Case for Redundant Arrays of Inexpensive Disks," *Proc. ACM SIGMOD Conference*, Chicago, IL, (June 1988).

[32] A. S. Hoagland, "Information STorage Technology -- A Look at the Future," *IEEE Computer Magazine*, V 18, N 7, (July 1985), pp. 60-67.

[33] "IBM 3380 Direct Access Storage Introduction," IBM GC 26-4491-0, September 1987.

[34] "M2361A Mini-Disk Drive Engineering Specifications (revised)," B03P-4825-0001A, (February 1987).

[35] Hamming, R. W., "Error Detecting and Correcting Codes," *The Bell System*

Technical Journal, Vol. XXVI, No. 2, (April 1950), pp. 147-160.

[36] Hillis, D., Personal Communication, 1987.

[37] Park, A., K. Balasubramanian, "Providing Fault Tolerance in Parallel Secondary Storage Systems," Department of Computer Science, Princeton University, CS-TR-057-86, (Nov. 1986).

[38] Maginnis, N. B., "Store More, Spend Less: Mid-Range Options Abound," *Computerworld*, Nov. 16, 1987, p. 71.

[39] Meador, W. E., "Disk Array Systems," *Proc. Spring Compcon Conference*, San Francisco, CA, (February 1989), pp. 143--146.

[40] Gibson, G. A., L. Hellerstein, R. M. Karp, R. H. Katz, D. A. Patterson, "Failure Correction Techniques for Large Disk Arrays," *Proc. Third Intr. Conf. on Architectural Support for Programming Languages and Operating Systems,* Boston, MA, (April 1989).

[41] Chen, Peter M., "An Evaluation of Redundant Arrays of Inexpensive Disks using an Amdahl 5890," M.S. Plan II, U.C. Berkeley Computer Science Division, (May 1989).

[42] Schulze, M., G. A. Gibson, R. H. Katz, D. A. Patterson, "How Reliable is a RAID?," *Proceedings of IEEE Spring COMPCON Conference*, San Francisco, CA, (February 1989).