Disk arrays were proposed in the 1980s as a way to use parallelism between multiple disks to improve aggregate I/O performance. Today they appear in the product lines of most major computer manufacturers. This article gives a comprehensive overview of disk arrays and provides a framework in which to organize current and future work. First, the article introduces disk technology and reviews the driving forces that have popularized disk arrays: performance and reliability. It discusses the two architectural techniques used in disk arrays: striping across multiple disks to improve performance and redundancy to improve reliability. Next, the article describes seven disk array architectures, called RAID (Redundant Arrays of Inexpensive Disks) levels 0–6 and compares their performance, cost, and reliability. It goes on to discuss advanced research and implementation topics such as refining the basic RAID levels to improve performance and designing algorithms to maintain data consistency. Last, the article describes six disk array prototypes or products and discusses future opportunities for research, with an annotated bibliography of disk array-related literature.
1. INTRODUCTION

In recent years, interest in RAID, Redundant Arrays of Inexpensive Disks,\(^1\) has grown explosively. The driving force behind this phenomenon is the sustained exponential improvements in the performance and density of semiconductor technology. Improvements in semiconductor technology make possible faster microprocessors and larger primary memory systems which in turn require larger, higher-performance secondary storage systems. These improvements have both quantitative and qualitative consequences.

On the quantitative side, Amdahl’s Law [Amdahl 1967] predicts that large improvements in microprocessors will result in only marginal improvements in overall system performance unless accompanied by corresponding improvements in secondary storage systems. Unfortunately, while RISC microprocessor performance has been improving 50% or more per year [Patterson and Hennessy 1994, p. 27], disk access times, which depend on improvements of mechanical systems, have been improving less than 10% per year. Disk transfer rates, which track improvements in both mechanical systems and magnetic-media densities, have improved at the faster rate of approximately 20% per year, but this is still far slower than the rate of processor improvement. Assuming that semiconductor and disk technologies continue their current trends, we must conclude that the performance gap between microprocessors and magnetic disks will continue to widen.

In addition to the quantitative effect, a second, perhaps more important, qualitative effect is driving the need for higher-performance secondary storage systems. As microprocessors become faster, they make possible new applications and greatly expand the scope of existing applications. In particular, image-intensive applications such as video, hypertext, and multimedia are becoming common. Even in existing application areas such as computer-aided design and scientific computing, faster microprocessors make it possible to tackle new problems requiring faster access to larger data sets. This shift in applications, along with a trend toward large, shared, high-performance, network-based storage systems, is causing us to reevaluate the way we design and use secondary storage systems [Katz 1992].

Disk arrays, which organize multiple, independent disks into a large, high-performance logical disk, are a natural solu-

\(^1\)Because of the restrictiveness of “Inexpensive,” sometimes RAID is said to stand for “Redundant Arrays of Independent Disks.”
tion to the problem. Disk arrays stripe data across multiple disks and access them in parallel to achieve both higher data transfer rates on large data accesses and higher I/O rates on small data accesses [Salem and Garcia-Molina 1986; Livny et al. 1987]. Data striping also results in uniform load balancing across all of the disks, eliminating hot spots that otherwise saturate a small number of disks while the majority of disks sit idle.

Large disk arrays, are highly vulnerable to disk failures however. A disk array with 100 disks is 100 times more likely to fail than a single-disk array. An MTTF (mean-time-to-failure) of 200,000 hours, or approximately 23 years, for a single disk implies an MTTF of 2000 hours, or approximately three months, for a disk array with 100 disks. The obvious solution is to employ redundancy in the form of error-correcting codes to tolerate disk failures. This allows a redundant disk array to avoid losing data for much longer than an unprotected single disk. However, redundancy has negative consequences. Since all write operations must update the redundant information, the performance of writes in redundant disk arrays can be significantly worse than the performance of writes in nonredundant disk arrays. Also, keeping the redundant information consistent in the face of concurrent I/O operations and system crashes can be difficult.

A number of different data-stripping and redundancy schemes have been developed. The combinations and arrangements of these schemes lead to a bewildering set of options for users and designers of disk arrays. Each option presents subtle tradeoffs among reliability, performance, and cost that are difficult to evaluate without understanding the alternatives. To address this problem, this article presents a systematic tutorial and survey of disk arrays. We describe seven basic disk array organizations along with their advantages and disadvantages and compare their reliability, performance, and cost. We draw attention to the general principles governing the design and configuration of disk arrays as well as practical issues that must be addressed in the implementation of disk arrays. A later section describes optimizations and variations to the seven basic disk array organizations. Finally, we discuss existing research in the modeling of disk arrays and fruitful avenues for future research. This article should be of value to anyone interested in disk arrays, including students, researchers, designers, and users of disk arrays.

2. BACKGROUND

This section provides basic background material on disks, I/O data paths, and disk technology trends for readers who are unfamiliar with secondary storage systems.

2.1 Disk Terminology

Figure 1 illustrates the basic components of a simplified magnetic disk drive. A disk consists of a set of platters coated with a magnetic medium rotating at a constant angular velocity and a set of disk arms with magnetic read/write heads that are moved radially across the platters' surfaces by an actuator. Once the heads are correctly positioned, data is read and written in small arcs called sectors on the platters' surfaces as the platters rotate relative to the heads. Although all heads are moved collectively, in almost every disk drive, only a single head can read or write data at any given time. A complete circular swath of data is referred to as a track, and each platter's surface consists of concentric rings of tracks. A vertical collection of tracks at the same radial position is logically referred to as a cylinder. Sectors are numbered so that a sequential scan of all sectors traverses the entire disk in the minimal possible time.

Given the simplified disk described above, disk service times can be broken into three primary components: seek time, rotational latency, and data transfer time. Seek time is the amount of time needed to move a head to the correct radial position and typically ranges from...
Figure 1. Disk terminology Heads reside on arms which are positioned by actuators. Tracks are concentric rings on a platter. A sector is the basic unit of reads and writes. A cylinder is a stack of tracks at one actuator position. An HDA (head-disk assembly) is everything in the figure plus the air-tight casing. In some devices it is possible to transfer data from multiple surfaces simultaneously, but this is both rare and expensive. The collection of heads that participate in a single logical transfer that is spread over multiple surfaces is called a head group.

1 to 30 milliseconds depending on the seek distance and the particular disk. Rotational latency is the amount of time needed for the desired sector to rotate under the disk head. Full rotation times for disks vary currently from 8 to 28 milliseconds. The data transfer time is dependent on the rate at which data can be transferred to/from a platter's surface and is a function of the platter's rate of rotation, the density of the magnetic media, and the radial distance of the head from the center of the platter—some disks use a technique called zone-bit-recording to store more data on the longer outside tracks than the shorter inside tracks. Typical data transfer rates range from 1 to 5 MB per second. The seek time and rotational latency are sometimes collectively referred to as the head-positioning time. Table 1 tabulates the statistics for a typical high-end disk available in 1993.

The slow head-positioning time and fast data transfer rate of disks lead to very different performance for a sequence of accesses depending on the size and relative location of each access. Suppose we need to transfer 1 MB from the disk in Table 1, and the data is laid out in two ways: sequential within a single cylinder or randomly placed in 8 KB blocks. In either case the time for the actual data transfer of 1 MB is about 200 ms. But the time for positioning the head goes from about 16 ms in the sequential layout to about 2000 ms in the random layout. This sensitivity to the workload is why I/O-intensive applications are cate-

Table 1. Specifications for the Seagate ST43401N Elite-3 SCSI Disk Drive

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form Factor/Disk Diameter</td>
<td>5.25 inch</td>
</tr>
<tr>
<td>Capacity</td>
<td>2.8 GB</td>
</tr>
<tr>
<td>Cylinders</td>
<td>2627</td>
</tr>
<tr>
<td>Tracks Per Cylinder</td>
<td>21</td>
</tr>
<tr>
<td>Sectors Per Track</td>
<td>99</td>
</tr>
<tr>
<td>Bytes Per Sector</td>
<td>512</td>
</tr>
<tr>
<td>Full Rotation Time</td>
<td>11.1 ms</td>
</tr>
<tr>
<td>Minimum Seek (single cylinder)</td>
<td>1.7 ms</td>
</tr>
<tr>
<td>Average Seek (random cylinder to cylinder)</td>
<td>11.0 ms</td>
</tr>
<tr>
<td>Maximum Seek (full stroke seek)</td>
<td>22.5 ms</td>
</tr>
<tr>
<td>Data Transfer Rate</td>
<td>4.6 MB/s</td>
</tr>
</tbody>
</table>

Average seek in this table is calculated assuming a uniform distribution of accesses. This is the standard way manufacturers report average seek times. In reality, measurements of production systems show that spatial locality significantly lowers the effective average seek distance [Hennessy and Patterson 1990, p 559].
RAID - 14'9

categorized as high data rate, meaning minimal head positioning via large, sequential accesses, or high I/O rate, meaning lots of head positioning via small, more random accesses. For example, scientific programs that manipulate large arrays of data fall in the high data rate category, while transaction-processing programs fall in the high I/O rate category.

2.2 Data Paths

A hierarchy of industry standard interfaces has been defined for transferring data recorded on a disk platter's surface to or from a host computer. In this section we review the complete data path, from a disk to a users' application (Figure 2). We assume a read operation for the purposes of this discussion.

On the disk platter's surface, information is represented as reversals in the direction of stored magnetic fields. These "flux reversals" are sensed, amplified, and digitized into pulses by the lowest-level read electronics. The protocol ST506/412 is one standard that defines an interface to disk systems at this lowest, most inflexible, and technology-dependent level. Above this level of the read electronics path, pulses are decoded to separate data bits from timing-related flux reversals. The bit-level ESDI (Enhanced Small Device Interface) and SMD (Storage Module Interface) standards define an interface at this more flexible, encoding-independent level. Then, to be transformed into the highest, most flexible packet-level, these bits are aligned into bytes, error-correcting codes applied, and the extracted data delivered to the host as data blocks over a peripheral bus interface such as SCSI (Small Computer Standard Interface), or IPI-3 (the third level of the Intelligent Peripheral Interface). These steps are performed today by intelligent on-disk controllers, which often include speed matching and caching "track buffers." SCSI and IPI-3 also include a level of data mapping: the computer specifies a logical block number, and the controller embedded on the disk maps that block number to a physical cylinder, track, and sector. This mapping allows the embedded disk controller to avoid bad areas of the disk by remapping logical blocks that are affected to new areas of the disk.

Topologies and devices on the data path between disk and host computer vary widely depending on the size and type of I/O system. Mainframes have the richest I/O systems, with many devices and complex interconnection schemes to access them. An IBM channel path, which encompasses 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. The collection of disks that share the same pathway to the head of string is called a string. In the workstation/file server world, the channel processor is usually called an I/O controller or host-bus adaptor (HBA), and the functionality of the storage director and head of string is contained in an embedded controller on the disk drive. As in the mainframe world, the use of high-level peripheral interfaces such as SCSI and IPI-3 allow multiple disks to share a single peripheral bus or string.

From the HBA, data is transferred via direct memory access, over a system bus, such as VME (Versa Module Eurocard), S-Bus, MicroChannel, EISA (Extended Industry Standard Architecture), or PCI (Peripheral Component Interconnect), to the host operating system's buffers. Then, in most operating systems, the CPU performs a memory-to-memory copy over a high-speed memory bus from the operating system buffers to buffers in the application's address space.

2.3 Technology Trends

Much of the motivation for disk arrays comes from the current trends in disk technology. As Table 2 shows, magnetic disk drives have been improving rapidly by some metrics and hardly at all by other metrics. Smaller distances between the magnetic read/write head and the disk surface, more accurate positioning
electronics, and more advanced magnetic media have increased the recording density on the disks dramatically. This increased density has improved disks in two ways. First, it has allowed disk capacities to stay constant or increase, even while disk sizes have decreased from 5.25 inches in 1983 to 1.3 inches in 1993. Second, the increased density, along with an increase in the rotational speed of the disk, has made possible a substantial increase in the transfer rate of disk drives.

On the other hand, seek times have improved very little, only decreasing from approximately 20 ms in 1980 to 10 ms today. Rotational speeds have increased at a similarly slow rate from 3600 revolutions per minute in 1980 to 5400-7200 today.

3. DISK ARRAY BASICS
This section examines basic issues in the design and implementation of disk

Figure 2. Host-to-device pathways. Data that is read from a magnetic disk must pass through many layers on its way to the requesting processor. Each dashed line marks a standard interface. Lower interfaces such as ST506 deal more closely with the raw magnetic fields and are highly technology dependent. Higher layers such as SCSI deal in packets or blocks of data and are more technology independent. A string connects multiple disks to a single I/O controller. Control of the string is distributed between the I/O and disk controllers.
Magnetic disks are improving rapidly in density and capacity, but more slowly in performance. A real
density is the recording density per square inch of magnetic media. In 1989, IBM demonstrated a 1
Gbit/sq. inch density in a laboratory environment. Linear density is the number of bits written along a
track. Intertrack density refers to the number of concentric tracks on a single platter.

<table>
<thead>
<tr>
<th>Table 2. Trends in Disk Technology.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>Areal Density</td>
</tr>
<tr>
<td>Linear Density</td>
</tr>
<tr>
<td>Inter-Track Density</td>
</tr>
<tr>
<td>Capacity (3.5&quot; form factor)</td>
</tr>
<tr>
<td>Transfer Rate</td>
</tr>
<tr>
<td>Seek Time</td>
</tr>
</tbody>
</table>

In particular, we examine the concepts of data striping and redundancy; basic RAID organizations; performance and cost comparisons between the basic RAID organizations; reliability of RAID-based systems in the face of system crashes, uncorrectable bit-errors, and correlated disk failures; and finally, issues in the implementation of block-interleaved, redundant disk arrays.

3.1 Data Striping and Redundancy

Redundant disk arrays employ two orthogonal concepts: data striping for improved performance and redundancy for improved reliability. Data striping distributes data transparently over multiple disks to make them appear as a single fast, large disk. Stripping improves aggregate I/O performance by allowing multiple I/Os to be serviced in parallel. There are two aspects to this parallelism. First, multiple independent requests can be serviced in parallel by separate disks. This decreases the queuing time seen by I/O requests. Second, single multiple-block requests can be serviced by multiple disks acting in coordination. This increases the effective transfer rate seen by a single request. The more disks in the disk array, the larger the potential performance benefits. Unfortunately, a large number of disks lowers the overall reliability of the disk array, as mentioned before. Assuming independent failures, 100 disks collectively have only 1/100th the reliability of a single disk. Thus, redundancy is necessary to tolerate disk failures and allow continuous operation without data loss.

We will see that the majority of redundant disk array organizations can be distinguished based on two features: (1) the granularity of data interleaving and (2) the method and pattern in which the redundant information is computed and distributed across the disk array. Data interleaving can be characterized as either fine grained or coarse grained. Fine-grained disk arrays conceptually interleave data in relatively small units so that all I/O requests, regardless of their size, access all of the disks in the disk array. This results in very high data transfer rates for all I/O requests but has the disadvantages that (1) only one logical I/O request can be in service at any given time and (2) all disks must waste time positioning for every request.
Coarse-grained disk arrays interleave data in relatively large units so that small I/O requests need access only a small number of disks while large requests can access all the disks in the disk array. This allows multiple small requests to be serviced simultaneously while still allowing large requests to see the higher transfer rates afforded by using multiple disks.

The incorporation of redundancy in disk arrays brings up two somewhat orthogonal problems. The first problem is to select the method for computing the redundant information. Most redundant disk arrays today use parity, though some use Hamming or Reed-Solomon codes. The second problem is that of selecting a method for distributing the redundant information across the disk array. Although there are an unlimited number of patterns in which redundant information can be distributed, we classify these patterns roughly into two different distributions schemes, those that concentrate redundant information on a small number of disks and those that distributed redundant information uniformly across all of the disks. Schemes that uniformly distribute redundant information are generally more desirable because they avoid hot spots and other load-balancing problems suffered by schemes that do not distribute redundant information uniformly. Although the basic concepts of data striping and redundancy are conceptually simple, selecting between the many possible data striping and redundancy schemes involves complex trade-offs between reliability, performance, and cost.

3.2 Basic RAID Organizations

This section describes the basic RAID organizations that will be used as the basis for further examinations of the performance, cost, and reliability of disk arrays. In addition to presenting RAID levels 1 through 5 that first appeared in the landmark paper by Patterson, Gibson, and Katz [Patterson et al. 1988], we present two other RAID organizations, RAID levels 0 and 6, that have since become generally accepted.\(^2\) For the benefit of those unfamiliar with the original numerical classification of RAID, we will use English phrases in preference to the numerical classifications. It should come as no surprise to the reader that even the original authors have been confused sometimes with regard to the disk array organization referred to by a particular RAID level! Figure 3 illustrates the seven RAID organizations schematically.

3.2.1 Nonredundant (RAID Level 0)

A nonredundant disk array, or RAID level 0, has the lowest cost of any RAID organization because it does not employ redundancy at all. This scheme offers the best write performance since it does not need to update redundant information. Surprisingly, it does not have the best read performance. Redundancy schemes that duplicate data, such as mirroring, can perform better on reads by selectively scheduling requests on the disk with the shortest expected seek and rotational delays [Bitton and Gray 1988]. Without redundancy, any single disk failure will result in data loss. Nonredundant disk arrays are widely used in supercomputing environments where performance and capacity, rather than reliability, are the primary concerns.

3.2.2 Mirrored (RAID Level 1)

The traditional solution, called mirroring or shadowing, uses twice as many disks as a nonredundant disk array [Bitton and Gray 1988]. Whenever data is written to a disk the same data is also written to a redundant disk, so that there are always two copies of the information. When data is read, it can be retrieved from the disk with the shorter queuing, seek, and rotational delays [Chen et al. 1990]. If a disk fails, the other copy is used to service requests. Mirroring is frequently used in

\(^2\)Strictly speaking, RAID level 0 is not a type of redundant array of inexpensive disks since it stores no error-correcting codes.
database applications where availability and transaction rate are more important than storage efficiency [Gray et al. 1990].

3.2.3 Memory-Style ECC (RAID Level 2)
Memory systems have provided recovery from failed components with much less cost than mirroring by using Hamming codes [Peterson and Weldon 1972]. Hamming codes contain parity for distinct overlapping subsets of components. In one version of this scheme, four data disks require three redundant disks, one less than mirroring. Since the number of redundant disks is proportional to the log
of the total number of disks in the system, storage efficiency increases as the number of data disks increases.

If a single component fails, several of the parity components will have inconsistent values, and the failed component is the one held in common by each incorrect subset. The lost information is recovered by reading the other components in a subset, including the parity component, and setting the missing bit to 0 or 1 to create the proper parity value for that subset. Thus, multiple redundant disks are needed to identify the failed disk, but only one is needed to recover the lost information.

Readers unfamiliar with parity can think of the redundant disk as having the sum of all the data in the other disks. When a disk fails, you can subtract all the data on the good disks from the parity disk; the remaining information must be the missing information. Parity is simply this sum modulo two.

3.2.4 Bit-Interleaved Parity (RAID Level 3)
One can improve upon memory-style ECC disk arrays by noting that, unlike memory component failures, disk controllers can easily identify which disk has failed. Thus, one can use a single parity disk rather than a set of parity disks to recover lost information.

In a bit-interleaved parity disk array, data is conceptually interleaved bit-wise over the data disks, and a single parity disk is added to tolerate any single disk failure. Each read request accesses all data disks, and each write request accesses all data disks and the parity disk. Thus, only one request can be serviced at a time. Because the parity disk contains only parity and no data, the parity disk cannot participate on reads, resulting in slightly lower read performance than for redundancy schemes that distribute the parity and data over all disks. Bit-interleaved parity disk arrays are frequently used in applications that require high bandwidth but not high I/O rates. Also they are simpler to implement than RAID Levels 4, 5, and 6.

3.2.5 Block-Interleaved Parity (RAID Level 4)
The block-interleaved parity disk array is similar to the bit-interleaved parity disk array except that data is interleaved across disks in blocks of arbitrary size rather than in bits. The size of these blocks is called the striping unit [Chen and Patterson 1990]. Read requests smaller than the striping unit are satisfied by accessing only a single data disk. Write requests must update the requested data blocks and must compute and update the parity block. For large writes that touch blocks on all disks, parity is easily computed by exclusive-oring the new data for each disk. For small write requests that update only one data disk, parity is computed by noting how the new data differs from the old data and applying those differences to the parity block. Small write requests thus require four disk I/Os: one to write the new data, two to read the old data and old parity for computing the new parity, and one to write the new parity. This is referred to as a read-modify-write procedure. Because a block-interleaved parity disk array has only one parity disk, which must be updated on all write operations, the parity disk can easily become a bottleneck. Because of this limitation, the block-interleaved distributed-parity disk array is universally preferred over the block-interleaved parity disk array.

3.2.6 Block-Interleaved Distributed-Parity (RAID Level 5)
The block-interleaved distributed-parity disk array eliminates the parity disk bottleneck present in the block-interleaved parity disk array by distributing the parity uniformly over all of the disks. An additional, frequently overlooked advantage to distributing the parity is that it also distributes data over all of the disks rather than over all but one. This allows all disks to participate in servicing read operations in contrast to redundancy schemes with dedicated parity disks in which the parity disk cannot participate in servicing read requests. Block-inter-
leaved distributed-parity disk arrays have the best small read, large read, and large write performance of any redundant disk array. Small write requests are somewhat inefficient compared with redundancy schemes such as mirroring however, due to the need to perform read-modify-write operations to update parity. This is the major performance weakness of RAID level-5 disk arrays and has been the subject of intensive research [Menon et al. 1993; Stodolsky and Gibson 1993].

The exact method used to distribute parity in block-interleaved distributed-parity disk arrays can affect performance. Figure 4 illustrates the best parity distribution of those investigated in [Lee and Katz 1991b], called the left-symmetric parity distribution. A useful property of the left-symmetric parity distribution is that whenever you traverse the striping units sequentially, you will access each disk once before accessing any disk twice. This property reduces disk conflicts when servicing large requests.

3.2.7 P + Q Redundancy (RAID Level 6)

Parity is a redundancy code capable of correcting any single self-identifying failure. As larger disk arrays are considered, multiple failures are possible, and stronger codes are needed [Burkhard and Menon 1993]. Moreover, when a disk fails in a parity-protected disk array, recovering the contents of the failed disk requires a successful reading of the contents of all nonfailed disks. As we will see in Section 3.4, the probability of encountering an uncorrectable read error during recovery can be significant. Thus, applications with more stringent reliability requirements require stronger error-correcting codes.

One such scheme, called P + Q redundancy, uses Reed-Solomon codes to protect against up to two disk failures using the bare minimum of two redundant disks. The P + Q redundant disk arrays are structurally very similar to the block-interleaved distributed-parity disk arrays and operate in much the same manner. In particular, P + Q redundant disk arrays also perform small write operations using a read-modify-write procedure, except that instead of four disk accesses per write requests, P + Q redundant disk arrays require six disk accesses due to the need to update both the “P” and “Q” information.

3.3 Performance and Cost Comparisons

The three primary metrics in the evaluation of disk arrays are reliability, performance, and cost. RAID levels 0 through 6 cover a wide range of tradeoffs among these metrics. It is important to consider all three metrics to understand fully the value and cost of each disk array organization. In this section, we compare RAID levels 0 through 6 based on performance and cost. The following section examines reliability.

3.3.1 Ground Rules and Observations

While there are only three primary metrics in the evaluation of disk arrays...
(reliability, performance, and cost), there are many different ways to measure each metric and an even larger number of ways of using them. For example, should performance be measured in I/Os per second, bytes per second, or response time? Would a hybrid metric such as I/Os per second per dollar be more appropriate? Once a metric is agreed upon, should we compare systems at the same cost, the same total user capacity, the same performance, or the same reliability? The method one uses depends largely on the purpose of the comparison and the intended use of the system. In time-sharing applications, the primary metric may be user capacity per dollar; in transaction-processing applications the primary metric may be I/Os per second per dollar; and in scientific applications, the primary metric may be bytes per second per dollar. In certain heterogeneous systems, such as file servers, both I/Os per second and bytes per second may be important. In many cases, these metrics may all be conditioned on meeting a reliability threshold.

Most large secondary storage systems, and disk arrays in particular, are throughput oriented. That is, generally we are more concerned with the aggregate throughput of the system than, for example, its response time on individual requests (as long as requests are satisfied within a specified time limit). Such a bias has a sound technical basis: as techniques such as asynchronous I/O, prefetching, read caching, and write buffering become more widely used, fast response time depends on sustaining a high throughput.

In throughput-oriented systems, performance can potentially increase linearly as additional components are added; if one disk provides 30 I/Os per second, 2 should provide 60 I/Os per second. Thus, in comparing the performance of disk arrays, we will normalize the performance of the system by its cost. In other words we will use performance metrics such as I/Os per second per dollar rather than the absolute number of I/Os per second.

Even after the metrics are agreed upon, one must decide whether to compare systems of equivalent capacity, cost, or some other metric. We chose to compare systems of equivalent file capacity where file capacity is the amount of information the file system can store on the device and excludes the storage used for redundancy. Comparing systems with the same file capacity makes it easy to choose equivalent workloads for two different redundancy schemes. Were we to compare systems with different file capacities, we would be confronted with tough choices such as how a workload on a system with user capacity X maps onto a system with user capacity 2X.

Finally, there is currently much confusion in comparisons of RAID levels 1 through 5. The confusion arises because a RAID level sometimes specifies not a specific implementation of a system but rather its configuration and use. For example, a RAID level-5 disk array (block-interleaved distributed parity) with a parity group size of two is comparable to RAID level 1 (mirroring) with the exception that in a mirrored disk array, certain disk-scheduling and data layout optimizations can be performed that, generally, are not implemented for RAID level-5 disk arrays [Hsiao and DeWitt 1990; Orji and Solworth 1993]. Analogously, a RAID level-5 disk array can be configured to operate equivalently to a RAID level-3 disk array by choosing a unit of data striping such that the smallest unit of array access always accesses a full parity stripe of data. In other words, RAID level-1 and RAID level-3 disk arrays can be viewed as a subclass of RAID level-5 disk arrays. Since RAID level-2 and RAID level-4 disk arrays are, practically speaking, in all ways inferior to RAID level-5 disk arrays, the problem of selecting among RAID levels 1 through 5 is a subset of the more general problem of choosing an appropriate parity group size and striping unit size for RAID level-5 disk arrays. A parity group size close to two may indicate the use of RAID level-1 disk arrays; a striping unit much smaller than the size of an average request may...
Table 3. Throughput per Dollar Relative to RAID Level 0.

<table>
<thead>
<tr>
<th>RAID level</th>
<th>Small Read</th>
<th>Small Write</th>
<th>Large Read</th>
<th>Large Write</th>
<th>Storage Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID level 0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAID level 1</td>
<td>1</td>
<td>1/2</td>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>RAID level 3</td>
<td>1/G</td>
<td>1/G</td>
<td>(G-1)/G</td>
<td>(G-1)/G</td>
<td>(G-1)/G</td>
</tr>
<tr>
<td>RAID level 5</td>
<td>1</td>
<td>max(1/G,1/4)</td>
<td>1</td>
<td>(G-1)/G</td>
<td>(G-1)/G</td>
</tr>
<tr>
<td>RAID level 6</td>
<td>1</td>
<td>max(1/G,1/6)</td>
<td>1</td>
<td>(G-2)/G</td>
<td>(G-2)/G</td>
</tr>
</tbody>
</table>

This table compares the throughputs of various redundancy schemes for four types of I/O requests. Small here refers to I/O requests of one striping unit; large refers to I/O requests of one full stripe (one stripe unit from each disk in an error correction group). G refers to the number of disks in an error correction group. In all cases, the higher the number the better. The entries in this table account for the major performance effects but not some of the second-order effects. For instance, since RAID level 1 stores two copies of the data, a common optimization is to read dynamically the disk whose positioning time to the data is smaller.

indicate the use of a RAID level-3 disk array.

3.3.2 Comparisons

Table 3 tabulates the maximum throughput per dollar relative to RAID level 0 for RAID levels 0, 1, 3, 5, and 6. The cost of each system is assumed to be proportional to the total number of disks in the disk array. Thus, the table illustrates that given equivalent cost RAID level-0 and RAID level-1 systems, the RAID level-1 system can sustain half the number of small writes per second that a RAID level-0 system can sustain. Equivalently, we can say that the cost of small writes is twice as expensive in a RAID level-1 system as in a RAID level-0 system. In addition to performance, the table shows the storage efficiency of each disk array organization. The storage efficiency is approximately inverse to the cost of each unit of user capacity relative to a RAID level-0 system. For the above disk array organizations, the storage efficiency is equal to the performance/cost metric for large writes.

Figure 5 graphs the performance/cost metrics from Table 3 for RAID levels 1, 3, 5, and 6 over a range of parity group sizes. The performance/cost of RAID level-1 systems is equivalent to the performance/cost of RAID level-5 systems when the parity group size is equal to two. The performance/cost of RAID level-3 systems is always less than or equal to the performance/cost of RAID level-5 systems. This is expected given that a RAID level-3 system is a subclass of RAID level-5 systems derived by restricting the striping unit size such that all requests access exactly a parity stripe of data. Since the configuration of RAID level-5 systems is not subject to such a restriction, the performance/cost of RAID level-5 systems can never be less than that of an equivalent RAID level-3 system. It is important to stress that these performance/cost observations apply only to the abstract models of disk arrays for which we have formulated performance/cost metrics. In reality, a specific implementation of a RAID level-3 system can have better performance/cost than a specific implementation of a RAID level-5 system.

As previously mentioned, the question of which RAID level to use is often better expressed as more general configuration questions concerning the size of the parity group and striping unit. If a parity group size of two is indicated, then mirroring is desirable. If a very small striping unit is indicated then a RAID level-3 system may be sufficient. To aid the reader in evaluating such decisions, Figure 6 plots the four performance/cost
metrics from Table 3 on the same graph for each of the RAID levels 3, 5, and 6. This makes the performance/cost trade-offs explicit in choosing an appropriate parity group size. Section 4.4 addresses how to choose the unit of striping.

3.4 Reliability

Reliability is as important a metric to many I/O systems as performance and cost, and it is perhaps the main reason for the popularity of redundant disk arrays. This section starts by reviewing the basic reliability provided by a block-interleaved parity disk array and then lists three factors that can undermine the potential reliability of disk arrays.

3.4.1 Basic Reliability

Redundancy in disk arrays is motivated by the need to overcome disk failures. When only independent disk failures are considered, a simple parity scheme works admirably. Patterson et al. [1988] derive the mean time between failures for a RAID level 5 to be

\[ \frac{MTTF(disk)^2}{N \times (G - 1) \times MTTR(disk)} \]

where \( MTTF(disk) \) is the mean-time-to-failure (MTTF) of a single disk, \( MTTR(disk) \) is the mean-time-to-repair (MTTR) of a single disk, \( N \) is the total
number of disks in the disk array, and $G$ is the parity group size. For illustration purposes, let us assume we have 100 disks that each had a mean time to failure of 200,000 hours and a mean time to repair of one hour. If we organized these 100 disks into parity groups of average size 16, then the mean time to failure of the system would be an astounding 3000 years. Mean times to failure of this magnitude lower the chances of failure over any given period of time.

For a disk array with two redundant disk per parity group, such as $P + Q$ redundancy, the mean time to failure is

$$MTTF^3(disk) \over N \times (G - 1) \times (G - 2) \times MTTR^3(disk)$$

Using the same values for our reliability parameters, this implies an astronomically large mean time to failure of 38 million years.

This is an idealistic picture, but it gives us an idea of the potential reliability afforded by disk arrays. The rest of this section takes a more realistic look at the reliability of block-interleaved disk arrays by considering factors such as system crashes, uncorrectable bit-errors, and correlated disk failures that can dramatically affect the reliability of disk arrays.

3.4.2 System Crashes and Parity Inconsistency

In this section, the term system crash refers to any event such as a power failure, operator error, hardware