FAST, SCALABLE, RELIABLE DATA STORAGE

Managing RAID *on* LINUX

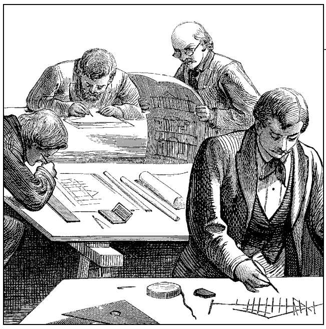
,

Managing RAID on LINUX

Derek Vadala

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo



CHAPTER 2

Planning and Architecture

Choosing the right RAID solution can be a daunting task. Buzzwords and marketing often cloud administrators' understanding of RAID technology. Conflicting information can cause inexperienced administrators to make mistakes. It is not unnatural to make mistakes when architecting a complicated system. But unfortunately, deadlines and financial considerations can make any mistakes catastrophic. I hope that this book, and this chapter in particular, will leave you informed enough to make as few mistakes as possible, so you can maximize both your time and the resources you have at your disposal. This chapter will help you pick the best RAID solution by first selecting which RAID level to use and then focusing on the following areas:

- Hardware costs
- Scalability
- Performance and redundancy

Hardware or Software?

RAID, like many other computer technologies, is divided into two camps: hardware and software. Software RAID uses the computer's CPU to perform RAID operations and is implemented in the kernel. Hardware RAID uses specialized processors, usually found on disk controllers, to perform array management functions. The choice between software and hardware is the first decision you need to make.

Software (Kernel-Managed) RAID

Software RAID means that an array is managed by the kernel, rather than by specialized hardware (see Figure 2-1). The kernel keeps track of how to organize data on many disks while presenting only a single virtual device to applications. This virtual device works just like any normal fixed disk.

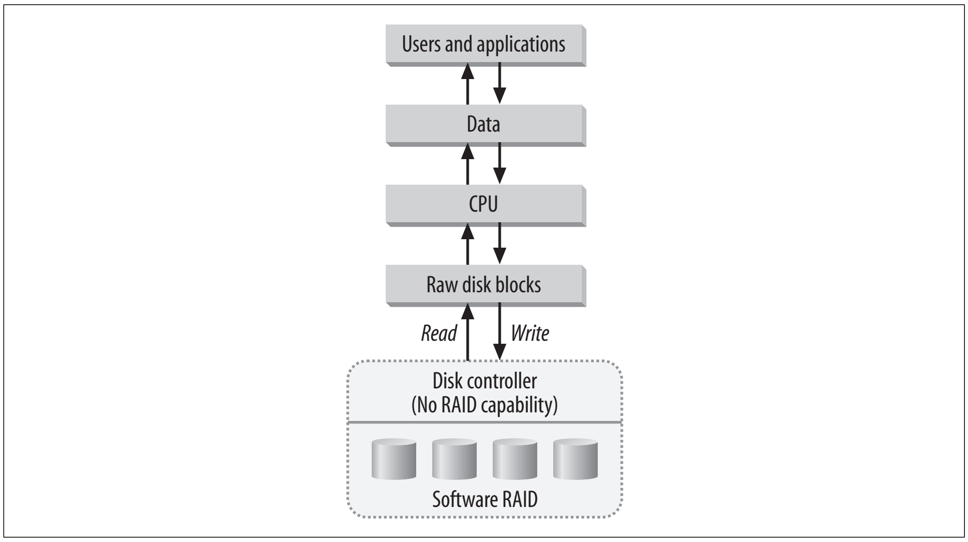


Figure 2-1. Software RAID uses the kernel to manage arrays.

Software RAID has unfortunately fallen victim to a FUD (fear, uncertainty, doubt) campaign in the system administrator community. I can't count the number of system administrators whom I've heard completely disparage all forms of software RAID, irrespective of platform. Many of these same people have admittedly not used software RAID in several years, if at all.

Why the stigma? Well, there are a couple of reasons. For one, when software RAID first saw the light of day, computers were still slow and expensive (at least by today's standards). Offloading a high-performance task like RAID I/O onto a CPU that was likely already heavily overused meant that performing fundamental tasks such as file operations required a tremendous amount of CPU overhead. So, on heavily saturated systems, the simple task of calling the *stat*[*] function could be extremely slow when compared to systems that didn't have the additional overhead of managing RAID arrays. But today, even multiprocessor systems are both inexpensive and common. Previously, multiprocessor systems were very expensive and unavailable to typical PC consumers. Today, anyone can build a multiprocessor system using affordable PC hardware. This shift in hardware cost and availability makes software RAID attractive because Linux runs well on common PC hardware. Thus, in cases when a single-processor system isn't enough, you can cost-effectively add a second processor to augment system performance.

Another big problem was that software RAID implementations were part of proprietary operating systems. The vendors promoted software RAID as a value-added

[*] The *stat(2)* system call reports information about files and is required for many commonplace activities like the *ls* command.

incentive for customers who couldn't afford hardware RAID, but who needed a way to increase disk performance and add redundancy. The problem here was that closed-source implementations, coupled with the fact that software RAID wasn't a priority in OS development, often left users with buggy and confusing packages.

Linux, on the other hand, has a really good chance to change the negative perceptions of software RAID. Not only is Linux's software RAID open source, the inexpensive hardware that runs Linux finally makes it easy and affordable to build reliable software RAID systems. Administrators can now build systems that have sufficient processing power to deal with day-to-day user tasks and high-performance system functions, like RAID, at the same time. Direct access to developers and a helpful user base doesn't hurt, either.

If you're still not convinced that software RAID is worth your time, then don't fret. There are also plenty of hardware solutions available for Linux.

Hardware

Hardware RAID means that arrays are managed by specialized disk controllers that contain RAID firmware (embedded software). Hardware solutions can appear in several forms. RAID controller cards that are directly attached to drives work like any normal PCI disk controller, with the exception that they are able to internally administer arrays. Also available are external storage cabinets that are connected to high-end SCSI controllers or network connections to form a *Storage Area Network (SAN)*. There is one common factor in all these solutions: the operating system accesses only a single block device because the array itself is hidden and managed by the controller.

Large-scale and expensive hardware RAID solutions are typically faster than software solutions and don't require additional CPU overhead to manage arrays. But Linux's software RAID can generally outperform low-end hardware controllers. That's partly because, when working with Linux's software RAID, the CPU is much faster than a RAID controller's onboard processor, and also because Linux's RAID code has had the benefit of optimization through peer review.

The major trade-off you have to make for improved performance is lack of support, although costs will also increase. While hardware RAID cards for Linux have become more ubiquitous and affordable, you may not have some things you traditionally get with Linux. Direct access to developers is one example. Mailing lists for the Linux kernel and for the RAID subsystem are easily accessible and carefully read by the developers who spend their days working on the code. With some exceptions, you probably won't get that level of support from any disk controller vendor—at least not without paying extra.

Another trade-off in choosing a hardware-based RAID solution is that it probably won't be open source. While many vendors have released cards that are supported under Linux, a lot of them require you to use closed-source components. This means that you won't be able to fix bugs yourself, add new features, or customize the code to meet your needs. Some manufacturers provide open source drivers while providing only closed-source, binary-only management tools, and vice versa. No vendors provide open source firmware. So if there is a problem with the software embedded on the controller, you are forced to wait for a fix from the vendor—and that could impact a data recovery effort! With software RAID, you could write your own patch or pay someone to write one for you straightaway.

RAID controllers

Some disk controllers internally support RAID and can manage disks without the help of the CPU (see Figure 2-2). These RAID cards handle all array functions and present the array as a standard block device to Linux. Hardware RAID cards usually contain an onboard BIOS that provides the management tools for configuring and maintaining arrays. Software packages that run at the OS level are usually provided as a means of post-installation array management. This allows administrators to maintain RAID devices without rebooting the system.

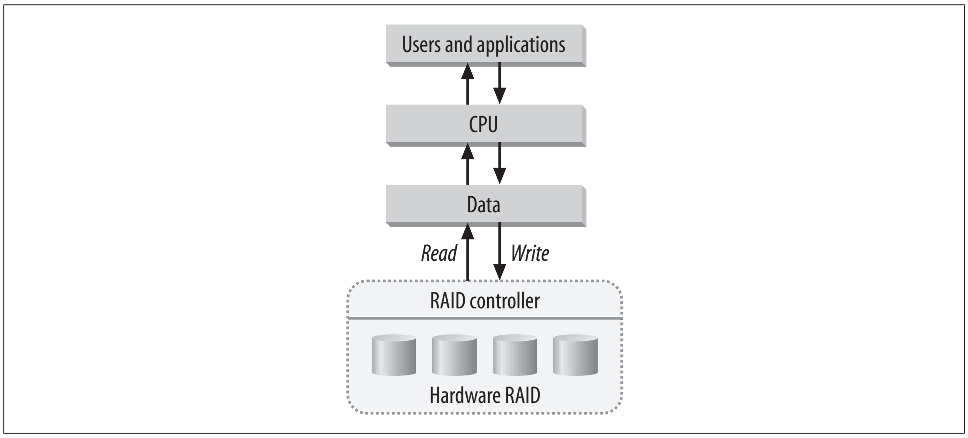


Figure 2-2. Disk controllers shift the array functions off the CPU, yielding an increase in performance.

While a lot of card manufacturers have recently begun to support Linux, it's important to make sure that the card you're planning to purchase is supported under Linux. Be sure that your manufacturer provides at least a loadable kernel module, or, ideally, open source drivers that can be statically compiled into the kernel. Open source drivers are always preferred over binary-only kernel modules. If you are stuck using a binary-only module, you won't get much support from the Linux community because without access to source code, it's quite impossible for them to diagnose interoperability problems between proprietary drivers and the Linux kernel. Luckily, several vendors either provide open source drivers or have allowed kernel hackers to develop their own. One shining example is Mylex, which sells RAID controllers. Their open source drivers are written by Leonard Zubkoff[*] of Dandelion Digital and can be managed through a convenient interface under the */proc* filesystem. Chapter 5 discusses some of the cards that are currently supported by Linux.

Outboard solutions

The second hardware alternative is a turnkey solution, usually found in outboard drive enclosures. These enclosures are typically connected to the system through a standard or high-performance SCSI controller. It's not uncommon for these specialized systems to support multiple SCSI connections to a single system, and many of them even provide directly accessible network storage, using NFS and other protocols.

These outboard solutions generally appear to an operating system as a standard SCSI block device or network mount point (see Figure 2-3) and therefore don't usually require any special kernel modules or device drivers to function. These solutions are often extremely expensive and operate as black box devices, in that they are almost always proprietary solutions. Outboard RAID boxes are nonetheless highly popular among organizations that can afford them. They are highly configurable and their modular construction provides quick and seamless, although costly, replacement options. Companies like EMC and Network Appliance specialize in this arena.

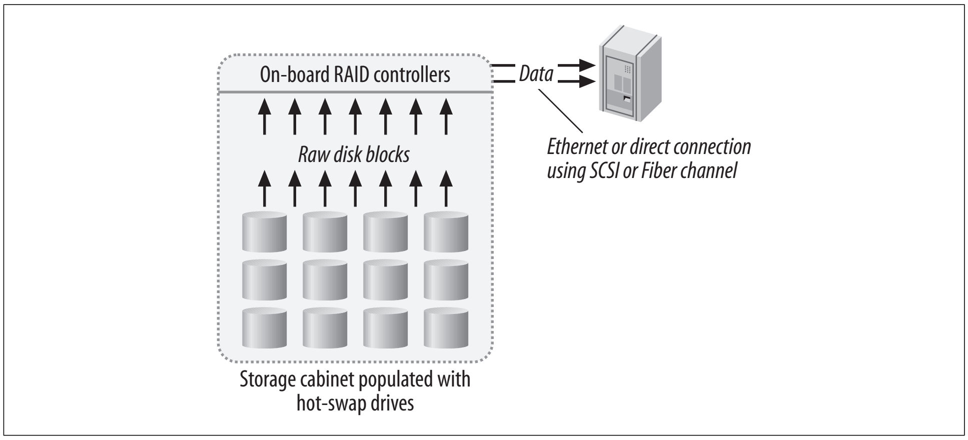

On-board RAID controllers
Data
Raw disk blocks
Ethernet or direct connection using SCSI or Fiber channel
Storage cabinet populated with hot-swap drives

Figure 2-3. Outboard RAID systems are internally managed and connected to a system to which they appear as a single hard disk.

[*] Leonard Zubkoff was very sadly killed in a helicopter crash on August 29, 2002. I learned of his death about a week later, as did many in the open source community. I didn't know Leonard personally. We'd had only one email exchange, earlier in the summer of 2002, in which he had graciously agreed to review material I had written about the Mylex driver. His site remains operational, but I have created a mirror at *http://dandelion.cynicism.com/*, which I will maintain indefinitely.

If you can afford an outboard RAID system and you think it's the best solution for your project, you will find them reliable performers. Do not forget to factor support costs into your budget. Outboard systems not only have a high entry cost, but they are also costly to maintain. You might also consider factoring spare parts into your budget, since a system failure could otherwise result in downtime while you are waiting for new parts to arrive. In most cases, you will not be able to find replacement parts for an outboard system at local computer stores, and even if they are available, using them will more than likely void your warranty and support contracts.

I hope you will find the architectural discussions later in this chapter helpful when choosing a vendor. I've compiled a list of organizations that provide hardware RAID systems in the Appendix. But I urge you to consider the software solutions discussed throughout this book. Administrators often spend enormous amounts of money on solutions that are well in excess of their needs. After reading this book, you may find that you can accomplish what you set out to do with a lot less money and a little more hard work.

Storage Area Network (SAN)

SAN is a relatively new method of storage management, in which various storage platforms are interconnected on a separate, usually high-speed, network (see Figure 2-4). The SAN is then connected to local area networks (LANs) throughout an organization. It is not uncommon for a SAN to be connected to several different parts of a LAN so that users do not share a single path to the SAN. This prevents a network bottleneck and allows better throughput between users and storage systems. Typically, a SAN might also be exposed to satellite offices using wide area network (WAN) connections.

Many companies that produce turnkey RAID solutions also offer services for planning and implementing a SAN. In fact, even drive manufacturers such as IBM and Western Digital, as well as large network and telecommunications companies such as Lucent and Nortel Networks, now provide SAN solutions.

SAN is very expensive, but is quickly becoming a necessity for large, distributed organizations. It has become vital in backup strategies for large businesses and will likely grow significantly over the next decade. SAN is not a replacement for RAID; rather, RAID is at the heart of SAN. A SAN could be comprised of a robotic tape backup solution and many RAID systems. SAN uses data and storage management in a world where enormous amounts of data need to be stored, organized, and recalled at a moment's notice. A SAN is usually designed and implemented by vendors as a top-down solution that is customized for each organization. It is therefore not discussed further in this book.

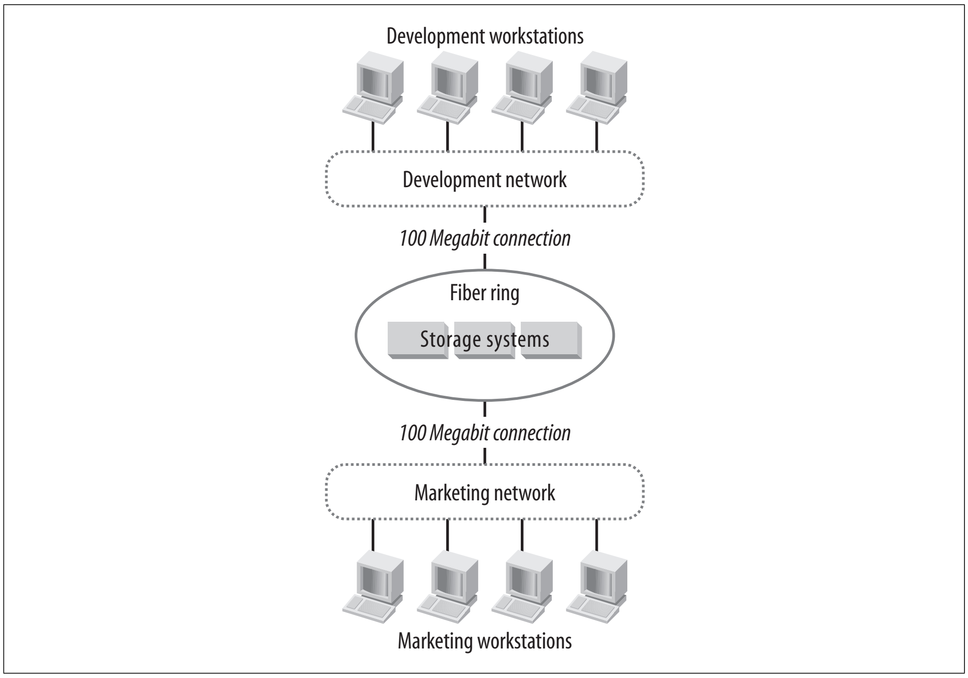


Figure 2-4. A simple SAN arrangement.

The RAID Levels: In Depth

It is important to realize that different implementations of RAID are suited to different applications and the wallets of different organizations. All implementations revolve around the basic levels first outlined in the Berkeley Papers. These core levels have been further expanded by software developers and hardware manufacturers. The RAID levels are not organized hierarchically, although vendors sometimes market their products to imply that there is a hierarchical advantage. As discussed in Chapter 1, the RAID levels offer varying compromises between performance and redundancy. For example, the fastest level offers no additional reliability when compared with a standalone hard disk. Choosing an appropriate level assumes that you have a good understanding of the needs of your applications and users. It may turn out that you have to sacrifice some performance to build an array that is more redundant. You can't have the best of both worlds.

The first decision you need to make when building or buying an array is how large it needs to be. This means talking to users and examining usage to determine how big your data is and how much you expect it to grow during the life of the array.

Table 2-1 briefly outlines the storage yield of the various RAID levels. It should give you a basic idea of how many drives you will need to purchase to build the initial array. Remember that RAID-2 and RAID-3 are now obsolete and therefore are not covered in this book.

Table 2-1. Realized RAID storage capacities

RAID level	Realized capacity
Linear mode	$DiskSize_0$+$DiskSize_1$+...$DiskSize_n$
RAID-0 (striping)	TotalDisks * DiskSize
RAID-1 (mirroring)	DiskSize
RAID-4	(TotalDisks-1) * DiskSize
RAID-5	(TotalDisks-1) * DiskSize
RAID-10 (striped mirror)	NumberOfMirrors * DiskSize
RAID-50 (striped parity)	(TotalDisks-ParityDisks) * DiskSize

Remember that you will eventually need to build a filesystem on your RAID device. Don't forget to take the size of the filesystem into account when figuring out how many disks you need to purchase. ext2 reserves five percent of the filesystem, for example. Chapter 6 covers filesystem tuning and high-performance filesystems, such as JFS, ext3, ReiserFS, XFS, and ext2.

The "RAID Case Studies: What Should I Choose?" section, later in this chapter, focuses on various environments in which different RAID levels make the most sense. Table 2-2 offers a quick comparison of the standard RAID levels.

Table 2-2. RAID level comparison

	RAID-1	Linear mode	RAID-0	RAID-4	RAID-5
Write performance	Slow writes, worse than a standalone disk; as disks are added, write performance declines	Same as a standalone disk	Best write performance; much better than a single disk	Comparable to RAID-0, with one less disk	Comparable to RAID-0, with one less disk for large write operations; potentially slower than a single disk for write operations that are smaller than the stripe size
Read performance	Fast read performance; as disks are added, read performance improves	Same as a standalone disk	Best read performance	Comparable to RAID-0, with one less disk	Comparable to RAID-0, with one less disk

Table 2-2. RAID level comparison (continued)

	RAID-1	Linear mode	RAID-0	RAID-4	RAID-5
Number of disk failures	N-1	0	0	1	1
Applications	Image servers; application servers; systems with little dynamic content/updates	Recycling old disks; no application-specific advantages		Same as RAID-5, which is a better alternative	File servers; databases

RAID-0 (Striping)

RAID-0 is sometimes referred to simply as striping; it was not included in the original Berkeley specification and is not, strictly speaking, a form of RAID because there is no redundancy. Under RAID-0, the host system or a separate controller breaks data into blocks and writes it to different disks in round-robin fashion (as shown in Figure 2-5).

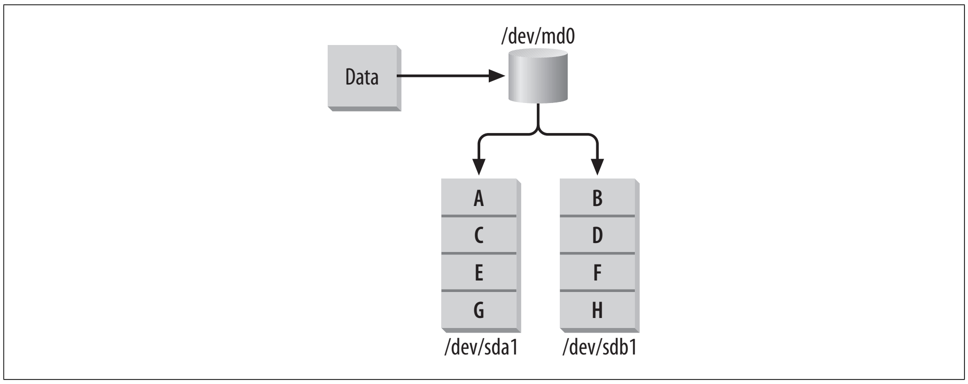


Figure 2-5. RAID-0 (striping) writes data consecutively across multiple drives.

This level yields the greatest performance and utilizes the maximum amount of available disk storage, as long as member disks are of identical sizes. Typically, if member disks are not of identical sizes, then each member of a striped array will be able to utilize only an amount of space equal to the size of the smallest member disk. Likewise, using member disks of differing speeds might introduce a bottleneck during periods of demanding I/O. See the "I/O Channels" and "Matched Drives" sections, later in this chaper, for more information on the importance of using identical disks and controllers in an array.

In some implementations, stripes are organized so that all available storage space is usable. To facilitate this, data is striped across all disks until the smallest disk is full. The process repeats until no space is left on the array. The Linux kernel implements stripes in this way, but if you are working with a hardware RAID controller, this behavior might vary. Check the available technical documentation or contact your vendor for clarification.

Because there is no redundancy in RAID-0, a single disk failure can wipe out all files. Striped arrays are best suited to applications that require intensive disk access, but where the potential for disk failure and data loss is also acceptable. RAID-O might therefore be appropriate for a situation where backups are easily accessible or where data is available elsewhere in the event of a system failure—on a load-balanced network, for example.

Disk striping is also well suited for video production applications because the high data transfer rates allow tremendous source files to be postprocessed easily. But users would be wise to keep copies of finished clips on another volume that is protected either by traditional backups or a more redundant RAID architecture. Usenet news sites have historically chosen RAID-0 because, while data is not critical, I/O throughput is essential for maintaining a large-volume news feed. Local groups and backbone sites can keep newsgroups for which they are responsible on separate fault-tolerant drives to additionally protect against data loss.

Linear Mode

Linux supports another non-RAID capability called linear (or sometimes append) mode. Linear mode sequentially concatenates disks, creating one large disk without data redundancy or increased performance (as shown in Figure 2-6).

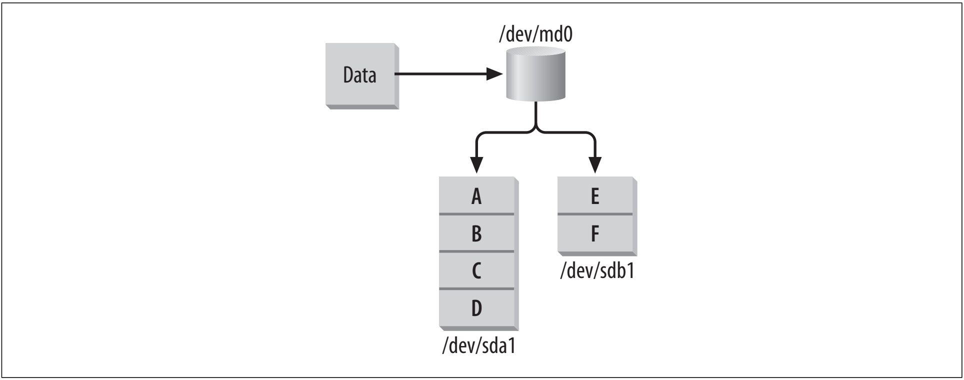


Figure 2-6. Linear (append) mode allows users to concatenate several smaller disks.

Linear arrays are most useful when working with disks and controllers of varying sizes, types, and speeds. Disks belonging to linear arrays are written to until they are full. Since data is not interleaved across the member disks, parallel operations that could be affected by a single disk bottleneck do not occur, as they can in RAID-0. No space is ever wasted when working with linear arrays, regardless of differing disk sizes. Over time, however, as data becomes more spread out over a linear array, you will see performance differences when accessing files that are on different disks of differing speeds and sizes, and when you access a file that spans more than one disk.

Like RAID-0, linear mode arrays offer no redundancy. A disk failure means complete data loss, although recovering data from a damaged array might be a bit easier than with RAID-0, because data is not interleaved across all disks. Because it offers no redundancy or performance improvement, linear mode is best left for desktop and hobbyist use.

Linear mode, and to a lesser degree, RAID-0, are also ideal for recycling old drives that might not have practical application when used individually. A spare disk controller can easily turn a stack of 2- or 3-gigabyte drives into a receptacle for storing movies and music to annoy the RIAA and MPAA.

RAID-1 (Mirroring)

RAID-1 provides the most complete form of redundancy because it can survive multiple disk failures without the need for special data recovery algorithms. Data is mirrored block-by-block onto each member disk (see Figure 2-7). So for every N disks in a RAID-1, the array can withstand a failure of N-1 disks without data loss. In a four-disk RAID-1, up to three disks could be lost without loss of data.

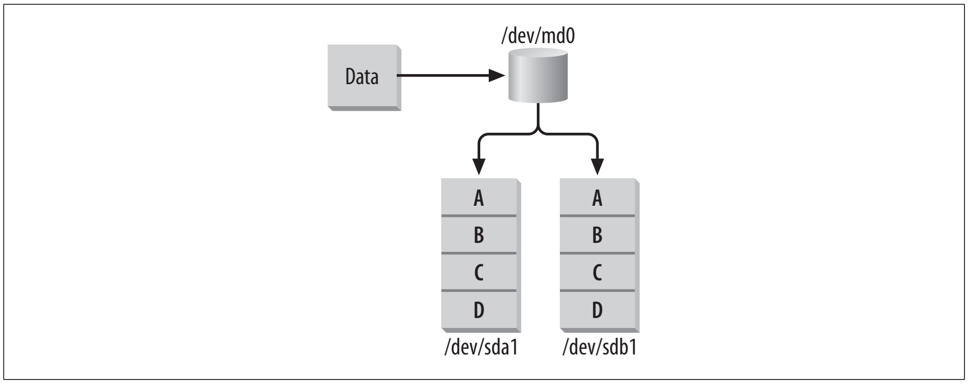


Figure 2-7. Fully redundant RAID-1.

As the number of member disks in a mirror increases, the write performance of the array decreases. Each write incurs a performance hit because each block must be written to each participating disk. However, a substantial advantage in read performance is achieved through parallel access. Duplicate copies of data on different hard drives allow the system to make concurrent read requests.

For example, let's examine the read and write operations of a two-disk RAID-1. Let's say that I'm going to perform a database query to display a list of all the customers that have ordered from my company this year. Fifty such customers exist, and each of their customer data records is 1 KB. My RAID-1 array receives a request to retrieve these fifty customer records and output them to my company's sales engineer. The drives in my array store data in 1 KB chunks and support a data throughput of 1 KB at a time. However, my controller card and system bus support a data throughput of 2 KB at a time. Because my data exists on more than one disk drive, I can utilize the full potential of my system bus and disk controller despite the limitation of my hard drives.

Suppose one of my sales engineers needs to change information about each of the same fifty customers. Now we need to write fifty records, each consisting of 1 KB. Unfortunately, we need to write each chunk of information to both drives in our array. So in this case, we need to write 100 KB of data to our disks, rather than 50 KB. The number of write operations increases with each disk added to a mirror array. In this case, if the array had four member disks, a total of 4 KB would be written to disk for each 1 KB of data passed to the array.

This example reveals an important distinction between hardware and software RAID-1. With software RAID, each write operation (one per disk) travels over the PCI bus to corresponding controllers and disks (see the sections "Motherboards and the PCI Bus" and "I/O Channels," later in this chapter). With hardware RAID, only a single write operation travels over the PCI bus. The RAID controller sends the proper number of write operations out to each disk. Thus, with hardware RAID-1, the PCI bus is less saturated with I/O requests.

Although RAID-1 provides complete fault tolerance, it is cost-prohibitive for some users because it at least doubles storage costs. However, for sites that require zero downtime, but are willing to take a slight hit on write performance, mirroring is ideal. Such sites might include online magazines and newspapers, which serve a large number of customers but have relatively static content. Online advertising aggregators that facilitate the distribution of banner ads to customers would also benefit from disk mirroring. If your content is nearly static, you won't suffer much from the write performance penalty, while you will benefit from the parallel read-as-you-serve image files. Full fault tolerance ensures that the revenue stream is never interrupted and that users can always access data.

RAID-1 works extremely well when servers are already load-balanced at the network level. This means usage can be distributed across multiple machines, each of which supports full redundancy. Typically, RAID-1 is deployed using two-disk mirrors. Although you could create mirrors with more disks, allowing the system to survive a multiple disk failure, there are other arrangements that allow comparable redundancy and read performance and much better write performance. See the "Hybrid Arrays" section, later in this chapter. RAID-1 is also well suited for system disks.

RAID-4

RAID-4 stripes block-sized chunks of data across each drive in the array marked as a data drive. In addition, one drive is designated as a dedicated parity drive (see Figure 2-8).

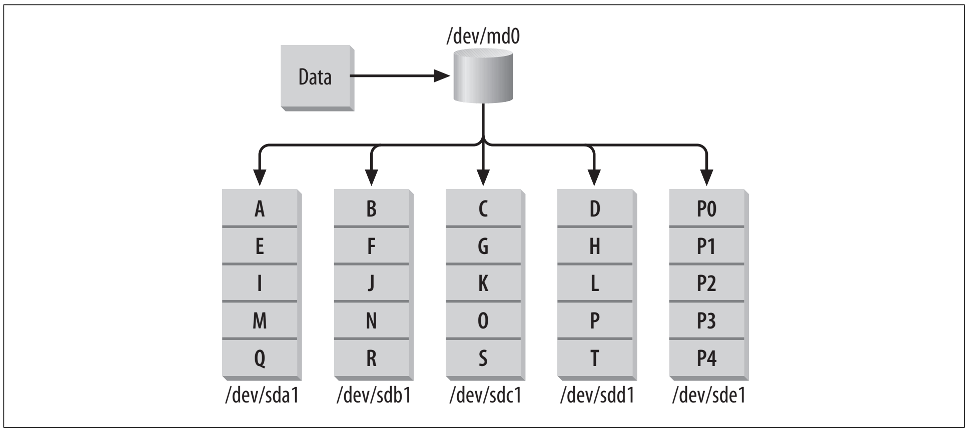


Figure 2-8. RAID-4 stripes data to all disks except a dedicated parity drive.

RAID-4 uses an exclusive OR (XOR) operation to generate checksum information that can be used for disaster recovery. Checksum information is generated during each write operation at the block level. The XOR operation uses the dedicated parity drive to store a block containing checksum information derived from the blocks on the other disks.

In the event of a disk failure, an XOR operation can be performed on the checksum information and the parallel data blocks on the remaining member disks. Users and applications can continue to access data in the array, but performance is degraded because the XOR operation must be called during each read to reconstruct the missing data. When the failed disk is replaced, administrators can rebuild the data from the failed drive using the parity information on the remaining disks. By sequentially performing an XOR on all parallel blocks and writing the result to the new drive, data is restored.

Although the original RAID specification called for only a single dedicated parity drive in RAID-4, some modern implementations allow the use of multiple dedicated parity drives. Since each write generates parity information, a bottleneck is inherent in RAID-4.

XOR

The exclusive OR (XOR) is a logical operation that returns a `TRUE` value if and only if one of the operands is `TRUE`. If both operands are `TRUE`, then a value of `FALSE` is returned.

p	q	p XOR q
T	T	F
T	F	T
F	T	T
F	F	T

When a parity RAID generates its checksum information, it performs the XOR on each data byte. For example, a RAID-5 with three member disks writes the byte 11011011 binary to the first disk and the byte 01101100 to the second disk. The first two bytes are user data. Next, a parity byte of 10110111 is written to the third disk. If a byte is lost because of the failure of either the first or the second disk, the array can perform the XOR operation on the other data byte and the parity information in order to retrieve the missing data byte. This holds true for any number of data bytes or, in our case, disks.

Placing the parity drive at the beginning of an I/O channel and giving it the lowest SCSI ID in that chain will help improve performance. Using a dedicated channel for the parity drive is also recommended.

It is very unlikely that RAID-4 makes sense for any modern setup. With the exception of some specialized, turnkey RAID hardware, RAID-4 is not often used. RAID-5 provides better performance and is likely a better choice for anyone who is considering RAID-4. It's prudent to mention here, however, that many NAS vendors still use RAID-4 simply because online array expansion is easier to implement and expansion is faster than with RAID-5. That's because you don't need to reposition all the parity blocks when you expand a RAID-4.

Dedicating a drive for parity information means that you lose one drive's worth of potential data storage when using RAID-4. When using N disk drives, each with space S, and dedicating one drive for parity storage, you are left with (N-1) * S space under RAID-4. When using more than one parity drive, you are left with (N-P) * S space, where P represents the total number of dedicated parity drives in the array.

RAID-5

RAID-5 eliminates the use of a dedicated parity drive and stripes parity information across each disk in the array, using the same XOR algorithm found in RAID-4 (see

Figure 2-9). During each write operation, one chunk worth of data in each stripe is used to store parity. The disk that stores parity alternates with each stripe, until each disk has one chunk worth of parity information. The process then repeats, beginning with the first disk.

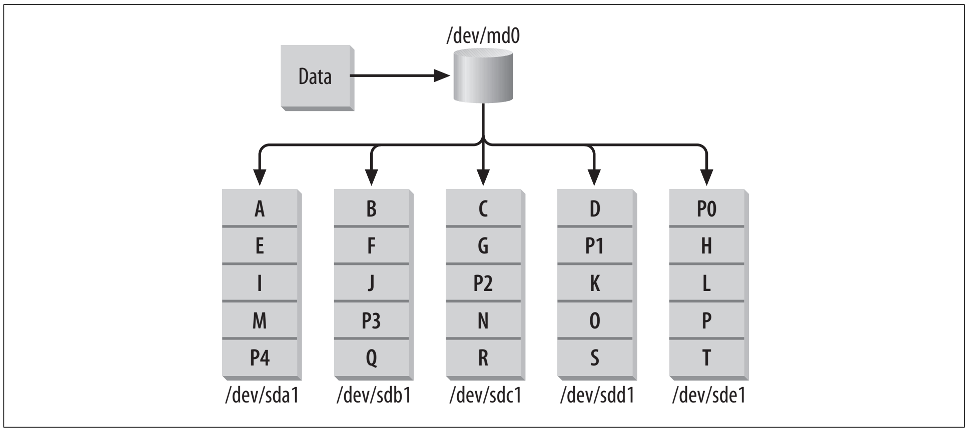


Figure 2-9. RAID-5 eliminates the dedicated parity disk by distributing parity across all drives.

Take the example of a RAID-5 with five member disks. In this case, every fifth chunk-sized block on each member disk will contain parity information for the other four disks. This means that, as in RAID-1 and RAID-4, a portion of your total storage space will be unusable. In an array with five disks, a single disk's worth of space is occupied by parity information, although the parity information is spread across every disk in the array. In general, if you have N disk drives in a RAID-5, each of size S, you will be left with (N-1) * S space available. So, RAID-4 and RAID-5 yield the same usable storage. Unfortunately, also like RAID-4, a RAID-5 can withstand only a single disk failure. If more than one drive fails, all data on the array is lost.

RAID-5 performs almost as well as a striped array for reads. Write performance on full stripe operations is also comparable, but when writes smaller than a single stripe occur, performance can be much slower. The slow performance results from prereading that must be performed so that corrected parity can be written for the stripe. During a disk failure, RAID-5 read performance slows down because each time data from the failed drive is needed, the parity algorithm must reconstruct the lost data. Writes during a disk failure do not take a performance hit and will actually be slightly faster. Once a failed disk is replaced, data reconstruction begins either automatically or after a system administrator intervenes, depending on the hardware.

RAID-5 has become extremely popular among Internet and e-commerce companies because it allows administrators to achieve a safe level of fault-tolerance without sacrificing the tremendous amount of disk space necessary in a RAID-1 configuration or suffering the bottleneck inherent in RAID-4. RAID-5 is especially useful in production environments where data is replicated across multiple servers, shifting the internal need for disk redundancy partially away from a single machine.

Hybrid Arrays

After the Berkeley Papers were published, many vendors began combining different RAID levels in an attempt to increase both performance and reliability. These hybrid arrays are supported by most hardware RAID controllers and external systems. The Linux kernel will also allow the combination of two or more RAID levels to form a hybrid array. In fact, it allows any combination of arrays, although some of them might not offer any benefit. The most common types of hybrid arrays, summarized in the following sections, are covered in this book.

RAID-10 (striping mirror)

The most widely used, and effective, hybrid array results from the combination of RAID-0 and RAID-1. The fast performance of striping, coupled with the redundant properties of mirroring, create a quick and reliable solution—although it is the most expensive solution.

A striped-mirror, or RAID-10, is simple. Two separate mirrors are created, each with a unique set of member disks. Then the two mirror arrays are added to a new striped array (see Figure 2-10). When data is written to the logical RAID device, it is striped across the two mirrors.

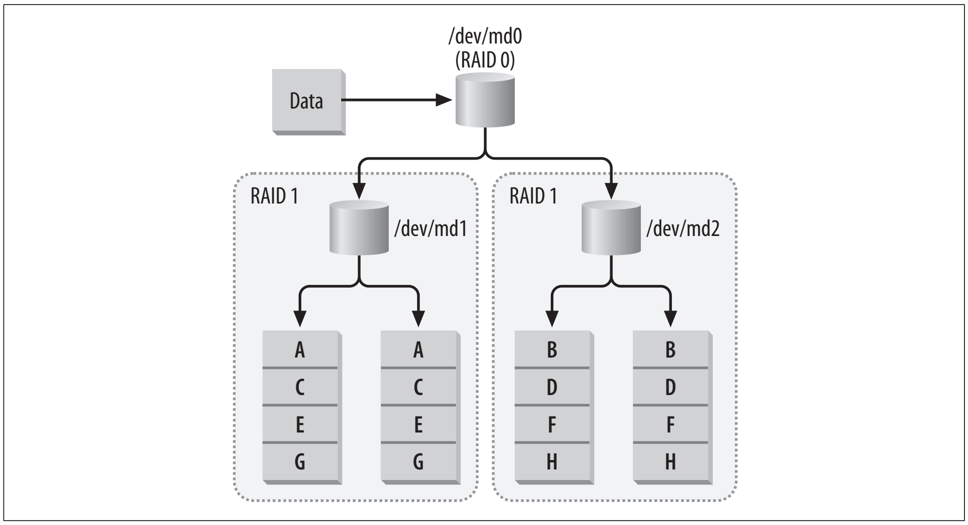


Figure 2-10. A hybrid array formed by combining two mirrors, which are then combined into a stripe.

Although this arrangement requires a lot of surplus disk hardware, it provides a fast and reliable solution. I/O approaches a throughput close to that of a standalone striped array. When any single disk in a RAID-10 fails, both sides of the hybrid (each mirror) may still operate, although the one with the failed disk will be operating in degraded mode. A RAID-10 arrangement could even withstand multiple disk failures on different sides of the stripe.

When creating a RAID-10, it's a good idea to distribute the mirroring arrays across multiple I/O channels. This will help the array withstand controller failures. For example, take the case of a RAID-10 consisting of two mirror sets, each containing two member disks. If each mirror is placed on its own I/O channel, then a failure of that channel will render the entire hybrid array useless. However, if each member disk of a single mirror is placed on a separate channel, then the array can withstand the failure of an entire I/O channel (see Figure 2-11).

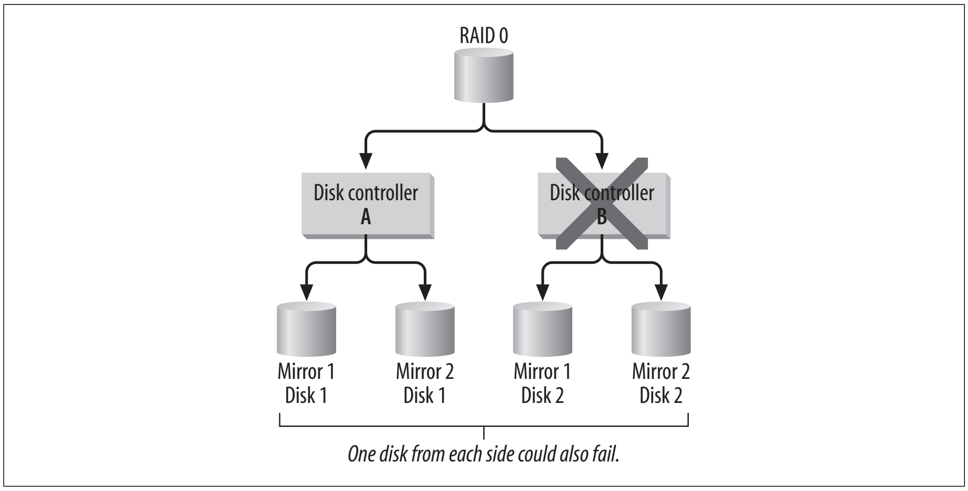


Figure 2-11. Spreading the mirrors across multiple I/O channels increases redundancy.

While you could combine two stripes into a mirror, this arrangement offers no increase in performance over RAID-10 and does not increase redundancy. In fact, RAID-10 can withstand more disk failures than what many manufacturers call RAID-0+1 (two stripes combined into a mirror). While it's true that a RAID-0+1 could survive two disk failures within the same stripe, that second disk failure is trivial because it's already part of a nonfunctioning stripe.

I've mentioned earlier that vendors often deviate from naming conventions when describing RAID. This is especially true with hybrid arrays. Make sure that your controller combines mirrors into a stripe (RAID-10) and not stripes into a mirror (RAID-0+1).

RAID-50 (striping parity)

Users who simply cannot afford to build a RAID-0+1 array because of the enormous disk overhead can combine two RAID-5 arrays into a striped array (see Figure 2-12). While read performance is slightly lower than a RAID-0+1, users will see increased write performance because each side of the stripe is made up of RAID-5 arrays, which also utilize disk striping. Each side of the RAID-50 array can survive a single disk failure. A failure of more than one disk in either RAID-5, though, would result in failure of the entire RAID-50.

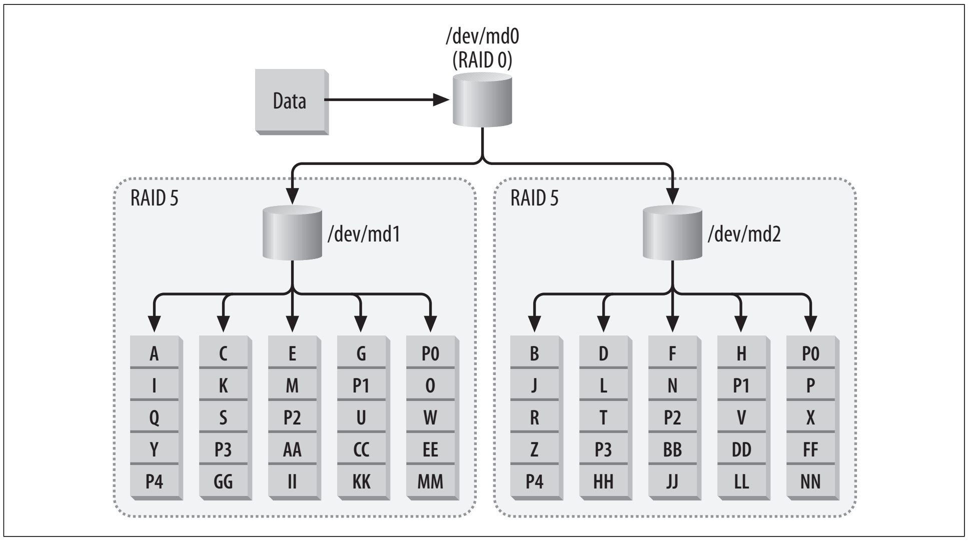


Figure 2-12. A hybrid array formed by combining RAID-5 arrays into a striped array.

RAID Case Studies: What Should I Choose?

Choosing an architecture can be extremely difficult. Trying to connect a specific technology to a specific application is one of the hardest tasks that system administrators face. Below are some examples of where RAID is useful in the real world.

Case 1: HTTP Image Server

Because RAID-1 supports parallel reads, it makes a great HTTP image server. Companies that sell products online and provide product photos to web surfers could use RAID-1 to serve images. Images are static content, and in this scenario, they will likely be read quite a bit more than they will be written. Although new product photos are frequently added, they are written to disk only once by a web developer, whereas they are viewed thousands of times by potential customers. Parallel read performance on RAID-1 helps facilitate the large number of hits, and the write performance loss with RAID-1 is largely irrelevant because writes are infrequent in this case. The redundancy aspect of RAID-1 also ensures that downtime is minimal in the event of a disk failure, although parallel read performance will be temporarily lost until the drive can be replaced. Using a hot-spare, of course, ensures that performance is affected for only a brief time.

Case 2: Usenet News

Striped arrays are clearly the best candidate for Internet news servers. Extremely fast read and write times are required to keep up with the enormous streams of data that a typical full-feed news server experiences. In many cases, the data on a news partition is inconsequential. Lost articles are frequent, even in normally operating feeds, and complete data loss usually means that only a few days' articles are lost.

Administrators could configure a single news server with both a striped array and mirrored array, as shown in Figure 2-13. The striped array could house newsgroups that are of no consequence and could easily withstand a day's worth of article loss without users complaining. Newsgroups that are read frequently, as well as local groups and system partitions, could be housed on the RAID-1 array. This would make the machine redundant in case of a disk failure.

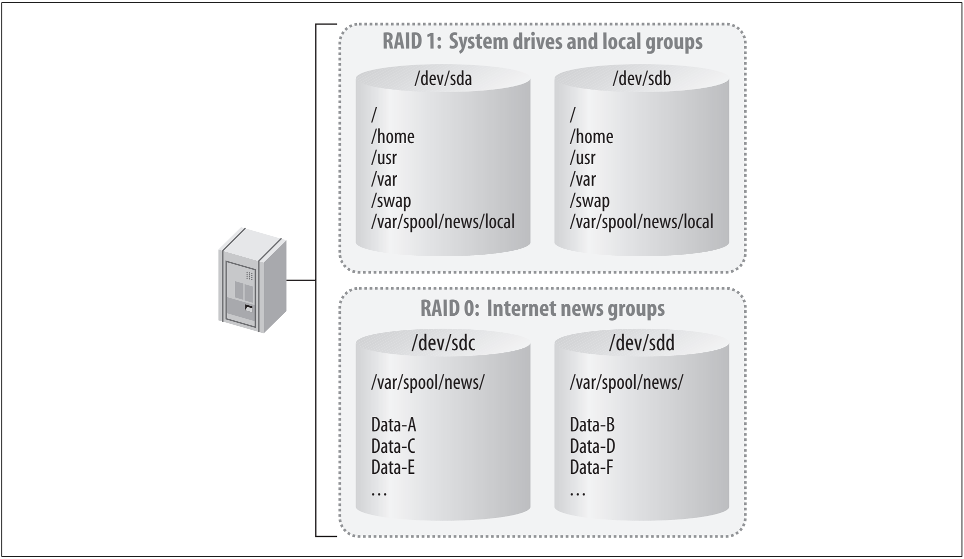


Figure 2-13. A Usenet news server with both a striped and mirror array.

Case 3: Home Use (Digital Audio, Video, and Images)

With the increasing capacity and availability of digital media, users will find it difficult to contain their files on a single hard disk. Linear mode and RAID-0 arrays provide a good storage architecture for storing MP3 audio, video, and image files. Often, these files are burned to CD or are easily replaceable, so the lack of redundancy in linear mode and RAID-0 can be overlooked. Users can opt to make backups of files that are either important or hard to replace.

A quick trip to a surplus warehouse or .COM auction might get you a supply of older, cheap hard disks that can be combined into a linear array. If you can find matched disks, then RAID-0 will work well in this case. A mix of different drives can be turned into a linear mode array. Both of these methods are perfect for home use because they maximize what might have become old and useless storage space and turn it into usable disk space.

Case 4: The Acme Motion Picture Company

People who produce motion pictures are faced with many storage problems. Accommodating giant source files, providing instant access to unedited footage, and storing a finished product that could easily exceed hundreds of gigabytes are just a few of the major storage issues that the film and television industries face.

Film production workstations would benefit greatly from RAID-5. While RAID-0 might seem like a good choice because of its fast performance, losing a work-in-progress might set work back by days, or even weeks. By using RAID-5, editors are able to achieve redundancy and see an improvement in performance. Likewise, RAID-1 might seem like a good choice because it offers redundancy without much of a performance hit during disk failures. But RAID-1, as discussed earlier, leads to an increase only in read performance, and editors will likely be writing postproduced clips often until the desired cut is achieved.

Source files and finished scenes would benefit most from RAID-1 setups. Workstations could read source files from these RAID-1 servers. Parallel reads would allow editors and production assistants to quickly pull in source video that could then be edited locally on the RAID-5 array, where write performance is better than on RAID-1. When a particular scene is completed, it could then be sent back to the RAID-1 array for safekeeping. Although write performance on RAID-1 isn't as fast as on RAID-5, the redundancy of RAID-1 is essential for ensuring that no data is ever lost. Reshooting a scene could be extremely costly and, in some cases, impossible.

Figure 2-14 shows how different RAID arrays could be used in film production.

Striping might also be a good candidate for film production workstations. If cost is a consideration, using RAID-0 will save slightly on drive costs and will outperform RAID-5. But a drive failure in a RAID-0 workstation would mean complete data loss.

Case 5: Video on Demand

This scenario offers the same considerations as Case 1, the site serving images. RAID-1, with multiple member disks, offers great read performance. Since writes aren't very frequent when working with video on demand, the write performance hit is okay.

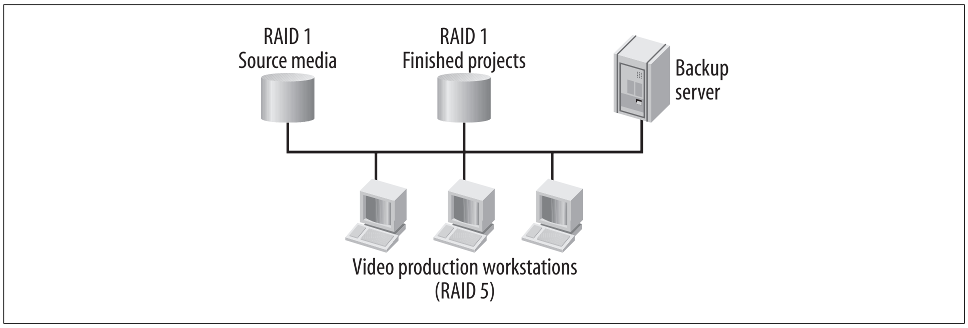


Figure 2-14. Workstations with RAID-5 arrays edit films while retrieving source films from a RAID-1 array. Finished products are sent to another RAID-1 array.

Disk Failures

Another benefit of RAID is its ability to handle disk failures without user intervention. Redundant arrays can not only remain running during a disk failure, but can also repair themselves if sufficient replacement hardware is available and was preconfigured when the array was created.

Degraded Mode

When an array member fails for any reason, the array is said to have gone into *degraded mode*. This means that the array is not performing optimally and redundancy has been compromised. Degraded mode therefore applies only to arrays that have redundant capabilities. A RAID-0, for example, has only two states: operational and failed. This interim state, available to redundant arrays, allows the array to continue operating until an administrator can resolve the problem—usually by replacing a failed disk.

Hot-Spares

As I mentioned earlier, some RAID levels can replace a failed drive with a new drive without user intervention. This functionality, known as *hot-spares*, is built into every hardware RAID controller and standalone array. It is also part of the Linux kernel. If you have hardware that supports hot-spares, then you can identify some extra disks to act as spares when a drive failure occurs. Once an array experiences a disk failure, and consequently enters into degraded mode, a hot-spare can automatically be introduced into the array. This makes the job of the administrator much easier, because the array immediately resumes normal operation, allowing the administrator to replace failed drives when convenient. In addition, having hot-spares decreases the chance that a second drive will fail and cause data loss.

Hot-spares can be used only with arrays that support redundancy: mirrors, RAID-4, and RAID-5. Striped and linear mode arrays do not support this feature.

Hot-Swap

All of the RAID levels that support redundancy are also capable of *hot-swap*. Hot-swap is the ability to removed a failed drive from a running system so that it can be replaced with a new working drive. This means drive replacement can occur without a reboot. Hot-swap is useful in two situations. First, you might not have enough space in your cases to support extra disks for the hot-spare feature. So when a disk failure occurs, you may want to immediately replace the failed drive in order to bring the array out of degraded mode and begin reconstruction. Second, although you might have hot-spares in a system, it is useful to replace the failed disk with a new hot-spare in anticipation of future failures.

Replacing a drive in a running system should not be attempted on a conventional system. While hot-swap is inherently supported by RAID, you need special hardware that supports it. This technology was originally available only to SCSI users through specially made hard drives and cases. However, some companies now make hot-swap ATA enclosures, as well as modules that allow you to safely hot-swap normal SCSI drives. For more information about hot-swap, see the "Cases, Cables, and Connectors" section, later in this chapter, and the "Managing Disk Failures" section in Chapter 7.

Although many people have successfully disconnected traditional drives from running systems, it is not a recommended practice. Do this at your own risk. You could wipe your array or electrocute yourself.

Hardware Considerations

Whether you choose to use kernel-based software RAID or buy a specialized RAID controller, there are some important decisions to make when buying components. Even if you plan to use software RAID, you will still need to purchase hard drives and disk controllers. The first step is to determine the ultimate size of your array and figure out how many drives are necessary to accommodate all the space you need, taking into account the extra space required by the level of RAID you choose. Don't forget to factor the eventual need for hot-spares into your plan.

Choosing the right components can be the hardest decision to make when building a RAID system. If you're building a production server, you should naturally buy the best hardware you can afford. If you're just experimenting, then use whatever you have at your disposal, but realize that you may have to shell out a few dollars to make things work properly.

Several factors will ultimately affect the performance and expandability of your arrays:

- Bus throughput
- I/O channels
- Disk protocol throughput
- Drive speed
- CPU speed and memory

Computer architecture is a vast and complicated topic, and although this book covers the factors that will most drastically impact array performance, I advise anyone who is planning to build large-scale production systems, or build RAID systems for resale, to familiarize themselves thoroughly with all of the issues at hand. A complete primer on computer architecture is well beyond the scope of this book. The "Bibliography" section of the Appendix contains a list of excellent books and web sites for readers who wish to expand their knowledge of computer hardware.

One essential concept that I do want to introduce is the *bottleneck*. Imagine the filtered water pitchers that have become so omnipresent over the last ten years. When you fill the chamber at the top of the pitcher with ordinary tap water, it slowly drips through the filter into another cache, from which you can pour a glass of water. The filtering process distributes water at a rate much slower than the pressure of an ordinary faucet. The filter has therefore introduced a bottleneck in your ability to fill your water glass, although it does provide some benefits. A more expensive filtration system might be able to yield better output and cleaner water. A cheaper system could offer quicker filtration with some sacrifices in quality, or better quality at a slower pace.

In computing, a bottleneck occurs when the inadequacies of a single component cause a slowdown of the entire system. The slowdown might be the result of poor system design, overuse, or both. Each component of your system has the potential to become a bottleneck if it's not chosen carefully. As you will learn throughout this chapter, some bottlenecks are simply beyond your control, while others begin to offer diminishing returns as you upgrade them.

An Organizational Overview

All systems are built around a motherboard. The motherboard integrates all the components of a computer by providing a means through which processors, memory, peripherals, and user devices (monitors, keyboards, and mice) can communicate. Specialized system controllers facilitate communication between these devices. This group of controllers is often referred to as the motherboard's *chipset*. In addition to facilitating communication, the chipset also determines factors that affect system expandability, such as maximum memory capacity and processor speed.

When an application needs data, the CPU first checks to see if the data is stored in memory. If the data is no longer in memory, the CPU asks the chipset to request the information from disk. The chipset sends a request to the *data bus*, where it is picked up by the appropriate disk controller and sent across the *disk bus* to the drive containing the data. The drive sends the information back to the controller card, which in turn passes it back to the CPU and main memory. Figure 2-15 illustrates the connections between various components of a modern PCI motherboard.

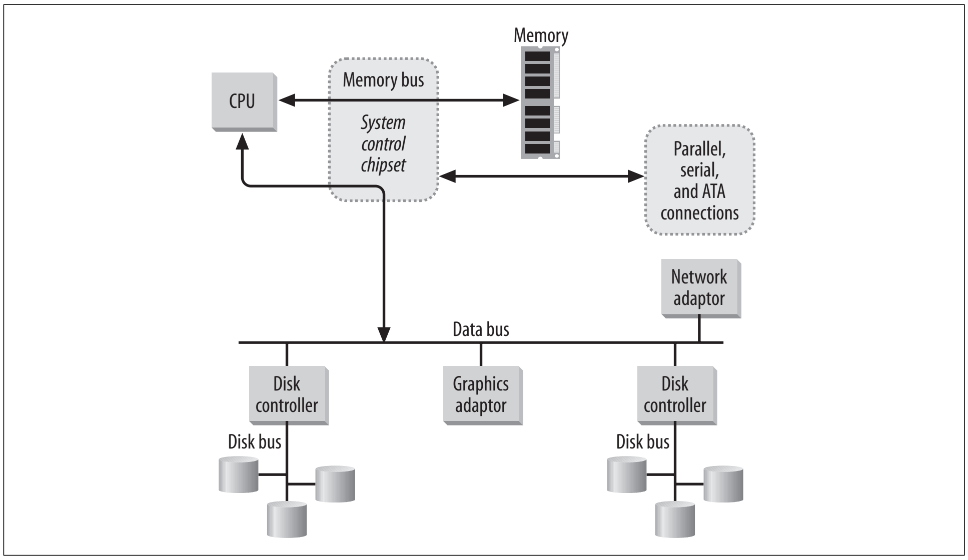


Figure 2-15. When disk I/O occurs, data travels over both the data bus and the disk bus, each a potential bottleneck.

The BIOS

Another important component of every computer system is the BIOS (basic input/output software). The BIOS is a chip on the motherboard that contains a simple set of drivers and instructions. When a machine is turned on, the software stored in the BIOS chip is loaded and executed. The BIOS has basic control over system components: hard disks, CD-ROMs, monitors, keyboards, etc. The BIOS looks for a particular disk sector and executes the program it finds there, usually an operating system. This is sometimes referred to as the *bootstrap* process.

The speeds of the data and disk buses have a direct impact on system performance, and each bus can become a bottleneck. While it's easy to add new disk controllers to a system, thereby increasing the overall number of disk buses, and consequently increasing the overall disk bus throughput, you only have one motherboard to work with. So choosing the right one for your application is essential.

Motherboards and the PCI Bus

Motherboards provide a way to interconnect the various components that make up a computer (memory, processors, and peripherals). Every motherboard has separate buses for communicating between these varied components. Disk controllers and, in turn, hard disks, communicate with the CPU and memory using the I/O bus, also called the data bus. The I/O bus is a standard interface through which peripheral cards (disk controllers, graphics adapters, network cards, etc.) can interface between peripherals (hard disks, monitors, Ethernet networks, etc.) and the CPU and memory.

The Peripheral Component Interconnect (PCI) bus is the most common data bus available today. In recent years, it has usurped the ubiquity of the now outdated Industry Standard Architecture (ISA) bus. Although ISA motherboards are still common, new motherboard purchases typically use the PCI bus. For backward compatibility, the PCI bus can handle ISA peripheral cards through the use of bridging, and many PCI motherboards provide an ISA slot for use with legacy cards.

Bus-width and bus-speed

The speed of the I/O bus is determined by two factors: *bus-width* and *bus-speed*. Bus-width describes how many bytes of data can be sent down the bus at a time. Bus-speed specifies how many times per second data can be transferred through the bus. Bus-width is measured in bits, and all motherboards support bus-widths in multiples of bytes. ISA motherboards support bus-widths of 8 and 16 bits (1 or 2 bytes), and modern PCI motherboards support bus-widths of up to 64 bits, or 8 bytes.

Bus-speed is measured by the number of *clock cycles* that can occur each second. Manufacturers now use the term Front Side Bus when referring to bus-speed. ISA boards run at 8.33 MHz, or 8.33 million clock cycles per second. The first PCI boards ran at 33.33 MHz, or 33.33 million clock cycles per second. A PCI motherboard with a 32-bit bus-width (4 bytes), operating at 33 MHz, has a maximum I/O throughput of 133.33 MB/s (4 bytes per cycle * 33.33 million cycles per second = 133.33 megabytes per second). Newer and faster PCI boards can operate at speeds of up to 533 MHz. Table 2-3 shows the various I/O throughputs of typical motherboards as a factor of bus-width and bus-speed.

Table 2-3. I/O bus throughput

Bus type	Width (bits)	Clock cycles (MHz)	Data throughput (MB/s)
ISA (XT)	8	8.33	8.33
ISA (AT)	16	8.33	16.66
PCI	32	33.33	133.33
PCI	64	66.66	533.33

The data throughput of your motherboard is the first bottleneck to consider when building a RAID system. If you are planning to use three SCSI cards, each with an advertised speed of 80 MB/s, you should quickly realize that a standard 32-bit PCI motherboard running at 33 MHz will become a bottleneck. The aggregate speed of your SCSI controllers (80 MB/s * 3 = 240 MB/s) is more than the overall speed of your I/O bus (133.33 MB/s).

64-Bit Motherboards

While some motherboards are advertised as having 64-bit PCI slots, usually only one or two of the PCI slots are usable by 64-bit PCI cards. Fortunately, many 64-bit cards can fit into 32-bit slots and operate in 32-bit mode. However, using 64-bit cards in 32-bit mode wastes their capability—they can operate at only half of their potential speed. So when choosing a 64-bit motherboard, be certain that it has enough 64-bit slots to meet your needs.

Not all motherboards are created equal. Be certain to check the manufacturer's specification when deciding which one to purchase, making careful note of the bus-width and bus-speed. Remember that all the expansion cards, including the graphics card, share the overall speed of the I/O bus. If you have a board that supports an overall bus throughput of 533 MB/s, then installing several high-end SCSI cards, a graphics adapter, and a network card might cause a bottleneck on the data bus. So for production file servers, it might make sense to configure a system without video (you could use the *console on serial port* features of Linux). Like most other aspects of technology, you should expect to see faster motherboards in the near future. Although 128-bit boards might be a year or two off, manufacturers are constantly working to increase the bus-speed.

In the same way that disks constantly fall behind the curve of storage needs, the I/O bus is always behind the curve when compared to the speed at which the CPU and main memory can interact. So the I/O bus will almost always become the most significant bottleneck on any motherboard. In response to this problem, it is common for high-end server boards to offer dedicated buses for one or more PCI slots. Some even offer a separate bus for each PCI slot, which allows you to place a RAID or SCSI card on its own I/O bus, separating other peripherals such as network and graphics cards. Using one of these dual-bus motherboards can effectively double the combined overall speed of your I/O bus.

I/O Channels

An I/O channel represents a single chain of devices attached to your machine, either internally or externally. Internal I/O channels are typically connected to a controller card (or to the motherboard) by ribbon cable. (Ribbon cables are flat cables, usually gray or blue, that interconnect hard drives and disk controllers inside a computer case.) Externally, you might connect drives or peripheral devices to a controller card using SCSI cables. The more identical, parallel I/O channels you have available for your array, the better performance you can expect out of it, as long as you are careful to identify and eliminate other bottlenecks.

The most common instance of parallel and identical I/O channels is the typical PC motherboard. Almost all i386-based motherboards include two onboard ATA/IDE disk controllers (see Figure 2-16).

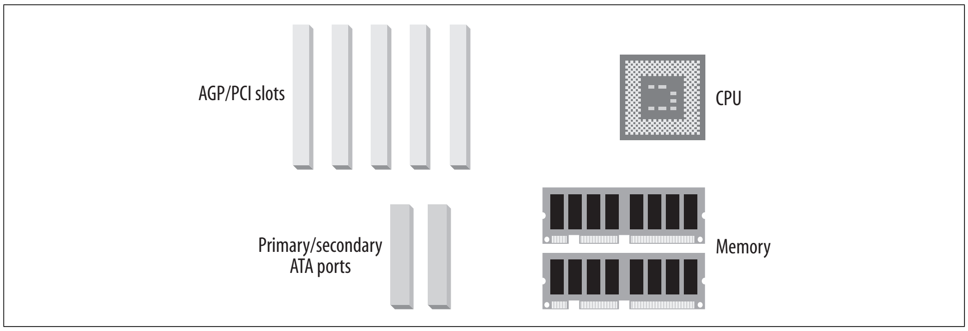


Figure 2-16. Major components on a motherboard.

When I say *identical*, I mean that each channel you select for use in your array supports the same architecture and protocols. *Parallel* means that each channel in the array can accept requests simultaneously. While you could theoretically use two different types of I/O channels in the same RAID array, you'd be wasting the performance of the faster channel because the faster chain needs to operate at slower rate in order to stay at the same pace as the other channels in the array. It's generally not a good idea to mix different iterations of the same disk protocol because their speeds vary.

It's also a bad idea to mix different disk protocols, such as SCSI and ATA, even though software RAID, in particular, allows both of these arrangements. The same is true for mixing hard drives of differing speeds, but I'll cover that issue in more detail in the "Choosing Hard Drives" section, later in this chapter.

In general, it is good practice to keep only one incarnation of any disk protocol on a single I/O channel. That might mean connecting devices such as CD-ROM drives and scanners, which operate at much slower speeds than current hard disks, on separate controllers. It is advisable to purchase a cheaper, slower controller to connect these devices, keeping them out of any I/O channel that contains faster devices that belong to an array.

For example, many SCSI controllers contain two separate, parallel channels that are not identical: a compact, high-density, 68-pin connector used to connect hard drives (wide SCSI) and a larger, low-density 50-pin connector often intended to connect

CD-ROM drives (see Figure 2-17). While both of these channels can be used in parallel, pairing them is a bad choice for RAID, because by combining the use of two channels in a single RAID array, we lose the performance associated with wide SCSI. Many cards, for example, provide two internal connectors: one that supports a 50-pin fast SCSI chain and another that supports a high-density 68-pin wide SCSI chain. If you are using the AT attachment (ATA), it's wise to connect your CD-ROM drive to a separate ATA controller when possible. ATA is discussed in the "Disk Access Protocols" section, later in this chapter.

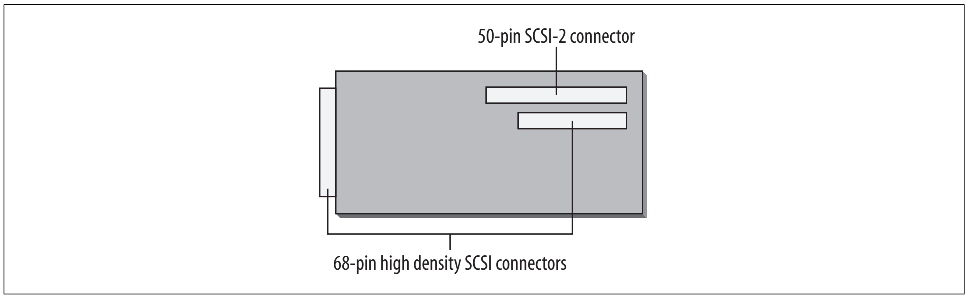


Figure 2-17. An SCSI controller with one external connector and two internal connectors (one 50-pin and one 68-pin).

It's also important to realize that while most SCSI cards provide external connectors, they are merely an extension of an internal channel. Therefore, the internal and external chains on a SCSI card do not operate in parallel. Space on your motherboard can quickly become scarce, and you might find that a single controller card with multiple I/O channels works better for you. Several manufacturers of SCSI cards make high-end versions of their consumer-grade cards that provide multiple distinct I/O channels. You might be able to get two or three I/O channels on a single PCI card.

You can also increase I/O bandwidth through a combination of two types of upgrades: buying high-density cards and adding several of them to your system to take advantage of the extra channels (see Figure 2-18).

Most hardware RAID cards are also available in models with multiple channels (see Figure 2-19). Some support as many as six separate channels on a single card, and most allow you to manage cards as a whole, so you can include devices connected to separate cards in the same array and manage them through a single interface. The number of cards that you can put in a single system is limited only by the number of slots available on your motherboard, but remember to consider the throughput of the motherboard when purchasing controller cards. Typical motherboards have a data bus throughput limit of 133 MB/s (32-bit) or 533 MB/s (64-bit). Adding three multichannel SCSI controllers that support speeds of 160 MB/s each would saturate the data bus on a heavily used system. Remember that network and graphics cards also use bandwidth on the I/O bus. Also recall that some high-end motherboards support dedicated PCI slots that can help avoid these problems.

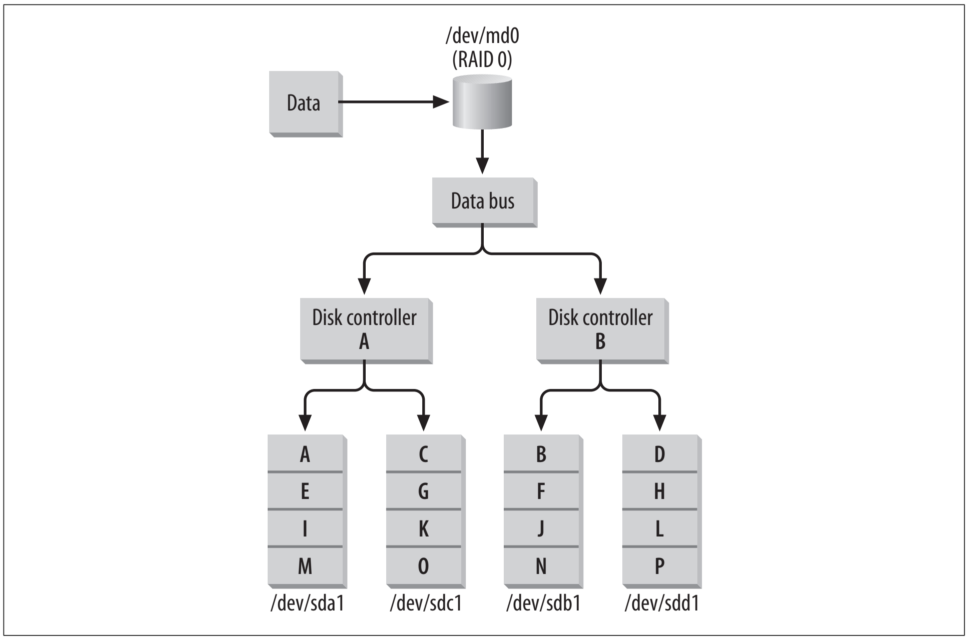


Figure 2-18. Using multiple disk controllers increases both throughput and the total number of usable drives.

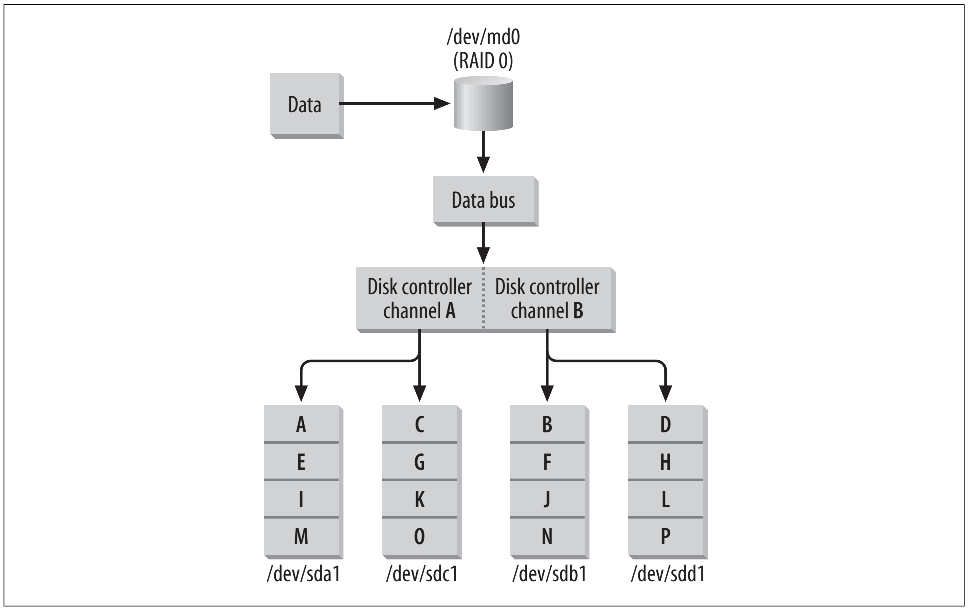


Figure 2-19. Some controllers have multiple channels on a single card.

When using more than one I/O channel, it's advisable to alternate between channels when adding disks to an array. That will help you to avoid overloading a single I/O channel. You typically don't need to worry about how you physically arrange your disks or how your hardware (or Linux) detects them. Just make certain that you balance them as equally as possible between each available channel. When you create the array, disks can be added in an arbitrary order so that their physical location can be taken into account. This process might be facilitated through a configuration file, command-line utility, GUI management package, or BIOS utility. I'll cover this process in more detail when I explain how to create new arrays in Chapters 3 and 4 (for software RAID) and in Chapter 5 (for hardware RAID).

There may also be some situations when it is necessary to use drives with slightly different performance. Let's say, for example, that you have a few ultra-wide SCSI drives from an assortment of different manufacturers. Since not all drives, regardless of protocol, are exactly the same, you will see slightly different speeds from each. In this case, it's best to arrange the drives so that the slowest has the smallest SCSI ID number and is closest to the controller. Likewise, the fastest should be placed farthest from the controller and should be set to the highest SCSI ID number. This will help to alleviate the performance differences. Users who are planning to create a linear mode array using several different drive sizes should arrange drives with this methodology in mind. This methodology may also be helpful for users who simply cannot afford to purchase new, matched drives.

Disk Access Protocols

The disk protocol of the hardware you choose has a tremendous impact on the performance and scalability of your array. Each protocol has its own hard limits on the maximum throughput of each I/O channel and the maximum number of devices you can attach to a single channel. So the disk protocol you select will have a direct impact on the maximum size of your array.

Although we traditionally think of RAID in terms of high-end SCSI systems, today it's not uncommon for consumer-marketed systems to come equipped with support for RAID on non-SCSI disks. In fact, Linux software RAID can support either SCSI or ATA devices as part of an array (see the following section). The kernel will even let you mix these protocols within a single RAID device, although that arrangement isn't recommended. (See the "Matched drives" section, later in this chapter, as well as the previous section, "I/O Channels.") Software RAID under Linux does not rely on the underlying disk architecture to work, so there is no reason why an array could not be built using a Firewire (IEEE 1394), Fiber-channel, or other disk architecture developed in the future, as long as you can find hardware and device drivers to support the architecture as a standalone device.

ATA (used interchangeably with the acronym IDE) and SCSI are discussed in detail throughout this book because they are the most common disk protocols in use today. ATA is a part of every modern motherboard, and SCSI is the most common choice for large servers.

The AT Attachment (ATA) and Integrated Disk Electronics (IDE)

Integrated disk electronics, or *IDE*, has had many incarnations and many names since its introduction in 1986. Originally, hard drives were small enough, in both size and capacity, to fit directly onto disk controllers. As storage requirements grew, manufacturers realized that housing drives on controller cards was an inefficient use of space. Soon drives and controllers became separate entities, connected by ribbon cable. This meant that drives could grow in size without interfering with the expandability of the motherboard. It was common for these integrated controller cards to make adjacent slots on the motherboard inaccessible. Manufacturers eventually decided that portions of the controller could be housed directly on the drives and that creating a standard drive interface would allow for both expandability and portability. Originally called IDE in several proprietary implementations, a standardized version called the *AT Attachment*, or *ATA*, was eventually ratified (although many people still use the terms IDE and ATA interchangeably). This new disk interface was called the AT Attachment because it was introduced with the ISA (AT) motherboard. It quickly grew in popularity, and today the ATA interface is the most widely deployed consumer disk interface. Figure 2-20 shows the ATA interface.

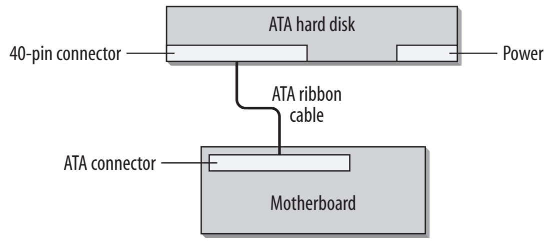


Figure 2-20. The ATA interface separated the drive and the controller.

ATA has evolved a great deal since its introduction. Its performance and scalability have improved over time. Table 2-4 outlines the various iterations of ATA.

Table 2-4. Overview of IDE/ATA types

ATA type	Maximum throughput (MB/s)	Common names
ATA-1	8.3	ATA, IDE, Fast ATA
ATA-2	16.66	EIDE, Fast ATA-2
ATA-3	16.66	Ultra DMA, Ultra ATA
ATA-4	33.33	Ultra ATA/33
ATA-5	66.66	Ultra ATA/66
ATA-6	100	Ultra ATA/100
ATA-7	133.33	Ultra ATA/133

Many of the names associated with various iterations of ATA represent departures from the ATA specification by a single manufacturer. Enhanced IDE (EIDE), for example, was an attempt by Western Digital to increase its market share by offering enhancements to the original ATA (ATA-1) specification before ATA-2 was ratified. This created a rash of vendor-enhanced ATA-compatible interfaces, resulting in many puzzling names. In general, ATA devices are compatible between iterations, but mixing old and new drives, like mixing different disk protocols, usually results in performance problems.

Master and slave

ATA devices have only two configuration settings: *master* and *slave*. Despite the unfortunate nomenclature, both drives can operate independently once the system initializes, although drives operating in slave mode won't perform as well. A single ATA disk operating as master on a dedicated channel will yield the best performance. So it's always recommended that you use only one disk per channel when working with ATA and RAID.

Only the master device can be used as a boot device, but you can use the master device from any ATA channel for booting. So on a standard system with two onboard ATA controllers, you have a maximum of two boot devices and a total of four ATA devices. Most users place their primary hard disk on the first interface and a CD-ROM drive on the second, so that either can be used as a boot device.

A simple jumper on the back of the drive determines whether an ATA device is operating in master or slave mode. Some devices also have a third setting called *cable select*. This jumper allows the system to determine which device is master or slave by its position on the cable. The first ATA device found on the cable is flagged as the master device and the second becomes the slave device. Unfortunately, many users report strange behavior when using this feature, such as disappearing drives or devices that won't boot properly. Because it's easy to manually set the devices, I recommend always setting up devices as master and slave and never using cable select.

Direct memory access (DMA)

Modern ATA devices support an I/O method called *direct memory access (DMA)* that allows two devices on the same channel to transfer data without direct CPU intervention. Using DMA relieves a lot of pressure on your CPU during array reconstruction, when large amounts of data need to be transferred between two drives. Sometimes DMA is not enabled by default. Chapter 7 discusses how to enable this feature and fine-tune ATA disks.

The drawbacks of ATA

By far the biggest drawback of ATA is its real limit of two devices per channel and its usable limit of one device per channel when performance is an issue. This limit hinders the scalability of any RAID built with ATA, in terms of performance and maximum storage. In fact, this limitation might be the determining factor in choosing SCSI over ATA. While most motherboards come with two onboard ATA controllers, the four-disk maximum (which really translates into two RAID disks) associated with a two-channel ATA system will likely warrant adding a low-end SCSI card or additional ATA controller.

Unlike SCSI, ATA does not support detached operations—a process that allows a disk controller to detach from the bus in between I/O requests so that the CPU can access another controller. In addition, drives connected to an ATA interface cannot, generally, interact with each other without CPU intervention. ATA does, however, have a simpler command set than SCSI, which helps decrease latency.

Using an ATA RAID controller should improve your performance a bit by offloading some of the load from the CPU and onto the controller. While ATA supports only two devices per channel, many of the ATA RAID cards available also provide built-in ATA controllers, so that you can add additional drives. For example, Promise Technologies and 3ware both sell controllers with more than two channels. The problem is fitting them all in a single case.

ATA, because of its ubiquity, might be the best solution for users who are unsure about building a RAID and want to test its effectiveness. It's also ideal for users who are on a budget or who simply do not need the best performance and reliability. Administrators who are considering software RAID might find it useful to experiment with some spare ATA drives; they're easy to come by.

SCSI

The *Small Computer Systems Interface*, or *SCSI*, has been around much longer than ATA, but has traditionally been priced out of consumer reach. This changed in 1986 with the introduction of the Apple Macintosh II, which came standard with an SCSI controller, but no hard disk. The following year, Apple introduced the Mac SE and the Mac II, both available with optional internal hard disks.*

Bus-width and signaling rates

SCSI, like the data bus of a motherboard, is defined by both a bus-width and a *signaling rate* (sometimes called the *clock rate*). Increasing either of these parameters increases the overall throughput of the SCSI bus. Bus-width is either *narrow* (8-bit) or *wide* (16-bit). As with motherboards, the bus-width determines how many bytes of data can be transmitted during each clock cycle. Bus-width also determines the number of devices that can be connected to a single SCSI bus. Narrow buses can handle eight devices and wide buses can handle sixteen. This gives each bus type 7 and 15 usable devices respectively (one device number is reserved for the controller).

* Thanks to *http://www.apple-history.com* for the time line.

The signaling rate measures how many times a second data can be pushed through the SCSI bus. Signaling rates are measured in megahertz. The first implementation of SCSI, also called SCSI-1, had a bus-width of 8 bits and a signaling rate of 5 MHz. One byte of data, transmitted five million times per second across the SCSI bus, gave SCSI-1 a data throughput of 5 MB/s. Since SCSI-1, more signaling rates have been added to the SCSI specification. *Fast SCSI* defined a 10 MHz signaling rate (yielding a 10 MB/s transfer rate) and from there, *Ultra SCSI* (20 MHz), *Ultra2 SCSI* (40 MHz), and *Ultra3 SCSI* (80 MHz) were eventually defined and implemented.

Although SCSI is governed by the American National Standards Institute (ANSI), some manufacturers, throughout SCSI's evolution, did not want to wait for newer and faster SCSI protocols to be standardized. In an attempt to gain market share, many SCSI manufacturers have prematurely released their own prestandardized implementations. The result, as with ATA, was a deviation in naming among manufacturers, although incompatibility was rare and today is generally a nonissue. Table 2-5 shows the various implementations of SCSI and their maximum data throughput rates.

Table 2-5. Overview of SCSI data throughput

Names	Bus width	Signaling rate (MHz)	Maximum data throughput (MB/s)
SCSI-1, SCSI, Narrow SCSI	8	5	5
Fast SCSI, Fast-Narrow SCSI	8	10	10
Fast Wide SCSI	16	10	20
Ultra SCSI	8	20	20
Ultra Wide SCSI	16	20	40
Ultra2 SCSI, Ultra2 Narrow SCSI	8	40	40
Ultra2 Wide SCSI	16	40	80
Ultra3 SCSI, Ultra 160 SCSI	16	80	160

There is already talk of yet higher signaling rates for SCSI. A wide bus with a signaling rate of 160 MHz, yielding a throughput of 320 MB/s, is currently under development. It is likely to be commonplace within the next year.

Transmission types

The final difference between SCSI implementations is found in the type of cabling used to interconnect devices. *Single-ended (SE)* devices transmit information over single wires. Using single wires for transmission on the disk bus limits the maximum cable length of the disk bus. It also limits the maximum data throughput because error correction requires a pair of wires for each signal.

Differential SCSI transmits information over a pair of wires, which requires more expensive cables, but solves the performance and cable length limitations imposed by single-ended SCSI. The first standard, *high-voltage differential (HVD)*, provided a faster disk bus and used an extremely high voltage. HVD also allowed a maximum cable length of 25 meters, compared with the 6-meter maximum of SE devices. However, manufacturing controllers and devices that supported HVD dramatically increased hardware costs. The drastic increase in voltage means that a separate chip was required to regulate the voltage of the SCSI bus. It also made HVD and SE incompatible, requiring older devices to be replaced or connected to a separate controller. Because of these limitations, HVD is extremely uncommon today, especially in the consumer market, although it is used in some specialized RAID systems.

Shortly afterward, *low-voltage differential (LVD)* devices were introduced. LVD devices provided an increased maximum throughput like HVD, but limited the overall cable length to 12 meters. However, LVD also dramatically decreased hardware costs when compared to HVD. By lowering the voltage of the SCSI bus, LVD allowed a single chip to control both the SCSI devices and the voltage. This decrease in voltage also allowed LVD and SE to coexist on the same bus. LVD is now the standard and is supported by all recent SCSI devices.

SCSI Versus ATA

Overall, SCSI is a much better choice than ATA, both as a standalone and as part of an array. It allows more devices per channel and provides higher throughput. It also has a much larger command set, compared to ATA, which translates into better performance and increased reliability. The only major drawback of SCSI is price. SCSI drives and controllers are generally more expensive, with SCSI drives typically costing two or three times as much per megabyte as their ATA equivalents. (Although today, some mid-range motherboards are available with built-in SCSI controllers at little extra cost.) If you plan to use external SCSI devices, you will need to spend extra money on cabling and external disk enclosures. On the other hand, ATA does not support external devices at all, so its expandability is limited.

Speed

In the past, SCSI outperformed ATA by leaps and bounds, but ATA has caught up substantially in recent years. Today, ATA disks perform as well as SCSI disks, so speed isn't as much of a factor as it was just three or four years ago. But, with SCSI, you can populate an I/O channel with enough devices to fully utilize the entire pipe. With ATA, you are really limited to one device per channel if you want decent performance from that device, and that's not enough to utilize the full pipe when working with the most recent ATA specifications.

High-end SCSI drives have data throughputs of about 40 MB/s. When using Ultra 160 SCSI, you would need three or four drives on a single chain to take full advantage of your bandwidth. ATA drives operate at much slower speeds, so if you were using Ultra ATA/100, you could not possibly populate a single channel with enough drives to take full advantage of your I/O pipe, even if you put two devices on the same channel. The “Choosing Hard Drives” section, later in this chapter, discusses hard disk bottlenecks in more detail.

Configuration

Many people complain about the complexities and pitfalls of SCSI termination. But it’s really quite simple. The beginning and end of every SCSI chain must be terminated. Figure 2-21 illustrates termination on a controller to which only internal devices are connected. The controller card is usually the last device on a channel and comes with built-in termination enabled.

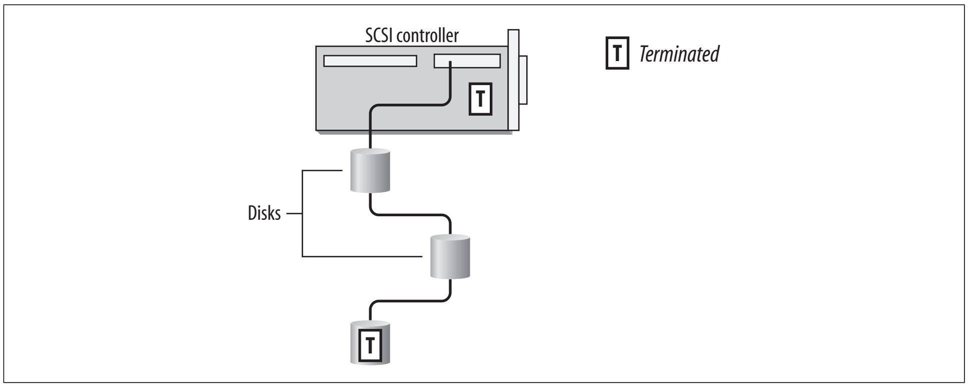


Figure 2-21. Modern controller cards provide onboard termination.

If you plan to use both internal and external devices on the same chain, then you will need to terminate the external portion of the chain. Figure 2-22 shows how to terminate a controller with both internal and external devices. Depending on your controller, you might also need to disable the controller’s termination in the SCSI BIOS, although many cards automatically do this once devices are connected to the external connector.

As shown in Figure 2-23, the same methodology applies if you are using only external devices.

Finally, specifications dictate that any unused connections on an internal cable appear after the last SCSI device on that chain. In practice, this recommendation is often ignored, and many users report no errors when breaking this rule. I have never had problems using cables with more connectors than drives internally. Caveat emptor.

Likewise, there are quite a number of reports about using autotermination of SCSI chains. Autotermination is built into controllers and disks. If you experience problems, you may wish to manually disable autotermination (which is a controller BIOS

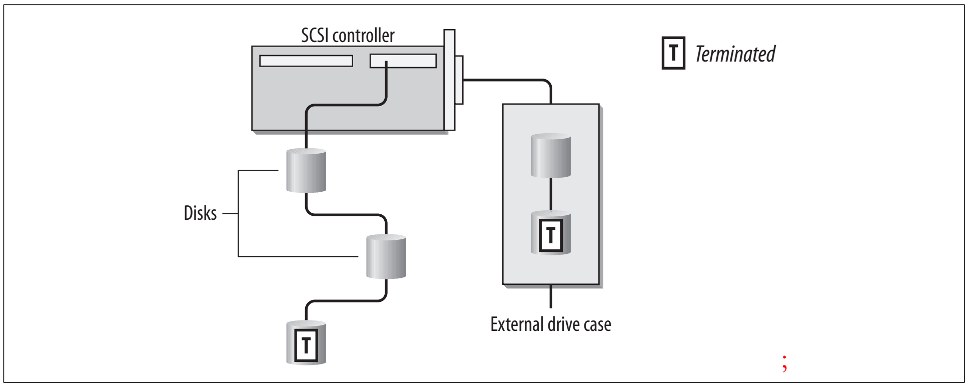


Figure 2-22. The last device on an external SCSI chain must be terminated.

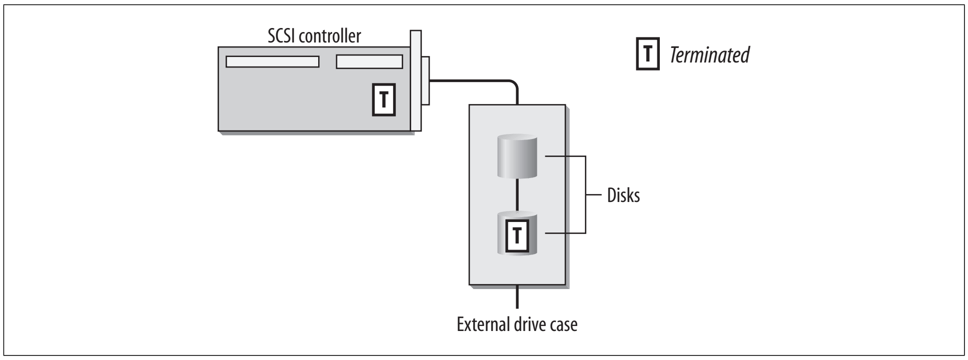


Figure 2-23. When using only external devices, the last disk on the chain and the SCSI card are terminated.

setting or a jumper on your hard disk) and actively terminate the chain at both ends. You can purchase terminators from the vendor who provides your SCSI cables, or even from a local computer store.

Growth

SCSI certainly has a much better upgrade path than ATA. Device-per-channel limits make SCSI much easier to deal with when you need to increase the size of an array. If you're on a budget, you might find it worthwhile to purchase an expensive SCSI controller, along with drives that are one or two technologies behind the current trends. Buying the latest and greatest SCSI card will increase the final price of a system by only a few hundred dollars. Buying the most cutting-edge disks, on the other hand, will affect system price by a few hundred dollars per drive. So while you can save costs by purchasing older drives initially, you won't have to discard your SCSI controller if you decide to upgrade to faster drives a few months later. Scaling back the original drive purchase initially might even place a hardware RAID controller within your budget.

Summary of SCSI versus ATA

SCSI supports more devices than ATA (although configuring many devices can be a challenge for many users and administrators). ATA is in more widespread use than SCSI, and that might make it easier to get hold of enough hardware to build a decent array. Some naysayers argue that SCSI is more confusing than ATA because SCSI users are faced with termination and drive placement considerations. Others are quick to point out the autodetection and block addressing problems with which ATA users must contend.

ATA can access only one device at a time, meaning that the benefits of parallel I/O under RAID are wasted. SCSI can address multiple devices concurrently and does not require the CPU to manage I/O, leaving more processing power for users and applications.

Table 2-6 summarizes the differences between ATA and SCSI. I think you will find that ATA is a cheap and usable way to quickly build arrays for both desktops and low-usage production systems, but that SCSI is the best choice for large systems and applications that require extremely intense I/O.

Table 2-6. The differences between SCSI and ATA

Feature	SCSI	ATA
Device limit	7 or 15 per channel	2 per channel
Maximum cable length	12 meters	~.5 meters
External devices	Yes	Not without special hardware
Termination required	Yes	No
Device ID	Yes	No (master/slave only)
Extra CPU load	No	Yes
Concurrent device access	Yes	No
Cost/availability	Expensive; need to add on	Cheap; built into most motherboards

I have excluded data throughput differences between SCSI and ATA from Table 2-6 because throughput with each protocol is typically limited by disk rather than by channel. Both SCSI and ATA will perform roughly equally in single-disk operations (assuming that similar specifications are compared). That being the case, SCSI supports many drives per channel, whereas ATA supports only one, from a usability standpoint. Thus, with SCSI, it's a lot easier to use the bandwidth you have available, while with ATA, it's really not possible.

Other Disk Access Protocols

Because RAID is oblivious to the hardware and disk architecture on which it is built, you can use any disk protocol that the Linux kernel supports to build an array. Indeed, if a newer, faster, and more reliable disk protocol (such as Serial ATA) were released this year or the next, it would only increase the usefulness of RAID. Furthermore, if a breakthrough in solid-state media happened in the next few years, these devices could also be grouped into arrays. While disk capacities and throughputs continue to increase, they nonetheless continue to fall behind the curve of increasing user needs.

Choosing Hard Drives

Hard drives represent the most challenging bottleneck in data storage. Unlike disk controllers, motherboards, and other components that make up a system, hard drives are unique because they contain mechanical components. This presents a complicated problem for engineers because the moving parts of hard drives limit the speed at which data can be stored and retrieved. Whereas memory and controllers, for example, are completely electronic and can operate at close to light speed, hard drives are much slower.

In general, it's a good idea to use the same disks in an array whenever possible. But using identical disks might not be an option all the time. Disks are made of several parts that affect their overall performance. If a situation arises in which you are forced to mix different disks, then you will want to know how to best evaluate a new disk to ensure that it will function appropriately when added to an array.

Platters, tracks, sectors, and cylinders

Two mechanical parts that affect performance are found on every hard drive. Inside each drive are magnetic *platters*, or disks, that store information. The platters, of which most common hard disks have several, sit on top of each other, with a minimum of space between each platter. They are bound by a spindle that turns them in unison. The surface of each platter (they are double-sided) has circular etchings called *tracks*, similar to a phonograph record, with the important difference being that tracks on a hard disk are concentric circles, while a record has a single spiral track.

Each track is made up of *sectors* that can store data (see Figure 2-24). The number of sectors on each track increases as you get closer to the edge of a platter. Sectors are generally 512 bytes in size, with some minor deviation that depends on the manufacturer.

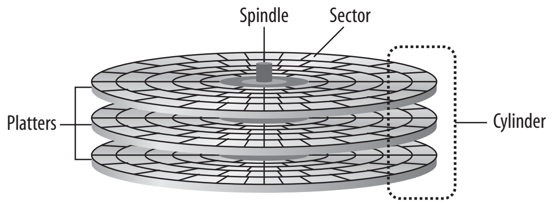


Figure 2-24. The surface of a hard disk platter.

The speed at which these disks spin affects how fast information can be read. The *rotation rate*, or *spin rate*, of a hard disk is measured in *revolutions per minute (RPM)*. Modern drives operate at speeds equal to or greater than 5400 RPM, with 7200 RPM being the most common consumer rotation rate. Drives operating at 7200 RPM are able to rotate through an entire track of data 120 times per second (7200 revolutions per minute / 60 seconds = 120 rotations per second). Most ATA drives spin at one of these rates, depending on the price of the drive. Older drives have slower spin rates. Faster SCSI drives, like those found in high-end servers or workstations, typically spin at rates of 10,000 RPM or higher.

Actuator arm

The second analog bottleneck is the *actuator arm*. The actuator arm sits on top of, or below, a platter and extends and retracts over its surface. At the end of the arm is a *read-write head* that sits a microscopic distance from the surface of the platter. The actuator arm extends across the radius of a platter so that different tracks can be accessed (see Figure 2-25). As the disk spins, the read-write head can access information on the current track without moving. When the end of the track is reached, it might seem logical for the actuator arm to move to the next track and continue writing. However, this would greatly increase the time needed to read or write data because the actuator arm moves much more slowly than the disk spins. Instead, data is written to the same track on the platter sitting directly above or below the current platter. A group of tracks, on different platters, that are the same distance from the spindle are called *cylinders*. Since the actuator arm moves every read-write head in unison, the read-write is already positioned to continue I/O. During a write, if there is no free space left on the current cylinder, the actuator arm moves the read-write heads to another track, and I/O resumes.

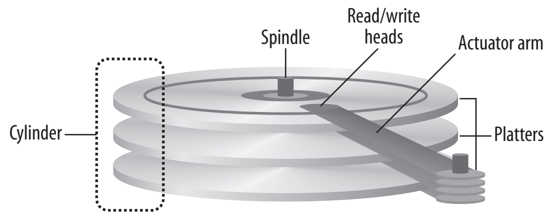


Figure 2-25. Actuator arms move heads across the surface of the disk.

When data is read or written along a single cylinder, and then along adjacent tracks, I/O is extremely fast. This is called *sequential access* because data is read from the drives in a linear fashion. When data is spread out among various tracks, sectors, and cylinders, the actuator arm must move frequently over the platters to perform I/O. This process is called *random access* and is much slower than sequential access.

Think again about the single spiral track of a phonographic record. That design makes records well suited for sequential data access, or audio playback. But it also makes random access impossible, hence the concentric circle design of hard disk platters, which are well suited for both types of data access.

RAID helps eliminate the analog bottlenecks present in hard disks. By striping data across multiple disks, RAID can circumvent the slow analog parts of hard disks.

Maximum data throughput

Unfortunately, hard disk throughput is difficult to measure consistently. The way data is arranged on the drive can affect performance. Data that is spread across many different parts of the disk takes more time to access than data that is grouped together, because the actuator arm has to move more frequently. The *average seek time* of a hard disk is a measurement of the time it takes for the actuator arm to position itself on a new cylinder or track. Once the actuator arm arrives at a new track, it must wait until the proper sector spins into place. The time it takes for the sector and the actuator arm to line up is called *latency*.

In addition to the rotation rate, average seek time and latency, hard disks also come equipped with a *data buffer*. Similar to cache memory on a processor, the data buffer allows a disk to anticipate and cache I/O, increasing the overall throughput of the drive. When selecting hard disks, the rotation rate, average seek time, and data buffer size are all important factors. Smaller seek times mean faster throughput, while higher rotation rates and larger data buffers also increase data throughput.

Doing the math to determine the maximum data throughput of a hard drive you're considering can be tedious. Therefore, manufacturers usually advertise the overall throughput of a drive in easy-to-understand terms. The throughput of a hard disk over time is measured in megabytes per second and is found in the technical documentation for each hard disk model. Unfortunately, there is no standard for measuring this value. Therefore, the name that references it can vary from vendor to vendor. IBM calls this measure the *sustained data rate*, whereas Seagate calls it the *average formatted transfer rate*. I'll use the term *transfer rate* throughout the rest of this book.

Hard disks are also capable of occasionally reaching speeds well beyond their sustained data rates. These increased speeds generally last only for a fraction of a second. This additional benchmark is known as the *burst rate*. Burst rate speeds are usually achieved only when the data bus is idle. If a system is idle most of the time and large chunks of data are written intermittently, you will see throughputs at the burst rate more often than on a busy system. It is also unlikely that these user-friendly measurements will be printed anywhere on the product packaging, so if you plan to buy drives off the shelf, be sure to check the manufacturers' web sites first.

Matched drives

Because different hard disks have different seek times, rotation rates, data buffers, and latency, they also have different data rates. Like mixing disk protocols, using hard drives of varying speeds can hinder array performance. The high performance of fast drives might be wasted while waiting for data from slower disks. Although the performance bottleneck is not as drastic when compared to mixing different SCSI implementations, you should still try to use *matched drives* (drives that are all the same model) whenever possible.

Hard disks also vary slightly in size. Although two disks from different vendors might both be advertised as 18 GB (gigabytes), the formatted capacity may vary slightly. If this occurs, you will need to take extra care when configuring disks to ensure that partitions for any arrays other than linear mode or RAID-0 are exactly the same size. Also note that some disk partitioning tools provide an option to create a partition using the rest of any available disk space. Be careful when choosing this option, as using it on different disks could result in partitions that vary in size.

During the life of your array, it's possible that even if you have taken great pains to make sure that all your disks are matched, you may be forced to introduce a disk that is slightly different. For example, what happens if a disk fails and your vendor no longer makes the drives with which you built the array? In that case, you might have no choice but to use a different drive because the cost of upgrading all the disks might be too high. Keeping spare disks on hand in anticipation of a failure is advisable whenever financially possible.

Cases, Cables, and Connectors

Just because you decide to build a software RAID or use an internal disk controller does not mean you need to fit all your drives into a single server or desktop case. In fact, you can chain as many devices as you want to your Linux system, keeping in mind the limits on devices per channel. Remember that ATA is limited to 2 devices per channel, whereas SCSI is limited to 7 or 15 devices per channel.

By housing drives in external cases and connecting them to the external port's disk controller, you can create a formidable storage device. Putting disks in different cases will not cause a noticeable performance hit. However, don't forget that there are maximum cable lengths between devices on individual channels. ATA has a cable length limit of about .5 meters. The cable length limits of an SCSI channel depend on the specific SCSI protocol and transmission type (see Table 2-7).

Table 2-7. SCSI cable length limits

SCSI type	Maximum data throughput (MB/s)	Maximum cable length (meters)
SCSI-1, SCSI, Narrow SCSI	5	6[a]
Fast SCSI, Fast Narrow SCSI	10	3[a]
Fast Wide SCSI	20	3[a]
Ultra SCSI	20	3[a]
Ultra Wide SCSI	40	3[a]
Ultra2 SCSI, Ultra2 Narrow SCSI	40	12[b]
Ultra2 Wide SCSI	80	12[b]
Ultra3 SCSI, Ultra 160 SCSI	160	12[b]

[a] Single-ended
[b] Low voltage differential

The cable length limit applies to the total number of devices on a single channel, including external devices. Remember to take into account not only the cable connecting your controller to the external casing, but also the internal ribbon cable found inside the external case. In the rare situation that you are working with HVD SCSI, remember that it has a maximum cable length of 25 meters, regardless of the SCSI implementation it uses.

Cables come in two types: cheap and expensive. I strongly recommend that you spare no expense when purchasing cables. I've seen countless system administrators drive themselves insane diagnosing an SCSI performance problem only to later realize that they've bought poor quality cables that could not handle the data load. This mantra applies when using both internal and external cabling. Controller card manufacturers often bundle an internal ribbon cable with new controller cards (unless you buy an OEM version). Use these cables at your own risk; their quality varies greatly between manufacturers. It's probably best to find a good source of reliable cables and use them in all your systems, even when cables come bundled with cases or controllers.

Finding the correct external drive cases can be difficult, especially when working with the latest SCSI protocols. Make sure that the connectors match your card, or you will have to buy an expensive converter cable that can hinder performance. It's also important to make sure that the case is rated for the protocol you are using. Some cases may come equipped with the proper external connectors, but the internal cable might be rated for an older SCSI implementation.

Drives come in two sizes: 3.5” and 5.25”. The 5.25” drives can only be placed in 5.25” bays. These drive bays are usually external, meaning that a plastic piece on the front of the case can be removed to expose the drive. 5.25” bays are full-height (3.25”). Full-height (5.25”) drives are uncommon in today’s PC market. You might find very large-capacity drives that have this form factor, but most disks are half-height (1.625”) and have a width of 3.25”. These smaller drives can be housed in 5.25” drive bays by using extension brackets that are usually bundled with cases. They can also be housed in 3.25” bays, which might be external or internal. When buying cases, external bays refer to spots that can be accessed without opening the case. Internal bays refer to drive mounts that can be accessed only when the case is opened.

Connectors

ATA cables use a standard 40-wire, 40-pin ribbon cable, while Ultra ATA (speeds of 33 MB/s and above) uses a 40-pin, 80-wire cable (as shown in Figure 2-26). The connectors and cables might look identical, but you must use the 80-wire with Ultra ATA disks. Be sure to check the specifications when purchasing cables.

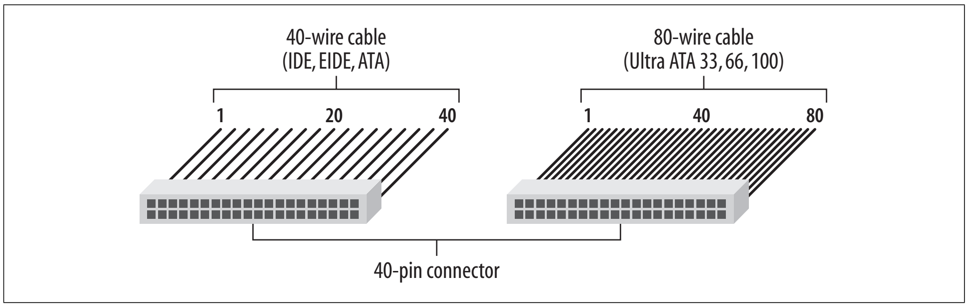


Figure 2-26. ATA cables all have the same 40-pin connectors, but Ultra AT (speeds greater than 33 MB/s) require newer 80-wire cables.

SCSI cables are much more confusing because SCSI cables have undergone more transformations than ATA cables. In most cases, you will be using a 68-pin ribbon cable for internal devices. Just make sure it’s rated for the bandwidth you’re using. Older external connectors have some variation (see Figure 2-27), but in most cases, 68-pin *high-density (HD)* connectors are used. However, newer 68-pin *very high-density (VHD)* connectors are making their way into the market. Decreasing the size of external connectors has made it easier for SCSI controller manufacturers to house multiple channels on a single card.

If you have different connectors on your controller card and your case, it’s easy to find cables that can accommodate you. Check out *http://www.scsipro.com* for custom SCSI cables.

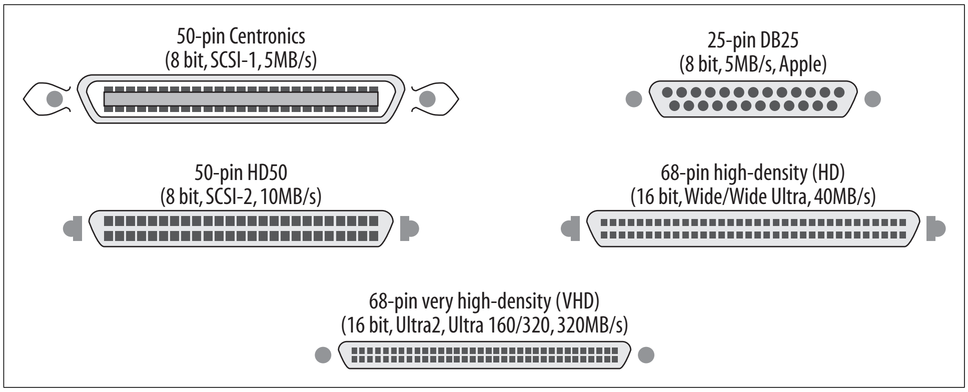


Figure 2-27. Most SCSI controllers use 68-pin high-density connectors for external connections. You may encounter some older connectors as well.

Single connector attachment (SCA)

To facilitate hot-swap disks, IBM introduced the single connector attachment (SCA) for SCSI hard disks. SCA integrates data transfer, power, and configurable options (such as SCSI ID) on a single 80-pin connector, as shown in Figure 2-28. Drives are plugged into an SCA backplane that is then connected to the SCSI bus (usually via SCSI ribbon cable) and the power supply. SCA drives are mounted in trays that slide into the backplane and lock into place, leaving the other side of the disk tray accessible from the outside of the case. These features make it easy to swap disks by eliminating the need to power down the system and dismantle the case.

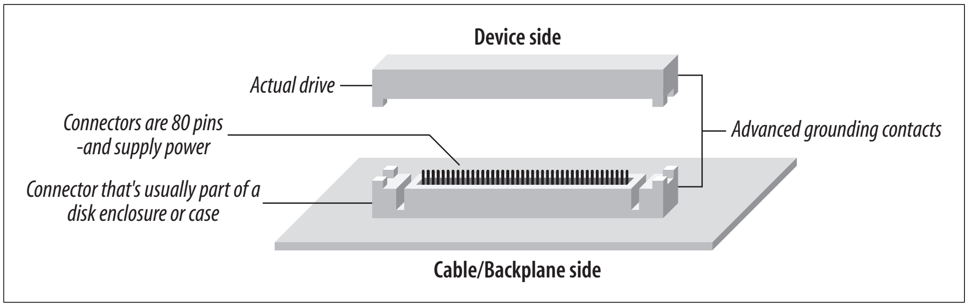

Device side
Actual drive
Connectors are 80 pins -and supply power
Advanced grounding contacts
Connector that's usually part of a disk enclosure or case
Cable/Backplane side

Figure 2-28. SCA disks use an 80-pin connector that facilitates data transfer, power, and configuration parameters.

Since its inception, SCA has been adopted by several manufacturers, and SCA-2 is the current implementation standard. SCA drives tend to be slightly more expensive than standard drives. Cases with SCA backplanes also run on the expensive side, but SCA is a necessity for any system that needs hot-swap capability because SCA is designed to allow power-on disk swapping. Recently, SCA chassis that fit into standard desktop cases have surfaced. Enlight Corporation (*http://www.enlightcorp.com*) manufactures a module that fits into three 5.25" drive bays. It supports up to five SCA disks and connects to an internal SCSI controller. Rackmount case makers also tend to sell custom drive cases that come equipped with an SCA backplane.

Individual adapter modules that allow the use of a single SCA disk with standard 68-pin SCSI cabling and power supply connection are also readily available. I've had mixed results using them, ranging from problem-free performance to SCSI channels running at less than optimal speeds. You will probably also have mixed results, but they do offer a pretty cheap way to get SCA functionality, especially on systems with only a couple of disks. Most cases support a minimum of five or six disks and are very pricey.

Power

Finally, make sure you have an adequate power supply in all your cases, whether they are dedicated drive cases or contain a system and disks. Most cases provide just enough internal power connectors so that the power supply cannot be overloaded. You can purchase power splitters if you run out of connectors, but remember that overloading a power supply can lead to fried hardware. If you find that you have more peripherals than power, you should considering upgrading your power supply. Most cases can be custom ordered with power supplies of up to 450 watts for a minimum of extra cash.

Making Sense of It All

In the final section of this chapter, I'd like to present an example RAID system that I built using parts available at most decent computer stores and online retailers.

The system in question was designed to replace a medium-volume web server that hosts content for video game enthusiasts. The original server was homegrown and quickly became inadequate as the site grew in popularity and moved out of its owner's workplace into a collocation facility. Connecting the system to a larger network pipe solved many of its initial problems, but eventually, the hardware itself became overworked.

The site is mostly static, except for a few moderators who post new articles and reviews each day. It's essential that the site have a 24×7 uptime, so RAID-0 is out of the question. And with my budget, RAID-1 wouldn't work either, because the site frequently distributes large video clips and demos of upcoming games. I simply couldn't afford the extra disks RAID-1 would require. That left RAID-5 as the best option.

In building the new RAID system, I needed to select a motherboard first because the old 32-bit PCI board was causing most of the performance problems on the original server. Because I was interested in high performance, I chose a motherboard that had two 64-bit PCI slots. Each of these 64-bit slots had a dedicated data bus with a throughput of 533 MB/s. (Remember that 64-bit PCI boards run at 66 MHz [66.6 million cycles per second * 8 bytes per cycle = 533 MB/s]). The remaining expansion slots are 32-bit and share a data bus with a throughput of 133 MB/s. The 32-bit bus wouldn't be used for anything except a low-end video card (for local administration), although in the future, a network card might be installed so that the system could be connected to a private administrative network.

In the first 64-bit PCI slot, I installed a high-speed networking card, which should alleviate any networking bottlenecks that the site was experiencing when it was using a 100-megabit Ethernet card. In the second slot, I installed a quad-channel Ultra SCSI 160 controller, giving me to a total disk bus throughput of 480 MB/s (3 * 160 MB/s). The unused bandwidth would help ensure that I didn't saturate the 533 MB/s data bus, while allowing for occasional burst rates that exceed the specifications of my disks and controller.

I found some reasonably priced hard disks that supported a sustained data rate of 40 MB/s and purchased a few external cases. Therefore, I didn't need to worry about cramming everything into a single desktop case. I knew that even the biggest desktop cases house only 7 or 8 disks, and that wouldn't allow me to take full advantage of my controller (480 MB/s ÷ 40 MB/s = 12 disks). After doing some thinking, I decided to purchase twelve drives, and I connected three of them to each controller channel. The drives are housed in the external cases I bought, externally connected to individual channels.

Although the average disk throughput was 40 MB/s, the manufacturer's specifications indicate that burst rates higher than that are common. Because I was using RAID-5, I could configure the array so that the system alternated between SCSI channels during I/O operations. That would help offset the potential for bottlenecks on an individual channel when the disks burst higher than 40 MB/s.

Once all the equipment (see Figure 2-29) was connected, I was left with three drives on each channel, with an aggregate disk bus throughput of 480 MB/s. That left some overhead on my data (PCI) bus to be safe, but didn't waste much of its potential, since I expected the disks would often outperform the 40 MB/s data rate by a small amount. I didn't need to worry about the graphics adapter or network cards interfering with disk throughput, either, because they were installed on separate data buses.

Hardware is always changing and the equipment you buy doesn't always meet your expectations, so it's always a good idea to do research before building or purchasing any system.

As an example of what can go wrong, a former collegue recently told me that he had to argue with his vendor in order to get a system with multiple SCSI backplanes. He had ordered a dual-channel RAID controller in a rackmount case with eight hard disks. But the vendor had designed the system so that there was only a single SCA

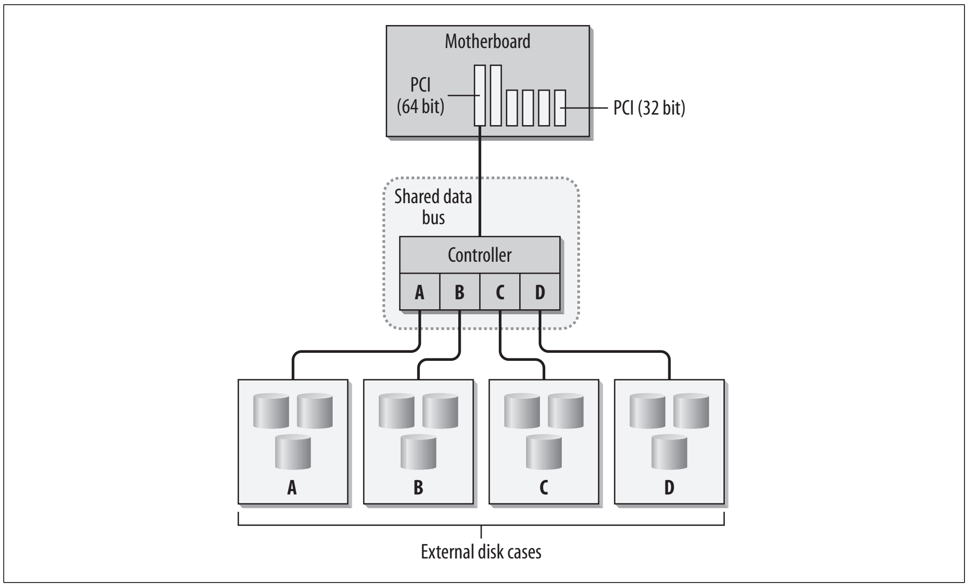


Figure 2-29. My web server contains a quad-channel SCSI controller. Three disks are connected to each channel.

backplane. This meant that all of his data would be travelling over a single SCSI channel and that the second channel would be wasted. The vendor offered the option of adding an external disk-only case for the second channel, but my colleague found that unfeasible due to the the high price of server colocation it added. In the end, my collegaue had to swap his components into a new case with two backplanes. The vendor ate the cost, but it took an extra week to get the system online.

Also, remember compatibility issues. I recommend checking relevant mailing lists and web sites to make certain that your disk controllers will work properly with your motherboard and network controller.