Storage Networks

DANIEL J. WORDEN

Apress™

Storage Networks

ISBN (pbk): 1-59059-298-0

Printed and bound in the United States of America 12345678910

Lead Editor: Jim Sumser

Technical Reviewer: Jonathan Hassell

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, John Franklin, Jason Gilmore, Chris Mills, Steven Rycroft, Dominic Shakeshaft, Jim Sumser, Karen Watterson, Gavin Wray, John Zukowski

Assistant Publisher: Grace Wong

Project Manager: Beth Christmas

Copy Manager: Nicole LeClerc

Copy Editor: Ami Knox

Production Manager: Kari Brooks

Production Editor: Janet Vail

Compositor: Diana Van Winkle

Proofreader: Elizabeth Berry

Indexer: Valerie Perry

Artist: Kinetic Publishing

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY, 10010 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, e-mail orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, e-mail orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

CHAPTER 5

RAID Levels and Logical Volumes

Direct-attached storage and Network Attached Storage have a great many components that work exactly alike. At this point in the discussion, I want to move to the storage stack itself and leave the implications of moving data across a network. In this chapter, you'll see the options that are available to a Storage Area Network for protecting and preserving data, as well as strategies for minimizing the performance penalty paid for that protection.

Specifically, I'll cover RAID levels from the perspectives of how they work and where you would use them. Then moving beyond the management of physical disks, we'll look at the role of the file system, and some of the options that are available.

This chapter will introduce some of the performance penalties to be paid for ensuring access to disk resources. The relative impact of these penalties will be illustrated in Chapter 10, where I review the performance of a specific 1GB file set and the performance metrics that various storage configurations yield.

I'll introduce logical volumes as supported in both Linux and Windows. The relationship of logical volumes to the hard disk devices and the file system is also explained in this chapter.

RAID Levels and What They Mean

The definition of RAID levels was introduced in Chapter 1. Given the review you've had for disk and networking technology, let's take a closer look at RAID levels.

Several terms are frequently used either interchangeably with RAID levels or in place of them. As they are descriptive and relevant, I thought it best we start with those concepts and then evaluate how they are implemented in various settings.

Striping—aka RAID Level 0

In Figure 5-1, you can see a representation of several hard disk drives with a physical volume cut away in each. *Striping* refers to the strategy of writing a given file to blocks on different volumes in a parallel fashion. The benefit results from minimizing the amount of time the heads are required to move over the platters and to put as many of the disk resources as possible to work concurrently.

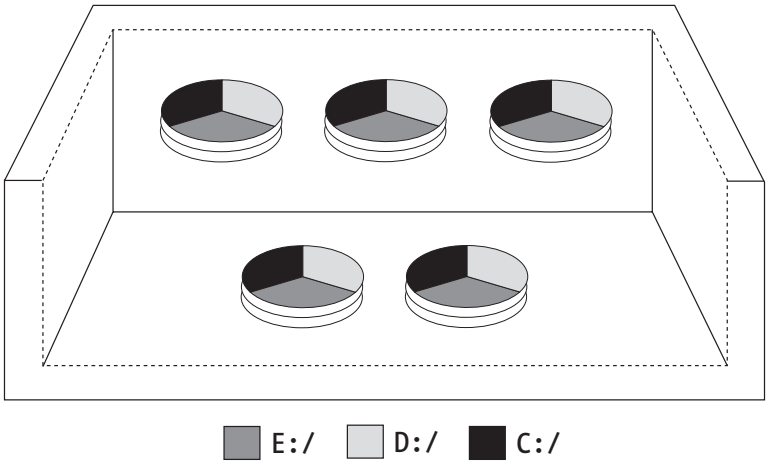

Figure 5-1. An array of disks with striping

Although read/write performance is enhanced with striping enabled, the cost is increased exposure to a head crash. Losing one of the disks of a stripe set results in the loss of the striped drive as a whole.

Spanning: When Only the Very Biggest Will Do

Figure 5-2 depicts the same array of disks created as a single logical drive. The logical drive spans all four disks and is represented as one storage pool. The advantage is that file sets larger than any one disk can then be stored intact.

A spanned drive may not take advantage of having four sets of read/write heads to make reading and writing more efficient; instead it may write sequentially, first filling up one drive, and then "spilling over" to the next. However, the risk or cost introduced by striping is also inherent in this storage strategy. Loss of one drive results in loss of access to all of the others.

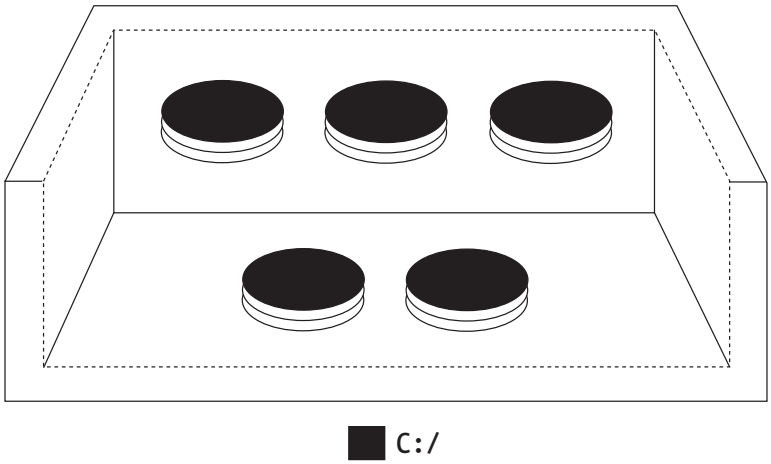

Figure 5-2. An array of disks spanned as one logical drive

Mirroring—aka RAID 1

Mirroring refers to the process of creating an exact duplicate image of a drive and providing a process to "fail over" to that mirror image in the event that the primary drive fails. Figure 5-3 shows the drive array as two drives with mirrors.

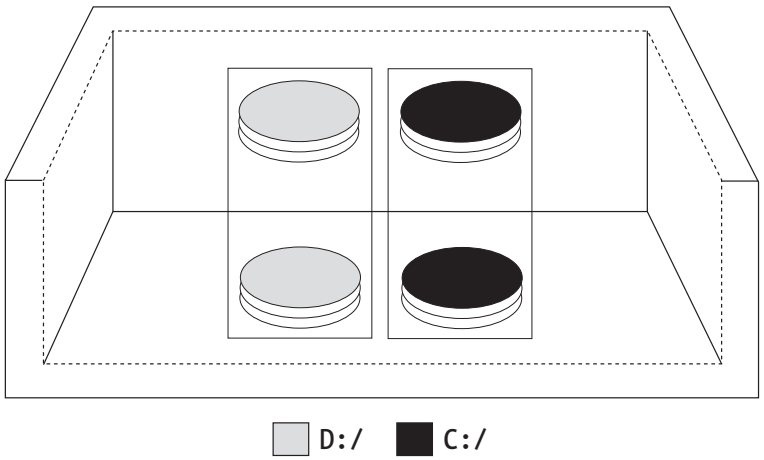

Figure 5-3. An array of two mirrored disks

Mirroring requires the completion of two sets of write activities before a transaction is completed. This necessarily increases the amount of work performed by the hardware for any write operation, which has a negative effect on its performance. Read transactions aren't affected by the mirroring process, which makes mirroring an effective strategy for any data you wish to make highly accessible for read-intensive applications.

Mirrored Stripe Set—aka RAID 0+1

To mitigate the performance hit from mirroring, you could configure your array as a combined stripe set with a mirror for that striped drive. This configuration is shown in Figure 5-4.

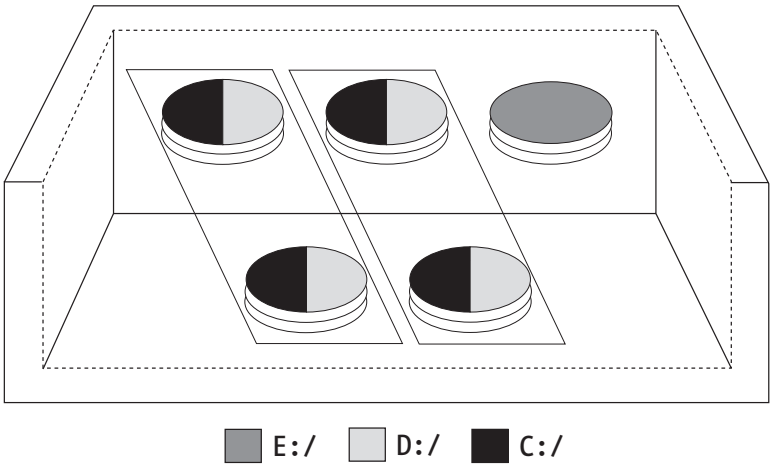

Figure 5-4. An array of striped and mirrored disks

Mirroring the stripe set mitigates the risk of data unavailability due to a disk failure. The performance penalty from the mirror is reduced to the extent possible by striping the write operations across multiple drives.

Stripe with Parity—aka RAID 3

Providing similar protection to RAID 0+1, but using a different mechanism, a stripe with parity dedicates a drive to maintaining extra information about the data that is written to the stripe set. In the event of a disk loss, the data can be re-created from the parity disk and remaining drives in the set.

Parity in this context is determined by invoking a calculation on each block of data written across the array. Using the exclusive OR operation (XOR), which says the value of the block may be odd or even but not both, the sum of the values is stored as a parity value. In the event that a disk is lost, the value of each block of data can be derived from recalculating the other values and the parity sum. Because the XOR value may only be true or false (odd or even), if all values are known plus the sum, the missing parity value can be determined. From that value, the data itself can be reconstituted on a new disk.

As shown in Figure 5-5, the key advantage to using a dedicated parity disk is the savings from not mirroring every data disk in a pair.

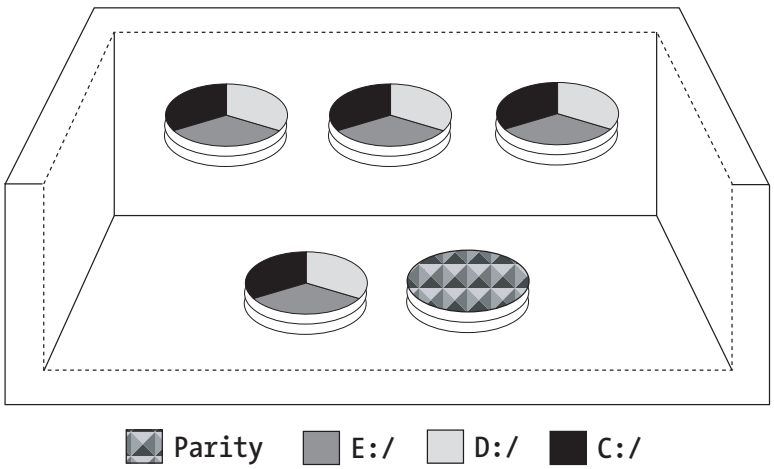

Figure 5-5. A stripe set with a dedicated parity disk

As you can see from the diagram, the cost is one disk for the entire array. This means that if you have 2 disks in a RAID array, it costs you one for parity. If you have 32 disks in the array, it costs you the same disk dedicated to parity, plus the performance hit on write operations.

RAID 5

RAID 5 operates in a similar fashion to RAID 3, with the key distinction that each drive maintains parity values as well as data. A RAID 5 array can reconstruct a data set if one or more of the disks in the array goes awry. Figure 5-6 depicts a RAID 5 array.

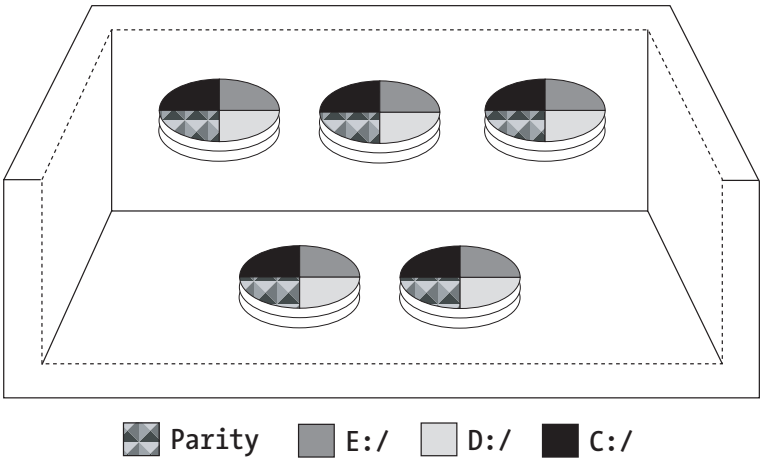

Figure 5-6. A RAID 5 array with several disks

By distributing the parity information across multiple disks, the RAID 5 array provides a high degree of fault tolerance. As you might expect, a penalty is incurred for write transactions in terms of the time compared to writing to a stripe set.

RAID Read/Write Operations

It should become increasingly clear that the main consideration for any storage strategy is the value of the data to be stored. Archival storage for historical purposes, record retention for legal compliance purposes, and casual lookup data require a different storage and redundancy strategy from high-volume transactions. Before you can map your requirement to a particular redundancy strategy, you should first be completely familiar with the inherent characteristics of the options available.

Stripe Setting

Let's build on the read/write functions of a single hard disk. When you create a stripe set, you do more than simply distribute the operation across two or more disks: you've harnessed the disks together, and that coordination must be taken into account as part of your configuration. True, you could simply choose the defaults and mark the job done on your to-do list, but to tune your storage network, you need to know how the various components work together, as well as which options to select.

When a hard disk leaves the factory, it has been low-level formatted with preset block sizes, regardless of the intended operating system that will perform a higher level format on installation. At the hardware level, block sizes for data read and written to disk remain constant—typically defined by manufacturers as sectors containing 512-byte blocks. Each of the drives in the stripe set must write a block for any given record, a stripe set with four drives has a logical block size for a record set at 2048 bytes.

Mirroring

A write so nice, they performed it twice: The duplication of effort involved in mirroring is perhaps the easiest form of RAID to visualize. As you would expect, the increased overhead of duplicating drives has an effect on performance. Mirroring requires two drives: a primary drive and a mirror drive. In the event of the failure of the primary drive, the system will fail over to using the mirrored drive instead of the primary. With hot-swappable hardware, you can remove the dead drive, install a new one, and create a new mirror, which then duplicates the contents of the new primary, thereby re-creating your mirrored set. There are some issues with being able to boot servers with mirrored boot disks. These are discussed in greater detail in the section "Hardware RAID" a little later in this chapter. Additionally, lab benchmarks for software and hardware mirroring performance under Gigabit Ethernet is included in Chapter 10.

Stripe with Parity

The metadata for the parity drive requires that for every write operation, a metadata write also occurs. Retrievals also require first a lookup of the metadata to obtain the disk addresses, and then a seek operation to position the heads to read the data where located. It isn't difficult to see that contention due to multiple requests wouldn't be the ideal situation for a RAID 3 drive. However, the ability to spread the storage of files across multiple hard disks and maintain that parity information in another location is especially suited for large files to be accessed by a single application at a time. Think medical imaging files or audio/visual content to be edited. Availability and fault tolerance are critical, but it's unlikely (though not impossible) that more than one application would want the data at a time. Of course, contention can result from multiple users looking for different files, but the advantages in this case should outweigh the costs.

Distributed Parity

RAID 5 devices determine how to interleave data on the basis of logical block definitions. Logical block sizes are set by the file system when the disk is formatted and can be defined as any size as long as it's a multiple of the physical disk block. Where the logical block size is the same as the physical block size, a single disk might be accessed to handle a particular block rather than requiring data to be accessed across multiple disks.

For some of the less-expensive RAID 5 offerings, parallel reads of the disks aren't supported in order to keep costs down. Instead, data is picked off sequentially from the chain of drives. The top-of-the-line RAID 5 implementations can concurrently pull data from each disk in the set as needed to satisfy the request.

Unlike a striped array, RAID 5 can accommodate the writing of a record set that isn't as long as the block size defined by the entire stripe group. To do this, a function known as *read-modify-write* is used. As a result of this function, the old data block, old parity block, and new data block are used to create a new parity block. The significance of this is that up to two read and two write operations must be performed in order to complete the writing of a single logical block. To ensure this doesn't become a performance nightmare, most RAID 5 hardware vendors provide caching to reduce the amount of time required to look up old block and parity information.

Hardware RAID

Hardware RAID is the term used to refer to a RAID array that is managed by software built into the RAID controller, rather than through the operating system or application software such as a database server. Hardware RAID for file servers is no longer an expensive option. Readily available RAID controllers from Rocket Raid, Highpointe, and Promise Technologies—not to mention solutions available from Adaptec and others—will provide RAID levels 0, 1, and 0+1.

Generally speaking, hardware RAID isn't mix and match. An IDE RAID controller will allow you to configure up to four drives on its two channels. However, because you're looking at risk and reward at the hardware level, it bears consideration that in this setting you've introduced a single point of failure—the RAID controller itself. A mirroring strategy that allows you to compensate for not just a failed drive, but also a failed controller may be the level of insurance you want. In that event, it would be a mistake to think that you were fully protected because you had mirrored the drive and not the card.

Hardware RAID allows your system to boot regardless of whether both the mirror and primary drives spin up successfully. Obviously, at least one of them has to be available at boot time, or there would be zero boot devices available.

Software RAID

Both Linux and Windows servers provide facilities to use the operating system to manage RAID disk management. A discussion of how this is implemented and the various performance findings for an identical hardware environment are presented in Chapter 10. Like any operation that involves the CPU, handling RAID activities through software necessarily creates competition for CPU and memory resources. In small workgroup settings, this will likely be a nonissue, as the performance differences between software and hardware RAID were negligible for levels 0, 1, and 0+1. However, you can expect that scalability issues will arise when relying on software RAID under larger workloads.

RAID Strategies

A hardware mirror of your /, /boot, and /swap directories will ensure that your system stays up and running during a boot disk failure. Electing to mirror your /winnt directory in software across two IDE controllers (as opposed to two channels on the same controller) can insure you against a controller and hard disk crash. User directories and disk resources declared for application data sets such as mail or database repositories can also provide a measure of availability in the event of a hardware failure.

It should be noted that RAID is a strategy for ensuring the system resources are available, not a replacement for backups. RAID does nothing to protect against file corruption or a runaway delete due to script or operator error. The job RAID is designed for is to faithfully reproduce data contained on more than one hard disk. There are other strategies and techniques that must be applied to provide more protection and insurance against other perils.

Logical Volume Management

If RAID manages the physical volumes of more than one disk, Logical Volume Management (LVM) is an intermediate step for the administration of those volumes to be presented to the file system. For purposes of this discussion, we'll assume that a RAID array has been designed and implemented. The next step is making those disk resources available to the file system.

Logical volumes can be considered named aggregates of the physical disk volumes. They serve as a mechanism for managing partitions, including their initial configuration and installation, and subsequent changes in size and allocations. File systems aren't defined on disks themselves (RAID'd or not), but instead on volume groups, which are created and allocated using the Logical Volume Manager.

Most of the terminology I use to refer to volume groups is descended from their initial implementations in HP-UX and AIX (which is where I first worked with them) and currently available in Linux. However, for those of you who are sensitive to perceived slights against Windows Servers, be aware that the use of dynamic disks under disk management in Windows since Windows Server 2000 is very much like using LVM. In some cases, however, there are features available as part of Logical Volume Group management and services that don't have a Windows equivalent. In such cases, I'll call out the difference explicitly.

Logical Volume Capabilities

Specifically, using logical groups, you can

- Increase and decrease the amount of disk available to a particular file system.
- Add physical storage to an existing volume group.
- Create snapshots of file systems in a point in time.
- Migrate volume groups to new devices.

These capabilities aren't exactly self-explanatory. A brief description of what each capability affords and where you might use it follows.

Allocating Disk Resources

Because file systems are mounted on volume groups, you may use the Logical Volume Group Manager to handle changes in your physical disk environment without affecting your data and files. This also provides an opportunity to separate roles and responsibilities among team members, where a published schedule of volume groups can be provided to, say, DBAs who aren't allowed access to the physical server environment. In the event that a systems administrator sees a hard

disk failure predicted under a SMART monitoring tool, the change can be made without impact on the other infrastructure components.

Unlike RAID, a volume group may have many file systems mounted on it. Although the dynamic disk manager under Windows 2000 will allow you to put different volumes across the hardware, you assign a single drive letter to the volume or mount it under a folder. This means that volume groups provide the additional benefit of being able to mount multiple file systems on the same volume group.

Adding and Removing Disks

Adding and removing physical storage while the system remains operating is something that your hardware environment must first support, of course. But with hot-swappable drive support available in Netfinity, Dell, and HP servers among others, the next question is how to manage that process. Using Logical Volume Management, you would first install your new disk, and then add it to a volume group. You might then elect to use the other interesting feature of volume groups: the ability to migrate volume groups to new devices. This way, you can use LVM to swap out devices without service interruption. The applicability to network storage administration should be obvious.

Snapshots

Backing up file systems and data sets that are in constant operational use is an interesting exercise in competing interests. Naturally everyone wants the data to be protected, and just as understandable is the desire to keep it available. Backing up open files is problematic, given the different locking mechanisms that exist. Or transaction dependency might exist between several databases, meaning that to back up one and then the other sequentially would result in an inconsistent state. One way this has been accommodated using logical volumes is the inherent ability to take a snapshot of a file system on it without having to allocate an identical amount of disk space for the snapshot. This is accomplished with a form of mirroring, except the only thing that is recorded in the snapshot are the changes made to the data from the time it was taken.

In this way, a duplicate copy of static or unchanged data isn't made, and the backup comes from the operational file system. Where changes are being made to the underlying data, the backup is taken from the snapshot.

It should be noted that this incurs some overhead, as it records changes being made to the system. Once the backup is completed, the snapshot file system needs to be removed, or it will continue logging changes until it fills up and breaks.

Summary

You should now have a better appreciation for the utilities that exist for managing multiple disk resources within any given server environment. Although much of the discussion on RAID and Logical Volume Management is applicable to direct-attached storage devices, most frequently those devices would themselves be used in Network Attached Storage solutions, such as file or application servers.

In the same way there is a stack of layers in the network from physical to application, so too is there a hierarchy of services in managing disk resources. In this chapter, you've been introduced to a number of key concepts and how those services can be implemented. Whereas RAID is a way to protect against disk failure, logical volume groups and dynamic disks are a way to abstract the physical devices into a collection of resources, which can then be managed by name rather than individual physical particulars.

The value of hot-swapping drives is readily apparent to anyone who wants a server with constant up time. From this chapter, you should have a sense of how you might go about designing these features in your storage network, and additionally how you might structure your environment to ensure good backups.

By approaching storage from the standpoint of first a single drive, then a collection of drives, and ultimately a dynamically managed pool of drives, you should be gaining a sense that a seemingly monolithic storage network is actually a nested set of components. Arriving at an understanding of these interdependencies will allow you to not only effectively manage a storage network, but also design one and tune its performance over time.