

CHAPTER 4

RAID

As it was originally proposed, the acronym RAID stood for Redundant Array of *Inexpensive* Disks. However, it has since come to be known as Redundant Array of *Independent* Disks. RAID was originally described in the paper “A Case for Redundant Arrays of Inexpensive Disks (RAID)” written by three University of California Berkeley researchers: David Patterson, Garth Gibson, and Randy Katz. The concept was presented at the 1988 conference of the ACM Special Interest Group on Management of Data (SIGMOD).

In the original Berkeley paper there were five RAID levels (1 – 5). RAID-6 was added later as an enhancement for RAID-5. In time other levels have been implemented by various companies using the concepts described in the original proposal. These include RAID-0 (striping without redundancy), and multilevel RAID (striping across RAID arrays), and others variants.

Section Goals

Upon completion of this section you should be able to:

- Understand the data redundancy and protection methods used for RAID
- Explain the different RAID levels
- Understand the advantages and disadvantages of each level
- Explain the different ways in which RAID can be implemented

Why RAID?

Patterson, Gibson and Katz proposed RAID as a solution to the impending imbalance in system performance. Although the speed of CPUs and memory size were undergoing exponential growth, the performance of system I/O was increasing slowly. Data storage access was, and is, limited by mechanical factors, such as rotational speed and latency. Purely electronic components such as a microprocessor or DRAM chip do not have these limitations.

Their solution to the impending crisis was the *disk array*. In the original proposal a disk array would use several low cost, low capacity drives in place of a single high capacity, high cost hard disk. The only problem with their solution is that the average MTBF (Mean Time Between Failure) for the array is lower than for a single disk. They overcame this by storing *redundant data* or *parity data*, which allows continued data access and even reconstruction in the event of a disk failure.

The large capacity, high-speed drives available today might seem to make the need for RAID irrelevant. However, the need for data storage capacity and high speed data access is growing faster than the capacity and transfer rates of individual drives. RAID is still the best solution for providing large amounts of data storage at a reasonable cost, with the added benefit of data protection.

RAID Basics

Before we begin to learn about the different types of RAID there are a several concepts you must understand. These concepts are fundamental to understanding how a RAID subsystem works, they are:

- Parity
- ECC
- Exclusive OR (XOR)
- Striping

Parity

Simple parity is a method by which a check bit is added to data so that errors can be detected. In a computer data is organized into bytes, which consist of a series of 8 bits. These bits are transmitted as a series of on or off signals that represent the binary values of 1 or 0. A series of 0's and 1's can represent an alphanumeric character or some other type of data. When the simplest form of parity is used it is usually referred to as *Even Parity* or *Odd Parity*. These two methods add an additional bit to each byte that indicates whether the number of 1's in the byte is an even or odd number. If the number of bits received does not match the parity indicator, an error has occurred.

Simple parity can only detect errors, they cannot be corrected. In addition, only odd-bit errors can be detected; even bit errors cancel out and remain undetected. However, the system has no way to know what bit was changed. The only way to correct the data is to retransmit the information. Error correction requires additional parity bits or more sophisticated calculations to transmit data that is both protected and recoverable. The two most common methods of this kind of data protection are *Error Correction Code* (ECC) and *XOR parity*.

ECC

Error Correcting Code is a process of generating parity values that can be used to both detect errors in data and to reconstruct the correct data. ECC allows data that is being transmitted to be checked for errors and corrected without retransmitting the data. ECC is now frequently designed into data storage and data transmission hardware as data rates, and therefore error rates, increase.

Adaptec SCSI RAID controllers use an Even Parity ECC algorithm for data error detection and correction in cache memory. For example, assume a single byte (8-bits) is represented by bit values 10000110. Adding a single 1 bit to the end of the byte results in an even number of 1's. This will immediately allow the system to determine if the data has been altered at the bit level. If the 1's bits add up to an even number of bits then no error is detected. In reality the data is protected by ECC in addition to the even parity. ECC allows for correction and not just detection of the error.

For cache memory, ECC is used as follows:

- When a unit of data (or word) is stored in RAM or peripheral storage, a code that describes the bit sequence in the word is calculated and stored along with the unit of data. Each 32-bit word needs an extra 7 bits to ensure single-bit error correction. Each 64-bit word requires 8 additional bits.
- When the a unit of data is requested, a code for the stored, and about-to-be-read, data word is again calculated using the original algorithm. The newly generated code is compared with the code generated when the word was stored.
- If the codes match, the data is free of errors.

If the codes do not match, the missing or erroneous bits are determined by comparing the codes which then allows the erroneous or missing bits to be corrected.

- No attempt is made to correct the data that is still in storage because it will be overlaid by new data and, if the errors were transient, the incorrect bits will be overwritten.

An error that occurs again at the same place in storage after the system has been turned off and on again indicates a permanent error.

In general, ECC increases the reliability of any computing system without adding significant cost, although there is some additional computational overhead for the ECC calculations. This overhead typically results in a 2% – 3% increase in memory access time, which is insignificant for any current application.

NOTE

Parity memory is not necessarily the same as ECC memory. To ensure that ECC protection is available, the memory module must be a true ECC-capable memory module.

XOR

Exclusive-OR (XOR) is a logical operation used to generate a unique value for each pair of bits in a data word. XOR is used to calculate RAID-5 parity for data and also by which the data can be reconstructed using the previously calculated parity value.

The following is the Boolean table for XOR calculations:

Bit 1	Bit 2	XOR Result
0	0	0
0	1	1
1	0	1
1	1	0

Where simple parity compares all of the bits in a byte, XOR compares every two bits. If the two bits are the same then an even XOR parity bit (0) is generated. If the bits are different then an odd parity bit (1) is created.

The benefit of XOR parity is that in addition to error detection, XOR provides error correction and data reconstruction. By comparing the remaining values in a data string and XORing the previously written parity values, the missing data can be reconstructed. This is possible because when the XOR operation is run twice on the data parity value, the result is the original value of the data.

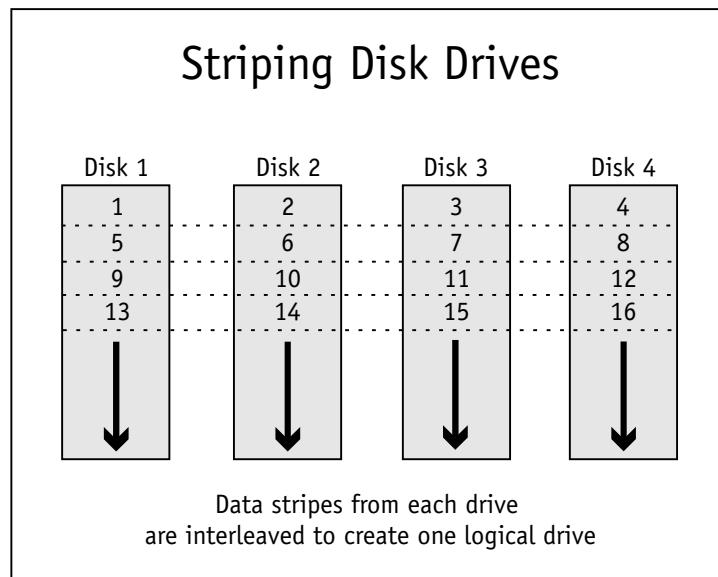
However, there is a substantial performance cost when using XOR parity. XOR parity calculations are more intensive than simpler parity calculations. As a result XOR parity can have a significant impact upon the performance of the host system. Any RAID which uses XOR parity should only be used on a controller that provides for hardware-based XOR through a processor or coprocessor that handles the XOR calculations in place of the host CPU.

Striping

Striping is a method by which a block of data can be broken down to smaller pieces and simultaneously written to multiple locations. By writing the data to multiple locations, in smaller increments, performance can be increased substantially.

In the example below the data is broken down into block size increments. Each block is then written to a separate drive. Because each of the four drives can write one block at the same time, the entire operation takes one fourth (1/4) of the time it would for a single drive to perform the same operation. In other words, if it takes 10ms to write each block, the four drives can write all the blocks in 40ms. It would take a single drive 120ms to perform the same operation.

Reading the data is also substantially faster, especially in parallel disk access RAID configurations. With parallel access, the transfer rate is the aggregate transfer rate of all drives participating in the I/O request. Mirrored disks can also deliver increased read performance, although they are not striped, by simultaneously responding to read requests from both disks.



RAID Levels

RAID-0

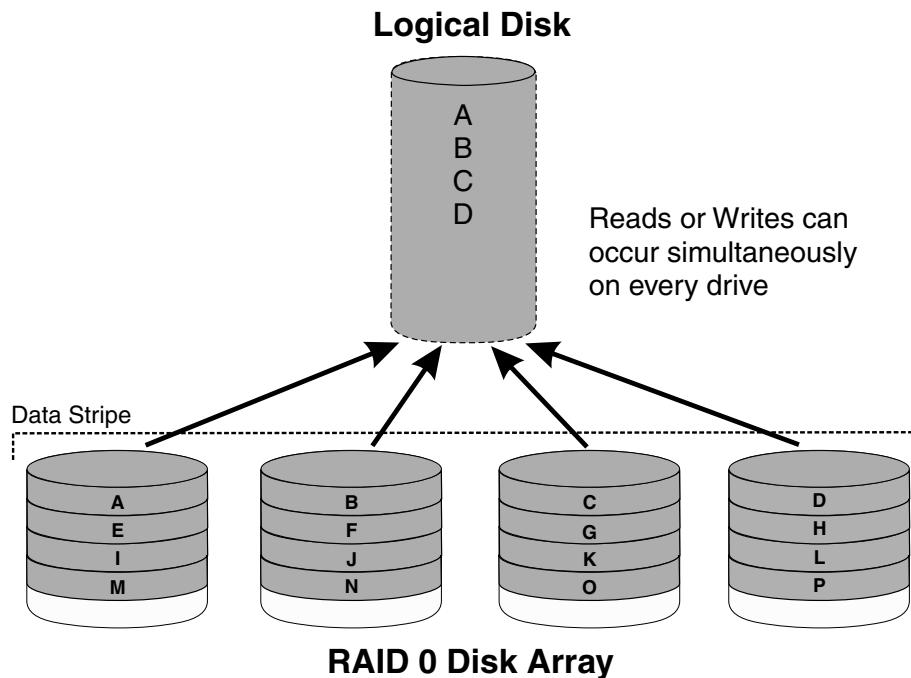
RAID-0 provides striping for performance, but does not offer any fault tolerance. The failure of a single drive will result in the loss of all data on the array.

RAID-0, was not defined by the Berkeley paper and is not true RAID because RAID-0 does not offer any redundancy. However, of all the RAID levels, RAID-0 offers the greatest increase in performance.

RAID-0 is known as *data striping*. In data striping the data being used is broken down into several smaller, equally sized pieces. Each piece is then written to or read from multiple drives. This process increases I/O bandwidth by simultaneously accessing multiple data paths.

Because RAID-0 does not offer any redundancy, a single drive failure can result in the loss of all data in a striped set. This means that all of the data on all of the drives could be lost if even a single drive fails. For this reason RAID-0 should never be used in mission critical environments and frequent data backups are recommended.

RAID-0 is ideal for high bandwidth applications, such as video production, editing, pre-press applications or any environment where read/write performance is paramount, but data protection is less of a concern.



RAID-1

Disk Mirroring - If one drive fails, all the data is available on the redundant drive with no significant loss in read/write performance.

Disk Duplexing - In addition to disk mirroring, a redundant controller card is added for additional subsystem fault tolerance.

RAID-1 defines *mirrored arrays*. In a mirrored array a complete copy of all of the data is written to two drives simultaneously. Because data can be retrieved from either drive, RAID-1 provides enhanced read performance. If the controller can perform two simultaneous reads per mirrored pair then a small increase in read performance can be achieved. However, the write performance is slightly less than a single disk because each data write generates two write commands that are directed to separate devices.

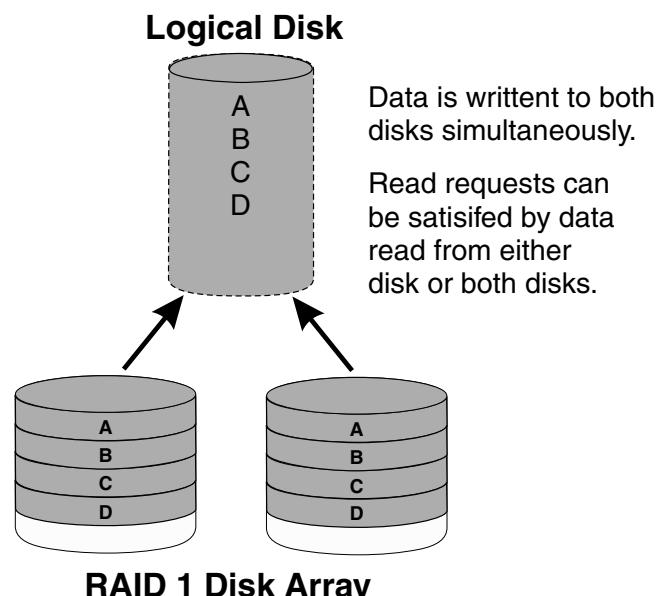
RAID-1 provides full fault tolerance for a single drive failure. Should one drive fail in a mirrored pair, then all of the data is still accessible on the duplicate. Because it is not necessary to rebuild data when a drive fails, there is no significant performance degradation when accessing the remaining drive. However, the remaining drive operates without any fault tolerance until the failed drive is replaced and the mirror set is rebuilt.

Additional fault tolerance can be achieved through the use of two controllers. This is known as *duplexing*. The difference between mirroring and duplexing is the number of controllers used. Duplexing uses a separate controller for each drive and the controllers must support having the drives on separate controllers. This adds additional fault tolerance in the event that the failure is on the controller rather than on the drive.

If the data path of the disk subsystem is heavily saturated, then duplexing can provide for a moderate performance gain over mirroring. By moving the mirrored data stream to a second bus, the available bandwidth on the first bus will be increased.

The most prominent objection to RAID-1 is that it is one of the most expensive methods of using RAID. This is because twice the amount of storage is required for each piece of data that is written. In other words, two gigabytes of storage space is required for each gigabyte of data. For this reason RAID-1 is typically used only for configurations where overall capacity is not a primary consideration.

RAID-1 is ideal for financial applications, such as payroll, where data availability and performance are critical. It is also frequently used for the boot device in servers. RAID-1 should be used where cost and overall capacity are not as important as overall performance and fault-tolerance.



RAID-2

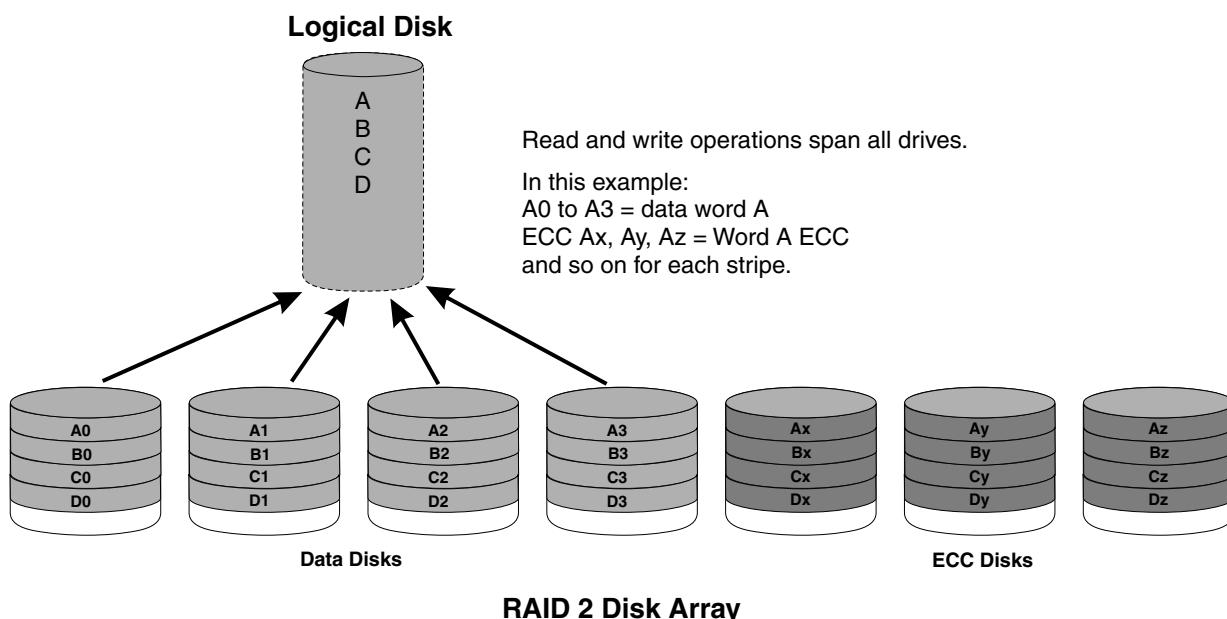
Parallel access with Hamming ECC

RAID-2 uses Hamming Error Correcting Codes (ECC) to achieve fault tolerance and parallel access for I/O operations. Hamming Error Correcting Codes were developed by Richard Hamming in 1950 to detect and correct bit-level errors in computer data.

In RAID-2 bytes or words of data are striped across an array of disks. This use of byte striping makes it possible for extremely high data transfer rates when accessing large amounts of data. While this RAID level is efficient for large amounts of data, the read/write of smaller data blocks provides unacceptable performance. This RAID level also requires one check disk for each data disk to detect and correct any errors. For these reasons RAID-2 is almost never used.

Hamming code error correction was used for early model drives without built in error detection. However, it has a high CPU overhead for the host system and SCSI drives now include built-in error detection and correction, which eliminates any benefits of this feature.

The following example is sufficient to detect and correct an error on a single disk. For multiple disk error correction more ECC disks are required. Actually using RAID 2 would typically require at least 14 disks, 10 for data and 4 for ECC.



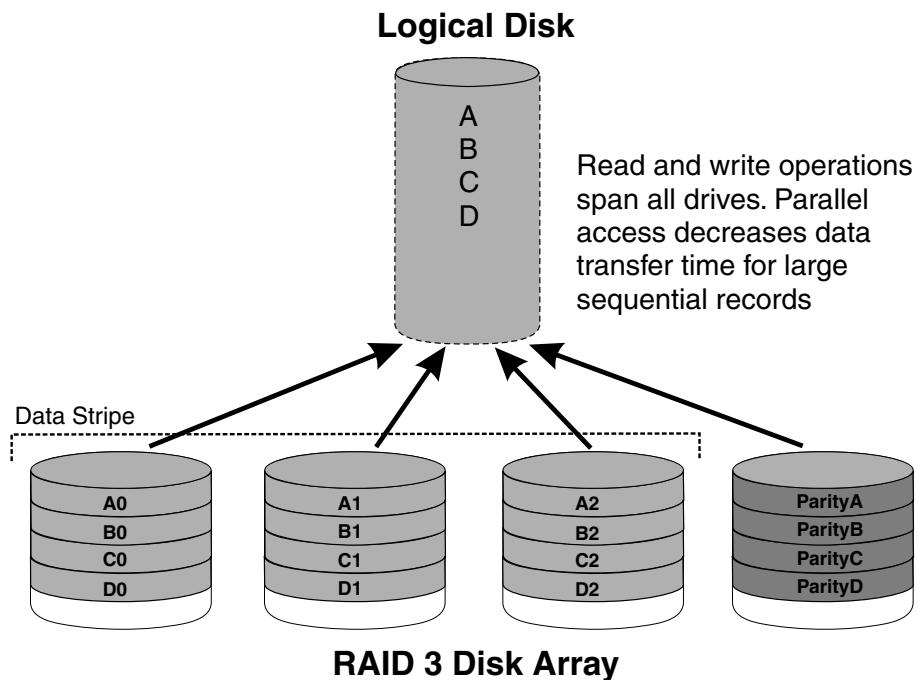
RAID-3

Parallel access with a dedicated parity disk.

RAID-3 is essentially a fault tolerant version of RAID-0 that trades some capacity, for the same number of drives, to provide a high level of fault tolerance. It takes advantage of data striping, except that one drive is reserved to store parity information. This parity data is used to maintain data integrity across all drives in the subsystem. The parity drive itself is divided up into stripes, and each parity stripe is used to store parity information for the corresponding data stripes across the array. RAID-3 achieves very high data transfer rates by reading from or writing to all of the drives in parallel while adding the ability to reconstruct data if a single drive fails. RAID-3 is most efficient for moving large sequential files.

The stripes of parity information stored on the dedicated parity drive are calculated using an Exclusive OR (XOR) function. If a data drive fails the data can be reconstructed using the parity information and data from the remaining drives. If the parity drive fails, the parity is recalculated from the data drives and written to the replacement drive.

RAID-3 is not commonly used because its optimum performance depends on having synchronous data access to all disks (parallel I/O). This requires use of a technique called *spindle synchronization*, where all the disks are spinning so that any data request can be satisfied by a single read operation of all disks simultaneously. Spindle synchronization is a special feature not generally available on most disk drives. Also, because the parity is on a separate disk, write operations face a performance bottleneck because parity must always be calculated written to that device.



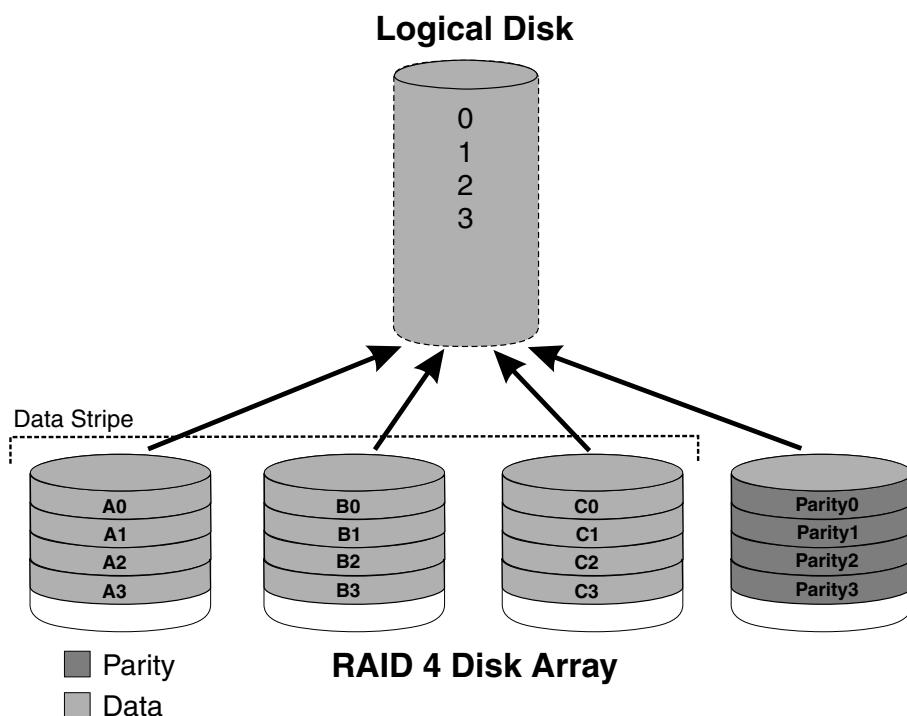
RAID-4

Independent access with a dedicated parity disk

RAID-4 is similar to RAID-3, but emphasizes performance for a different type of application. This RAID level performs better for smaller file access, such as database transaction processing, rather than large sequential files. Another difference is a larger stripe depth, usually two blocks, which allows the RAID management software to access the disks independently rather than being optimized for simultaneous, parallel access. This essentially replaces the high data throughput capability of RAID-3 with faster data access in applications that primarily issue multiple requests to read small blocks of data.

RAID-4 is similar to RAID-3, except that in RAID-4 sector interleaving is used in place of byte interleaving. This is much more efficient than RAID-3. RAID-4 makes use of block-level striping with a dedicated parity disk. Because it uses block level data striping instead of bit level, RAID-4 is also more efficient than RAID-2.

As in RAID-3, RAID-4 has an inherent bottleneck on the dedicated parity drive. As data is written to the array, the parity encoding is part of each write operation. This relegates RAID-4 to applications that primarily read data. Consequently, this RAID level is rarely implemented.

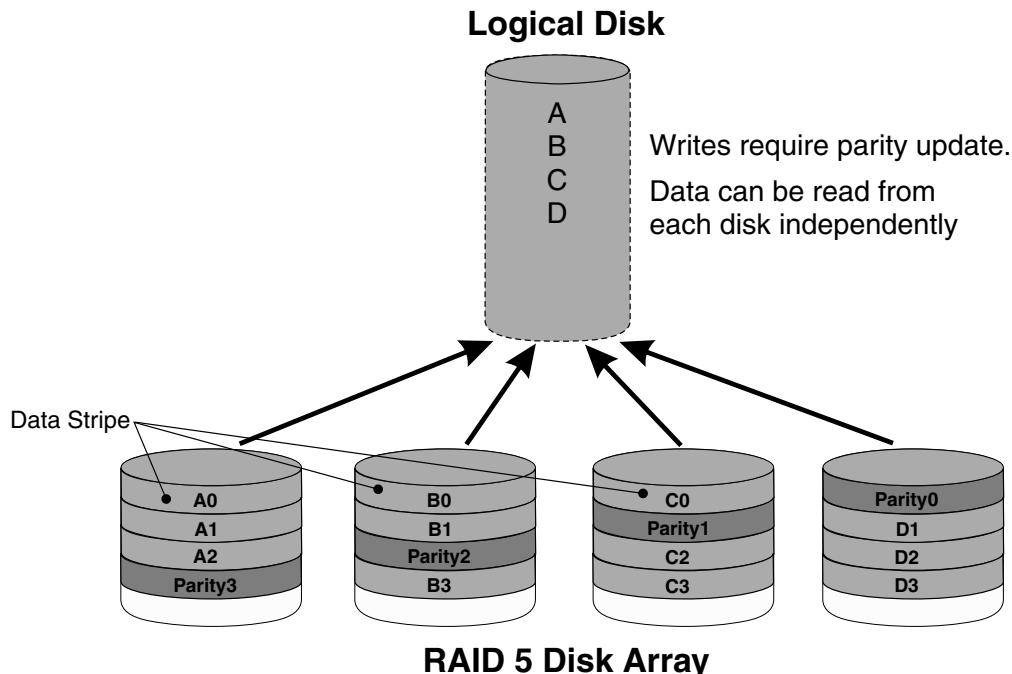


RAID-5

Independent access with distributed parity

RAID-5 is similar to RAID-4, except that the parity is also striped across all of the drives. For large data read/write functions the performance is the same as RAID-4, but because the parity disk is not a bottleneck, smaller I/O functions do not affect performance. However, because of the parity striping there is significant overhead in doing data rebuilds or reading data from an array with a failed disk.

Due to trade off between performance, fault tolerance, and cost, RAID-5 is probably the most common RAID implementation. This RAID level is a good choice for file and application servers. It is also commonly used for Internet and intranet servers.



When data is written to a RAID-5 array, the parity information must be updated. This is accomplished by finding out which data bits were changed by the write operation and then changing the corresponding parity bits in the following process:

1. Read the old data.

2. Write the new data.

3. XOR the old data with the new data.

The result is a bit mask which has a one in the position of every bit which has changed.

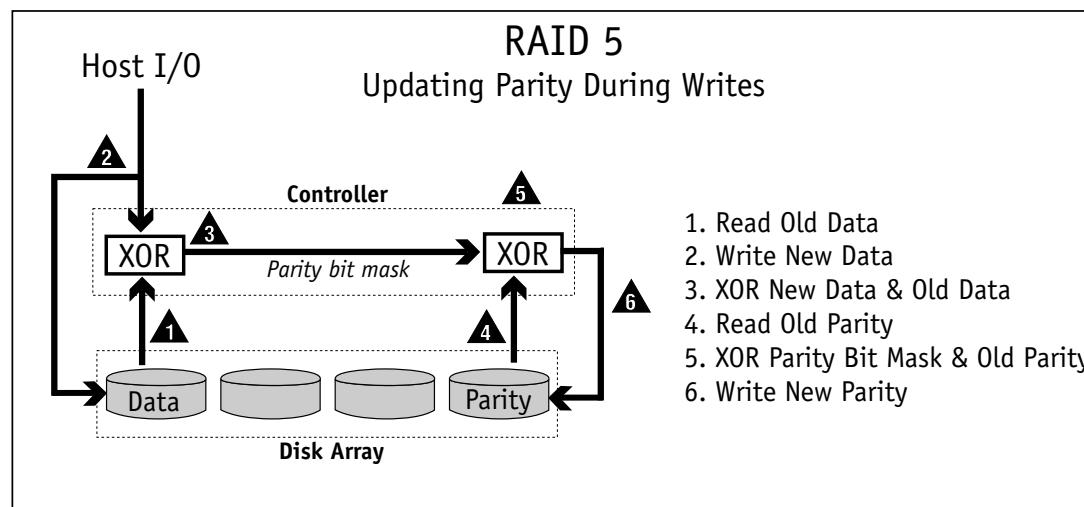
4. Read old parity from the array.

5. XOR the bit mask with the old parity information.

This results in the corresponding bits being changed in the parity information.

6. Write the updated parity back to the array.

Therefore, for every application write request, a RAID-5 array must perform two reads, two writes and two XOR operations to complete the original write operation.



The cost of calculating and storing parity, rather than redundant data, is the extra time taken during write operations to regenerate the parity information. This additional time results in a degradation of write performance for RAID-5 arrays over RAID-1 arrays. A RAID-5 write operation is 33% to 60% slower than a RAID-1 write operation. Because of this, RAID-5 arrays are not recommended for applications in which write performance is critically important or is the dominant type of I/O operation.

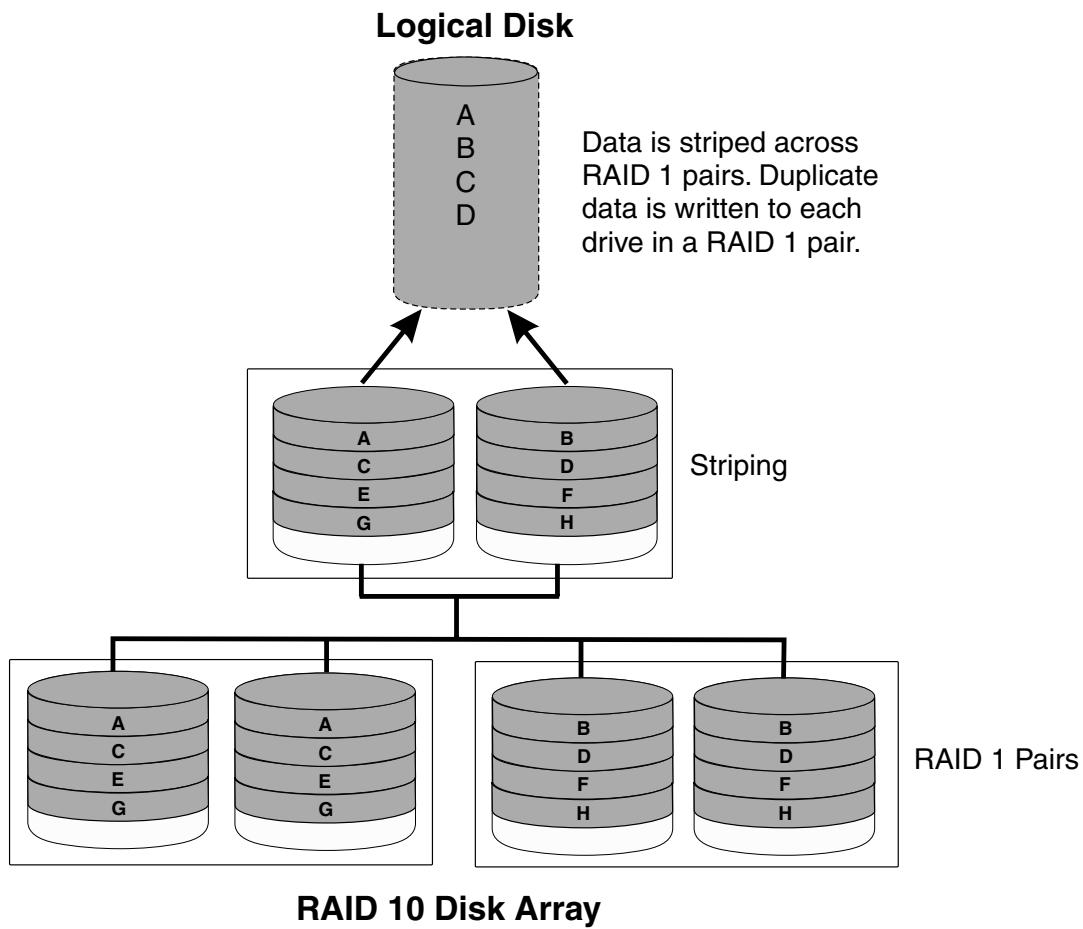
Overall performance is also affected by the number of drives in the array, where a larger number of drives can result in lower performance. This is especially noticeable when the array is degraded (a drive has failed), because all the remaining drives must be read and the XOR parity used to reconstruct the missing data. Therefore both read and write operations will be significantly affected when an array is operating in degraded mode.

RAID-6

Striped data with dual distributed parity

RAID-6 is the same as RAID-5 except that it uses a second level of independently calculated and distributed parity information for additional fault tolerance. This extra fault tolerance provides data security in the event two drives fail before a drive can be replaced.

While this RAID level does provide greater fault tolerance than level 5, there is a significant loss in write performance due to the requirement for storing parity twice for each write operation. A RAID-6 configuration also requires N+2 drives to accommodate the additional parity data, which makes it less cost effective than RAID-5 for an equivalent storage capacity.

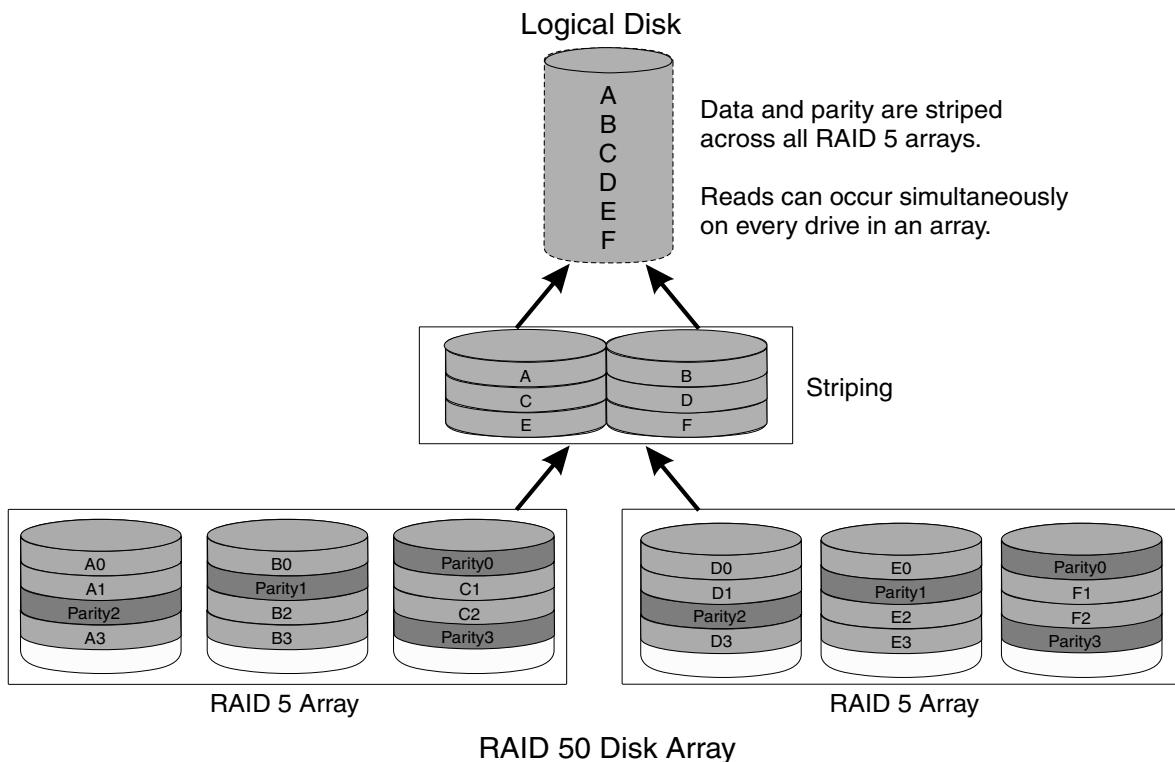
RAID 10*Strip set of mirrored arrays*

RAID 10 (also called RAID 0/1) is a combination of RAID levels 0 and 1. In this type of implementation a RAID-0 stripe set of the data is created across a 2-disk array for performance benefits. A duplicate of the first stripe set is then mirrored on another 2-disk array for fault tolerance. While this configuration provides all of the performance benefits of RAID-0 and the redundancy of RAID-1, this level is very costly to implement because a minimum of four disks are necessary to create a RAID 10 configuration.

NOTE *A RAID 10 configuration can continue operations even when two disks have failed, provided that the two disks not part of the same RAID-1 mirror set.*

RAID 50

Stripe set of parity arrays



RAID level 50 (also called RAID 0/5) is a combination of RAID levels 0 and 5. Multiple RAID-5 arrays are striped together using RAID-0. Parity is maintained separately for each RAID-5 group in the striped array. This level provides the same advantages of RAID-5 for small data transfers with the added performance of striping for disk read/write operations. Also, because parity is calculated independently for each RAID-5 component, if one array is degraded the effect on overall operations is not as significant as for a single RAID-5 array.

However, the overhead incurred by RAID-5 parity generation is still present. Normally this does not cause noticeable degradation unless you are dependent on software-based XOR functionality or have a large number of disks in the array. RAID subsystems that support hardware-based XOR should provide performance nearly equal to a RAID-0 configuration with the added protection of data parity information in the event of a disk failure.

A minimum of six disks are required for a RAID 50 configuration.

NOTE *A RAID 50 configuration can continue operations even when two disks have failed, provided that the two disks are not part of the same RAID-5 parity group.*

Creating a RAID Subsystem

Disk storage subsystems can be created through several different methods. These include a software-based implementation, a bus-based model, and an external controller system. Each of these methods has its own unique advantages and disadvantages. The goals of the individual installation, including network size, cost, and performance needs, will determine which disk subsystem should be used.

A major decision facing most system administrators is whether to choose controller-based (hardware) RAID or allow the operating system to perform the RAID functions. Both options give you the necessary protection that RAID offers; however, there are distinct differences in the usability and performance between hardware and software RAID.

Software-based RAID is implemented by the server's operating system or through a special software driver. All array functions are implemented in this software running on the server CPU. Just like any other application, software-based arrays occupy system memory, consume host CPU cycles and system bus bandwidth. Software-based arrays generally degrade overall server performance because of the additional demands on system resources.

Hardware-based RAID is implemented directly on a specialized RAID controller. All array functions and processes are managed by embedded processors and specialized ASICs and firmware on the controller. The performance benefits of hardware RAID over software RAID is clearly demonstrated by any industry-standard benchmark, especially in RAID-5 environments where parity must be calculated for each I/O operation.

Chapter 6, *"Implementing RAID"* discusses the advantages and disadvantages of these methods in more detail.