How safe is your data?

Dr. Nicholas Jones
Storage Systems Development
IBM UK Laboratories

Introduction
Despite the advances in modern disk drive technology, drives do fail. Since so much of your data is stored on hard disks, when I talk about data loss I really mean data loss as the result of hard disk failure.

Types of Failure
Hard disks have become amazingly sophisticated. Yet they have also become a commodity and, especially at the consumer end of the market, they are remarkably cheap.

Consider for a moment the pessimist's view of a hard disk: a thin glass plate; smeared with a layer of glorified rust; rotating 15,000 times per minute; with the heads flying above the surface of the disk at a height of less than 1/3 the wavelength of light. This requires phenomenally accurate manufacturing, but is mass-produced at the lowest possible price, so cannot prove 100% reliable. Broadly speaking there are two ways in which it can fail.

The first type of failure is where you, the user, know that a problem has occurred. In many cases this is total loss of access to the drive, as access to the data requires every component on the drive to work. If the motor fails, the data is still on the platters, but access to it is lost; the same is true if the power circuit fails; or the drive microcontroller. Other examples of reported errors are data corruption and medium errors, which occur when valid data just cannot be read from the surface of the disk. You know that an error has occurred because the read command fails. Believe it or not, known errors are the preferred type of failure.

The other type of failure is when the disk drive lies and appears to be behaving normally when it is in fact failing. One such failure would be when the host computer issues a write; the drive completes the command with success, but the data is never actually written to the disk. This nightmare scenario is known as a dropped write. Another example of an unreported error would be when the data passed to the drive is written, but to the wrong place on the disk. In both of these cases subsequent reads to the disk would return valid but wrong data.

This second type of error may not sound to be too disastrous; after all losing the last set of changes made to a document may not be considered a catastrophe. I will discuss the impact of data loss later, but rather than thinking about a document, consider the consequences if the failing drive was in a server used by a bank, and the stale data returned was the balance of your bank account…

The Probability of Failure
The Mean Time Between Failure (MTBF) quoted by Seagate for their current enterprise class drive, the Cheetah, is 1.2 million hours[(i)]. For their consumer class drive, the Barracuda, this drops to 600,000 hours[(i)]. Looking at these two numbers it is quite reasonable to believe that, for a given drive, the risk of it failing is remarkably small. It is only when you consider the vast number of drives on which you depend that the risk becomes more apparent.

Consider the academic staff at the University of Southampton, of which there are approximately 1200; say that each uses a computer containing a consumer grade drive. You can now expect drive failure, followed by a cry of anguish, once every 500 hours (21 days). Next consider the data centre of a large bank or retail chain, containing 10,000 enterprise class drives. In this situation a drive failure is anticipated once a week.

It is difficult to get a measurement of the rate of dropped writes, as many of them may never be detected. The stale data could be overwritten by a later write without ever having being read; or a subsequent read could return the data, and the application not detect that it is stale.

Field data suggests that the most likely error is data loss, or known errors, which is the area I will concentrate on.

The True Cost of Failure
When it does occur, the consequences of data loss can be devastating. The rate of increase in hard disk capacity has shown no sign of slowing, so should total drive failure occur, you have the opportunity to lose more data than ever before!

In summer 2002 the *Disaster Recovery Journal*[(ii)] reported that 43% of firms which suffer a massive data loss never re-open, and 51% will reopen only to shut down permanently within two years.

Addressing the Problem
For many people, the obvious solution is to make regular backups of their important data. This could range from something as simple a copying your documents to another PC over the network; to periodically burning an image of your hard drive onto CD; through to automated nightly backups to tape, with off-site storage. There are, however, several drawbacks to relying solely on backups to protect your data.

The first is that you only recover data from the time of your last backup; hence the amount of data lost depends on the amount of work done since. If you are a home user or small business, and take backups nightly, this may not be a problem. If however you are a large financial institution, processing hundreds of thousands of transactions a day, losing a day's worth of changes could prove catastrophic.

The second is that this is really disaster recovery. Although being able to recover data following a failure is excellent, data loss has still occurred, and it takes time and effort to recover. A preferable solution would be to avoid data loss in the first place.

Redundant Information on Disk
In order to recover from an error, you must first be able to detect that the error has occurred. The basis of all error detection and correction in hard disks is the inclusion of

redundant information, along with hardware and software to make use of it.

Viewed from the outside, a hard disk is made up of sectors, each containing 512 bytes. On the disk surface itself however, each sector contains around 20 bytes of additional information which is used for an error correcting code (ECC). Several different codes have been developed over the years, but the most prevalent is the Reed-Solomon algorithm[iii]. This is used in applications ranging from data transmission lines to computer memory, and of course hard disks.

When a sector of client data is written, the ECC is calculated and stored in the extra bytes. When the data is read back, the ECC is used to verify that the read worked. If an error is detected, the drive will attempt to fix it. If this fails then, at the very least, the error has been detected and reported. The capability of the Reed-Solomon code depends upon the number of bytes used for ECC data. There is a balance to be struck between error detection and correction, and the storage and processing this requires.

Redundant Disks

Given that many errors are detectable, you need a system such that the failure does not result in the loss of data. The concept of using extra disks to solve the problem was an area of ongoing development, but the first systematic overview of the entire subject was published in 1988 by three researchers from the University of California Berkeley: Patterson, Gibson and Katz. Their paper was titled *A Case for Redundant Arrays of Inexpensive Disks (RAID)*[iv].

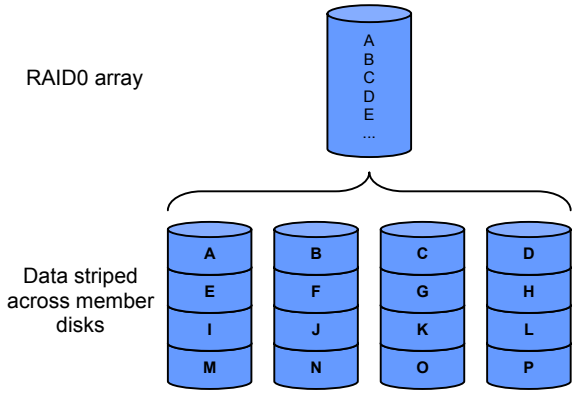
A RAID array is a collection of hard disks which are grouped together and appear to the host computer as a single logical device.

At the time of the paper reliable hard disks were hugely expensive, but cheap ones were unreliable. Part of the motivation was to replace a single large expensive disk with an array of inexpensive smaller ones, and to cope with the reliability problems this presented. As disk prices have fallen, the acronym RAID is often referred to today as a Redundant Array of Independent Disks.

The paper presented five RAID levels, titled RAID1 through to RAID5. Not all of these are in common use today, and new ones have been added. I will concentrate only on those RAID levels of interest today.

RAID0 – Striping

This level was not included in the original RAID paper. It offers no redundancy, and actually increases the risk of data loss, so it is not true RAID - hence the name RAID0. To the cautiously minded it could be considered pointless, but it does have certain merits and occurs again in later RAID levels, so is discussed here.



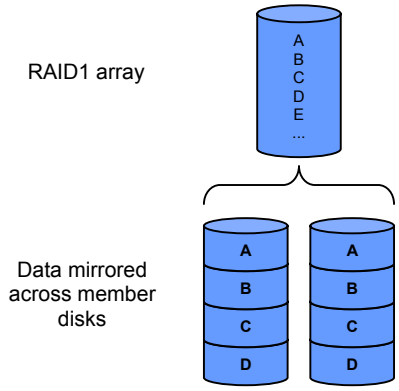
In a RAID0 array the data is broken down into small pieces, called strips, which are then distributed across the member disks.

The advantage of RAID0 is that it offers increased bandwidth and throughput over a single disk. By reading and writing data to multiple disks, each may be accessed simultaneously. It is therefore ideal for high bandwidth applications, such as video editing, where reliability is of less concern than performance.

RAID1 – Mirroring

This is perhaps the most straight forward RAID level, and is analogous to keeping a constantly up to date backup of the data.

In a RAID1 array, the member disks are configured in mirrored pairs, as shown below. In a mirrored array a complete copy of all the data is written to both drives simultaneously.



RAID1 provides full fault tolerance for a single drive failure and detected read errors, as a complete copy of the data is available on the mirror disk.

RAID1 offers a significant improvement in read performance over a single disk, as the reads can be submitted in parallel to the two disks in the mirror pair. It does however suffer from a slight decrease in write performance, as the command will only complete back to the host computer when the writes to both disks in the mirror pair have completed.

The biggest disadvantage of RAID1 is the poor efficiency, as only 50% of the storage in the system is available to the user.

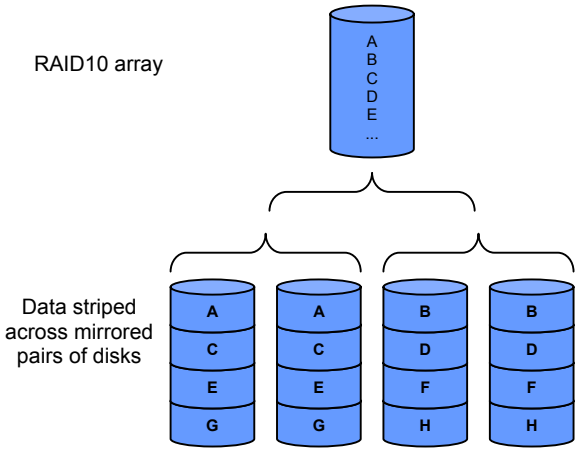
RAID1 is appropriate when data reliability and fault tolerance are more important than cost.

RAID10 – Mirroring and Striping

RAID10 is a combination of levels 0 and 1. The data is broken down into small strips, as in RAID0, and is then spread across pairs of mirrored disks, as in RAID1.

This configuration offers the performance benefits of RAID1, as the work is distributed amongst the member disks and operations can be carried out in parallel, along with the redundancy provided by RAID1, as the array contains two complete copies of all data.

The downside is the same as that for RAID1 - RAID10 is costly to implement because the storage efficiency is only 50%.



It is suitable for similar applications to RAID1, where data reliability and fault tolerance are more important than cost. An additional benefit of RAID10 over RAID1 is increased performance.

XOR Based Parity

In order to understand the next levels of RAID, an introduction to exclusive-or (XOR) based parity is needed. This is nothing particularly complex, and those who have studied physics or electronics may have a vague recollection of XOR logic gates and truth tables.

The XOR is a bitwise operator, whose truth table is show below. The behaviour is that if the two inputs are different, the output is 1, otherwise the output is 0.

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

Table 1 - Truth table for the XOR operator

Consider the example shown below. The parity is the bitwise XOR of the two data bytes.

Data 1	0	1	1	0	0	1	0	1
Data 2	0	0	1	1	0	0	1	1
Parity	0	1	0	1	0	1	1	0

Table 2 - An example of parity

The key property of XOR based parity is that, should data loss occur, the calculation can be repeated using the parity and remaining data to reconstruct the missing data. As an example, imagine that one row of data from the previous table had been lost, leaving you with the information given below.

Data 1	0	1	1	0	0	1	0	1
Data 2								
Parity	0	1	0	1	0	1	1	0

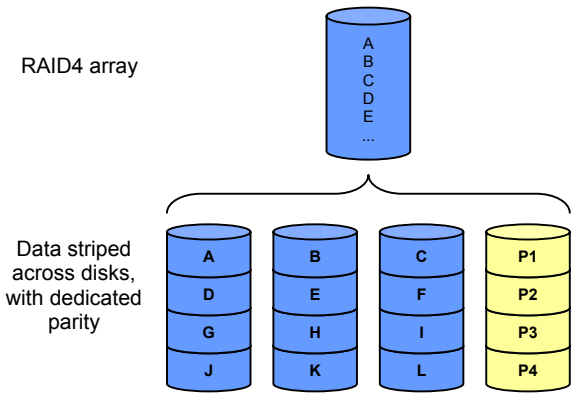
Table 3 – Data loss has occurred

If you were to work through the remaining data and parity, XORing them together, you would be able to reconstruct Data2.

The example given here has just two inputs, but in reality any number of bits can be XORed together. It is also worth noting that the operator is associative, meaning that the order of the inputs does not matter. A XOR B XOR C is the same as A XOR C XOR B.

RAID4 – Parity

RAID4 provides fault tolerance using a dedicated parity disk, rather than through the mirroring technique used for RAID1. The layout of the member disks is shown in the following diagram.



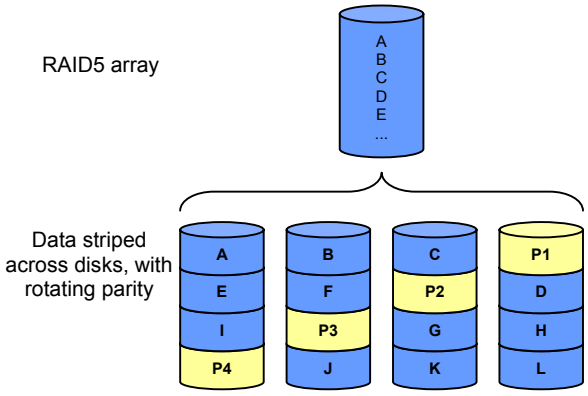
The data is broken down into strips and distributed across all but one of the member disks. The remaining one being the parity, which is the XOR of the data strips. In the layout shown above, P1 is the XOR of A, B and C; P2 is the XOR of D, E and F; and so on.

The storage efficiency of RAID4 is far greater than that of RAID1 or 10. It is in fact optimal for redundancy against a single drive failure, as all but one of the array members are available for customer data. The efficiency therefore increases with the number of disks in the array.

RAID4 offers good read performance, as the data is distributed across the member disks, exactly as in RAID0. The write performance however is abysmal! Whenever new data is written to any member disk, the corresponding parity strip must be updated. This means that the parity disk becomes a bottleneck, and it is for this reason that RAID4 is seldom (if ever) used.

RAID5 – Rotated Parity

The main drawback to RAID4 is that the parity disk is a performance bottleneck. RAID5 solves this problem by rotating the parity strip along with the data strips, thus spreading the burden of the parity write amongst all the array members. The write performance is still not as good as RAID1 or 10, as each time data is written the corresponding parity strip must also be updated. The storage efficiency and read performance of RAID5 are the same as that of RAID4.



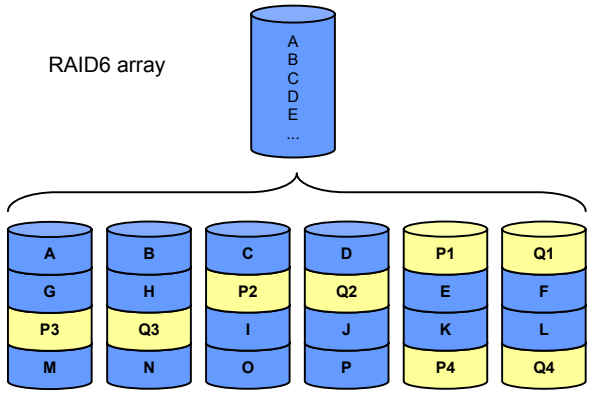
The optimal storage efficiency, reasonable performance, and fault tolerance means that RAID5 is the most commonly used RAID level.

RAID5 provides the redundancy required to cope with the failure of any single disk within an array, whether this be a medium error or total drive failure. When an error is encountered, the reconstruct algorithm will only succeed if all the remaining array members can be read. If one of these reads fails, data is lost.

RAID6 – Dual Parity

RAID6 offers protection against any two drive failures, but at the expense of performance and storage efficiency. It was not included in the original RAID paper, and even today there is no standard implementation. A number of algorithms have been published and patented, and a RAID6 implementation was even introduced into the 2.6 Linux kernel[v].

RAID6 is similar to RAID5 in that the redundancy is provided via parity, although to protect against two drive failures two parities are needed. The data is split into strips which are distributed across the member disks, and the parities rotate to avoid the performance bottleneck suffered by RAID4. The two parities, labelled P and Q in the diagram below, are calculated using different algorithms, as two copies of the same parity would be of very little use!



Putting it into Perspective

It is all very well spending a lot of time and money to implement and maintain data storage using RAID arrays, but you cannot rely on RAID alone.

For continued access to your data, your system must not be susceptible to any single point of failure. Thus far I have only considered disk failure, and have explained how RAID can be used to offer protection. There are, however, many other failures which could result in data loss.

Should the building in which your server and disks are located catch fire, your data will be lost. The fact that it was stored on a RAID array will not offer much consolation. Likewise (and this has happened), a fire alarm could trigger the sprinkler system, flooding your server and disks along with the rest of the machine room. Regular backups are still good practice, as they offer a second level of protection against such equipment failure.

As a final thought, do not neglect the impact of users. According to a report by the Wall Street Journal[vi], human error was the culprit in 64% of all critical data lost.

Further Information & References

A good starting point for finding out more about RAID is Google.

A selection of other web sites with plenty of information to get started from is:

- IBM TotalStorage: www.storage.ibm.com
- Adaptec: www.adaptec.com
- Storage Network Industry Association: www.snia.org
- IBM Technical Journals: www.research.ibm.com/journals

---

[i] http://www.seagate.com/products/datasheet
[ii] http://www.drj.com
[iii] http://www.eccpage.com is a good starting place. Then *Error-Correcting Codes*, 2nd edition, by W.W. Peterson and E.J. Weldon.
[iv] *A Case for Redundant Arrays of Inexpensive Disks*, by D.A Patterson, G. Gibson and R.H. Katz. Published in the proceedings of the 1988 ACM SIGMOD conference on Management of data. Start at www.acm.org
[v] http://kernel.org/pub/linux/kernel/people/hpa/raid6.pdf
[vi] http://www.wsj.com (subscription only)