



WIKIPEDIA
The Free Encyclopedia

Navigation

[Main page](#)
[Contents](#)
[Featured content](#)
[Current events](#)
[Random article](#)

Interaction

[About Wikipedia](#)
[Community portal](#)
[Recent changes](#)
[Contact Wikipedia](#)
[Donate to Wikipedia](#)
[Help](#)

Toolbox

[What links here](#)
[Related changes](#)
[Upload file](#)
[Special pages](#)
[Permanent link](#)
[Cite this page](#)

Print/export

[Create a book](#)
[Download as PDF](#)
[Printable version](#)

Article [Discussion](#)

[Read](#) [Edit](#) [View history](#)



Standard RAID levels

From Wikipedia, the free encyclopedia



This article **needs additional citations for verification**.

Please help [improve this article](#) by adding [reliable references](#). Unsourced material may be [challenged](#) and [removed](#). *(February 2008)*



This article **may be too technical for most readers to understand**. Please [improve](#) this article to [make it accessible to non-experts](#), without removing the technical details.

Main article: [RAID](#)

The **standard RAID levels** are a basic set of [RAID](#) configurations and employ [striping](#), [mirroring](#), or [parity](#). The standard RAID levels can be nested for other benefits (see *[Nested RAID levels for modes like 1+0 or 0+1](#)*); there are also [non-standard RAID levels](#), and [non-RAID drive architectures](#), which may be offered as alternatives to RAID architectures.

Contents

- 1 RAID 0
 - 1.1 RAID 0 failure rate
 - 1.2 RAID 0 performance
- 2 RAID 1
 - 2.1 RAID 1 failure rate
 - 2.2 RAID 1 performance
- 3 RAID 2
- 4 RAID 3
- 5 RAID 4
- 6 RAID 5
 - 6.1 RAID 5 parity handling
 - 6.2 RAID 5 disk failure rate
 - 6.3 RAID 5 performance
 - 6.4 RAID 5 usable size
- 7 RAID 6
 - 7.1 Redundancy and data loss recovery capability
 - 7.2 Performance (speed)
 - 7.3 Efficiency (potential waste of storage)
 - 7.4 Implementation
 - 7.4.1 Computing Parity
- 8 Non-standard RAID levels and non-RAID drive architectures
- 9 See also
- 10 References
- 11 External links

RAID 0

[\[edit\]](#)

A **RAID 0** (also known as a **stripe set** or **striped volume**) splits data evenly across two or more disks ([striped](#)) with no parity information for redundancy. It is important to note that RAID 0 was not one of the original RAID levels and provides no [data redundancy](#). RAID 0 is normally used to increase performance, although it can also be used as a way to create a small number of large virtual disks out of a large number of small physical ones.

A RAID 0 can be created with disks of differing sizes, but the storage space added to the array by each disk is limited to the size of the smallest disk. For example, if a 120 GB disk is striped together with a 100 GB disk, the size of the array will be 200 GB.

$$\begin{aligned}\text{Size} &= 2 \cdot \min(120\text{ GB}, 100\text{ GB}) \\ &= 2 \cdot 100\text{ GB} \\ &= 200\text{ GB}\end{aligned}$$

RAID 0 failure rate

[\[edit\]](#)



This section **does not cite any references or sources**.

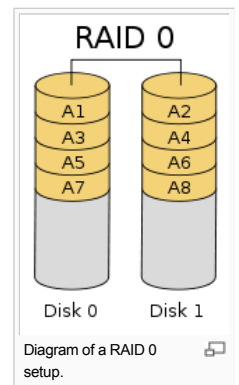
Please help [improve this article](#) by adding citations to [reliable sources](#). Unsourced material may be [challenged](#) and [removed](#). *(March 2010)*

Although RAID 0 was not specified in the original RAID paper, an idealized implementation of RAID 0 would split I/O operations into equal-sized blocks and spread them evenly across two disks. RAID 0 implementations with more than two disks are also possible, though the group reliability decreases with member size.

Reliability of a given RAID 0 set is equal to the average reliability of each disk divided by the number of disks in the set:

$$\text{MTTF}_{\text{group}} \approx \frac{\text{MTTF}_{\text{disk}}}{\text{number}}$$

That is, reliability (as measured by [mean time to failure](#) (MTTF) or [mean time between failures](#) (MTBF) is roughly inversely proportional to the



number of members – so a set of two disks is roughly half as reliable as a single disk. If there were a probability of 5% that the disk would fail within three years, in a two disk array, that probability would be upped to

$$\mathbb{P}(\text{at least one fails}) = 1 - \mathbb{P}(\text{neither fails}) = 1 - (1 - 0.05)^2 = 0.0975 = 9.75\%$$

The reason for this is that the [file system](#) is distributed across all disks. When a drive fails the file system cannot cope with such a large loss of data and coherency since the data is "striped" across all drives (the data cannot be recovered without the missing disk). Data can be recovered using special tools, however, this data will be incomplete and most likely corrupt, and data recovery is typically very costly and not guaranteed.

RAID 0 performance

[\[edit\]](#)

While the block size can technically be as small as a byte, it is almost always a multiple of the hard disk sector size of 512 bytes. This lets each drive seek independently when randomly reading or writing data on the disk. How much the drives act independently depends on the access pattern from the file system level. For reads and writes that are larger than the stripe size, such as copying files or video playback, the disks will be seeking to the same position on each disk, so the seek time of the array will be the same as that of a single drive. For reads and writes that are smaller than the stripe size, such as database access, the drives will be able to seek independently. If the sectors accessed are spread evenly between the two drives, the apparent seek time of the array will be half that of a single drive (assuming the disks in the array have identical access time characteristics). The transfer speed of the array will be the transfer speed of all the disks added together, limited only by the speed of the RAID controller. Note that these performance scenarios are in the best case with optimal access patterns.

RAID 0 is useful for setups such as large [read-only NFS server](#) where [mounting](#) many disks is time-consuming or impossible and redundancy is irrelevant.

RAID 0 is also used in some gaming systems where performance is desired and data integrity is not very important. However, real-world tests with games have shown that RAID-0 performance gains are minimal, although some desktop applications will benefit.^{[1][2]} Another article examined these claims and concludes: "Striping does not always increase performance (in certain situations it will actually be slower than a non-RAID setup), but in most situations it will yield a significant improvement in performance."^[3]

RAID 1

[\[edit\]](#)

A **RAID 1** creates an exact copy (or **mirror**) of a set of data on two or more disks. This is useful when read performance or reliability are more important than data storage [capacity](#). Such an array can only be as big as the smallest member disk. A classic RAID 1 mirrored pair contains two disks (see diagram), which increases reliability [geometrically](#) over a single disk. Since each member contains a complete copy of the data, and can be addressed independently, ordinary wear-and-tear reliability is raised by the power of the number of self-contained copies.

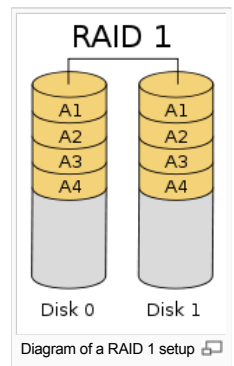
RAID 1 failure rate

[\[edit\]](#)



This section **does not cite any references or sources**.

Please help [improve this article](#) by adding citations to [reliable sources](#). Unsourced material may be [challenged](#) and [removed](#). (March 2010)



As a trivial example, consider a RAID 1 with two identical models of a disk drive with a 5% probability that the disk would fail within three years. Provided that the failures are [statistically independent](#), then the probability of both disks failing during the three year lifetime is

$$P(\text{dual failure}) = (0.05)^2 = 0.0025 = 0.25\%$$

Thus, the probability of losing all data is 0.25% over a three year period if nothing is done to the array. If the first disk fails and is never replaced, then there is a 5% chance the data will be lost. If only one of the disks fails, no data would be lost. As long as a failed disk is replaced before the second disk fails, the data is safe.

However, since two identical disks are used and since their usage patterns are also identical, their failures cannot be assumed to be independent. Thus, the probability of losing all data, if the first failed disk is not replaced, may be considerably higher than 5%.

As a practical matter, in a well managed system the above is irrelevant because the failed hard drive will not be ignored. It will be replaced. The reliability of the overall system is determined by the probability the remaining drive will continue to operate through the repair period, that is the total time it takes to detect a failure, replace the failed hard drive, and for that drive to be rebuilt. If, for example, it takes one hour to replace the failed drive, the overall system reliability is defined by the probability the remaining drive will operate for one hour without failure.

While RAID 1 can be an effective protection against physical disk failure, it does not provide protection against data corruption due to viruses, accidental file changes or deletions, or any other data-specific changes. By design, any such changes will be instantly mirrored to every drive in the array segment. A virus, for example, that damages data on one drive in a RAID 1 array will damage the same data on all other drives in the array at the same time. For this reason systems using RAID 1 to protect against physical drive failure should also have a traditional data backup process in place to allow data restoration to previous points in time. It would seem self-evident that any system critical enough to require disk redundancy also needs the protection of reliable data backups.

RAID 1 performance

[\[edit\]](#)

Since all the data exists in two or more copies, each with its own hardware, the read performance can go up roughly as a linear multiple of the number of copies. That is, a RAID 1 array of two drives can be reading in two different places at the same time, though not all implementations of RAID 1 do this.^[4] To maximize performance benefits of RAID 1, independent disk controllers are recommended, one for each disk. Some refer to this practice as **splitting** or **duplexing**. When reading, both disks can be accessed independently and requested sectors can be split evenly between the disks. For the usual mirror of two disks, this would, in theory, double the transfer rate when reading. The apparent access time of the array would be half that of a single drive. Unlike RAID 0, this would be for all access patterns, as all the data are present on all the disks. In reality, the need to move the drive heads to the next block (to skip blocks already read by the other drives) can effectively mitigate speed advantages for sequential access. Read performance can be further improved by adding drives to the mirror. Many older IDE RAID 1 controllers read only from one disk in the pair, so their read performance is always that of a single disk. Some older RAID 1 implementations would also read both disks simultaneously and compare the data to detect errors. The [error detection and correction](#) on modern disks makes this less useful in environments requiring normal availability. When writing, the array performs like a single disk, as all mirrors must be written with the data. Note that these performance scenarios are in the best case with optimal access patterns.

RAID 1 has many administrative advantages. For instance, in some environments, it is possible to "split the mirror": declare one disk as inactive, do

RAID 1 has many administrative advantages. For instance, in some environments, it is possible to split the mirror. Restore one disk as inactive, do a backup of that disk, and then "rebuild" the mirror. This is useful in situations where the file system must be constantly available. This requires that the application supports recovery from the image of data on the disk at the point of the mirror split. This procedure is less critical in the presence of the "snapshot" feature of some file systems, in which some space is reserved for changes, presenting a static point-in-time view of the file system. Alternatively, a new disk can be substituted so that the inactive disk can be kept in much the same way as traditional backup. To keep redundancy during the backup process, some controllers support adding a third disk to an active pair. After a rebuild to the third disk completes, it is made inactive and backed up as described above.

RAID 2

[[edit](#)]

A **RAID 2** stripes data at the **bit** (rather than block) level, and uses a **Hamming code** for **error correction**. The disks are synchronized by the controller to spin in perfect tandem. Extremely high data transfer rates are possible. This is the only original level of RAID that is not currently used.

The use of the Hamming(7,4) code (four data bits plus three parity bits) also permits using 7 disks in RAID 2, with 4 being used for data storage and 3 being used for error correction.

RAID 2 is the only standard RAID level, other than some implementations of RAID 6, which can automatically recover accurate data from single-bit corruption in data. Other RAID levels can detect single-bit corruption in data, or can sometimes reconstruct missing data, but cannot reliably resolve contradictions between parity bits and data bits without human intervention.

(Multiple-bit corruption is possible though extremely rare. RAID 2 can detect but not repair double-bit corruption.)

All hard disks soon after implemented an error correction code that also used Hamming code, so RAID 2's error correction was now redundant and added unnecessary complexity. Like RAID 3, this level quickly became useless and it is now obsolete. There are no commercial applications of RAID 2.^{[5][6]}

RAID 3

[[edit](#)]

A **RAID 3** uses **byte-level** striping with a dedicated parity disk. RAID 3 is very rare in practice. One of the side effects of RAID 3 is that it generally cannot service multiple requests simultaneously. This comes about because any single block of data will, by definition, be spread across all members of the set and will reside in the same location. So, any I/O operation requires activity on every disk and usually requires synchronized spindles.

In our example, a request for block "A" consisting of bytes A1-A6 would require all three data disks to seek to the beginning (A1) and reply with their contents. A simultaneous request for block B would have to wait.

However, the performance characteristic of RAID 3 is very consistent, unlike higher RAID levels.^[clarification needed] the size of a stripe is less than the size of a sector or OS block so that, for both reading and writing, the entire stripe is accessed every time. The performance of the array is therefore identical to the performance of one disk in the array except for the transfer rate, which is multiplied by the number of data drives (i.e., less parity drives).

This makes it best for applications that demand the highest transfer rates in long sequential reads and writes, for example uncompressed video editing. Applications that make small reads and writes from random places over the disk will get the worst performance out of this level.^[6]

The requirement that all disks spin synchronously, aka in **lockstep**, added design considerations to a level that didn't give significant advantages over other RAID levels, so it quickly became useless and is now obsolete.^[5] Both RAID 3 and RAID 4 were quickly replaced by RAID 5.^[7] However, this level has commercial vendors making implementations of it. It's usually implemented in hardware, and the performance issues are addressed by using large disks.^[6]

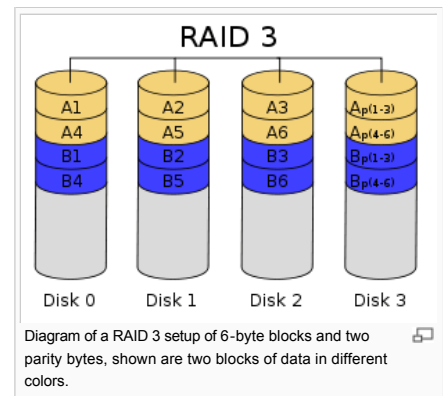


Diagram of a RAID 3 setup of 6-byte blocks and two parity bytes, shown are two blocks of data in different colors.

RAID 4

[[edit](#)]

A **RAID 4** uses **block-level** striping with a dedicated parity disk. This allows each member of the set to act independently when only a single block is requested. If the disk controller allows it, a RAID 4 set can service multiple read requests simultaneously. RAID 4 looks similar to RAID 5 except that it does not use distributed parity, and similar to RAID 3 except that it stripes at the block level, rather than the byte level. Generally, RAID 4 is implemented with hardware support for parity calculations, and a minimum of 3 disks is required for a complete RAID 4 configuration.

In the example on the right, a read request for block A1 would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

For writing the parity disk becomes a bottleneck, as simultaneous writes to A1 and B2 would in addition to the writes to their respective drives also both need to write to the parity drive. In this way RAID example 4 places a very high load on the parity drive in an array.

The performance of RAID 4 in this configuration can be very poor, but unlike RAID 3 it does not need synchronized spindles. However, if RAID 4 is implemented on synchronized drives and the size of a stripe is reduced below the OS block size a RAID 4 array then has the same performance pattern as a RAID 3 array.

Currently, RAID 4 is only implemented at the enterprise level by one single company, **NetApp**, who solved the performance problems discussed above with their proprietary **WAFL** filesystem.^[citation needed]

Both RAID 3 and RAID 4 were quickly replaced by RAID 5.^[7]

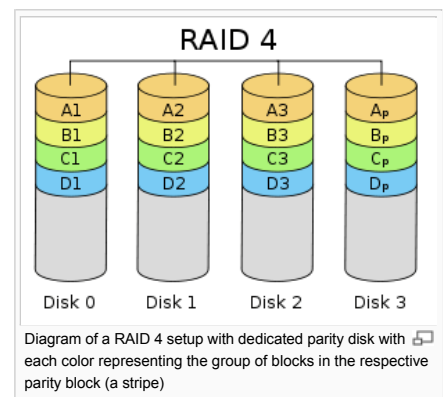
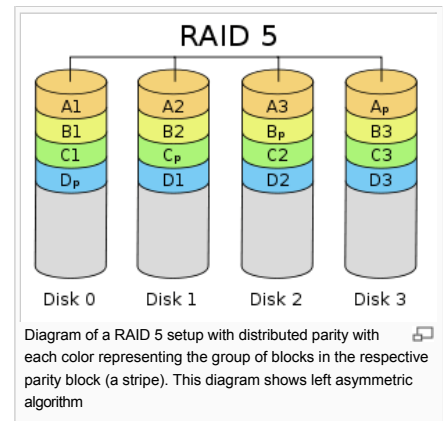


Diagram of a RAID 4 setup with dedicated parity disk with each color representing the group of blocks in the respective parity block (a stripe)

RAID 5

[[edit](#)]

A **RAID 5** uses [block-level](#) striping with parity data distributed across all member disks. RAID 5 has achieved popularity because of its low cost of redundancy. This can be seen by comparing the number of drives needed to achieve a given capacity. RAID 1 or RAID 1+0, which yield redundancy, give only $s / 2$ storage capacity, where s is the sum of the capacities of n drives used. In RAID 5, the yield is $S_{\min} \times (n - 1)$ where S_{\min} is the size of the smallest disk in the array. As an example, four 1-TB drives can be made into a 2-TB redundant array under RAID 1 or RAID 1+0, but the same four drives can be used to build a 3-TB array under RAID 5. Although RAID 5 is commonly implemented in a disk controller, some with hardware support for parity calculations (hardware RAID cards) and some using the main system processor (motherboard based RAID controllers), it can also be done at the operating system level, e.g., using Windows *Dynamic Disks* or with [mdadm](#) in Linux. A minimum of three disks is required for a complete RAID 5 configuration. In some implementations a degraded RAID 5 disk set can be made (three disk set of which only two are online), while mdadm supports a fully-functional (non-degraded) RAID 5 setup with two disks - which function as a slow RAID-1, but can be expanded with further volumes.



In the example, a read request for block A1 would be serviced by disk 0. A simultaneous read request for block B1 would have to wait, but a read request for B2 could be serviced concurrently by disk 1.

RAID 5 parity handling

[\[edit\]](#)

A concurrent series of blocks (one on each of the disks in an array) is collectively called a **stripe**. If another block, or some portion thereof, is written on that same stripe, the parity block, or some portion thereof, is recalculated and rewritten. For small writes, this requires:

- Read the old data block
- Read the old parity block
- Compare the old data block with the write request. For each bit that has flipped (changed from 0 to 1, or from 1 to 0) in the data block, flip the corresponding bit in the parity block
- Write the new data block
- Write the new parity block

The disk used for the parity block is staggered from one stripe to the next, hence the term **distributed parity blocks**. RAID 5 writes are expensive in terms of disk operations and traffic between the disks and the controller.

The parity blocks are not read on data reads, since this would add unnecessary overhead and would diminish performance. The parity blocks are read, however, when a read of blocks in the stripe and within the parity block in the stripe are used to reconstruct the errant sector. The CRC error is thus hidden from the main computer. Likewise, should a disk fail in the array, the parity blocks from the surviving disks are combined mathematically with the data blocks from the surviving disks to reconstruct the data on the failed drive on-the-fly.

This is sometimes called Interim Data Recovery Mode. The computer knows that a disk drive has failed, but this is only so that the operating system can notify the administrator that a drive needs replacement; applications running on the computer are unaware of the failure. Reading and writing to the drive array continues seamlessly, though with some performance degradation.

RAID 5 disk failure rate

[\[edit\]](#)

The maximum number of drives in a RAID 5 redundancy group is theoretically unlimited. The tradeoffs of larger redundancy groups are greater probability of a simultaneous double disk failure, the increased time to rebuild a redundancy group, and the greater probability of encountering an unrecoverable sector during RAID reconstruction. As the number of disks in a RAID 5 group increases, the [mean time between failures](#) (MTBF, the reciprocal of the [failure rate](#)) can become lower than that of a single disk. This happens when the likelihood of a second disk's failing out of $N - 1$ dependent disks, within the time it takes to detect, replace and recreate a first failed disk, becomes larger than the likelihood of a single disk's failing.

[Solid-state drives](#) (SSDs) may present a revolutionary instead of evolutionary way of dealing with increasing RAID-5 rebuild limitations. With encouragement from many flash-SSD manufacturers, [JEDEC](#) is preparing to set standards in 2009 for measuring UBER (uncorrectable bit error rates) and "raw" bit error rates (error rates before ECC, error correction code).^[8] But even the economy-class Intel X25-M SSD claims an unrecoverable error rate of 1 sector in 10^{15} bits and an MTBF of two million hours.^[9] Ironically, the much-faster throughput of SSDs (STEC claims its enterprise-class Zeus SSDs exceed 200 times the transactional performance of today's 15k-RPM, enterprise-class HDDs)^[10] suggests that a similar error rate (1 in 10^{15}) will result a two-magnitude shortening of MTBF.

In the event of a system failure while there are active writes, the parity of a stripe may become inconsistent with the data. If this is not detected and repaired before a disk or block fails, data loss may ensue as incorrect parity will be used to reconstruct the missing block in that stripe. This potential vulnerability is sometimes known as the **write hole**. Battery-backed cache and similar techniques are commonly used to reduce the window of opportunity for this to occur. The same issue occurs for RAID-6.

RAID 5 performance

[\[edit\]](#)

RAID 5 implementations suffer from poor performance when faced with a workload which includes many writes which are smaller than the capacity of a single stripe. This is because parity must be updated on each write, requiring read-modify-write sequences for both the data block and the parity block. More complex implementations may include a non-volatile write back cache to reduce the performance impact of incremental parity updates.

Random write performance is poor, especially at high concurrency levels common in large multi-user databases. The read-modify-write cycle requirement of RAID 5's parity implementation penalizes random writes by as much as an [order of magnitude](#) compared to RAID 0.^[11]

Performance problems can be so severe that some database experts have formed a group called BAARF — the Battle Against Any Raid Five.^[12]

The read performance of RAID 5 is almost as good as RAID 0 for the same number of disks. Except for the parity blocks, the distribution of data over the drives follows the same pattern as RAID 0. The reason RAID 5 is slightly slower is that the disks must skip over the parity blocks.

RAID 5 usable size

[\[edit\]](#)

For example, a RAID 5 array of four 1-TB drives would have a usable capacity of 3 TB. RAID 5 is not recommended for arrays of less than four drives.

Parity data uses up the capacity of one drive in the array (this can be seen by comparing it with [RAID 4](#): RAID 5 distributes the parity data across the disks, while RAID 4 centralizes it on one disk, but the amount of parity data is the same). If the drives vary in capacity, the smallest of them sets the limit. Therefore, the usable capacity of a RAID 5 array is $(N - 1) \cdot S_{\min}$, where N is the total number of drives in the array and S_{\min} is the capacity of the smallest drive in the array.

The number of hard disks that can belong to a single array is limited only by the capacity of the storage controller in hardware implementations, or by the OS in software RAID. One caveat is that unlike RAID 1, as the number of disks in an array increases, the chance of data loss due to multiple drive failures is increased. This is because there is a reduced ratio of "losable" drives (the number of drives which may fail before data is lost) to total drives.

RAID 6

[\[edit\]](#)

Redundancy and data loss recovery capability

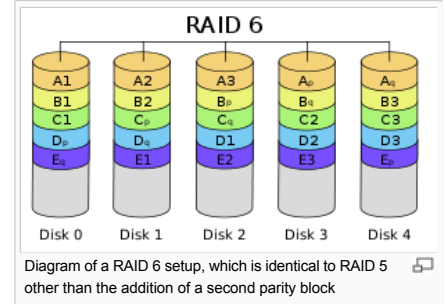
[\[edit\]](#)

RAID 6 extends RAID 5 by adding an additional parity block; thus it uses [block-level](#) striping with two parity blocks distributed across all member disks.

Performance (speed)

[\[edit\]](#)

RAID 6 does not have a performance penalty for read operations, but it does have a performance penalty on write operations because of the overhead associated with parity calculations. Performance varies greatly depending on how RAID 6 is implemented in the manufacturer's storage architecture – in software, firmware or by using firmware and specialized [ASICs](#) for intensive parity calculations. It can be as fast as a RAID-5 system with one less drive (same number of data drives).^[13]



Efficiency (potential waste of storage)

[\[edit\]](#)

RAID 6 is no more space inefficient than RAID 5 with a hot spare drive when used with a small number of drives, but as arrays become bigger and have more drives the loss in storage capacity becomes less important and the probability of data loss is greater. RAID 6 provides protection against data loss during an array rebuild, when a second drive is lost, a bad block read is encountered, or when a human operator accidentally removes and replaces the wrong disk drive when attempting to replace a failed drive.

The usable capacity of a RAID 6 array is $(N - 2) \cdot S_{\min}$, where N is the total number of drives in the array and S_{\min} is the capacity of the smallest drive in the array.

Implementation

[\[edit\]](#)

According to the Storage Networking Industry Association (SNIA), the definition of RAID 6 is: "Any form of RAID that can continue to execute read and write requests to all of a RAID array's virtual disks in the presence of any two concurrent disk failures. Several methods, including dual check data computations (parity and [Reed-Solomon](#)), orthogonal dual parity check data and diagonal parity, have been used to implement RAID Level 6."^[14]

Computing Parity

[\[edit\]](#)

Two different *syndromes* need to be computed in order to allow the loss of any two drives. One of them, **P** can be the simple XOR of the data across the stripes, as with RAID 5. A second, independent syndrome is more complicated and requires the assistance of [field theory](#).

To deal with this, the [Galois field](#) $GF(m)$ is introduced with $m = 2^k$. An important result from field theory says that $GF(m) \cong F_2[x]/(p(x))$ where $p(x)$ is an [irreducible polynomial](#) of degree k . That is to say, an element of the field is a binary value multiplied by some power of an abstract variable x . Typically a chunk of data may be written as $d_{k-1}d_{k-2}...d_0$ in base 2 where each d_i is either 0 or 1. This is chosen to correspond with the element $d_{k-1}x^{k-1} + d_{k-2}x^{k-2} + ... + d_1x + d_0$ in the Galois field. Let $D_0, ..., D_{n-1} \in GF(m)$ correspond to the stripes of data across hard drives encoded as field elements in this manner (in practice they would probably be broken into byte-sized chunks). If g is some [generator](#) of the field and \oplus denotes addition in the field while concatenation denotes multiplication, then **P** and **Q** may be computed as follows:

$$\mathbf{P} = \bigoplus_i D_i = D_0 \oplus D_1 \oplus D_2 \oplus ... \oplus D_{n-1}$$

$$\mathbf{Q} = \bigoplus_i g^i D_i = g^0 D_0 \oplus g^1 D_1 \oplus g^2 D_2 \oplus ... \oplus g^{n-1} D_{n-1}$$

For a computer scientist, a good way to think about this is that \oplus is a bitwise XOR operator and g^i is the action of a [linear feedback shift register](#) on a chunk of data. Thus, in the formula above,^[15] the calculation of **P** is just the XOR of each stripe. This is because addition in any characteristic two finite field reduces to the XOR operation. The computation of **Q** is the XOR of a shifted version of each stripe.

Mathematically, the *generator* is an element of the field such that g^i is different for each nonnegative i satisfying $i < n$.

If one data drive is lost, the data can be recomputed from **P** just like with RAID 5. If two data drives are lost or the drive containing **P** is lost the data can be recovered from **P** and **Q** using a more complex process. Working out the details is not hard with field theory. Suppose that D_i and D_j are the lost values with $i \neq j$. Using the other values of D , constants A and B may be found so that $D_i \oplus D_j = A$ and $g^i D_i \oplus g^j D_j = B$. Multiplying both sides of the latter equation by g^{n-i} and adding to the former equation yields $(g^{n-i+j} \oplus 1) D_j = g^{n-i+j} B$ and thus a solution for D_j which may be used to compute D_i .

The computation of **Q** is CPU intensive compared to the simplicity of **P**. Thus, a RAID 6 implemented in software will have a more significant effect on system performance, and a hardware solution will be more complex.

Non-standard RAID levels and non-RAID drive architectures

[\[edit\]](#)

Main articles: [Non-standard RAID levels](#) and [Non-RAID drive architectures](#)

There are other RAID levels that are promoted by individual vendors, but not generally standardized. The non-standard RAID levels 5E, 5EE and 6E

There are other RAID levels that are promoted by individual vendors, but not generally standardized. The non-standard RAID refers 0E, 0EE and 0L extend RAID 5 and 6 with [hot-spare](#) drives.

Other non-standard RAID levels include: RAID 1.5, RAID 7, RAID-DP, RAID S or parity RAID, [Matrix RAID](#), RAID-K, RAID-Z, RAIDn, Linux MD RAID 10, IBM ServeRAID 1E, unRAID, and [Drobo BeyondRAID](#).

There are also [non-RAID drive architectures](#), which are referred to by similar acronyms, notably SLED, JBOD, SPAN/BIG, and MAID.

See also

[\[edit\]](#)

- [RAID](#)
- [Nested RAID levels](#)
- [Non-RAID drive architectures](#)
- [Non-standard RAID levels](#)

References

[\[edit\]](#)

- ↑ "Western Digital's Raptors in RAID-0: Are two drives better than one?". AnandTech. July 1, 2004. Retrieved 2007-11-24.
- ↑ "Hitachi Deskstar 7K1000: Two Terabyte RAID Redux". AnandTech. April 23, 2007. Retrieved 2007-11-24.
- ↑ "RAID 0: Hype or blessing?". Tweakers.net. August 7, 2004. Retrieved 2008-07-23.
- ↑ "Mac OS X, Mac OS X Server: How to Use Apple-Supplied RAID Software". Apple.com. Retrieved 2007-11-24.
- ↑ *a* *b* Derek Vadala (2003). *Managing RAID on Linux. O'Reilly Series* (illustrated ed.). O'Reilly. p. 6. ISBN 1565927303, 9781565927308.
- ↑ *a* *b* *c* Evan Marcus, Hal Stern (2003). *Blueprints for high availability* (2, illustrated ed.). John Wiley and Sons. p. 167. ISBN 0471430269, 9780471430261.
- ↑ *a* *b* Michael Meyers, Scott Jernigan (2003). *Mike Meyers' A+ Guide to Managing and Troubleshooting PCs* (illustrated ed.). McGraw-Hill Professional. p. 321. ISBN 0072231467, 9780072231465.
- ↑ Shilov, Anton (November 20, 2008). "JEDEC to Set Solid State Drive Standards in 2009". X-bit labs. Retrieved 2009-03-24.
- ↑ "Intel X18-M/X25-M SATA Solid State Drive (SSDSA1MH080G2, SSDSA2MH080G2, SSDSA1MH160G2, SSDSA2MH160G2) Product Manual" (PDF). Intel Corporation. July 2009. pp. 1. Retrieved 2009-07-23.
- ↑ "Zeus^{IOPS} Solid State Drive". STEC, Inc.. 2005-04-02. Retrieved 2009-03-24.
- ↑ *Cary Millsap* (21 August 1996) (PDF). *Configuring Oracle Server for VLDB*.
- ↑ "BAARF - Battle Against Any Raid Five". Retrieved 2008-07-11.
- ↑ Rickard E. Faith (13 May 2009). *A Comparison of Software RAID Types*.
- ↑ "Dictionary R". Storage Networking Industry Association. Retrieved 2007-11-24.
- ↑ Anvin, H.Peter (21 May 2009). "The mathematics of RAID-6". Retrieved November 4, 2009.

External links

[\[edit\]](#)

- [RAID Calculator for Standard RAID Levels and Other RAID Tools](#)
- [Tutorial on "RAID 6 Essentials — reduced performance or not?"](#) - Requires Flash.
- [IBM summary on RAID levels](#)
- [RAID 5 Parity explanation and checking tool.](#)
- [Dell animations and details on RAID levels 0, 1, 5](#)

Categories: [RAID](#)

This page was last modified on 31 July 2010 at 08:39.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. See [Terms of Use](#) for details. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

[Contact us](#)

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#)

