

THE RAID TUTORIAL

- [An Introduction to RAID](#)
- [The Need for RAID](#)
- [Data Striping & Redundancy](#)
- [Different Types of RAID](#)
- [Tool for Storage Efficiency](#)
- [Cost & Performance Issues](#)
- [Reliability Issues in RAID](#)
- [Tool for Reliability](#)
- [Glossary](#)
- [References](#)

AN INTRODUCTION TO RAID

[HOME](#) / [NEXT](#)

RAID stands for **R**edundant **A**rray of **I**nexpensive **D**isks.

RAID is the organization of multiple disks into a large, high performance logical disk.

Disk arrays stripe data across multiple disks and access them in parallel to achieve:

- Higher data transfer rates on large data accesses and
- Higher I/O rates on small data accesses.

Data striping also results in uniform load balancing across all of the disks, eliminating hot spots that otherwise saturate a small number of disks, while the majority of disks sit idle.

BUT....

Large disk arrays, however are highly **vulnerable to disk failures**. A disk array with a hundred disks is a hundred times more likely to fail than a single disk. An MTTF (mean-time-to-failure) 500,000 hours for a single disk implies an [MTTF](#) of 500,000/100 i.e. 5000 hours for a disk array with a hundred disks.

So....

The solution to the problem of lower reliability in disk arrays is to improve the [availability](#) of the system. This can be achieved **by employing redundancy** in the form of error-correcting codes to tolerate disk failures. A redundant disk array can now retain data for much longer time than an unprotected single disk.

Do not confuse between reliability and availability.

Reliability is how well a system can work without any failures in its components. If there is a failure, the system was not reliable.

Availability is how well a system can work in times of a failure. If a system is able to work even in the presence of a failure of one or more system components, the system is said to be available.

Redundancy improves the availability of a system, but cannot improve the reliability. Reliability can only be increased by improving manufacturing technologies or using lesser individual components in a system.

DISADVANTAGES DUE TO REDUNDANCY

Every time there is a write operation, there is a change of data. This change also, has to be reflected in the disks storing redundant information. This **worsens the performance of writes** in redundant disk arrays significantly compared to the performance of writes in non redundant disk arrays.

Also, keeping the redundant information consistent in the presence of concurrent I/O operation and the possibility of system crashes can be difficult.

THE NEED FOR RAID

[BACK](#) / [HOME](#) / [NEXT](#)

The need for RAID can be summarized in two points given below. The two keywords are Redundant and Array.

- An **array** of multiple disks accessed in **parallel** will give **greater throughput** than a single disk.
- **Redundant** data on multiple disks provides **fault tolerance**.

Provided that the RAID hardware and software perform true parallel accesses on multiple drives, there will be a **performance improvement over a single disk**.

With a single hard disk, you cannot protect yourself against the costs of a disk failure, the time required to obtain and install a replacement disk, reinstall the operating system, restore files from backup tapes, and repeat all the data entry performed since the last backup was made.

With multiple disks and a suitable redundancy scheme, your system can stay up and running when a disk fails, and even while the replacement disk is being installed and its data restored.

To create an optimal cost-effective RAID configuration, we need to simultaneously achieve the following goals:

- Maximize the number of disks being accessed in parallel.
- Minimize the amount of disk space being used for redundant data.
- Minimize the overhead required to achieve the above goals.

[BACK](#) / [HOME](#) / [NEXT](#)

DATA STRIPING AND REDUNDANCY

[BACK](#) / [HOME](#) / [NEXT](#)

There are 2 important concepts to be understood in the design and implementation of disk arrays:

1. Data striping, for improved performance.
2. Redundancy for improved reliability.

DATA STRIPING

Data striping transparently distributes data over multiple disks to make them appear as a single fast, large disk. Striping improves aggregate I/O performance by allowing multiple I/Os to be serviced in parallel. There are 2 aspects to this parallelism.

- Multiple, independent requests can be serviced in parallel by separate disks. This decreases the queueing time seen by I/O requests.
- Single, multiple block requests can be serviced by multiple disks acting in co-ordination. This increases the effective transfer rate seen by a single request. The performance benefits increase with the number of disks in the array. Unfortunately, a large number of disks lowers the overall reliability of the disk array.

Most of the redundant disk array organizations can be distinguished based on 2 features:

1. the granularity of data interleaving and
2. the way in which the redundant data is computed and stored across the disk array.

Data interleaving can be either fine grained or coarse grained.

Fine grained disk arrays conceptually interleave data in relatively small units so that

all I/O requests, regardless of their size, access all of the disks in the disk array. This results in very high data transfer rate for all I/O requests but has the disadvantages that only one logical I/O request can be in service at any given time and all disks must waste time positioning for every request.

Coarse grained disk arrays interleave data in relatively large units so that small I/O requests need access only a small number of disks while large requests can access all the disks in the disk array. This allows multiple small requests to be serviced simultaneously while still allowing large requests to see the higher transfer rates afforded by using multiple disks.

REDUNDANCY

Since larger number of disks lower the overall reliability of the array of disks, it is important to incorporate redundancy in the array of disks to tolerate disk failures and allow for the continuous operation of the system without any loss of data.

The incorporation of redundancy in disk arrays brings up two problems:

1. Selecting the method for computing the redundant information. Most redundant disks arrays today use parity, though some use Hamming or Reed-Solomon codes.
2. Selecting a method for distribution of the redundant information across the disk array. The distribution method can be classified into 2 different schemes:
 - Schemes that concentrate redundant information on a small number of disks.
 - Schemes that distribute redundant information uniformly across all of the disks.

Such schemes are generally more desirable because they avoid hot spots and other load balancing problems suffered by schemes that do not uniformly distribute redundant information.

Finally, it is important to mention that selecting between the many possible data striping and redundancy schemes involves complex tradeoffs between reliability, performance and cost, which have been discussed in the next few sections.

[BACK](#) / [HOME](#) / [NEXT](#)

BASIC RAID ORGANIZATIONS

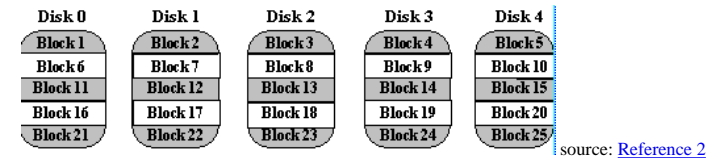
[BACK](#) / [HOME](#) / [NEXT](#)

There are many types of RAID and some of the important ones are introduced below:

NON-REDUNDANT (RAID LEVEL 0)

A non-redundant disk array, or RAID level 0, has the lowest cost of any RAID organization because it does not employ redundancy at all. This scheme offers the best performance since it never needs to update redundant information. Surprisingly, it does not have the best performance. Redundancy schemes that duplicate data, such as mirroring, can perform better on reads by selectively scheduling requests on the disk with the shortest expected [seek](#) and [rotational](#) delays. Without, redundancy, any single disk failure will result in data-loss. Non-redundant disk arrays are widely used in super-computing environments where performance and capacity, rather than reliability, are the primary concerns.

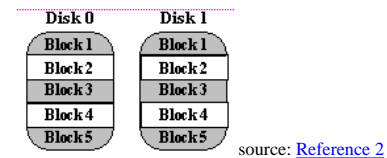
Sequential blocks of data are written across multiple disks in stripes, as follows:



The size of a data block, which is known as the "stripe width", varies with the implementation, but is always at least as large as a disk's sector size. When it comes time to read back this sequential data, all disks can be read in parallel. In a multi-tasking operating system, there is a high probability that even non-sequential disk accesses will keep all of the disks working in parallel.

MIRRORED (RAID LEVEL 1)

The traditional solution, called mirroring or shadowing, uses twice as many disks as a non-redundant disk array. whenever data is written to a disk the same data is also written to a redundant disk, so that there are always two copies of the information. When data is read, it can be retrieved from the disk with the shorter queuing, seek and rotational delays. If a disk fails, the other copy is used to service requests. Mirroring is frequently used in database applications where availability and transaction time are more important than storage efficiency.



MEMORY-STYLE(RAID LEVEL 2)

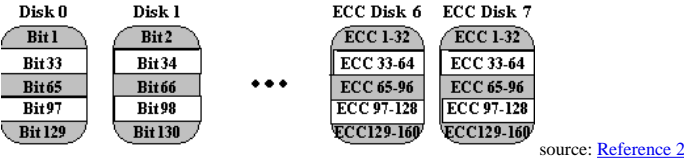
Memory systems have provided recovery from failed components with much less cost than mirroring by using Hamming codes. Hamming codes contain parity for distinct overlapping subsets of components. In one version of this scheme, four disks require three redundant disks, one less than mirroring. Since the number of redundant disks is proportional to the log of the total number of the disks on the system, storage efficiency increases as the number of data disks increases.

If a single component fails, several of the parity components will have inconsistent values, and the failed component is the one held in common by each incorrect subset. The lost information is recovered by reading the other components in a subset, including the parity component, and setting the missing bit to 0 or 1 to create proper parity value for that subset. Thus, multiple redundant disks are needed to identify the failed disk, but only one is needed to recover the lost information.

In you are unaware of parity, you can think of the redundant disk as having the sum of all data in the other disks. When a disk fails, you can subtract all the data on the good disks from the parity disk; the remaining information must be the missing information. Parity is simply this sum modulo 2.

A RAID 2 system would normally have as many data disks as the word size of the computer, typically 32. In addition, RAID 2 requires the use of extra disks to store an error-correcting code for redundancy. With 32 data disks, a RAID 2 system would require 7 additional disks for a Hamming-code ECC. Such an array of 39 disks was the subject of a U.S. patent granted to Unisys Corporation in 1988, but no commercial product was ever released.

For a number of reasons, including the fact that modern disk drives contain their own internal ECC, RAID 2 is not a practical disk array scheme.

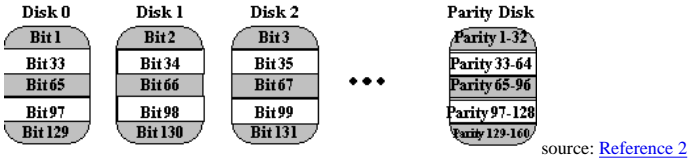


BIT-INTERLEAVED PARITY (RAID LEVEL 3)

One can improve upon memory-style ECC disk arrays by noting that, unlike memory component failures, disk controllers can easily identify which disk has failed. Thus, one can use a single parity rather than a set of parity disks to recover lost information.

In a bit-interleaved, parity disk array, data is conceptually interleaved bit-wise over the data disks, and a single parity disk is added to tolerate any single disk failure. Each read request accesses all data disks and each write request accesses all data disks and the parity disk. Thus, only one request can be serviced at a time. Because the parity disk contains only parity and no data, the parity disk cannot participate on reads, resulting in slightly lower read performance than for redundancy schemes that distribute the parity and data over all disks. Bit-interleaved, parity disk arrays are frequently used in applications that require high bandwidth but not high I/O rates. They are also simpler to implement than RAID levels 4, 5, and 6.

Here, the parity disk is written in the same way as the parity bit in normal Random Access Memory (RAM), where it is the Exclusive Or of the 8, 16 or 32 data bits. In RAM, parity is used to detect single-bit data errors, but it cannot correct them because there is no information available to determine which bit is incorrect. With disk drives, however, we rely on the disk controller to report a data read error. Knowing which disk's data is missing, we can reconstruct it as the Exclusive Or (XOR) of all remaining data disks plus the parity disk.



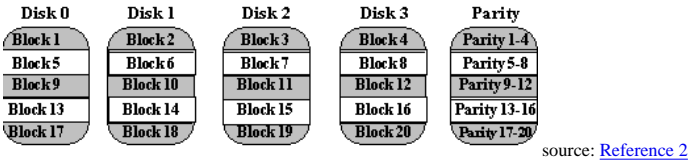
As a simple example, suppose we have 4 data disks and one parity disk. The sample bits are:

Disk 0	Disk 1	Disk 2	Disk 3	Parity
0	1	1	1	1

The parity bit is the XOR of these four data bits, which can be calculated by adding them up and writing a 0 if the sum is even and a 1 if it is odd. Here the sum of Disk 0 through Disk 3 is "3", so the parity is 1. Now if we attempt to read back this data, and find that Disk 2 gives a read error, we can reconstruct Disk 2 as the XOR of all the other disks, including the parity. In the example, the sum of Disk 0, 1, 3 and Parity is "3", so the data on Disk 2 must be 1.

BLOCK-INTERLEAVED PARITY (RAID LEVEL 4)

The block-interleaved, parity disk array is similar to the bit-interleaved, parity disk array except that data is interleaved across disks of arbitrary size rather than in bits. The size of these blocks is called the striping unit. Read requests smaller than the striping unit access only a single data disk. Write requests must update the requested data blocks and must also compute and update the parity block. For large writes that touch blocks on all disks, parity is easily computed by exclusive-or'ing the new data for each disk. For small write requests that update only one data disk, parity is computed by noting how the new data differs from the old data and applying those differences to the parity block. Small write requests thus require four disk I/Os: one to write the new data, two to read the old data and old parity for computing the new parity, and one to write the new parity. This is referred to as a read-modify-write procedure. Because a block-interleaved, parity disk array has only one parity disk, which must be updated on all write operations, the parity disk can easily become a bottleneck. Because of this limitation, the block-interleaved distributed parity disk array is universally preferred over the block-interleaved, parity disk array.



BLOCK-INTERLEAVED DISTRIBUTED-PARITY (RAID LEVEL 5)

The block-interleaved distributed-parity disk array eliminates the parity disk bottleneck present in the block-interleaved parity disk array by distributing the parity uniformly over all of the disks. An additional, frequently overlooked advantage to distributing the parity is that it also distributes data over all of the disks rather than over all but one. This allows all disks to participate in servicing read operations in contrast to redundancy schemes with dedicated parity disks in which the parity disk cannot participate in servicing read requests. Block-interleaved distributed-parity disk array have the best small read, large write performance of any redundancy disk array. Small write requests are somewhat inefficient compared with redundancy schemes such as mirroring however, due to the need to perform read-modify-write operations to update parity. This is the major performance weakness of RAID level 5 disk arrays.

The exact method used to distribute parity in block-interleaved distributed-parity disk arrays can affect performance. Following

figure illustrates left-symmetric parity distribution.

0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

(Left-Symmetric)

Each square corresponds to a stripe unit. Each column of squares corresponds to a disk. P0 computes the parity over stripe units 0, 1, 2 and 3; P1 computes parity over stripe units 4, 5, 6, and 7 etc. (source: [Reference 1](#))

A useful property of the left-symmetric parity distribution is that whenever you traverse the striping units sequentially, you will access each disk once before accessing any disk device. This property reduces disk conflicts when servicing large requests.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
Block 1	Block 2	Block 3	Block 4	Parity 1-4
Block 6	Block 7	Block 8	Parity 5-8	Block 5
Block 11	Block 12	Parity 9-12	Block 9	Block 10
Block 16	Parity 13-16	Block 13	Block 14	Block 15
Parity 17-20	Block 17	Block 18	Block 19	Block 20

source: [Reference 2](#)

P+Q REDUNDANCY (RAID LEVEL 6)

Parity is a redundancy code capable of correcting any single, self-identifying failure. As large disk arrays are considered, multiple failures are possible and stronger codes are needed. Moreover, when a disk fails in parity-protected disk array, recovering the contents of the failed disk requires successfully reading the contents of all non-failed disks. The probability of encountering an uncorrectable read error during recovery can be significant. Thus, applications with more stringent reliability requirements require stronger error correcting codes.

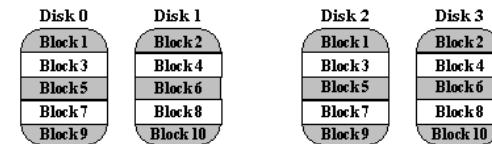
Once such scheme, called P+Q redundancy, uses Reed-Solomon codes to protect against up to two disk failures using the bare minimum of two redundant disk arrays. The P+Q redundant disk arrays are structurally very similar to the [block-interleaved distributed-parity disk arrays](#) and operate in much the same manner. In particular, P+Q redundant disk arrays also perform small write operations using a read-modify-write procedure, except that instead of four disk accesses per write requests, P+Q redundant disk arrays require six disk accesses due to the need to update both the 'P' and 'Q' information.

STRIPED MIRRORS (RAID LEVEL 10)

RAID 10 was not mentioned in the original 1988 article that defined RAID 1 through RAID 5. The term is now used to mean the combination of RAID 0 (striping) and RAID 1 (mirroring). Disks are mirrored in pairs for redundancy and improved performance, then data is striped across multiple disks for maximum performance. In the diagram below, Disks 0 & 2 and Disks 1 & 3 are mirrored pairs.

Obviously, RAID 10 uses more disk space to provide redundant data than RAID 5. However, it also provides a performance advantage by reading from all disks in parallel while eliminating the write penalty of RAID 5. In addition, RAID 10 gives better performance than RAID 5 while a failed drive remains unreplaced. Under RAID 5, each attempted read of the failed drive can

be performed only by reading all of the other disks. On RAID 10, a failed disk can be recovered by a single read of its mirrored pair.



source: [Reference 2](#)

[Tool to calculate storage efficiency given the number of disks and the RAID level](#) (source: [Reference 3](#))

RAID Systems Need Tape Backups

It is worth remembering an important point about RAID systems. Even when you use a redundancy scheme like mirroring or RAID 5 or RAID 10, you must still do regular tape backups of your system. There are several reasons for insisting on this, among them:

- RAID does not protect you from multiple disk failures. While one disk is off line for any reason, your disk array is not fully redundant.
- Regular tape backups allow you to recover from data loss that is not related to a disk failure. This includes human errors, hardware errors, and software errors.

[BACK](#) / [HOME](#) / [NEXT](#)

COST & PERFORMANCE ISSUES

[BACK](#) / [HOME](#) / [NEXT](#)

There are three important considerations while making a selection as to which RAID level is to be used for a system viz. cost, performance and reliability.

There are many different ways to measure these parameters for eg. performance could be measured as I/Os per second per dollar, bytes per second or response time. We could also compare systems at the same cost, the same total user capacity, the same performance or the same reliability. The method used largely depends on the application and the reason to compare. For example, in transaction processing applications the primary base for comparison would be I/Os per second per dollar while in scientific applications we would be more interested in bytes per second per dollar. In some heterogeneous systems like file servers both I/O per second and bytes per second may be important. Sometimes it is important to consider reliability as the base for comparison.

Taking a closer look at the RAID levels we observe that most of the levels are similar to each other. RAID level 1 and RAID level 3 disk arrays can be viewed as a subclass of RAID level 5 disk arrays. Also RAID level 2 and RAID level 4 disk arrays are generally found to be inferior to RAID level 5 disk arrays. Hence the problem of selecting among RAID levels 1 through 5 is a subset of the more general problem of choosing an appropriate parity group size and striping unit for RAID level 5 disk arrays.

Some Comparisons

Given below is a table that compares the throughput of various redundancy schemes for four types of I/O requests. The I/O requests are basically reads and writes which are divided into small (reads & writes) and large ones. Remembering the fact that our data has been spread over multiple disks (data striping), a small refers to an I/O request of one striping unit while a large I/O request refers to requests of one full stripe (one stripe unit from each disk in an error correcting group).

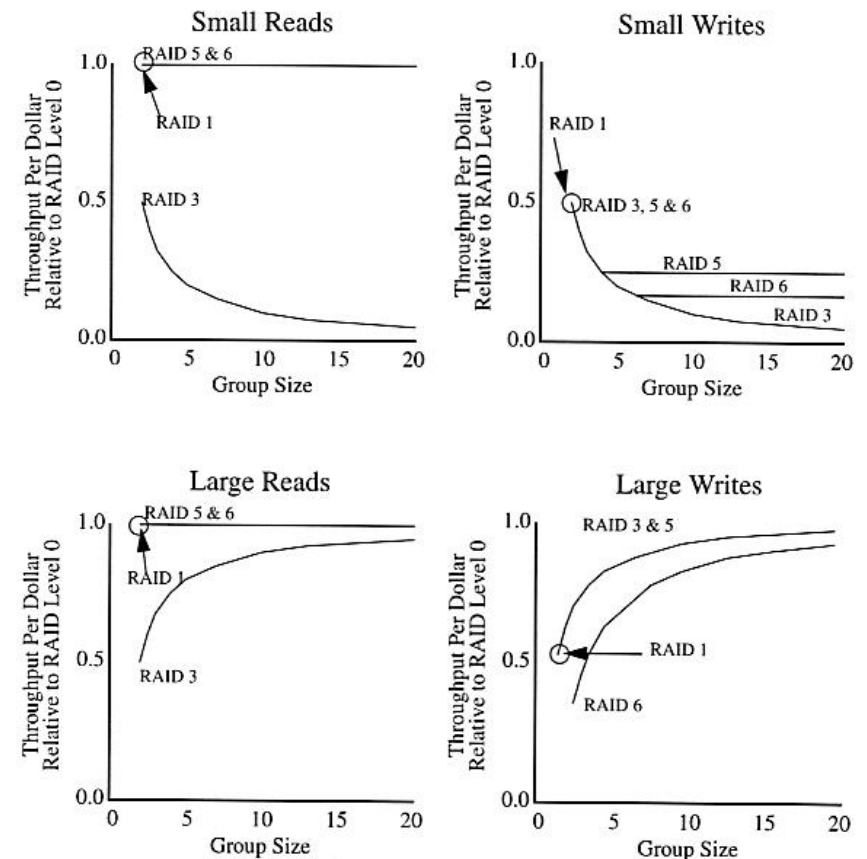
RAID Type	Small Read	Small Write	Large Read	Large Write	Storage Efficiency
RAID Level 0	1	1	1	1	1
RAID Level 1	1	1/2	1	1/2	1/2
RAID Level 3	1/G	1/G	$(G-1)/G$	$(G-1)/G$	$(G-1)/G$
RAID Level 5	1	$\max(1/G, 1/4)$	1	$(G-1)/G$	$(G-1)/G$

RAID Level 6	1	$\max(1/G, 1/6)$	1	$(G-2)/G$	$(G-2)/G$
------------------------------	---	------------------	---	-----------	-----------

G : The number of disks in an error correction group.

The table above tabulates the maximum throughput per dollar relative level 0 for RAID levels 0, 1, 3, 5 and 6. For practical purposes we consider RAID levels 2 & 4 inferior to RAID level 5 disk arrays, so we don't show the comparisons. The cost of a system is directly proportional to the number of disks it uses in the disk array. Thus the table shows us that given equivalent cost RAID level 0 and RAID level 1 systems, the RAID level 1 system can sustain half the number of small writes per second that a RAID level 0 system can sustain. Equivalently the cost of small writes is twice as expensive in a RAID level 1 system as in a RAID level 0 system.

The table also shows storage efficiency of each RAID level. The storage efficiency is approximately inverse the cost of each unit of user capacity relative to a RAID level 0 system. The [storage efficiency](#) is equal to the performance/cost metric for large writes.

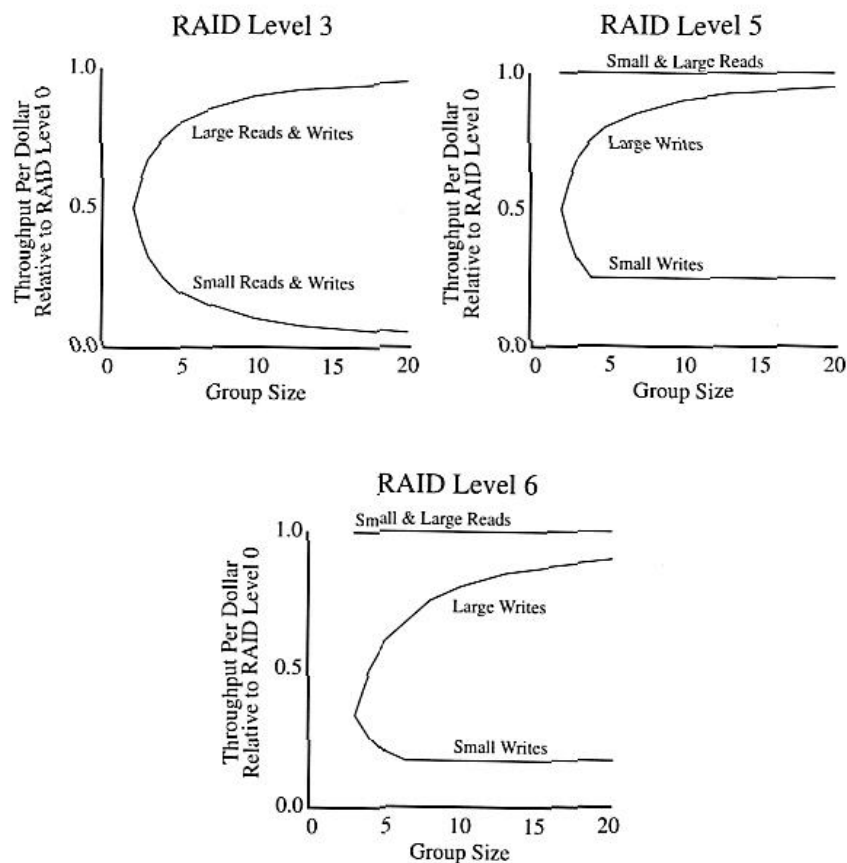


source: [Reference 1](#)

The figures above graph the performance/cost metrics from the table above for RAID levels 1, 3, 5 and 6 over a range of parity group sizes. The performance/cost of RAID level 1 systems is equivalent to the performance/cost of RAID level 5 systems when the parity group size is equal to 2. The performance/cost of RAID level 3 systems is always less than or equal to the performance/cost of RAID level 5 systems. This is expected given that a RAID level 3 system is a subclass of RAID level 5 systems derived by restricting the striping unit size such that all requests access exactly a parity stripe of data. Since the configuration of RAID level 5 systems is not subject to such a restriction, the performance/cost of RAID level 5 systems can never be less than that of an equivalent RAID level 3 system. Of course such generalizations are specific to the models of disk arrays used in the above experiments. In reality, a specific implementation of a RAID level 3 system can have better performance/cost than a specific implementation of a RAID level 5 system.

The question of which RAID level to use is better expressed as more general configuration questions concerning the size of the parity group and striping unit. For a parity group size of 2, mirroring is desirable, while for a very small striping unit RAID level 3 would be suited.

The figure below plots the performance/cost metrics from the table above for RAID levels 3, 5 & 6.



RELIABILITY

[BACK](#) / [HOME](#)

Reliability of any I/O system has become as important as its performance and cost. This part of the tutorial:

- Reviews the basic reliability provided by a [block-interleaved parity disk array](#)
- Lists and discusses three factors that can determine the potential reliability of disk arrays.

Redundancy in disk arrays is motivated by the need to fight disk failures. Two key factors MTTF(Mean-Time-to-Failure) and MTTR(Mean-Time-to-Repair) are of primary concern in estimating the reliability of any disk. Following are some formulae for the mean time between failures :

RAID level 5

$$\frac{\text{MTTF}(\text{disk})^2}{N*(G-1)*\text{MTTR}(\text{disk})}$$

Disk array with two redundant disk per parity group (eg: P+Q redundancy)

$$\frac{\text{MTTF}(\text{disk})^3}{N*(G-1)*(G-2)*(\text{MTTR}(\text{disk})^2)}$$

N - total number of disks in the system

G - number of disks in the parity group

FACTORS AFFECTING RELIABILITY

Three factors that can dramatically affect the reliability of disk arrays are:

- System crashes
- Uncorrectable bit-errors
- Correlated disk failures

SYSTEM CRASHES

System crash refers to any event such as a power failure, operator error, hardware breakdown, or software crash that can interrupt an I/O operation to a disk array.

Such **crashes can interrupt** write operations, resulting in states where the data is updated and the parity is not updated or vice versa. In either case, parity is inconsistent and cannot be used in the event of a disk failure. Techniques such as **redundant hardware** and power supplies can be applied to make such **crashes less frequent**.

System crashes can cause parity inconsistencies in both bit-interleaved and block-interleaved disk arrays, but the problem is of practical concern only in block-interleaved disk arrays.

For, reliability purposes, **system crashes** in block-interleaved disk arrays are similar to disk failures in that they may **result in the loss of the correct parity for stripes that were modified during the crash**.

UNCORRECTABLE BIT-ERRORS

Most [uncorrectable bit-errors](#) are generated because data is incorrectly written or gradually damaged as the magnetic media ages. These errors are detected only when we attempt to read the data.

Our interpretation of [uncorrectable bit error rates](#) is that they represent the rate at which errors are detected during reads from the disk during the normal operation of the disk drive.

One [approach](#) that can be used with or without redundancy is to try [to protect against bit errors](#) by predicting when a disk is about to fail. VAXsimPLUS, a product from DEC, monitors the warnings issued by disks and notifies an operator when it feels the disk is about to fail.

CORRELATED DISK FAILURES

Causes: Common environmental and manufacturing factors.

For example, an accident might sharply increase the failure rate for all disks in a disk array for a short period of time. In general, **power surges, power failures and simply switching the disks on and off** can place stress on the electrical components of all affected disks. Disks also share common support hardware; when this hardware fails, it can lead to multiple, simultaneous disk failures.

Disks are generally more likely to fail either very early or very late in their lifetimes.

[Early failures](#) are frequently caused by transient defects which may not have been detected during the manufacturer's burn-in process.
[Late failures](#) occur when a disk wears out. Correlated disk failures greatly reduce the reliability of disk arrays by making it much more likely that an initial disk failure will be closely followed by additional disk failures before the failed disk can be reconstructed.

MEAN-TIME-TO-DATA-LOSS(MTTDL)

Following are some formulae to calculate the mean-time-to-data-loss([MTTDL](#)). In a block-interleaved parity-protected disk array, data loss is possible through the following three common ways:

- double disk failures
- system crash followed by a disk failure
- disk failure followed by an uncorrectable bit error during reconstruction

The above three failure modes are the hardest failure combinations, in that we, currently, don't have any techniques to protect against them without sacrificing performance.

RAID Level 5

Double Disk Failure	$\frac{\text{MTTF}(\text{disk1}) * \text{MTTF}(\text{disk2})}{N * (G-1) * \text{MTTR}(\text{disk})}$
System Crash + Disk Failure	$\frac{\text{MTTF}(\text{system}) * \text{MTTF}(\text{disk})}{N * \text{MTTR}(\text{system})}$

Disk Failure + Bit Error	$\frac{\text{MTTF}(\text{disk})}{N * (1 - (p(\text{disk}))^{(G-1)})}$
Software RAID	harmonic sum of the above
Hardware RAID	harmonic sum of above excluding system crash + disk failure

Failure Characteristics for [RAID Level 5](#) Disk Arrays (source: [Reference 1](#))

P+Q disk Array

Triple Disk Failures	$\frac{\text{MTTF}(\text{disk}) * (\text{MTTF}(\text{disk2}) * \text{MTTF}(\text{disk3}))}{N * (G-1) * (G-2) * \text{MTTR}(\text{disk})^2}$
System Crash + Disk Failure	$\frac{\text{MTTF}(\text{system}) * \text{MTTF}(\text{disk})}{N * \text{MTTR}(\text{system})}$
Double disk failure + Bit error	$\frac{\text{MTTF}(\text{disk}) * \text{MTTF}(\text{disk2})}{N * (G-1) * (1 - (p(\text{disk}))^{(G-2)}) * \text{MTTR}(\text{disk})}$
Software RAID	harmonic sum of the above
Hardware RAID	harmonic sum excluding system crash +disk failure

Failure characteristics for a [P+Q disk array](#) (source: [Reference 1](#))

p(disk) = The probability of reading all sectors on a disk (derived from disk size, sector size, and BER)

[Tool for Reliability](#) Using the Above Equations. (source: [Reference 3](#))

Glossary

[HOME](#)

RAID -- Redundant Array of Inexpensive Disks

MTTF -- Mean-Time-To-Failure

Average interval of time that a component will operate before failing. See Failure Functions below.

MTTR -- Mean-Time-To-Repair

Average amount of time needed to repair a component, recover a system, or otherwise restore service after a failure. See Failure Functions below.

MTBF -- Mean-Time-Between-Failures

MTBF = MTTR + MTTR

MTTDL -- Mean-Time-To-Data-Loss

A reliability metric. MTTDL is a function of MTTF, MTTR, total number of disks in the system (N), and parity group size (G).

Failure Functions.

When a time basis is determined, failures can be expressed in several ways: the cumulative failure function, the failure intensity function, the failure rate function, and the mean time to failure function. The cumulative failure function (also called the mean value function) denotes the average cumulative failures associated with each point of time. The failure intensity function represents the rate of change of the cumulative failure function. The failure rate function (or called the hazard rate, or the rate of occurrence of failures) is defined as the instantaneous failure rate at a time t, given that the system has not failed up to t. The mean time to failure (MTTF) function represents the expected time that the next failure will be observed. (MTTF is also known as MTBF, mean time between failures.) Note that the above three measures are closely-related and could be translated with one another.

Mean Time To Repair and Availability. Another quantity related to time is mean time to repair (MTTR), which represents the expected time until a system will be repaired after a failure is observed. When the MTTF and MTTR for a system are measured, its availability can be obtained.

Availability -- is the probability that a system is available when needed. Typically, it is measured by

$$\text{Availability} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$$

ECC -- Error Correction Code

BER -- Bit Error Rate

p(disk) -- The probability of reading all sectors on a disk (derived from disk size, sector size, and BER)

G -- parity group size i.e., number of disks in parity group

N -- total number of disks in a redundant array system

seek time -- Amount of time needed to move the head to the correct radial position of the disk.

rotational latency -- Amount of time needed for the desired sector to rotate under the disk head.

[HOME](#)

References

[1] Peter M. Chen, Edward K. Lee, Garth A. Gibson, Randy H. Katz and David A. Patterson. ***RAID: High-Performance, Reliable Secondary Storage***. ACM Computing Surveys.

[2] http://www.consensys.com/html/rzma1 RAID disk array__overvie.html

[3] <http://umunhum.stanford.edu/tools/jsdt.html>

[4] D.A. Patterson and J.L. Hennessy, ***Computer Architecture: A Quantitative Approach***, 2nd edition, Morgan-Kaufmann, 1996.

[5] E K Lee, R H Katz, "**Performance Consequences of Parity Placement in Disk Arrays**"

[6] E K Lee, R H Katz, "**An Analytic Performance Model of Disk Arrays and its Applicaton**"

[7] E K Lee, "**Software and Performance issues in the Implementation of a RAID prototype**"

[8] P M chen, G A Gibson, R H Katz, D A Patterson, "**An Evaluation of Redundant Array of Disks using an Amdhal 5890**"

[9] P M Chen, D A Patterson, "**Maximizing Performance in a striped Disk Array**"

The Tool for Reliability

[HOME](#)

Input Value(s):

Mean-time-to-failure of disk1	MTTF(disk) =	<input type="text"/> hours
Mean-time-to-failure of disk2	MTTF(disk2) =	<input type="text"/> hours
Mean-time-to-failure of disk3	MTTF(disk3) =	<input type="text"/> hours
Mean-time-to-failure of the system	MTTF(sys) =	<input type="text"/> hours
Mean-time-to-repair of a single disk	MTTR =	<input type="text"/> hours
Mean-time-to-repair of the system	MTTR(sys) =	<input type="text"/> hours
Number of disks in the disk array	N =	<input type="text"/>
p(disk)	p =	<input type="text"/>
Parity group size	G =	<input type="text"/>
RAID Level	RAID Level	<input type="text" value="RAIDlevel5"/>
Failure Characteristics	Failure Type	<input type="text" value="Double/Triple Disk Failure"/>

Result(s):

Reliability of the disk array	R =	<input type="text"/> hours	or	<input type="text"/> years
-------------------------------	-----	----------------------------	----	----------------------------

MTTDL	MTTDL =	<input type="text"/> hours	or	<input type="text"/> years
-------	---------	----------------------------	----	----------------------------

Reset Values

Evaluate

[HOME](#)

Tool to Calculate Storage Efficiency

[HOME](#) / [NEXT](#)

This tool allows an architect to design a RAID (redundant array of inexpensive disks) storage system. Given the total number of disks available and the level of RAID desired the designer is given the storage efficiency.

RAID Levels

RAID 1 -- Disk Mirroring
RAID 2 -- Bit Interleaved Redundancy
RAID 3 -- Bitwise Parity
RAID 4 -- Block Parity
RAID 5 -- Interleaved Parity Sectors
RAID 6 -- P+Q Redundancy

Input Values:

Total number of disks available	disks =	<input type="text"/>	disks
RAID level (1-5)	level =	<input type="text"/>	

Results:

Storage Disks	storage =	<input type="text"/>	Disks
Redundant Disks	redund =	<input type="text"/>	Disks
Storage efficiency	effic =	<input type="text"/>	%

<input type="button" value="Reset Values"/>	<input type="button" value="Evaluate"/>
---	---