Overview

Let's explore **RAID**, a strategy for using several disks together to produce a quicker, bigger, and more dependable disk system.

This section should help us answer the following questions:

- **How can we build a storage system that is fast, large, and dependable?**
- **What are the most important techniques?**
- **What are the trade-offs between approaches?**

Introduction

I/O can be slow and bottleneck the system. Sometimes we want it to be bigger to store more data, or more reliable so we can backup our data and keep it secure.

A **Redundant Array of Independent Disks (RAID)** protects against drive failure by storing the same data in different places on multiple hard drives. A group of physical disks can then function as one logical disk on your system.

There are different RAID levels, and not all provide redundancy.

Inside, a RAID contains:
* Several disks
* Memory (volatile and non-volatile), and
* One or more processors.

A hardware RAID is a computer system designed to manage a set of disks.

RAIDs outperform single disks in many ways, including:
* **Performance** - Parallelizing disks can drastically reduce I/O times.
* **Capacity** - Big data requires big disks.
* **Reliability** - Spreading data across multiple drives without RAID techniques makes data **vulnerable to a single disk failure**. With **redundancy**, RAIDs may survive a disk failure and keep working like nothing happened.

RAID technology has three basic functions:
1. Block access to data is achieved by **striping the data** on the disk, reducing the mechanical seek time and increasing data access speed.
1. **Reading multiple disks** in an array reduces mechanical seek time and increases data access speed at the same time.
1. **Mirroring** and storing **parity** information helps in data protection.

RAIDs do their job **transparent** to the systems they are used on, so it looks like a single big disk representing itself as a linear array of blocks. It can replace a disk without changing a single line of software while the OS and user applications continue to work as before.

Transparency increases RAID **deployability** by letting users and administrators use RAID without worrying about software compatibility.

Interface And RAID Internals

A file system sees a RAID as a single disk, a linear array of blocks that the file system or client can read or write to.

The RAID will issue one or more physical I/Os once it knows which disk (or disks) to access. The physical I/Os vary depending on the RAID level.

If a RAID stores each block twice (one on each disk), two physical I/Os would be required for each logical I/O.

A RAID system connects to a host using a normal interface (SCSI or SATA).

Inside, RAIDs contain:
* **Microcontrollers** that run firmware to control the RAID
* **Volatile memory (DRAM)** to buffer data blocks being read and written
* **Non-volatile memory (NVM)** to buffer writes safely, and
***** Parity calculations might even be done with special logic

In a sense, a RAID system is a tailored computer system that contains a processor, memory, and disks Instead of running applications, it runs RAID-specific software.

Fault Model

RAIDs are designed to sense and recover from certain kinds of disk failures. For this, it needs to know which faults to expect.

Fail-Stop

With the **fail-stop** fault model, a disk can be in one of two states.
- **Working**
- All blocks can be read or written.
- **Failed**
- When a disk has failed, we assume the data is gone for good.

The **fail-stop** model assumes assume that a failing disk is quickly sensed by fault detection. In a RAID array, we would expect the RAID controller hardware (or software) to detect a failed disk instantly.

For now, we can ignore more complex errors like disk corruption. We also don't have to worry about a single block on a functional drive becoming inaccessible. We'll consider more realistic disk faults like these a little later.

Evaluating A RAID

There are many ways to construct a RAID, each with its own goals, advantages, and disadvantages.

We will evaluate each RAID design on three categories.
* **Capacity** - How much usable space is in N drives with B blocks each available to RAID clients?
* $N \cdot B$ without redundancy. Keeping two copies of each block, or **mirroring** gives us $(N \cdot B)/2$.

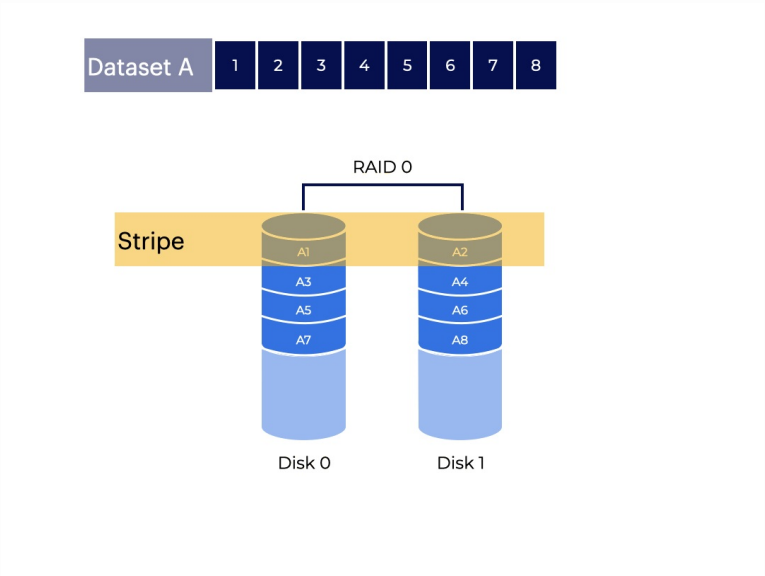
- **Reliability** - Tolerance for disk failures for specified design. Our current fault model assumes only a complete disk can fail.
- **Performance** - Performance is difficult to evaluate because it really depends on the workload transmitted to the disk array. Let's look at some typical workloads before our analysis.
-
- - **Availability** - How available is the system for actual use?

RAID Level 0: Striping

RAID Level 0 uses block level **striping**, or splitting data up across multiple areas. It chooses reasonable strips on N hard disks to make a strip set.

The idea is to split a piece of data into several strips and write it to all of the hard drives at the same time, allowing reading and writing to happen simultaneously. It performs well, but doesn't protect us against a disk failure.

Say we have 2 physical hard disks of $1TB$ each and these disks store our data, $A1 - 8$.



.guides/img/raid1

In this figure, $A1$ & $A2$ create a **stripe**. Instead of only placing one block into a disk at a time, we can deal with two (or more) at a time.

We can illustrate the advantage of storing our data like this with an example.

We could ask a person to write down a dataset of numbers, $1 - 20$. This should take them a few seconds.

We could then ask two people to write down this number set and split the work, with person 1 writing $1 - 10$ and person 2 writing $11 - 20$. This would take us less time that our single person request.

Let's say we then lost one of our data writers responsible for a piece of our data. If we lose them or they fail, we would also lose our dataset.

This is the same case with RAID Level 0. If we lose one of our disks, we lose all of our data because our drives were operating as one logical drive.

Evaluation

- **Reliability** - RAID Level 0 doesn't stand up against disk failure, so it's not very reliable.
- **Capacity** - Since the whole space is used to store original data (not copies), N disks, each with B blocks is fully used.

Chunk Sizes

The **chunk size** influences array performance. Because the positioning time for the entire request is decided by the maximum of all positioning times across drives, accessing blocks across many disks increases positioning times.

Large **chunk sizes** restrict parallelism within files, requiring numerous simultaneous searches in order to get high speeds. However, they also reduce positioning time. When only one file fits on a single disk and it is accessed, the positioning time is only that of a single disk positioning.

To find the *"best"* chunk size, you need to know about how much work the disk system does. For now, we'll assume the size of one block chunk is $4KB$ and stick to a single block.

Evaluating RAID Performance

Two performance measures can be used to evaluate RAID performance:
1. **Single-request latency** - The time it takes for a single logical I/O request to be addressed by a RAID shows how much parallelism there can be.
1. **Steady-state throughput** - The overall bandwidth of multiple queries. Because RAIDs are often used in high-performance situations, we will focus on steady-state bandwidth.

Welcome back our sequential and random style workload examples!

- A **sequential request** (sequence of requests) can accesses $1MB$ of data, starting at block x and finishing at block ($x + 1MB$)
- For **random workloads** each request is small and to a different random disk location.

Sequential and random workloads need different disk performance.
* **Sequential access** allows a disk to work at its best, spending less time seeking and waiting for rotation and **more time delivering data**.
* With **random access**, time is spent seeking and waiting for rotation, not delivering data.

To account for this, we'll assume a disk can transport data at $S\ MB/s$ sequentially and $R\ MB/srandomly$. Generally, $S >> R$.

Let's calculate S and R with:
* A $10MB$ sequential transfer, and
* A $10KB$ random transfer.

Say we have the following disk properties:

- Average seek time: $7ms$
- Average rotational delay: 43 ms4
- Transfer rate of disk: $50MB/s$

To calculate S we need to:
* Figure out how long a $10MB$ transfer takes
* Seek for 7 milliseconds
* Spin for 3 milliseconds, then
* The transfer begins

A transfer of $10MB$ at $50MB/s$ takes $200milliseconds$. So, each $10 - MB$ request takes $210milliseconds$ to complete.

$$S = \frac{Data_{amount}}{Time_{Access}} = \frac{10MB}{210ms} = 47.62MB/s$$

S is very close to the disk's peak bandwidth because of the time spend transferring data.

We can compute R similarly:

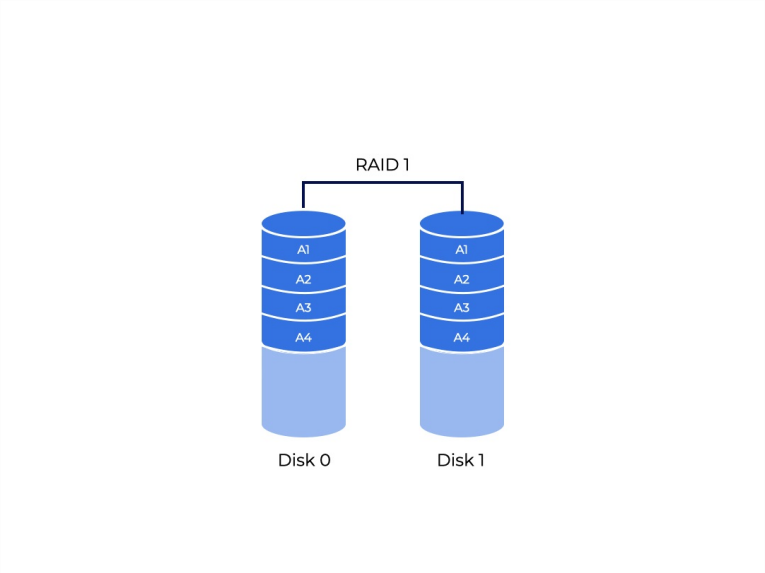
$$R = \frac{Data_{amount}}{Time_{Access}} = \frac{10KB}{10.195ms} = 0.981MB/s$$

RAID-1: Mirroring

RAID level 1 is also known as **mirroring**. This level doesn't have **striping**. Instead is uses at least two drives that copy the data storage, so we won't lose all of our data if one disk fails.

Reading from both disks at the same time will improve read performance. Write performance is the same for single disk storage.

In a mirrored system, each logical block has two physical copies.



.guides/img/raid2

Here, we won't have as much space as we'd like. Of our 2 total terabytes, only half is usable because we need the other half for the mirrored copy of data.

This also means that we have the **reliability** of if one disk was to fail, the raid controller can simply move over to the working version. We could then replace the faulty hard drive, and the data would begin to copy again to the new drive.

Evaluation

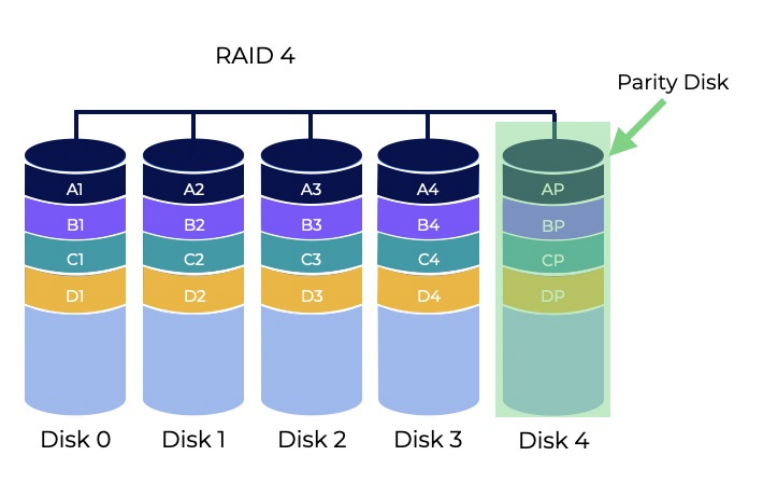
- **Reliability** - This RAID level can handle at least 1 disk failure because the failed disk would have a copy on another disk. Our reliability would be 1 to N/2 because it can handle $N/2$ disk failures.
- **Capacity** - Only half of our available space can be used for storage. The

other half of our space is used for copies of what we already have, so our capacity would be $N \cdot B/2$

RAID-4: Space Saving With Parity

RAID Level 4 is striped at the block level and has a dedicated **parity** disk. **Parity** is a way of providing redundancy to a disk array.

To avoid the significant space cost we get with mirrored systems, parity-based techniques aim to use less capacity, by sacrificing performance.



.guides/img/raid3

With this RAID Level, we have one column solely for parity. This parity is calculated using an **XOR** function. One simple approach is to assign data bits with even numbers of ones to parity 0, and data bits with odd numbers of ones to parity 1.

If we have data bits 0, 0, 0, 1 the parity bit would be:

$$XOR(0, 0, 0, 1) = 1$$

If we have data bits 0, 0, 1, 1 the parity bit would be:

$$XOR(0, 0, 1, 1) = 0$$

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	0	0	1	1
0	0	1	1	0

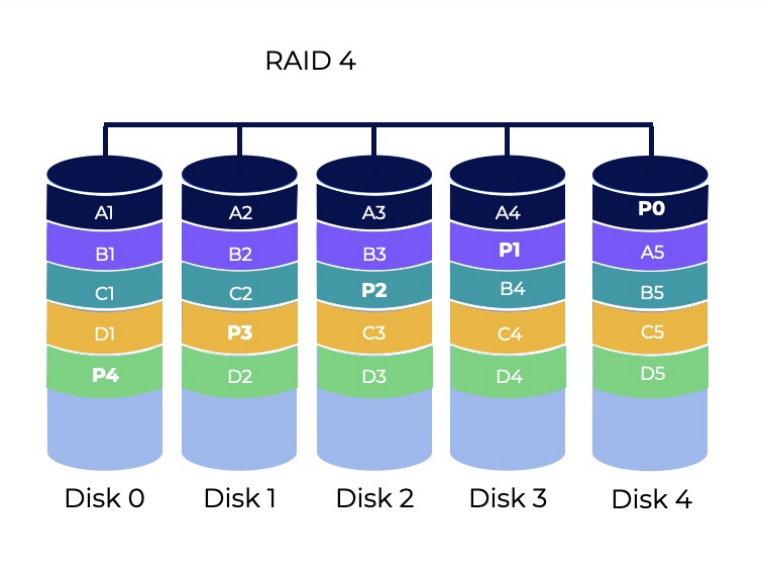
Above is a table with all of our disks and imaginary data bits. If we lost disk 3 to a disk failure, we can look at the values in the other columns as well as the parity bit and recompute the data bit that was stored in Disk 3. This is how we would recover lost data.

Evaluation

- **Reliability** - This RAID level lets us recover 1 disk failure at the most. If we lose more than one disk, we can't recover the data.
- **Capacity** - One whole disk in this system is reserved to storing the parity information, so $N - 1$ disks are available for us to store data, with each disk having B blocks.

RAID-5: Rotating Parity

RAID-5 is similar to RAID-4 except instead of having one dedicated parity drive, the parity is distributed across the drives, increasing space efficiency and improving write performance.



.guides/img/raid4

Each stripe's parity block is now distributed the disks, removing the parity-disk bottleneck.

RAID-5 analysis is similar to RAID-4. Both levels have the same effective capacity and fault tolerance. A single request (read or write) has the same latency as RAID-4.

Random read speed improves because we can use all disks now.

Evaluation

- **Reliability** - This RAID level lets us recover 1 disk failure at the most. If we lose more than one disk, we can't recover the data.
- **Capacity** - The space that we use for storing the parity information is equal to one whole disk. $N - 1$ disks are available for us to store data, with each disk having B blocks, or $(N - 1) \cdot B$

RAID-10: 1+0

RAID-10 is sometimes referred to as a nested level 1 + 0. We have a level one set up where it's mirroring data, but it's then striped across like in a RAID 0

If Disk 1 were to fail, the system won't mess up because we have the backup with disk 0 in raid 1. We also have the level of performance that comes with using a RAID 0, but we can only actually use half of the available space.

This system is efficient because an entire pair of drives would need to fail for this drive to mess up.

Other RAID Levels

There are several other types of RAID levels that we didn't cover in detail.

- **RAID-2** is made up of bit-level stripping with a Hamming Code parity.
- **RAID-3** is made up of byte-level striping with a dedicated parity.

These two aren't used very often.

- **RAID-6** is a newer version that has a distributed double parity. This means that instead of just one parity bit being distributed across all the disks, there are two spread out across each block.

There are also hybrid RAIDs, which use more than one RAID level nested one after the other to meet certain needs.

Summary

RAID combines multiple separate disks into a larger, more dependable unit; it does so transparently, so the hardware and software above are unaffected.

- The exact RAID level you choose depends greatly on what is relevant to the end-user.
- Mirrored RAID, for example, is simple, stable, and gives decent performance but at a significant cost.
- RAID-5, on the other hand, is more dependable and has better capacity, although it struggles with tiny writes.

Choosing the right RAID and configuring its parameters (chunk size, number of drives, etc.) for a given workload is an art and often difficult.