An Architectural and Performance Analysis of RAID Systems

Derived from Lecture on Computing Infrastructures

June 29, 2025

Contents

1 Introduction to RAID

The proliferation of data-intensive applications necessitates storage solutions that transcend the limitations of a single disk drive. A single disk is constrained by its physical capacity, its data transfer rate, and its inherent susceptibility to failure. To address these challenges, a storage architecture known as **RAID (Redundant Array of Independent Disks)** was proposed by Patterson, Gibson, and Katz at Berkeley in the late 1980s.

Definition 1.1 (RAID)**.** *RAID is a storage virtualization technology that combines multiple physical disk drives into one or more logical units for the purposes of data redundancy, performance improvement, or both.*

The core premise of RAID is to create the illusion of a single, large, fast, and reliable disk from an array of smaller, less expensive, and less reliable commodity disks. This is achieved through a specialized hardware or software component known as the **RAID controller**, which manages the distribution and organization of data across the physical disks transparently to the host operating system.

The primary objectives of a RAID system are threefold:

1. **Increased Capacity:** By aggregating multiple disks, the total storage capacity can exceed that of the largest available single disk.
2. **Enhanced Performance:** By accessing multiple disks in parallel, the aggregate data transfer bandwidth can be significantly increased.
3. **Improved Reliability:** By storing redundant information, the system can tolerate the failure of one or more disks without data loss.

The RAID Controller. The RAID controller is a sophisticated embedded system, often equipped with its own CPU, memory (DRAM cache), and I/O interfaces. It intercepts I/O requests from the operating system and translates them into coordinated operations on the physical disks according to a specific **RAID level**.

2 Core RAID Techniques

RAID architectures are built upon two fundamental techniques: data striping and redundancy. Different RAID levels represent distinct combinations of these techniques.

2.1 Data Striping

Data striping is a technique for distributing data sequentially across multiple disks. A logical, contiguous stream of data is broken down into smaller segments, which are then written to different disks in a round-robin fashion.

Definition 2.1 (Stripe Unit and Stripe Width)**.**

- *The **stripe unit** (or chunk size) is the smallest contiguous block of data written to a single disk before the RAID controller moves to the next disk in the array. This is a configurable parameter.*
- *The **stripe width** is the total number of disks across which a full stripe of data is written. In non-redundant arrays, this is simply the number of disks, N.*

The primary benefit of striping is performance. For large, sequential I/O requests, the data can be read or written from all disks simultaneously, multiplying the effective bandwidth. For small, random I/O requests, the requests are statistically likely to be directed to different disks, allowing them to be serviced in parallel.

Logical to Physical Mapping. Let N be the number of disks in the array and C be the chunk size in blocks. A logical block address, LBA, can be mapped to a physical disk index, D_{idx}, and a local block address on that disk, D_{LBA}, using the following relations:

$$D_{idx} = (LBA/C) \pmod N \tag{1}$$

$$D_{LBA} = (LBA/(C \cdot N)) \cdot C + (LBA \pmod C) \tag{2}$$

The choice of chunk size involves a trade-off:

- **Small Chunks:** Maximize parallelism for small, random requests, leading to higher I/O operations per second (IOPS). This is ideal for transactional workloads.
- **Large Chunks:** Improve single-disk performance for sequential workloads by reducing seek time overhead relative to data transfer time. This is better for streaming large files.

2.2 Redundancy

While striping improves performance, it decreases reliability. The Mean Time To Failure (MTTF) of an array of N disks is approximately $\frac{\text{MTTF}_{disk}}{N}$, assuming failures are independent. If any single disk fails in a striped array without redundancy, the entire logical volume is lost. To counteract this, redundancy is introduced.

2.2.1 Data Duplication (Mirroring)

The simplest form of redundancy is to create an exact, block-for-block copy of the data on a separate disk or set of disks.

- **Pros:** Conceptually simple. Fast data reconstruction, as data can be directly copied from the mirror. Read performance can be enhanced by servicing read requests from any available disk in the mirror set.
- **Cons:** High storage overhead. A simple two-disk mirror has a 50% capacity cost (i.e., only 50% of the total physical capacity is usable for data storage).

2.2.2 Data Reconstruction (Parity)

A more space-efficient method involves storing a "signature" of the data from which lost information can be reconstructed. The most common technique is parity, based on the exclusive OR (XOR) operation.

Lemma 2.1 (XOR Parity)**.** *For a set of bits $d_0, d_1, \ldots, d_{N-1}$, the parity bit P is calculated as:*

$$P = d_0 \oplus d_1 \oplus \cdots \oplus d_{N-1} \tag{3}$$

If any single bit d_i is lost, it can be reconstructed using the remaining bits and the parity bit:

$$d_i = d_0 \oplus \cdots \oplus d_{i-1} \oplus d_{i+1} \oplus \cdots \oplus d_{N-1} \oplus P \tag{4}$$

Proof. This follows from the properties of XOR, namely that $x \oplus x = 0$ and $x \oplus 0 = x$. □

Reconstruction Cost. While parity is space-efficient (requiring only one disk's worth of capacity for redundancy in a simple setup), it imposes a higher computational and I/O cost during reconstruction. To rebuild a failed disk, the controller must read the corresponding data blocks from *all* other surviving disks in the stripe to recalculate the lost data. This is significantly more I/O-intensive than the simple copy operation in a mirrored system.

3 Standard RAID Levels

RAID levels are standardized configurations that combine striping and redundancy in different ways. We will analyze the performance and reliability of the most common levels. Let N be the number of disks, S be the sequential bandwidth of a single disk, and R be the random I/O rate of a single disk.

3.1 RAID 0: Pure Striping

RAID 0 implements data striping with no redundancy. Its sole purpose is to maximize performance and capacity.

Table 1: RAID 0 Characteristics

Attribute	**Description**
Usable Capacity	$N \times$ DiskSize (100% efficiency)
Fault Tolerance	None. Failure of any single disk results in total data loss.
Sequential Read/Write	$N \times S$. Excellent performance due to full parallelism.
Random Read/Write	$N \times R$. Excellent performance.
Pros	Highest performance, full capacity utilization, simple design.
Cons	No reliability. Less reliable than a single disk.
Use Cases	Non-critical data requiring high speed, such as video editing scratch disks or temporary scientific computing storage.

3.2 RAID 1: Pure Mirroring

RAID 1 implements data mirroring with no striping. Its primary purpose is reliability. In its most common form, it uses two disks to create a single mirrored pair.

Table 2: RAID 1 Characteristics (2-Disk Mirror)

Attribute	Description
Usable Capacity	$\frac{N}{2} \times$ DiskSize (50% efficiency for a 2-disk set)
Fault Tolerance	Can tolerate the failure of one disk.
Sequential Read	Up to $N \times S$ (controller can read from both disks in parallel).
Sequential Write	S (data must be written to all disks).
Random Read	Up to $N \times R$.
Random Write	R.
Pros	High reliability, simple, fast recovery.
Cons	Very high storage cost (50% overhead).
Use Cases	Operating system disks, databases, and other applications requiring high availability and fault tolerance over capacity.

3.3 Nested RAID Levels: RAID 1+0 and RAID 0+1

To combine the performance of striping with the reliability of mirroring, nested (or hybrid) RAID levels are used. The two most common are RAID 1+0 and RAID 0+1. Both require an even number of disks, with a minimum of four.

Definition 3.1 (Nested RAID Naming)**.** *A nested RAID level denoted as 'X+Y' (e.g., 1+0) implies that the RAID Y policy is applied first (closer to the host request), and its output is then directed to one or more underlying arrays configured with the RAID X policy.*

3.3.1 RAID 1+0 (Stripe of Mirrors)

In RAID 1+0, data is first striped across multiple mirrored pairs.

- **Architecture:** The array is constructed from $N/2$ mirrored pairs (RAID 1 groups). A top-level RAID 0 controller then stripes data across these pairs.
- **Fault Tolerance:** Highly robust. The system can tolerate the failure of at least one disk in every mirrored pair. The entire array only fails if *both* disks in a single mirrored pair fail.

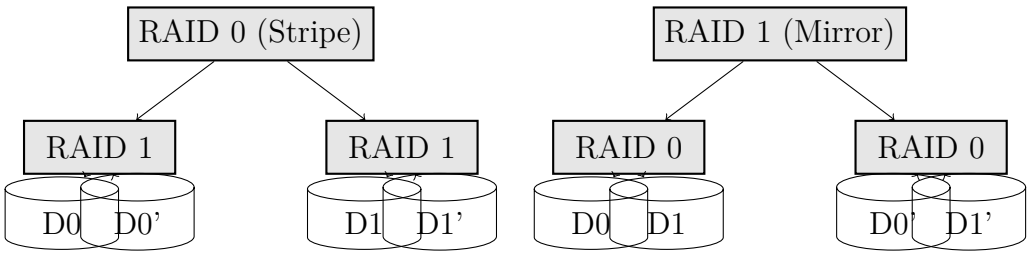
3.3.2 RAID 0+1 (Mirror of Stripes)

In RAID 0+1, data is first striped across a set of disks, and this entire striped set is then mirrored.

- **Architecture:** The array is constructed from two RAID 0 sets of $N/2$ disks each. A top-level RAID 1 controller mirrors all writes to both sets.

- **Fault Tolerance:** Less robust than RAID 1+0. If any single disk fails, its entire RAID 0 set is considered degraded. The system then relies completely on the other striped set. If a subsequent failure occurs on *any disk* in the second set, the entire array fails.

Key difference!



(a) RAID 1+0: Stripe of Mirrors **(b) RAID 0+1: Mirror of Stripes**

Figure 1: Architectural comparison of RAID 1+0 and RAID 0+1.

Table 3: Comparison of Nested RAID Levels

Attribute	**RAID 1+0**	**RAID 0+1**
Usable Capacity	$N/2 \times$ DiskSize	$N/2 \times$ DiskSize
Performance	Generally similar, high for all workloads.	Generally similar, high for all workload
Reliability	**Higher**. Survives multiple failures if they occur in different mirrored pairs.	**Lower**. A second failure on the mirrored side causes total array failure.
Rebuild Cost	Lower. Only the failed disk's mirror is needed for rebuild, reducing I/O load.	Higher. The entire striped set must be read to rebuild the mirror.

Conclusion. Due to its superior fault tolerance and more efficient rebuild process, **RAID 1+0 is almost always preferred over RAID 0+1** in modern system design.

3.4 Parity-Based RAID: Levels 4, 5, and 6

These levels use parity for redundancy instead of mirroring, offering better storage efficiency.

- **RAID 4:** Stripes data across $N-1$ disks and stores all parity information on a single, dedicated parity disk. This parity disk becomes a write bottleneck, as every write operation requires an update to it.
- **RAID 5:** Solves the RAID 4 bottleneck by distributing the parity blocks across all disks in the array, along with the data blocks. This balances the write load more effectively. It can tolerate a single disk failure.
- **RAID 6:** Extends RAID 5 by adding a second, independent parity block. This allows the array to tolerate the failure of any **two** disks, providing significantly higher reliability. This is critical for large arrays where the probability of a second failure occurring during the lengthy rebuild of a first failure is non-trivial.

The performance of these levels, particularly for writes, is more complex due to the "read-modify-write" penalty associated with updating parity information.

4 Performance Analysis Formalism

To properly evaluate RAID performance, we must distinguish between sequential and random access patterns, as their impact on disk mechanics is vastly different.

4.1 Single Disk Baseline

Let's establish a baseline for a typical mechanical hard drive.

- Average Seek Time (T_{seek}): 7 ms
- Rotational Delay ($T_{rotation}$): 3 ms (for a 10,000 RPM disk, avg. is half rotation)
- Transfer Rate ($R_{transfer}$): 50 MB/s

4.1.1 Sequential Access Bandwidth (S)

For a large sequential transfer (e.g., 10 MB), the mechanical overhead is paid only once.

$$T_{total} = T_{seek} + T_{rotation} + T_{transfer} = 7ms + 3ms + \frac{10\,\text{MB}}{50\,\text{MB/s}} = 10ms + 200ms = 210ms \quad (5)$$

The effective bandwidth, S, is:

$$S = \frac{\text{Data Size}}{T_{total}} = \frac{10\,\text{MB}}{0.21\,\text{s}} \approx 47.6\,\text{MB/s} \quad (6)$$

For large transfers, $S \approx R_{transfer}$

4.1.2 Random Access Rate (R)

For small random transfers (e.g., 10 KB), the mechanical overhead dominates each operation.

$$T_{total} = T_{seek} + T_{rotation} + T_{transfer} = 7ms + 3ms + \frac{0.01\,\text{MB}}{50\,\text{MB/s}} = 10ms + 0.2ms = 10.2ms \quad (7)$$

The effective bandwidth for a single request is low. A more useful metric is the number of I/O Operations Per Second (IOPS), which is $1/T_{total}$. The random access bandwidth, R, is:

$$R = \frac{\text{Data Size}}{T_{total}} = \frac{10\,\text{KB}}{0.0102\,\text{s}} \approx 0.98\,\text{MB/s} \quad (8)$$

For small random transfers, $R \ll S$

4.2 RAID Performance Summary

The following table summarizes the theoretical peak performance of various RAID levels in terms of the single-disk bandwidths S and R.

Table 4: RAID Performance Characteristics (N disks)

RAID Level	**Seq. Read**	**Seq. Write**	**Rand. Read**	**Rand. Write**
RAID 0	$N \times S$	$N \times S$	$N \times R$	$N \times R$
RAID 1	$N \times S$	S	$N \times R$	R
RAID 1+0	$N \times S$	$\frac{N}{2} \times S$	$N \times R$	$\frac{N}{2} \times R$
RAID 5	$(N-1) \times S$	$(N-1) \times S$	$N \times R$	$\frac{N}{4} \times R$ †

Table 5: *

†The random write performance for RAID 5 is significantly penalized by the read-modify-write cycle. A single logical write requires four physical I/O operations: read old data, read old parity, write new data, write new parity.