RAID: Redundant Arrays of Inexpensive Disks

Team Members: Yuhan Rui, Yibin Lou , Youcheng Fang

December 26, 2025

Basics of RAID (from "Three Easy Pieces, CH38")

What is RAID?

- **Definition**: **R**edundant **A**rray of **I**nexpensive **D**isks.
- **Origins**:
  - Introduced in the late 1980s.
  - By researchers at **U.C. Berkeley**: David Patterson, Randy Katz, and Garth Gibson.

The Crux: How to make a Large, Fast, Reliable Disk?

- **Faster**: I/O operations are slow and can be the system bottleneck.
- **Larger**: Data grows rapidly; single disks get full quickly.
- **More Reliable**: If a disk fails without backup, valuable data is gone.

- **Solution**: Use multiple disks in concert to build a better storage system.

RAID: Interface and Transparency

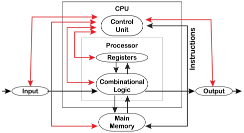
- **Externally**: Looks like a typical disk (a linear array of blocks).
- **Internally**: A complex system (a "computer" itself):
  - Multiple disks.
  - Memory (Volatile and Non-volatile).
  - One or more processors to manage the array.
- **Advantages**:
  - **Performance**: Parallel I/O speeds up operations.
  - **Capacity**: Aggregates multiple disks for large datasets.
  - **Reliability**: Uses **redundancy** to tolerate disk loss.

Tip: Transparency Enables Deployment

- RAID replaces a disk **transparently**: No changes needed to the host computer, OS, or applications.
- This compatibility was key to RAID's massive success ("Deployability").

Hardware RAID: A Specialized Computer

Under the Hood: Components & Architecture



- **I/O Processor**: Runs firmware, offloads RAID logic (e.g., parity calc) from host CPU.
- **DRAM Cache**: GBs of memory buffer to accelerate R/W performance.
- **BBU (Battery Backup)**: Solves **"Consistency Update Problem"** by preserving cache on power loss.
- **Host Interface**: Connects to motherboard (PCIe); presents "Single Large Disk" to OS.
- **XOR Engine**: Dedicated hardware circuit for extreme speed XOR parity calc.
- **Disk Interface**: Connects backend physical drives (SATA / SAS).

Enterprise Storage: External Disk Arrays

Scaling Up: When a Card Isn't Enough

- **Independence**: Standalone hardware cabinet connected via Fiber/SAS.
- **Backplane & Hot-Swap**: Massive circuit board; allows replacing drives *without* powering down.
- **Ultimate Transparency**:
  - Internally: 100+ drives, complex RAID-6.
  - Externally: Server sees just **one cable** and **one giant disk**. (Storage Virtualization)

"Essentially, it strips the 'RAID Controller' and 'Batch of Disks' from the host to create a standalone, specialized 'Storage Server'."

Fault Model

To understand RAID, we must first define the **Fail-Stop Fault Model**.

- **Two States Only**:
  - **Working**: All blocks can be read or written perfectly.
  - **Failed**: The disk is permanently lost.
- **Critical Assumption: Immediate Detection**
  - We assume the RAID controller can **immediately observe** when a disk has failed.
  - *Example*: The controller receives an error code or detects a timeout.
- **What we ignore (for now)**:
  - *Silent Data Corruption*: Disk returns bad data without error.
  - *Latent Sector Errors*: Only part of the disk fails.

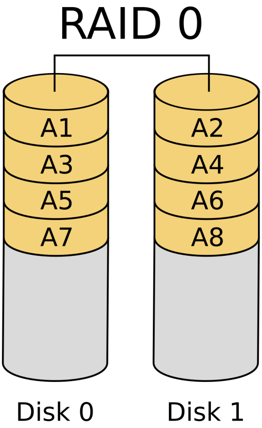
Evaluation Metrics

Three Axes of Analysis

- **1. Capacity**: How much *useful* space is available?
  - Given N disks with B blocks each.
  - **Range**: From $N \cdot B$ (No redundancy) to $N \cdot B/2$ (Mirroring).
- **2. Reliability**: How many disk faults can we tolerate?
  - Based on our **Fail-Stop** model (entire disk fails).
- **3. Performance**: The most challenging metric.
  - *Why?* It depends heavily on the **Workload**.
  - We must test against distinct patterns: Sequential vs. Random, Read vs. Write.

RAID Level 0: Striping

Performance at all costs



- **Not "True" RAID**:
  - No redundancy at all.
  - Excellent **upper-bound** for performance and capacity.
- **Round-Robin Distribution**:
  - Spreads blocks across disks to extract max **parallelism**.
- **Terminology**:
  - Blocks in the same row (e.g., 0, 1, 2, 3) form a **"Stripe"**.

RAID 0: Chunk Sizes & Mapping

Fine-tuning the Striping Strategy

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Chunk Size = 1 Block (4KB)

Disk 0	Disk 1	Disk 2	Disk 3	
0	2	4	6	chunk size:
1	3	5	7	2 blocks
8	10	12	14	
9	11	13	15	

Chunk Size = 2 Blocks (8KB)

Impact of Chunk Size

- **Small**: High Parallelism.
- **Big**: High Concurrency.

Mapping Formula (Generalized)

Let $S =$ Chunk Size, $N =$ Disks.

- $\text{ChunkIdx} = \lfloor A/S \rfloor$
- **Disk** $= \text{ChunkIdx}\%N$
- **Offset** $= \lfloor \text{ChunkIdx}/N \rfloor \cdot S + (A\%S)$

Example ($S = 2, N = 4$, Block 14)

- $\text{ChunkIdx} = 14/2 = 7$
- **Disk** $= 7\%4 = \mathbf{3}$ (Disk 3)
- **Offset** $= \lfloor 7/4 \rfloor \cdot 2 + (14\%2) = 2$
- *Result: Disk 3, Offset 2*

Chunk Size Trade-offs

Parallelism vs. Positioning Time

- **Small Chunk Size** (e.g., 4KB):
  - **Pro**: Increases **Parallelism**. A single file stripes across many disks (fast transfer).
  - **Con**: Increases **Positioning Time**.
    - Accessing multiple disks means waiting for the *slowest* disk to position.
    - $T_{position} = \max(T_{disk1}, T_{disk2}, \dots)$
- **Big Chunk Size** (e.g., 64KB):
  - **Pro**: Reduces **Positioning Time**. A file fits on fewer disks (or just one).
  - **Con**: Reduces Intra-file Parallelism.
    - Throughput relies on **Concurrency** (multiple independent requests).
- **Conclusion**:
  - Determining the "best" size is hard; it requires knowing the **Workload**.
  - *Note: Real arrays often use 64KB, but we assume 4KB for simplicity.*

RAID 0 Analysis & Performance Metrics

Evaluating Striping and Defining Workloads

RAID 0 Analysis

- **Capacity**: **Perfect** ($N \cdot B$). No space wasted.
- **Reliability**: **Zero**. Any disk failure $\rightarrow$ Data Loss.
- **Performance**: **Excellent**. All disks utilize in parallel.

- **Performance Metrics**:
  - **Single-request Latency**: Time for one logical I/O operation.
  - **Steady-state Throughput**: Total bandwidth of concurrent requests (RAID is typically used in high-performance environments, so this is our main focus).

Sequential vs. Random Workloads

Why $S \gg R$?

Disk I/O Time Components

For each request, I/O time can be decomposed into:

- **Seek Time**: Move the head to the right track.
- **Rotational Latency**: Wait for the sector to rotate under the head.
- **Transfer Time**: Actually move data between disk and memory.

Workload Types

- **Sequential Access** (S MB/s):
  - Large, contiguous requests (e.g., scan a file from block x to $x + 1$MB).
  - Disk stays on nearby tracks; little seeking/rotation, mostly transfer.
- **Random Access** (R MB/s):
  - Many small requests to unrelated locations (e.g., 4KB at block 10, then 550000, then 20100...).
  - Most time is spent in seek + rotation; little time in transfer.

In practice, sequential throughput S is **much larger** than random throughput R (i.e., $S \gg R$).

Example: Computing S and R

A simple disk exercise

Assume disk parameters: seek 7 ms, rotation 3 ms, transfer rate 50 MB/s.

Sequential throughput S

Request size: 10 MB

- Seek 7 ms, rotation 3 ms
- Transfer $10\,\text{MB}/50\,\text{MB/s} = 0.2\,\text{s} = 200\,\text{ms}$

Total time $\approx 7 + 3 + 200 = 210\,\text{ms}$

$S = 10\,\text{MB}/210\,\text{ms} \approx 47.6\,\text{MB/s}$.

Most time is spent transferring data $\Rightarrow$ S is near peak bandwidth.

Random throughput R

Request size: 10 KB

- Seek 7 ms, rotation 3 ms
- Transfer $10\,\text{KB}/50\,\text{MB/s} \approx 0.195\,\text{ms}$

Total time $\approx 7 + 3 + 0.195 \approx 10.195\,\text{ms}$

$R = 10\,\text{KB}/10.195\,\text{ms} \approx 0.98\,\text{MB/s}$.

Seek + rotation dominate $\Rightarrow$ $R < 1\,\text{MB/s}$, and $S/R \approx 50$.

Takeaways: Large sequential requests amortize seek/rotation and achieve high S; tiny random requests are dominated by positioning time, so R is tiny and $S \gg R$.

Back to RAID-0 Analysis

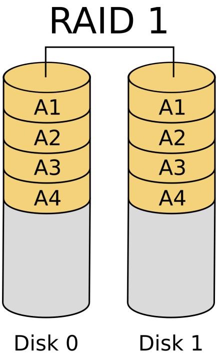
Putting S and R to use

- **Latency (single request)**
  - For a single-block logical I/O, RAID-0 just sends the request to one physical disk.
  - So latency is **about the same** as a single disk (seek + rotation + transfer on one disk).
- **Steady-state throughput**
  - **Sequential ($N \cdot S$ MB/s)**: use all N disks in parallel for large sequential requests.
  - **Random ($N \cdot R$ MB/s)**: with many independent random I/Os, all disks can still be busy.
- **Upper bound for comparison**
  - These simple formulas ($N \cdot S$ and $N \cdot R$) are easy to reason about.
  - They serve as an **upper bound** when we compare to other RAID levels later.

RAID Level 1: Mirroring

The expensive safety net

- **Layout**: Keep **two copies** of each logical block on **different** disks; disks form mirrored pairs (0–1, 2–3, ...), and data is **striped across pairs**.
- **Read/Write**: Reads can go to **either** copy of a block; writes must update **both** copies but can be issued in **parallel**.



Simple RAID-1 layout

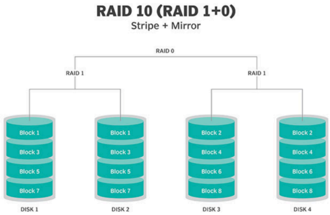
Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

Mirrored pairs across disks

RAID-10 vs. RAID-01

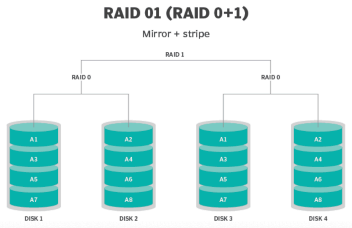
Stripe of mirrors vs. mirror of stripes

RAID-10 (RAID 1+0)



- Structure: combination of RAID 1 and RAID 0 → first **mirror** disks, then **stripe across mirrors**.
- Pros: RAID 0 performance + RAID 1 reliability.
- Cons: many disks (high cost), low space efficiency.

RAID-01 (RAID 0+1)



- Structure: combination of RAID 0 and RAID 1 → build RAID-0 stripes first, then **mirror the stripes**.
- Pros: RAID 0 performance + RAID 1 redundancy between stripes.
- Cons: **weaker** fault tolerance (any disk in a stripe kills that stripe) and low space efficiency.

RAID 1 Analysis (I)

Capacity, reliability, and latency

- **Capacity**: Mirroring level $= 2 \Rightarrow$ only half of raw space is useful.
  - With N disks of B blocks, useful capacity is $\dfrac{N \cdot B}{2}$.
- **Reliability**:
  - Can **tolerate 1 disk failure for sure** (per mirror pair).
  - In some lucky cases, can tolerate up to $N/2$ failures (depending on which disks fail).
  - In practice, we assume it is "good for handling a single failure" and don't rely on luck.
- **Single-request latency**:
  - **Read**: Same as a single disk. RAID-1 just directs the read to one copy.
  - **Write**: Needs two physical writes (to both copies) in parallel.
  - Worst-case seek/rotation of the two disks dominates $\Rightarrow$ slightly worse than a single-disk write.

RAID 1 Analysis (II)

Steady-state throughput with S and R

- **Sequential workloads**:
  - Each logical write $\Rightarrow$ two physical writes (to both disks in a mirror pair).
  - Maximum sequential write bandwidth: $\frac{N}{2} \cdot S$ MB/s (half of peak).
  - Sequential reads get the **same** bandwidth: still only $\frac{N}{2} \cdot S$ MB/s.
    - Intuition: each disk only serves every other block (0,4,8,...) $\Rightarrow$ each disk delivers only half of its peak.
- **Random workloads**:
  - Random reads: best case for RAID-1.
  - We can distribute requests across all N disks $\Rightarrow$ throughput $\approx N \cdot R$ MB/s.
  - Random writes: each logical write $\Rightarrow$ two physical writes.
  - Effective random-write bandwidth $\approx \frac{N}{2} \cdot R$ MB/s (half of what striping achieved).

Aside: The RAID Consistent-Update Problem

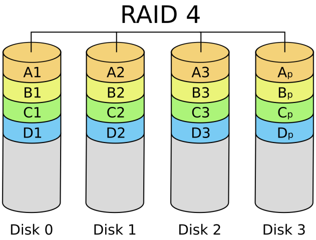
Making multi-disk writes atomic

- **Problem** (mirrored RAID): one logical write must update two disks. If power fails after Disk 0 is written but before Disk 1, the two copies become **inconsistent**.
- **Goal**: make multi-disk updates **atomic** – either both copies are updated, or neither is.
- **Solution**: use a **write-ahead log** in non-volatile storage.
  - Log what the RAID is about to do, then perform the disk writes.
  - After a crash, a **recovery** procedure replays or rolls back logged operations so mirrors end up consistent.

RAID Level 4: Saving Space With Parity

Structure and basic idea

- **Goal**: Add redundancy like RAID-1, but use **less capacity**.
- **Design**: Data is striped across $N-1$ disks; the last disk stores a **parity block** for each stripe.
- **Parity idea**: For each stripe, compute 1 parity block P from data blocks using XOR.



RAID-4 layout: $N-1$ data disks + 1 parity disk

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

Example: one parity block per stripe ($P0, P1, \ldots$)

RAID Level 4: Parity Math

How parity and recovery work

- **Bit-level invariant**: For each row (data + parity), XOR of all bits is 0: $C0 \oplus C1 \oplus C2 \oplus C3 \oplus P = 0$.
- **Reconstruction**: If one disk/column (e.g., C2) is lost, read the others + P and XOR them; the result is exactly the missing value (same XOR we used to create parity).
- **Blocks**: Real disks store 4KB blocks, so RAID just does this XOR bitwise across blocks: for each bit position, XOR data bits and write the result into the parity block.

C0	C1	C2	C3	P
0	0	1	1	XOR(0,0,1,1) = 0
0	1	0	0	XOR(0,1,0,0) = 1

Block0	Block1	Block2	Block3	Parity
00	10	11	10	11
10	01	00	01	10

RAID 4 Analysis (I)

Capacity and reliability

- **Capacity**: RAID-4 uses 1 disk for parity per RAID group.
  - With N disks of B blocks, useful capacity is $(N-1) \cdot B$.
- **Reliability**:
  - Can tolerate **1 disk failure** in the group (any single disk).
  - If more than one disk in the group is lost, we cannot reconstruct the data.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

RAID 4 Analysis (II)

Steady-state throughput: reads and full-stripe writes

- **Sequential reads**:
  - All $N-1$ data disks can stream data in parallel (parity disk mostly idle).
  - Peak sequential read bandwidth: $(N-1) \cdot S$ MB/s.
- **Sequential writes**: implemented as **full-stripe writes** (e.g., blocks 0,1,2,3 written together).
  - RAID computes new parity (e.g., $P0$) as XOR of all new data blocks in the stripe.
  - Then writes all N blocks (data + parity) in parallel.
  - Effective sequential write bandwidth: also $(N-1) \cdot S$ MB/s.
- **Random reads**:
  - One-block random reads are spread across the $N-1$ data disks (not the parity disk).
  - Effective random-read throughput: $(N-1) \cdot R$ MB/s.

RAID 4 Analysis (III)

Random writes and parity update

- **Random writes**: overwrite a single data block in a stripe.
  - Must update both the data block and the parity block in that stripe.
- **Additive vs. subtractive parity** (bit view):
  - Additive: read all other data blocks in stripe, XOR with new data to form new parity (many reads when N is large).
  - **Subtractive**: use old data + new data + old parity:

$$P_{new} = (C_{old} \oplus C_{new}) \oplus P_{old}$$

  - Same formula applies bitwise across an entire block (4KB, etc.).

RAID 4 Analysis (IV)

Small-write I/O cost and parity bottleneck

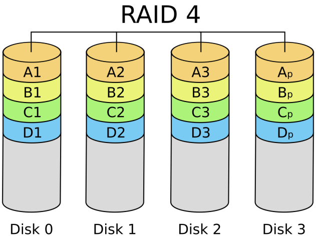
- **I/O cost for small random writes** (using subtractive parity):
  - Each logical write $\Rightarrow$ 2 reads (old data, old parity) + 2 writes (new data, new parity).
  - Parity disk does 2 I/Os per logical write (1 read, 1 write) and becomes a **bottleneck**.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
*4	5	6	7	+P1
8	9	10	11	P2
12	*13	14	15	+P3

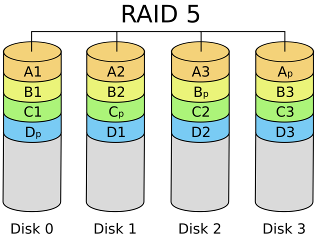
RAID Level 5: Rotating Parity

From RAID-4 to RAID-5

- **Motivation**: RAID-4 small writes are limited by a single parity disk (bottleneck at $R/2$ MB/s).
- **Key idea**: RAID-5 works like RAID-4, but **rotates the parity block across disks** for each stripe.
- **Effect**: spreads parity updates over all disks, removing the dedicated parity-disk bottleneck.



RAID-4: fixed parity disk



RAID-5: parity blocks rotated across disks

RAID Level 5: Rotating Parity

Fixing the RAID 4 Bottleneck

- **Solution**: Don't use a dedicated parity disk. **Rotate** the parity block assignment across all disks.
- **Impact**:
  - The "Parity Update" load is distributed evenly.
  - Random Write performance improves significantly because multiple stripes can be written in parallel.
- **Comparison**:
  - Same capacity as RAID 4: $(N-1) \times B$.
  - Same reliability: Tolerates 1 failure.
  - **Much better random write concurrency.**

RAID-5 Analysis

How it compares to RAID-4

- **Capacity & reliability**: Same as RAID-4.
  - Effective capacity: $(N-1) \cdot B$; can tolerate 1 disk failure.
- **Sequential performance**:
  - Sequential read/write bandwidth and single-request latency are essentially the same as RAID-4.
- **Random reads**:
  - Slightly better than RAID-4, since all disks (including those holding parity for some stripes) can serve data for other stripes.
- **Random writes** (small writes):
  - Still need 4 I/Os per logical write (2 reads + 2 writes), but parity work is **spread across disks**.
  - With many independent writes, we can keep all N disks roughly busy.
  - Effective small-write bandwidth $\approx \frac{N}{4} \cdot R$ MB/s – factor 1/4 comes from 4 I/Os per logical write.

RAID Levels in Practice

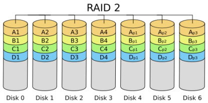
Choosing the “right” level

	RAID-0	RAID-1	RAID-4	RAID-5
Capacity	$N \cdot B$	$(N \cdot B)/2$	$(N-1) \cdot B$	$(N-1) \cdot B$
Reliability	0	1 (for sure) $\frac{N}{2}$ (if lucky)	1	1
Throughput				
Sequential Read	$N \cdot S$	$(N/2) \cdot S^1$	$(N-1) \cdot S$	$(N-1) \cdot S$
Sequential Write	$N \cdot S$	$(N/2) \cdot S^1$	$(N-1) \cdot S$	$(N-1) \cdot S$
Random Read	$N \cdot R$	$N \cdot R$	$(N-1) \cdot R$	$N \cdot R$
Random Write	$N \cdot R$	$(N/2) \cdot R$	$\frac{1}{2} \cdot R$	$\frac{N}{4} R$
Latency				
Read	T	T	T	T
Write	T	T	$2T$	$2T$

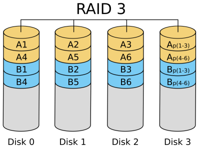
In practice, **different RAID levels** are chosen based on **workload, capacity, reliability, and cost trade-offs**, rather than a **single “best” option**.

Other RAID Levels

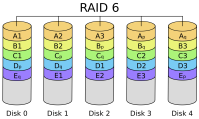
A quick structural tour



RAID-2:
Uses bit-level striping with multiple **Hamming-code parity** disks;
rarely used in practice.



RAID-3:
Byte/bit-level striping with a **single shared parity** disk;
good for large sequential I/O.



RAID-6:
Like RAID-5 but with **two independent parity blocks** per stripe;
tolerates 2 disk failures.

Other Interesting RAID Issues

And a quick Part 1 wrap-up

- **Failure handling**: Hot spares, rebuild performance, and what happens while a disk is rebuilding.
- **More realistic faults**: Latent sector errors or block corruption, and techniques to detect/correct them.
- **Software RAID**: Implementing RAID in software can be cheaper but has its own issues (e.g., consistent-update).

Part 1 takeaway: RAID gives us powerful tools to trade off **capacity**, **reliability**, and **performance**, but picking the *right* level and parameters for a real workload is still **more of an art than a science**.

Part 2 — The Case for RAID (from “A Case for Redundant Arrays of Inexpensive Disks”)

Abstract

- **CPU & Memory Performance** increase.
- **I/O Performance**: If not matched, the performance gains of CPUs and memories will be wasted.
- **SLED (Single Large Expensive Disk)**:
  - Capacity has grown rapidly.
  - Performance improvement has been modest.
- **RAID (Redundant Arrays of Inexpensive Disks)**:
  - Based on magnetic disk technology developed for personal computers.
  - Offers an attractive alternative to SLED.
  - Promises improvements of an order of magnitude in:
    - Performance
    - Reliability
    - Power consumption
    - Scalability

Growth of Computing and CPU Performance

- Gordon Bell's observation: 40% performance growth in single-chip computers per year (1974-1984).
- Bill Joy's prediction of even faster growth in 1985.
- The formula for MIPS growth:

$$\mathrm{MIPS} = 2^{\mathrm{Year}-1984}$$

- Amdahl: Each CPU instruction per second requires one byte of main memory

Memory Growth

- Moore's Law: Transistor count doubles approximately every two years.
- Memory capacity has quadrupled every 2-3 years.
- The ratio of memory to MIPS is known as α, with Amdahl's constant meaning $\alpha = 1$.
- Due to falling memory prices, the α ratio has risen above 1 in modern systems.

The Pending I/O Crisis

- Amdahl's Law:

$$S = \frac{1}{(1-f) + \frac{f}{k}}$$

  where:

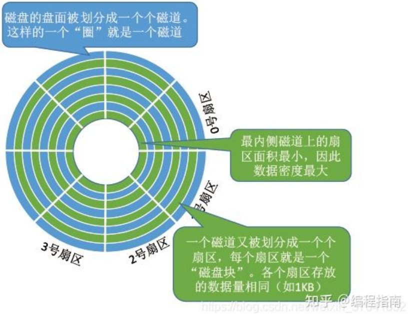
  - $S =$ effective speedup,
  - $f =$ fraction of work in faster mode,
  - $k =$ speedup in the faster mode.

Secondary Storage and Disk Technology

- The growth of secondary storage must match CPU and memory improvements.
- MAD (Maximal Areal Density): the maximum number of bits that can be stored per square inch, or the bits per inch in a track times the number of tracks per inch
- $$\text{MAD} = 10^{(\text{Year}-1971)/10}$$
- Growth in disk capacity and corresponding price reduction.

SLED Limitations

- The performance of SLED is dominated by **seek time** and **rotation latency**:
  - From 1971 to 1981, the raw seek time for a high-end IBM disk improved by only a factor of two.
  - Rotation latency remained unchanged.



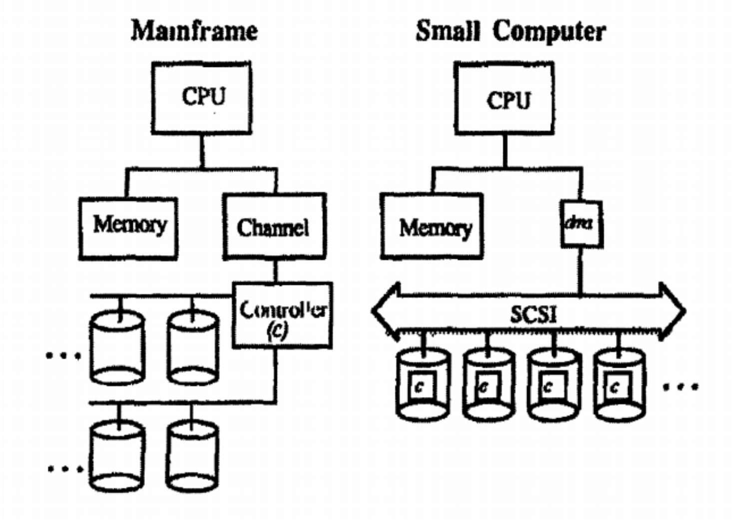
A Solution: Arrays of Inexpensive Disks

Inexpensive disks achieve similar I/O performance per actuator as large disks.

Characteristics	*IBM 3380*	*Fujitsu M2361A*	*Conners CP3100*	*3380 v 3100 (>1 means 3100 is better)*	*2361 v 3100*
Disk diameter (inches)	14	10 5	3 5	4	3
Formatted Data Capacity (MB)	7500	600	100	01	2
Price/MB(controller incl)	\$18-\$10	\$20-\$17	\$10-\$7	1-2 5	1 7-3
MTTF Rated (hours)	30,000	20,000	30,000	1	1 5
MTTF in practice (hours)	100,000	?	?	?	?
No Actuators	4	1	1	2	1
Maximum I/O's/second/Actuator	50	40	30	6	8
Typical I/O's/second/Actuator	30	24	20	7	8
Maximum I/O's/second/box	200	40	30	2	8
Typical I/O's/second/box	120	24	20	2	8
Transfer Rate (MB/sec)	3	2 5	1	3	4
Power/box (W)	6,600	640	10	660	64
Volume (cu ft)	24	3 4	03	800	110

A Solution: Arrays of Inexpensive Disks

SCSI make these disks highly cost-effective.



The Bad News: Reliability

- MTTF (Mean Time To Failure) The average time a disk operates before failure.
- Assuming the failures of disks are independent and follow an exponential distribution.
- Since the failure rate of the array is the sum of the failure rates of individual disks:

$$\lambda_{\text{array}} = N \times \lambda_{\text{disk}}$$

- Since failure rate and MTTF are inversely proportional, the MTTF of the array is:

$$\text{MTTF of Array} = \frac{\text{MTTF of Single Disk}}{\text{Number of Disks in Array}}$$

The Bad News: Reliability

1. **Exponential CDF (Single Disk Failure Probability within t)**

$$F(t) = P(X \leq t) = 1 - e^{-\lambda t}$$

2. **Single Disk Reliability (Probability of Working within t)**

$$P_{\text{single}}(t) = P(X > t) = e^{-\lambda t}$$

3. **Total Reliability of N-Disk Array**

$$P_{\text{array}}(t) = \prod_{i=1}^{N} P_{\text{single},i}(t) = e^{-N\lambda t}$$

4. **Total Failure Rate of Array**

$$\lambda_{\text{array}} = N\lambda$$

RAID MTTF Calculation

$$MTTF_{RAID} = \frac{MTTF_{Disk}}{G+C} * \frac{MTTF_{Disk}}{(G+C-1)*MTTR} * \frac{1}{n_G}$$

$$= \frac{(MTTF_{Disk})^2}{(G+C)*n_G * (G+C-1)*MTTR}$$

$$MTTF_{RAID} = \frac{(MTTF_{Disk})^2}{(D+C*n_G)*(G+C-1)*MTTR}$$

Reliability Overhead Cost and Usable Storage Capacity Percentage

Reliability Overhead Cost: This is the extra check disks, expressed as a percentage of the number of data disks D.
Usable Storage Capacity Percentage: The percentage of total capacity used for data storage.

Performance in Supercomputers and Transaction-Processing Systems

Supercomputers:

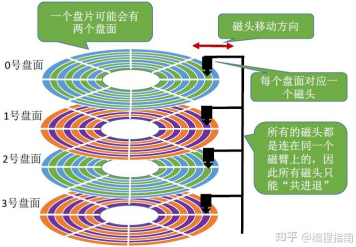
- The number of reads and writes per second for large blocks of data.

Transaction-Processing Systems:

- The number of individual reads or writes per second.

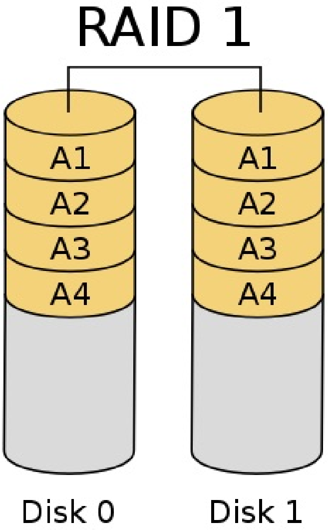
Summary:

- Supercomputers: High data rate.
- Transaction-processing: High I/O rate.



RAID 1: Mirrored Disks

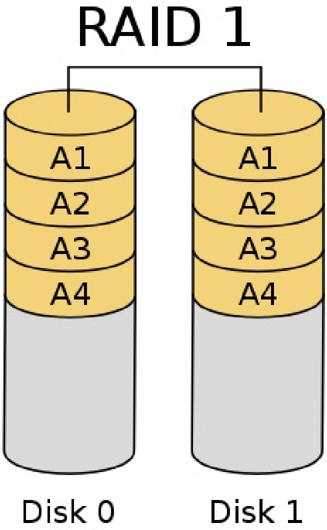
- Each data disk has an identical mirror, ensuring redundancy.
- Every write to a data disk is also written to the check disk.
- Provides high fault tolerance but requires double the disk capacity.



RAID 1: Mirrored Disks

Tandem's optimization allows for:

- Parallel reads from both disks.
- Multiple controllers enabling optimized reading without waiting for the second write to complete.

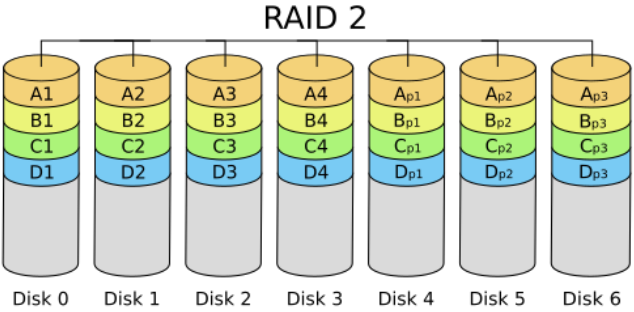


Slowdown Factor: A slowdown factor (S) is introduced when more than two disks are involved in the group.

- $1 < S < 2$, depending on the number of disks and the parallelism in use.

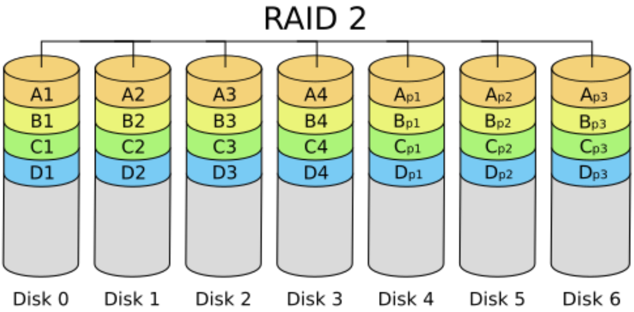
RAID 2: Hamming Code for ECC

- It uses bit-interleaving and adds check disks to detect and correct errors.
- A single parity disk detects errors, but multiple check disks are required to correct errors.
- For 10 data disks (G), 4 check disks (C) are required; for 25 data disks, 5 check disks.



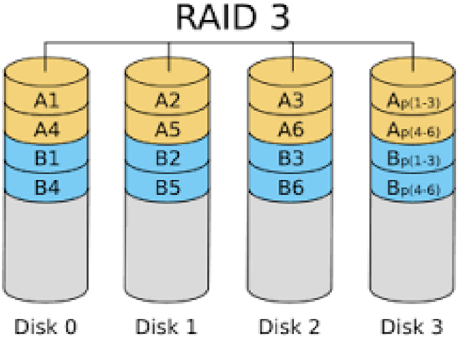
RAID 2: Hamming Code for ECC

- Each data transfer unit is a sector.
- Bit-interleaved disks mean a large transfer requires at least G sectors.
- For small data transfers, reads and writes still imply reading the full sector from each bit-interleaved disk.
- Writes involve the read-modify-write (R-M-W) cycle across all disks in the group.



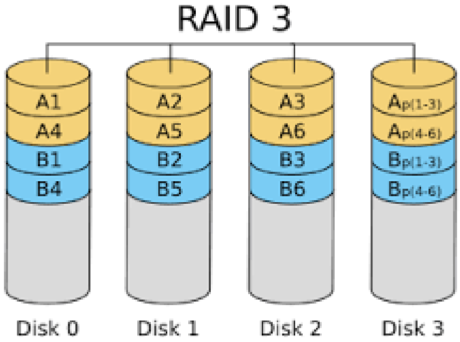
Reducing Check Disks in RAID 3

- Reducing check disks to one per group (C = 1).
- The performance of RAID 3 is the same as Level 2 RAID.
- The reduction in total disks increases reliability.



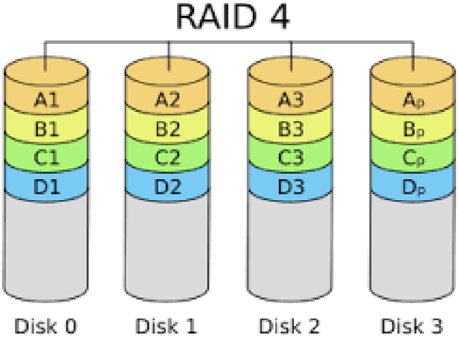
Error Reconstruction

- **Error Detection:** Special signals provided in the disk interface or extra checking information at the end of a sector to detect and correct soft errors.
- **Error Correction:** Information on the failed disk can be reconstructed by calculating the parity of the remaining good disks.



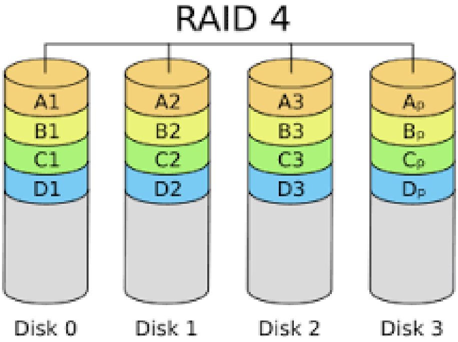
RAID Level 4 - Independent Reads/Writes

- Unlike RAID 3, data is not spread across multiple disks at the bit level. Instead, data is stored in sectors, and each disk can handle its individual transfer.
- Reads can be performed independently on a single disk at maximum disk rate, while still detecting errors.

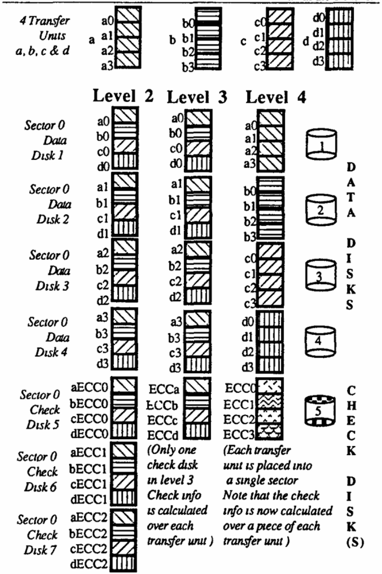


RAID 4 Performance and Bottleneck

- Small writes improve since only 2 disks are involved in reading and writing the data and parity.
- The check disk is critical in limiting the number of simultaneous writes to the RAID system.

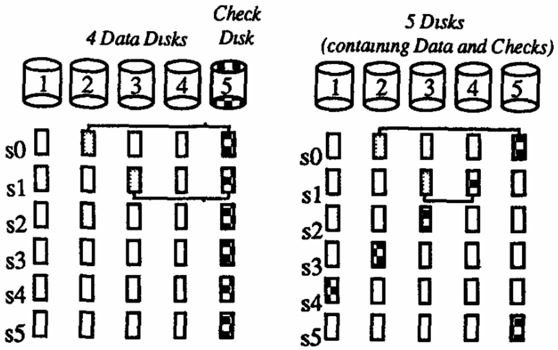


Comparison of level 2,3,4



RAID 5 Check Information Placement

- In RAID 4, check information is stored in a single check disk.
- In RAID 5, check information is distributed across all disks, including data disks.

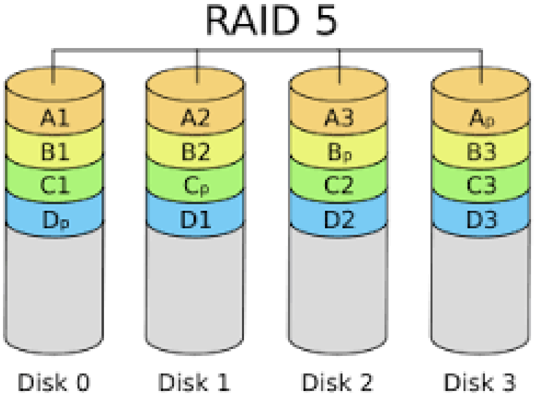


(a) Check information for Level 4 RAID for G=4 and C=1 The sectors are shown below the disks (The checked areas indicate the check information) Writes

(b) Check information for Level 5 RAID for G=4 and C=1 The sectors are shown below the disks, with the check information and data spread evenly through all the

RAID 5 Parallel Write Advantage

- RAID 5 supports multiple independent writes per group.
- In RAID 4, writing to sectors on different disks requires sequential writes to the check disk.
- In RAID 5, writes to different sectors can occur in parallel, significantly improving write performance.



RAID 5 vs SLED Comparison

- **Performance:** RAID 5 provides up to 10x better performance compared to IBM 3380, and 5x better than Fujitsu M2361A.
- **Reliability:** RAID 5 has significantly higher MTTF due to redundancy and data reconstruction.
- **Power Consumption and Size:** RAID 5 is more energy-efficient and smaller in size, reducing cooling costs.
- **Storage Capacity and Expandability:** RAID 5 offers better storage utilization and modular expansion compared to SLED.
- **Advantages:** RAID 5 is ideal for supercomputer applications, transaction processing, and systems with limited storage.

Challenges: Data Corruption

(Based on: "An Analysis of Data Corruption in the Storage Stack", FAST '08)

Data Integrity Segment (DIS)

Building the Foundation for End-to-End Protection

1. **Data Structure:**
   - Every **4KB** data block is mandatorily attached with **64 bytes** of metadata.

2. **Physical Implementation (Layout):**

Enterprise Class (FC)

- Uses **520-byte** sectors.
- $8 \times 520 = 4160$ bytes.
- *Result:* Perfect embedding, zero performance overhead.

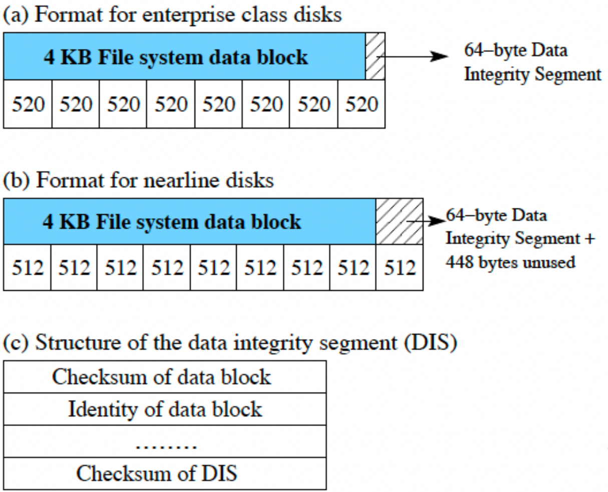
Nearline Class (SATA)

- Uses standard **512-byte** sectors.
- *Result:* Requires consuming extra sector space for DIS.

3. **DIS Components:**
   - **Checksum:** Verifies physical bits.
   - **Identity:** Verifies logical context (Inode, Offset).

Data Integrity Segment (DIS)

Building the Foundation for End-to-End Protection



Real-Time Defense: Detection on Read

1. Checksum Mismatch (CM)
   - **Layer:** RAID / Adapter.
   - **Logic:** $Hash(Data) \neq Stored$.
   - **Target:** Physical media corruption, bit flips.

2. Identity Discrepancy (ID)
   - **Layer:** File System.
   - **Logic:** Checksum is OK, but DIS Inode $\neq$ Request.
   - **Target:** Lost Writes, Misdirected Writes.

Recovery Action:
Failure in *either* check $\rightarrow$ Trigger **RAID Reconstruction**.

Data Scrubbing

Scrubbing Workflow:

1. **Verify Integrity:** Check Checksums of all blocks (finds latent CMs).
2. **Verify Logic:** Check RAID Parity Consistency.

Unique Error: Parity Inconsistency (PI)

- **Phenomenon:** All Data Checksums are OK, but $Data + Data \neq Parity$.
- **Causes:** Memory corruption, Software bugs, non-atomic updates.
- **Resolution:** Rewrite Parity.

Core Challenge: Silent Corruption

Fail-Stop vs. Silent Corruption

Traditional RAID (Fail-Stop)

- Disk Fails
- **Error Reported**
- RAID Rebuilds

Reality (Silent Corruption)

- Disk Fails (Bit flip)
- **NO Error Reported**
- RAID propagates bad data

Type 1: Checksum Mismatches (CMs)

Definition

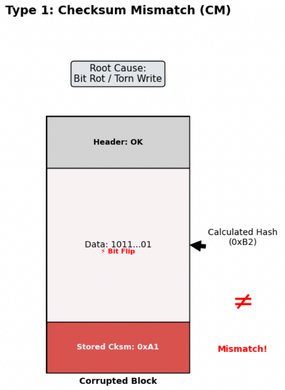
Data is read, hash is calculated, but **Calculated Checksum $\neq$ Stored Checksum**.

Root Causes:

- **Bit-level Corruption:** Media aging, head interference, bit rot.
- **Torn Writes:** Power failure during write (partial data written).
- **Misdirected Writes:** Overwriting good data at wrong location.

Detection Mechanism:

- Detected during **ANY RAID Read** (User Read, Reconstruction, or Scrubbing).



Type 2: Identity Discrepancies (IDs)

Definition

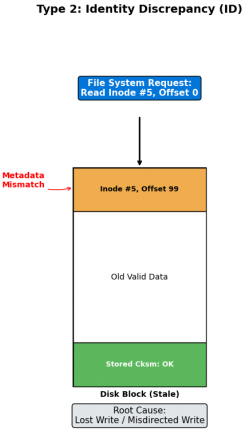
The Checksum is **Valid** (Physical bits are intact), but the block's metadata (Inode/Offset) does not match the File System request.

Root Causes:

- **Lost Writes:** Disk reports "Success" but data was never written. System reads back *old, valid data*.
- **Misdirected Writes:** Data written to the wrong address.

Detection Mechanism:

- Only detected during a **File System Read**.



Type 3: Parity Inconsistencies (PIs)

Definition

All Data Block Checksums are **Valid**, but:

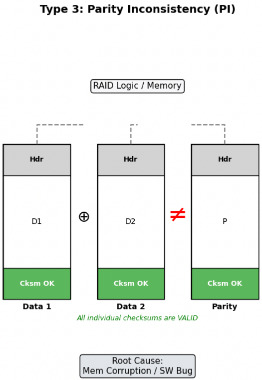
$$\text{Data}_1 \oplus \text{Data}_2 \oplus \ldots \neq \text{Parity}$$

Root Causes:

- **Memory Corruption:** Bit flips in RAID controller cache.
- **Software Bugs:** Errors in RAID algo.
- **Non-atomic Updates:** Power loss between Data write & Parity update.

Detection Mechanism:

- Only detected during **Data Scrubbing** (Background verification).



Challenge I: Prevalence Challenge

Observation Data:

- Total Checksum Mismatches: **400,000+**
- SATA (Nearline) error rate is **10x** higher than FC (Enterprise).

RAID Challenge 1: The Normalized Threat

- *RAID Assumption:* Data corruption is treated as a **rare and exceptional anomaly**.
- *Study Conclusion:* Data corruption is actually a **frequent and persistent operational threat**.
- **Impact:** RAID must evolve from an occasional recovery mechanism to an all-weather defense system.

Challenge II: Reconstruction Hazard

Observation Data:

- **8%** of errors were discovered during **RAID Reconstruction**.

Logical Deduction:

- Reconstruction = System's most vulnerable moment (Redundancy Lost).
- Error during Reconstruction = **Double Failure**.

RAID Challenge 2: Single Parity Failure

- RAID 5 cannot handle this 8% scenario.
- **Result:** Permanent Data Loss.

Challenge III: Spatial Locality

Observation: Distribution of Errors

Observation Data:

- Errors tend to cluster in **consecutive physical blocks**.
- Specific disk models fail at **specific block addresses**.

RAID Challenge 3: Layout Vulnerability

- Traditional RAID uses **Aligned Stripes** (Same offset across disks).
- **Risk:** A firmware bug targeting a specific address can corrupt the *entire stripe* (Data + Parity) simultaneously.
- **Need:** Staggered Striping layouts.

Challenge IV: System Correlation

Testing the Independence Assumption

Hypothesis Testing:

- H_0: Disk failures are independent.
- **Result:** Reject H_0. Failures are highly correlated.

Evidence:

- Extreme case: **92 disks** in a single system failed simultaneously.

RAID Challenge 4: The Independence Deficit

- **Cause:** Shared hardware (Controllers/Adapters) leads to common-mode failures.
- **Impact:** RAID groups must be distributed across physical failure domains.

Lessons Learned I: Protection & Redundancy

1. The Value of Checksums
   - **Observation:** Corruption is a reality, not a theory. We observed **400,000+** checksum mismatches. Up to **4%** of specific drive models developed errors within 17 months.
   - **Conclusion:** The protection provided by **Checksums and Block Identity** is fully worth the extra storage space required.

2. The Reconstruction Hazard
   - **Observation:** A significant portion (**8%**) of corruption is detected during RAID reconstruction.
   - **Implication:**
     - Protection against **Double Disk Failures** (e.g., RAID 6) is necessary to prevent data loss.
     - **Aggressive Scrubbing** is crucial to detect errors *before* a rebuild is needed.

Lessons Learned II: Policy & Layout

3. Enterprise Drive Replacement Policy

- **Observation:** Enterprise drives have a low probability of initial corruption, but once they fail, they are highly likely to fail again.
- **Strategy: "One Strike Policy"** It makes sense to **replace Enterprise drives immediately** upon the first detected corruption. Since the initial probability is low, the total replacement cost remains manageable.

4. Physical Layout Design

- **Observation:** Certain block numbers are more prone to corruption than others (due to firmware/hardware bugs affecting specific addresses).
- **Design Requirement: Staggered Striping** RAID designers should ensure stripe blocks are **NOT** stored at the same or nearby physical block numbers across disks.

Thank You!