I/O

CISC 3310 — CUNY Brooklyn College
Lecture Notes

I/O

In this chapter, we will:

1. Appreciate the need for input/output (I/O) operations in a computer.
2. Learn about various I/O handling methods.
3. Understand the difference between character I/O and block I/O.
4. Introduce the idea of RAID: Redundant Arrays of Inexpensive Disks and its purpose.

I/O

So far, we learned that a computer can calculate things and make decisions, and store information, both in short term and long term fashion.

What is missing in this whole picture? Aha! We haven't talked about how the computer would communicate all the results it computed with users, which is the topic of I/O: input and output. In fact, without I/O, all those computations that a computer makes would be quite useless if we, as users, don't get to benefit from them.

Just like the CPU is a device used for computations, and RAM, cache, and disks are storage devices, computers also rely on I/O devices, most of which we use daily: a mouse, a keyboard, a screen/display, speakers, printers, scanners, etc. We learned back in Topic 1 that I/O devices split into two groups: input devices vs. output devices.

Making I/O devices work in a timely manner is critical to the efficiency of the computer: even when the CPU computes things fast, it would be a slow I/O device or slow communication between the computer and the I/O device that would 'ruin' the experience of using a computer. Just imagine playing a game, with each mouse click processed only 1 second later: ouch.

I/O Handling Methods: Program-Controlled I/O

To what extent is the CPU involved in transferring data into and out of the computer? Here are 4 methods:

1. **Program-Controlled I/O** (also called **Polling I/O**) is when the CPU runs a single program in which it continuously **polls** (= checks) a special control register on each I/O device that is connected to the computer to see if that I/O device is ready to transfer data. This is literally the only action the CPU does in its free time: it doesn't run any other programs.

   Only when data becomes available, the CPU performs the transfer, and then returns to that busy-waiting state.

   Pros: We can easily change or add more I/O devices to the computer just by changing a few code lines in the program.
   Cons: The CPU is busy waiting or just doing I/O and can't run any other program (needless to say multiple programs.)

   One example of a device that might use a program-controlled I/O is a simple vending machine whose sole purpose is to wait for customers to purchase something and dispense the goods when the customer enters the right amount of money.

I/O Handling Methods: Interrupt-Driven I/O

2. In a computer that uses **Interrupt-Driven I/O**, the CPU can perform other activities while waiting for I/O events.

   An **interrupt** is a signal that a hardware device issues to the CPU. A device could issue an interrupt in case of an error (such as dividing by 0), but also to indicate that an I/O event is happening.

   When the CPU is interrupted, it pauses the execution of whichever program it currently runs, and instead executes a function called **service routine** whose purpose is to handle the interrupt. After the service routine returns, the CPU resumes running code from the point where it paused.

   Since each interrupt must be handled differently (depending on the interrupt type,) the service routine uses an array called the **interrupt vector** that is stored in memory in order to find and call a **handler**, which is a program that handles a specific type of an interrupt. Each array index contains an address to a handler program.

I/O Handling Methods: Interrupt-Driven I/O

2. The interrupt mechanism goes as follows:

    a. A **device controller**, which is a piece of hardware in the device whose purpose is to control the device's I/O signals, sends a signal to the CPU via a wire called an **interrupt-request line**.
    b. In-between the execution of a program's instructions, the CPU notices that the controller issued a signal, and reads the interrupt number (which indicates what the interrupt type is.)
    c. The CPU uses this number as an index into the interrupt vector to call the appropriate interrupt handler.
    d. The handler first saves the current execution state, which includes the values inside the registers, back to main memory (so that the CPU could return executing code after the interrupt is handled,) finds the reason for the interrupt, and performs all the actions needed to resolve the interrupt.
    e. The CPU then restores the saved execution state from RAM and continues running the instructions of the program that was running before the interrupt occurred.

I/O Handling Methods: Interrupt-Driven I/O

2. Here are various Intel CPU interrupt types and their corresponding numbers in the interrupt vector:

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Intel interrupt types and the interrupt vector. Taken from Bell, John T. "I/O Systems." University of Illinois, Chicago.

I/O Handling Methods: Interrupt-Driven I/O

2. Most CPUs have two types of interrupt request lines: a nonmaskable one and a maskable one, each of which is used to transfer signals for two types of interrupts.

   A **nonmaskable** interrupt is one that a CPU cannot ignore, and must take measures in order to handle the interrupt. Events that could issue a nonmaskable interrupt are usually unrecoverable memory errors, such as division by zero.

   A **maskable** interrupt might be ignored by the CPU when critical OS code that must not be interrupted is executed.

   While executing such critical code, the CPU would turn off the maskable interrupt request lines, so that no interrupt signals are registered while such code is running.

   In the diagram on the previous slide, all I/O-related interrupts have numbers ranging from 32 to 255. This is because all the described interrupts in numbers 0 to 31 are nonmaskable, while I/O interrupts are maskable and can be ignored by the CPU when needed.

I/O Handling Methods: Interrupt-Driven I/O

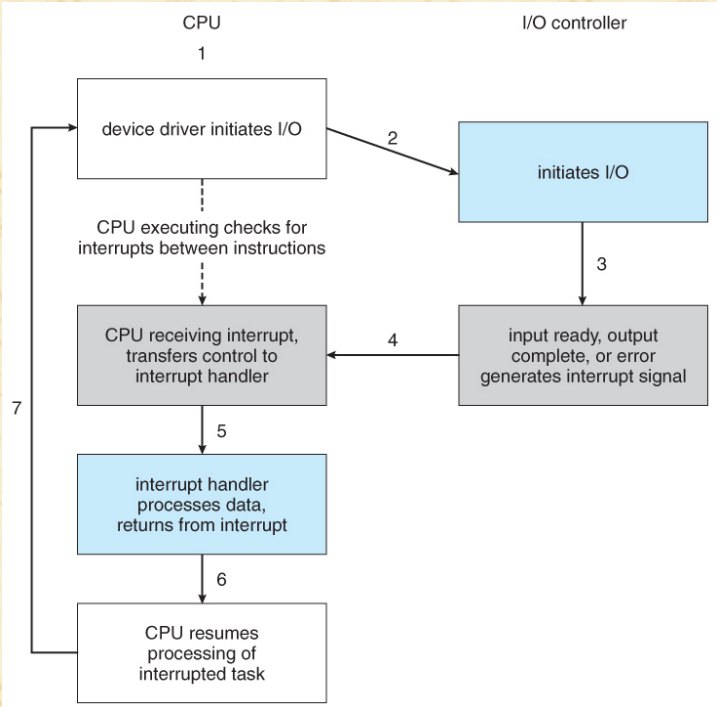
2. As we said from before, with an **interrupt-driven I/O** approach, the CPU can perform other activities while waiting for the I/O device. After every instruction that the CPU executes, it will check if a signal came via one of its interrupt-request lines. If not, the CPU will continue to execute the next instruction. If yes, the service routine, which the CPU executes, will run an interrupt handler which will handle the I/O operation. We say that:
    - A device's controller ***raises*** an interrupt.
    - The CPU ***catches*** the interrupt and ***dispatches*** the interrupt handler.
    - The interrupt handler ***clears*** the interrupt by servicing the device.

    Pros: The CPU isn't limited anymore to running just one firmware: it will run other programs until an interrupt occurs.
    Cons: Still, frequent I/O requests can reduce the efficiency since the CPU is responsible for handing all the I/O activities.

    The flow chart on the next slide illustrates what occurs during an interrupt.

I/O Handling Methods: Interrupt-Driven I/O

2.



Interrupt-driven I/O cycle. Taken from Bell, John T. "I/O Systems." University of Illinois, Chicago.

I/O Handling Methods

Any questions?

I/O Handling Methods: DMA I/O

3. A computer using **Direct Memory Access I/O** introduces an additional processor (CPU-like device) called a **direct-memory access controller** or **DMA controller** whose sole purpose is to manage I/O data.

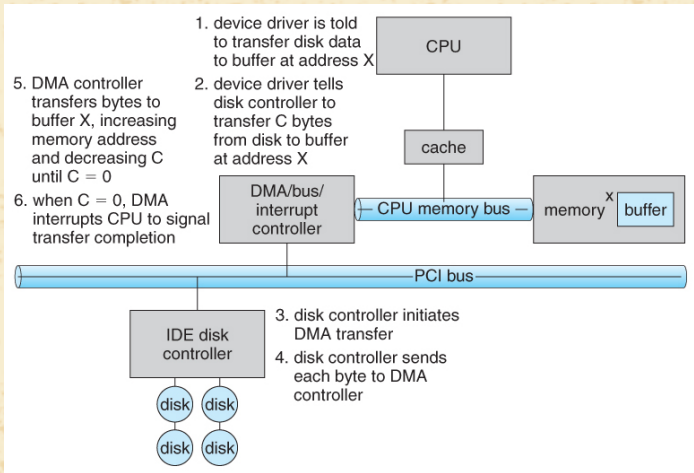
   This is done in order to free the CPU from reading all the data from the input device or writing all the data to an output device, which is what the CPU does during program-controlled and interrupt-based I/O. Here is the process:

   a. The main CPU issues a command to the DMA controller with the following info:
      1. The ID or number of the device to which data should be transferred,
      2. The location (address) to the beginning of the data within memory, and
      3. The total number of bytes to be transferred.

      Note that (2) and (3) tell precisely what chunk of data should be transferred.
   b. The DMA controller proceeds to handle the I/O, while the main CPU continues running other programs.
   c. When the DMA finishes its work, it raises an interrupt to inform the CPU that the data transfer is complete.

I/O Handling Methods: DMA I/O

3. **Pros**: When no errors or interrupts happen, the CPU is not responsible for anything that has to do with handing I/O besides deligating I/O jobs to the DMA and noticing when I/O activity occurs/ends. This lets the CPU concentrate its power to running instructions of other programs to an even greater extent compared to interrupt driven I/O.
   Cons: The DMA chip is useful only for casual I/O fetching and writing, but when I/O-related failures happen, only the CPU is responsible for fixing them and resuming normal activity, which reduces the CPU's performance.



Steps in a DMA transfer. Taken from Bell, John T. "I/O Systems." University of Illinois, Chicago.

I/O Handling Methods: Channel I/O

4. I/O Processors are small and smart CPU-like devices that control channel paths connected to I/O devices.

   They are similar to a smart DMA that can perform complex I/O-related activities, such as calculations and jumps, which means that they can resolve various I/O errors if such happen during an I/O event. This method is called **Channel I/O**.

   The main CPU still needs to dispatch the job to an I/O processor, specifically, by telling where the code that the processor needs to run is located in RAM, but that's it: the CPU isn't involved anymore in any part of an I/O event.

   Pros: The essence of the I/O Processor(s) grants independence from the CPU in terms of handling various events, thus freeing the CPU for doing other, more meaningful activities.
   Cons: Building such a complex system with several small CPUs has financial costs (more time, expensive materials, and smarter specialists are needed for constructing this kind of computer.)

I/O Handling Methods

Any questions?

Block I/O Devices vs. Character I/O Devices

We can further divide computer devices into the following two categories: **Block Devices** and **Character Devices**.

- This division is about the amount of data that the computer can transfer to or from a device at once.
- **Block** devices can be accessed one block at a time. A **block** is a fixed size of data, usually standing at a multiple of 512 bytes. Block sizes of 4k (= 4096) and 8k (= 8192) bytes are common.
  - The blocks we are talking about here are different from cache blocks. The RAM's page size, on the other hand, is usually a multiple of a block size.
  - Examples of block devices include hard disk drives, solid-state drives, and USB flash drives.
- **Character** devices can be accessed one byte at a time (remember that 1 byte can store exactly one ASCII character or a small integer between 0-255.)
  - Examples of character devices include the mouse, the keyboard, and the monitor. CPU, RAM, and cache are character devices, too, because we can access a single byte on each of these.
- As a general rule, every non-storage peripheral device is a character device.

Block I/O Devices vs. Character I/O Devices

Any questions?

RAID

RAID stands for Redundant Array of Independent Disks.

This concept was invented by David A. Patterson, Garth A. Gibson, and Randy H. Katz in 1987 for the sake of improving data reliability and performance by using multiple physical disks.

RAID relies on the concept of redundancy: storing multiple copies of data increases the chances that, if one of the copies is damaged, the other copy remains available. Data is distributed across the disks in specific configurations designed to enhance redundancy or speed.

A **level** in RAID defines a particular configuration for how data is stored and accessed on the disks: we'll talk about RAID levels 0, 1, 2, 3, 4, 5, 6, 10, 50, and DP in these lecture notes.

Each **level** offers a tradeoff between redundancy, performance, and storage capacity. Understanding these tradeoffs helps in choosing the appropriate **RAID** configuration for specific use cases.

RAID: Level 0

RAID 0 is configured by striping data across multiple disks without any redundancy. **Striping** is dividing files into chunks and writing these chunks to different disks in parallel.

While this configuration excels in read and write speeds, it provides no fault tolerance; a single disk failure results in complete data loss.

Since there is no space reserved for redundancy, all available disk capacity is used for storage.

Although RAID 0 is cost-effective in terms of storage efficiency, it is unsuitable for critical systems where data reliability is essential. Typically, this level is used for tasks requiring high throughput and low latency, such as video editing, gaming, or scientific simulations where data can be regenerated if lost.

It is also often employed as a temporary workspace for non-persistent data, where speed outweighs risk.

RAID: Level 0

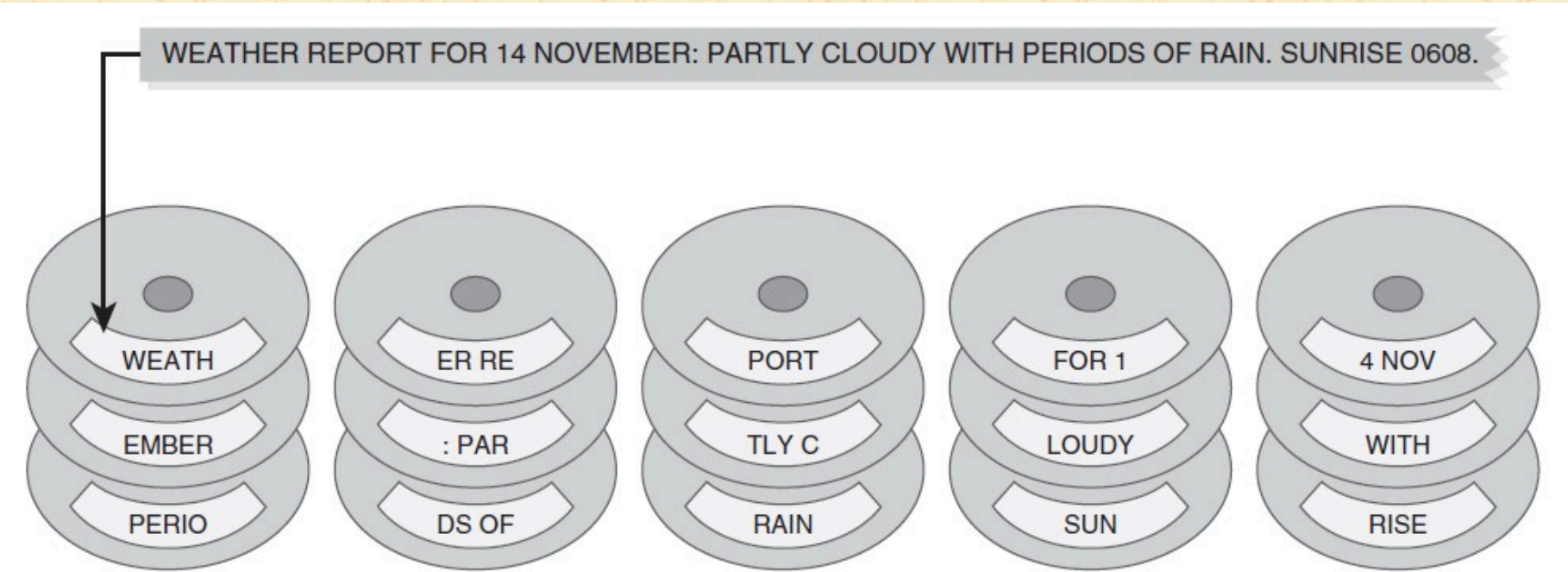


FIGURE 7.19 A Record Written Using RAID-0, Block Interleave Data Striping with No Redundancy

A RAID Level 0 disk configuration. Figure 7.19 on page 459 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 1

RAID 1, referred to as **mirroring**, duplicates data across two or more disks, ensuring that each disk contains an identical copy.

This redundancy allows uninterrupted operation even if one disk fails, as the data remains accessible from the mirror.

Although read speeds may improve since multiple disks can be accessed simultaneously, write speeds are not significantly enhanced due to the need to write the same data to all disks.

Since half the total storage is reserved for redundancy, the usable capacity is effectively halved.

While this configuration offers excellent fault tolerance, it may not be cost-efficient for systems requiring high storage capacity.

RAID 1 is particularly suitable for applications where reliability is critical, such as transactional databases, backup systems, and small business servers. Moreover, its simplicity makes it an attractive choice for users who value ease of maintenance and data protection over performance optimization.

RAID: Level 1

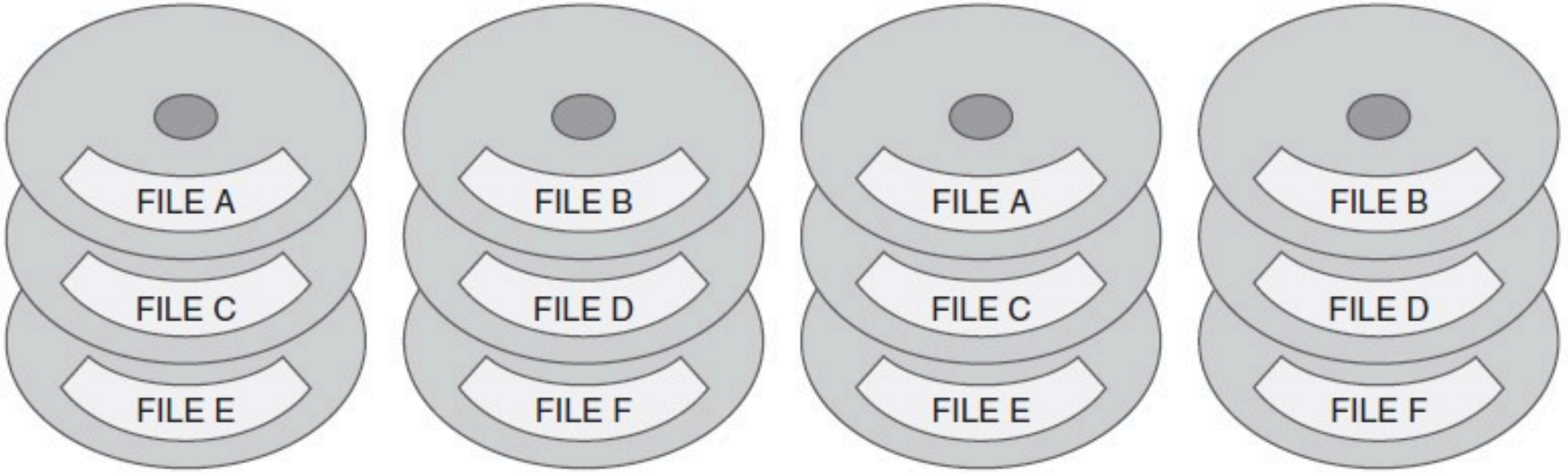


FIGURE 7.20 RAID-1: Disk Mirroring

A RAID Level 1 disk configuration. Figure 7.20 on page 460 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 2

RAID 2 is an uncommon configuration that utilizes bit-level striping in conjunction with dedicated disks for storing **error correction codes** (**ECC**). Since it requires multiple dedicated disks for ECC, it provides high fault tolerance but offers minimal storage efficiency.

This level ensures data reliability by detecting and correcting errors during reads, although its complexity often outweighs its benefits in modern applications.

Unlike other levels, RAID 2 relies on synchronized spinning disks, which increases its hardware requirements and costs.

Although its error-correcting capabilities were revolutionary at the time of its invention, advancements in disk technology have rendered this level largely obsolete.

RAID 2 was historically significant for scientific and industrial applications, where data accuracy was prioritized over cost or simplicity. However, due to its limited practicality, nowadays it has been largely replaced by more efficient configurations.

RAID: Level 2

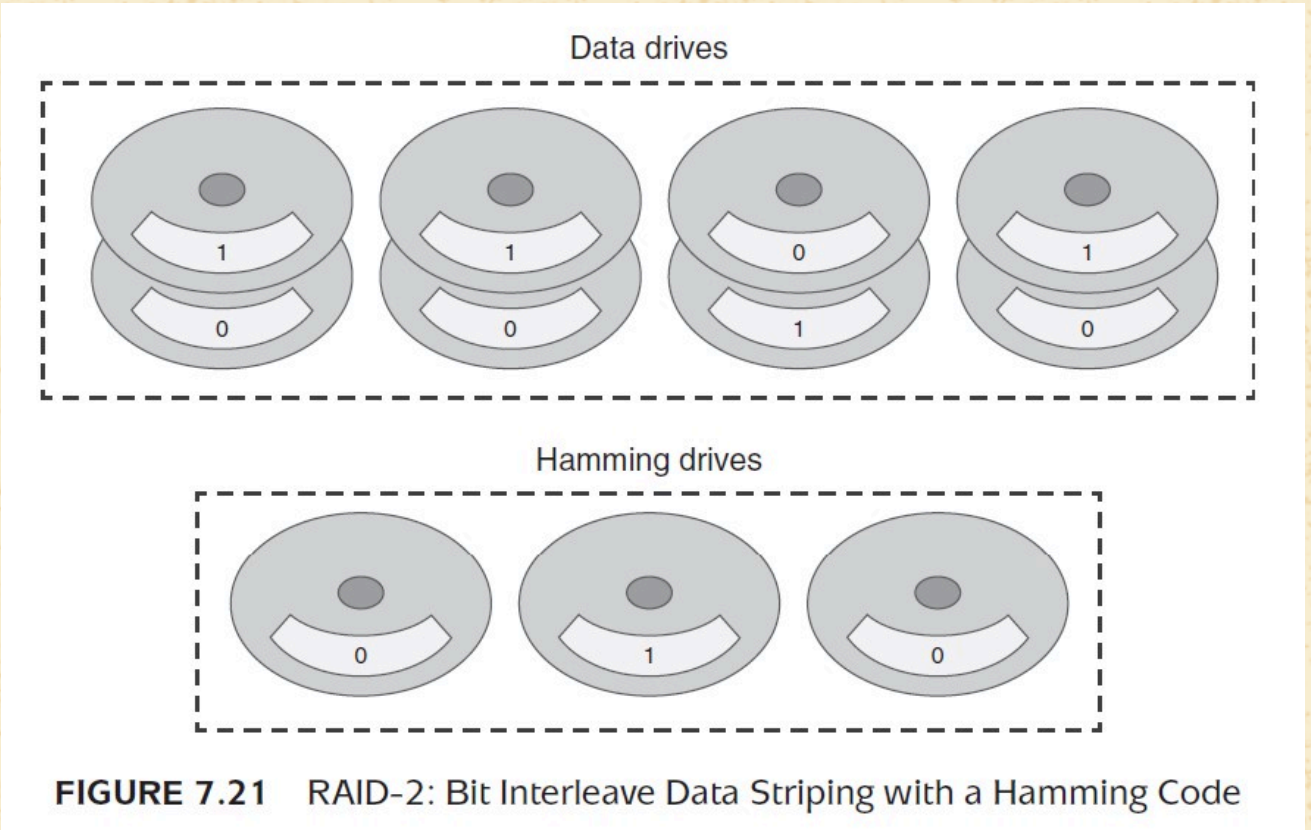


FIGURE 7.21 RAID-2: Bit Interleave Data Striping with a Hamming Code

A RAID Level 2 disk configuration. Figure 7.21 on page 461 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 3

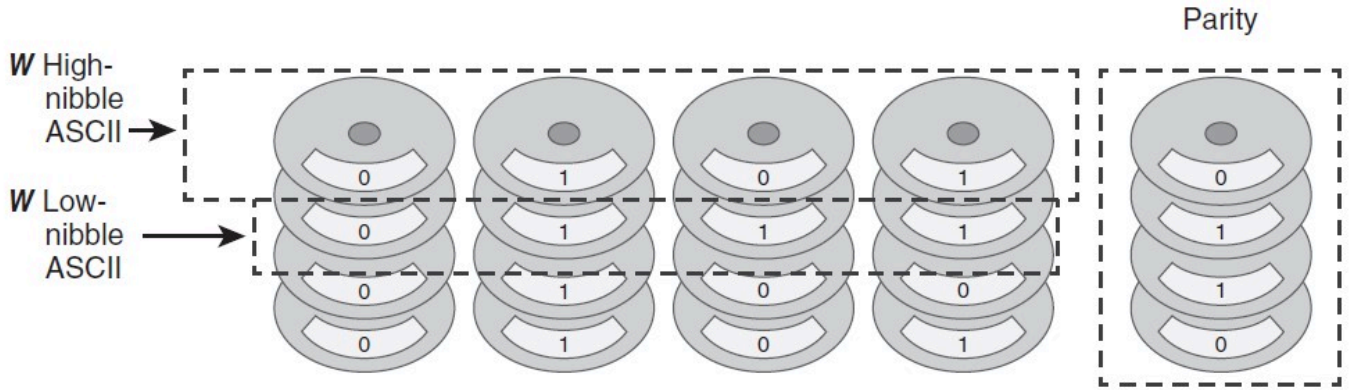
RAID 3 combines byte-level striping with a single dedicated parity disk to ensure redundancy. Data is divided into small units, which are spread across multiple disks, while the parity disk stores information necessary for reconstructing lost data in case of a failure. A **parity bit** tells whether the sum of the 1 bits in some data is even or odd (e.g., the parity of 100101 is 1 = odd.)

Performance is excellent due to parallel access: since only one disk is used for parity, the remaining disks contribute fully to storage capacity, making RAID 3 more efficient than mirrored configurations like RAID 1.

This level is particularly suitable for applications requiring sequential data access, such as video or audio streaming, where write speed is less critical. However, the reliance on a single parity disk creates a potential point of failure, reducing its overall reliability compared to more advanced levels.

RAID 3 is rarely used today due to its limitations and the availability of more robust alternatives.

RAID: Level 3



Letter	ASCII	Parity(even)	
		High Nibble	Low Nibble
W	0101 0111	0	1
E	0100 0101	1	0
A	0100 0001	1	1
T	0101 0100	0	1
H	0100 1000	1	1
E	0100 0101	1	0
R	0101 0010	0	1

FIGURE 7.22 RAID-3: Bit Interleave Data Striping with Parity Disk

A RAID Level 3 disk configuration. Figure 7.22 on page 462 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 4

RAID 4 employs block-level striping and a dedicated parity disk to provide redundancy and performance improvements.

Unlike RAID 3, data is written in blocks, which allows for more efficient random access, making it suitable for mixed workloads.

The parity disk stores recovery data, enabling the system to rebuild information from a failed disk, but this configuration also risks a bottleneck during write operations.

Since all parity calculations are directed to a single disk, it may become a performance limitation under heavy write-intensive workloads.

This level is ideal for archival systems or environments with more reads than writes, such as document storage systems.

While offering better random access than RAID 3, RAID 4 is less commonly implemented due to the bottleneck introduced by its parity disk. Nevertheless, it laid the foundation for more advanced configurations like RAID 5 and RAID 6.

RAID: Level 4

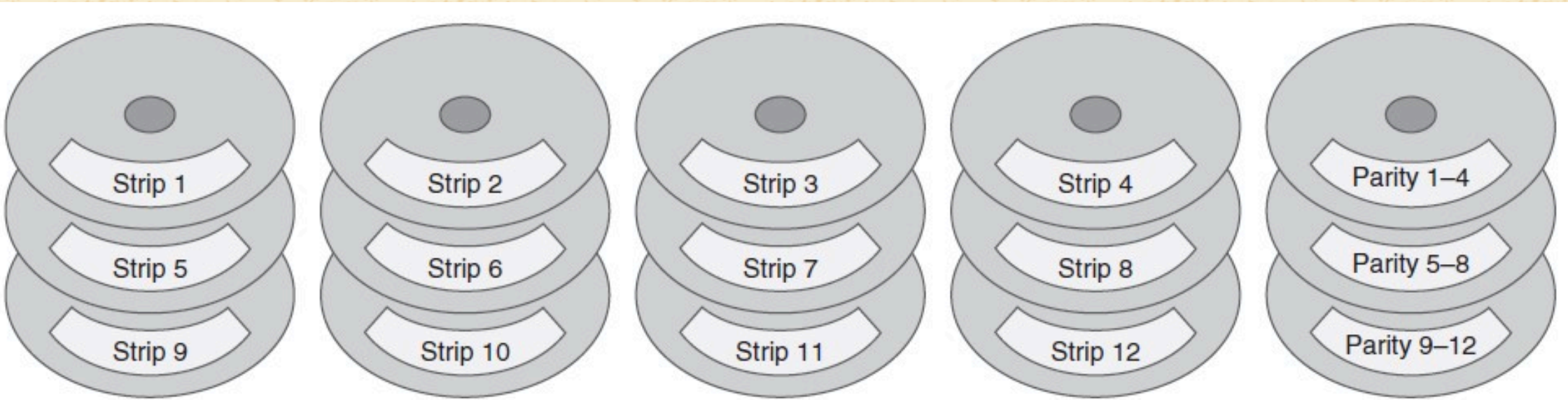


FIGURE 7.23 RAID-4: Block Interleave Data Striping with One Parity Disk

A RAID Level 4 disk configuration. Figure 7.23 on page 463 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 5

RAID 5 uses block-level striping combined with distributed parity to achieve a balance of performance, redundancy, and storage efficiency.

Parity information is distributed across all disks in the array, ensuring that no single disk acts as a bottleneck or point of failure. This configuration allows the system to recover from the failure of one disk by using the parity data on the remaining disks.

Although read operations are efficient due to parallel access to multiple disks, write performance is slightly reduced due to the parity calculations required.

RAID 5 requires at least 3 disks, and the total usable storage is equal to the sum of all disks minus the one parity disk.

It is a versatile and cost-effective option, making it one of the most widely used configurations in environments like file servers, database systems, and small-to-medium business storage solutions. Still, rebuilding the array after a disk failure can be time-consuming, especially with large-capacity disks, which may increase the risk of additional failures during recovery.

RAID: Level 5

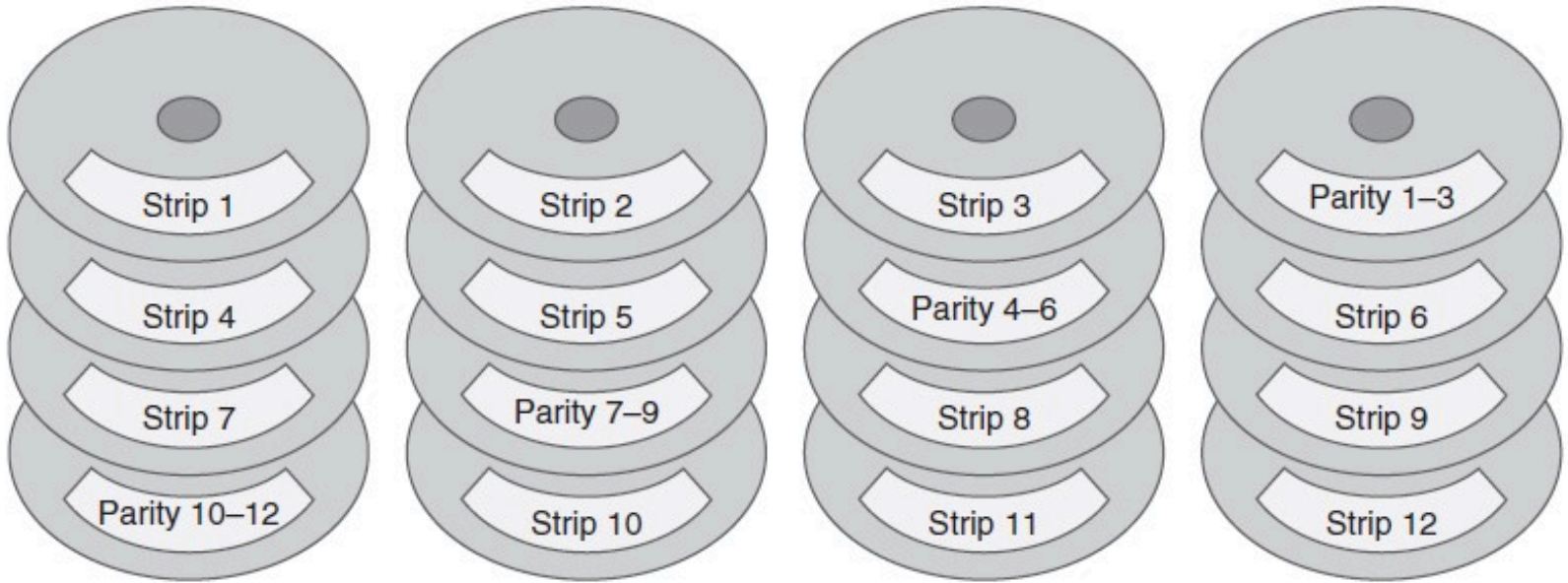


FIGURE 7.24 RAID-5: Block Interleave Data Striping with Distributed Parity

A RAID Level 5 disk configuration. Figure 7.24 on page 464 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 6

RAID 6 extends RAID 5 by adding an additional layer of redundancy through a second parity block, allowing for the failure of two disks simultaneously.

Like RAID 5, data and parity are distributed across all disks, but the second parity calculation increases the complexity and time required for writes. This enhanced fault tolerance makes RAID 6 a reliable choice for critical systems, especially those with large arrays of high-capacity disks where multiple failures are more likely.

The configuration requires at least 4 disks, and the usable capacity is the total capacity of all disks minus two parity disks. Although write performance is slower than RAID 5, read speeds remain high, and the additional parity makes it highly reliable for systems that prioritize data integrity.

RAID 6 is often used in enterprise environments, such as large-scale storage systems, virtualization platforms, and mission-critical applications where downtime must be minimized. However, rebuilding a failed array can still be resource-intensive, and administrators should ensure proper hardware support for parity calculations.

RAID: Level 6

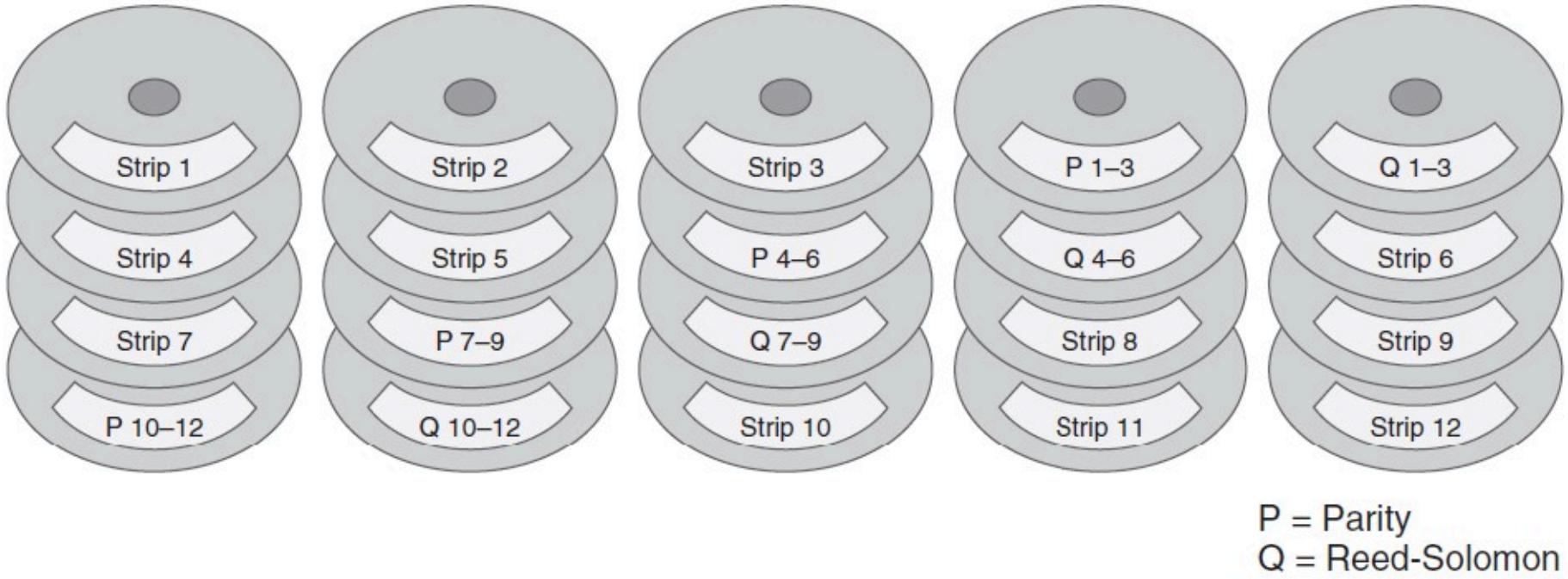


FIGURE 7.25 RAID-6: Block Interleave Data Striping with Dual-Error Protection

A RAID Level 6 disk configuration. Figure 7.25 on page 465 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: DP

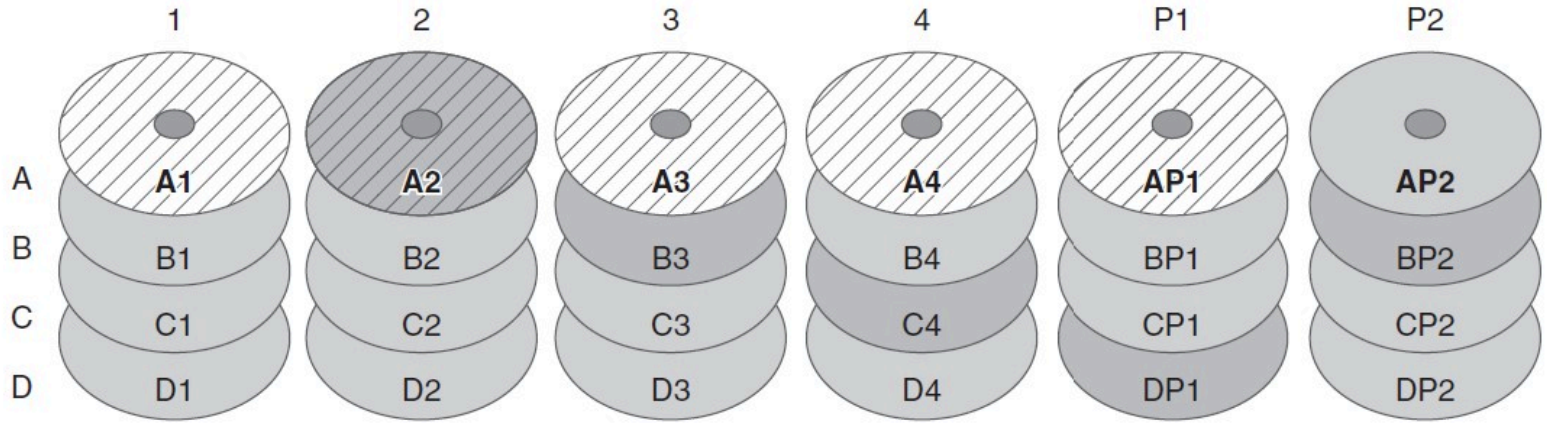
RAID DP (Dual Parity) offers advanced fault tolerance by using two independent parity calculations across all disks. Data and parity are distributed evenly, which balances read and write performance while maintaining redundancy.

This setup provides protection against the simultaneous failure of two disks, which is critical in high-capacity arrays where disk failures are more likely. While the dual parity calculations slightly reduce write speeds, the configuration ensures minimal data loss risk, even during disk rebuilds.

Unlike RAID 6, RAID DP is often optimized for specific hardware or software platforms, offering better performance in certain environments. Although it requires advanced hardware support and more complex management, its enhanced reliability makes it a preferred choice for critical applications with stringent performance demands.

RAID DP is ideal for enterprise storage systems, such as large-scale NAS or SAN solutions, where uptime and data protection are top priorities. Administrators should consider its cost and system requirements, as it often involves proprietary implementations that may limit compatibility.

RAID: DP



$$AP1 = A1 \oplus A2 \oplus A3 \oplus A4 \qquad AP2 = A1 \oplus B2 \oplus C3 \oplus D4$$
$$BP1 = B1 \oplus B2 \oplus B3 \oplus B4 \qquad BP2 = A2 \oplus B3 \oplus C4 \oplus DP1$$
$$CP1 = C1 \oplus C2 \oplus C3 \oplus C4 \qquad CP2 = A3 \oplus B4 \oplus CP1 \oplus D1$$
$$DP1 = D1 \oplus D2 \oplus D3 \oplus D4 \qquad DP2 = A4 \oplus BP1 \oplus C1 \oplus D2$$

FIGURE 7.26 Error Recovery Pattern for RAID DP
The recovery of A2 is provided by the overlap of equations AP1 and BP2.

A RAID DP disk configuration. Figure 7.26 on page 466 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: DP

Any questions?

RAID: Level 10

RAID 10, a hybrid configuration, combines the performance benefits of RAID 0 with the redundancy of RAID 1.

Data is first mirrored across pairs of disks and then striped across those mirrored sets, providing both speed and fault tolerance. If one disk in a mirrored pair fails, its twin maintains data integrity, while striping ensures high read and write performance.

This configuration requires at least four disks and reduces usable storage to 50% of the total disk capacity, as half is reserved for redundancy.

Although it is more expensive due to its high disk requirements, it offers superior rebuild times compared to parity-based levels, as only the failed disk within a mirror needs recovery.

RAID 10 is highly suitable for environments that demand both speed and reliability, such as high-performance database servers, transaction-heavy systems, or virtualization platforms. Its simplicity and efficiency make it a popular choice for critical systems, despite the higher cost per gigabyte.

RAID: Level 10

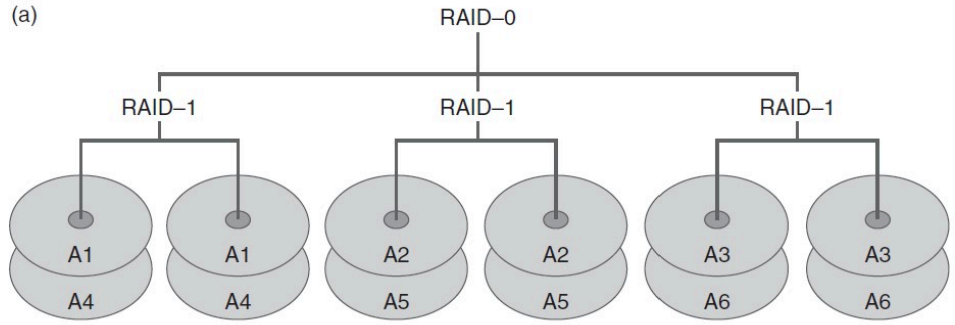


FIGURE 7.28 Hybrid RAID Levels
(a) RAID-10: Stripe of Mirrors
(b) RAID-01: Mirror of Stripes

A RAID Level 10 disk configuration. Figure 7.28 on page 468 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Level 50

RAID 50 combines the striping of RAID 0 with multiple RAID 5 arrays to deliver enhanced performance and fault tolerance.

The configuration stripes data across several RAID 5 groups, distributing parity within each group, which ensures data redundancy while improving speed.

This level requires at least 6 disks, and its fault tolerance depends on the ability of each RAID 5 group to survive a single disk failure. By leveraging the performance of striping and the redundancy of distributed parity, RAID 50 balances speed, reliability, and storage efficiency better than either level alone.

It is particularly suitable for large-scale applications, such as data centers and high-transaction environments, where both high throughput and fault tolerance are required. Nonetheless, rebuilding an array can still be resource-intensive, and the configuration is more complex to implement and manage compared to simpler levels.

Despite its higher cost and complexity, RAID 50 is favored for handling both random and sequential workloads efficiently.

RAID: Level 50

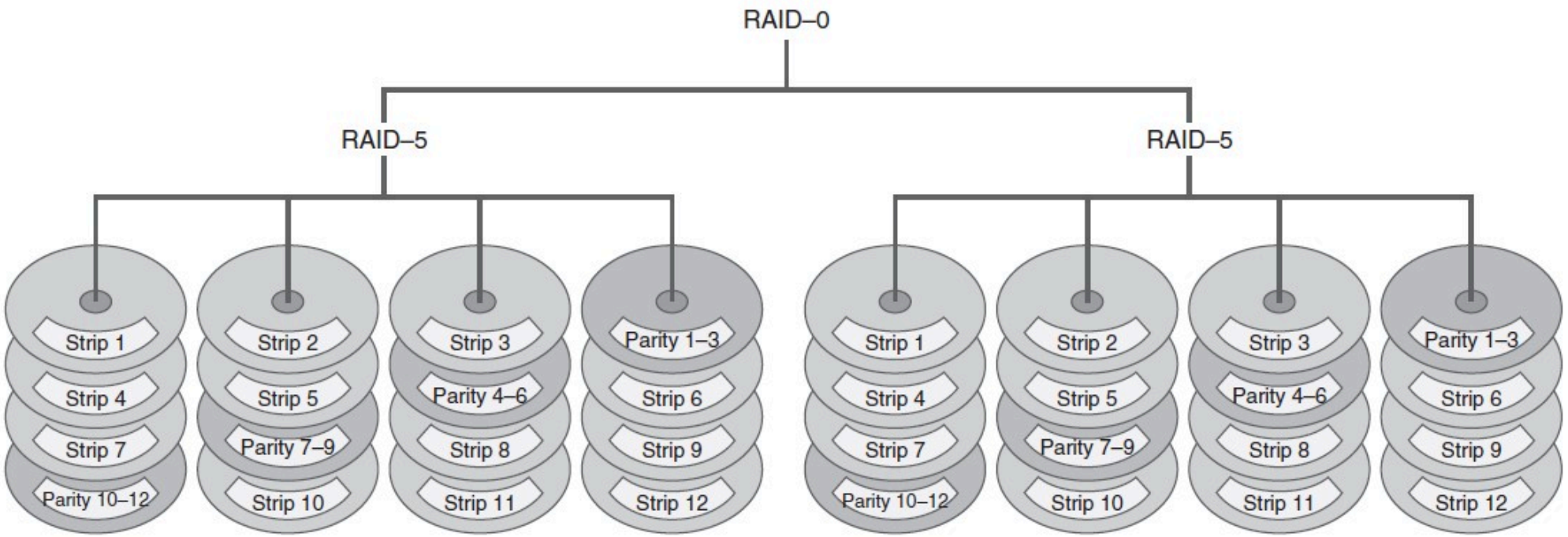


FIGURE 7.29 RAID-50: Striping and Parity

A RAID Level 50 disk configuration. Figure 7.29 on page 468 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Summary

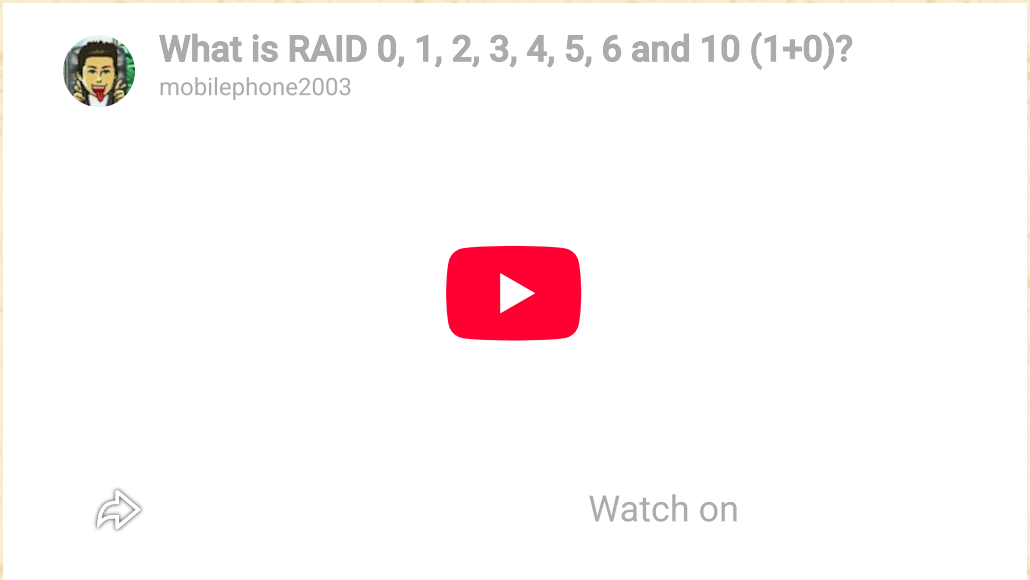
RAID Level	Description	Reliability	Throughput	Pros and Cons
0	Block interleave data striping	Worse than single disk	Very good	Least cost, no protection
1	Data mirrored on second identical set	Excellent	Better than single disk on reads, worse on writes	Excellent protection, high cost
2	Bit interleave data striping with Hamming code	Good	Very good	Good performance, high cost, not used in practice
3	Bit interleave data striping with parity disk	Good	Very good	Good performance, reasonable cost
4	Block interleave data striping with one parity disk	Very good	Much worse on writes as single disk, very good on reads	Reasonable cost, poor performance, not used in practice
5	Block interleave data striping with distributed parity	Very good	On writes not as good as single disk, very good on reads	Good performance, reasonable cost
6	Block interleave data striping with dual-error protection	Excellent	On writes much worse than single disk, very good on reads	Good performance, reasonable cost, complex to implement
10	Mirrored disk striping	Excellent	Better than single disk on reads, not as good as single disk on writes	Good performance, high cost, excellent protection
50	Parity with striping	Excellent	Excellent read performance. Better than RAID-5; not as good as RAID 10	Good performance, high cost, good protection
DP	Block interleave data striping with dual parity disks	Excellent	Better than single disk on reads, not as good as single disk on writes	Good performance, reasonable cost, excellent protection

TABLE 7.1 Summary of RAID Capabilities

RAID Levels: Summary. Table 7.1 on page 469 of *Essentials of Computer Organization and Architecture* by Linda Null, 2023.

RAID: Summary

Video explaining most of the RAID levels we covered:



https://www.youtube.com/watch?v=wTcxRObq738

RAID: Summary

Any questions?

I/O: Sources

CS301: Computer Architecture. Unit 7.1, Unit 7.2, and Unit 7.4. *Saylor Academy*, 2010. License: CC BY: Attribution.

Wienand, Ian. (2019). Section 3.3. *Computer Science from the Bottom Up*. License: CC BY-SA: Attribution-ShareAlike.