# UNIT-II

Data Protection, Intelligent Storage system: Implementation of RAID, RAID Array Components, RAID Levels, RAID Comparison, RAID Impact on Disk Performance, Hot Spares Components of an Intelligent Storage System, Intelligent Storage Array

Business-critical applications require high levels of performance, availability, security, and scalability.

A hard disk drive is a core element of storage that governs the performance of any storage system.

Some of the older disk array technologies could not overcome performance constraints due to the limitations of a hard disk and its mechanical components.

**RAID** technology made an important contribution to enhancing storage performance and reliability, but hard disk drives even with a RAID implementation could not meet performance requirements of today’s applications.

With advancements in technology, a new breed of storage solutions known as an intelligent storage system has evolved.

The intelligent storage systems detailed in this chapter are the feature-rich **RAID** arrays that provide highly optimized **I/O** processing capabilities.

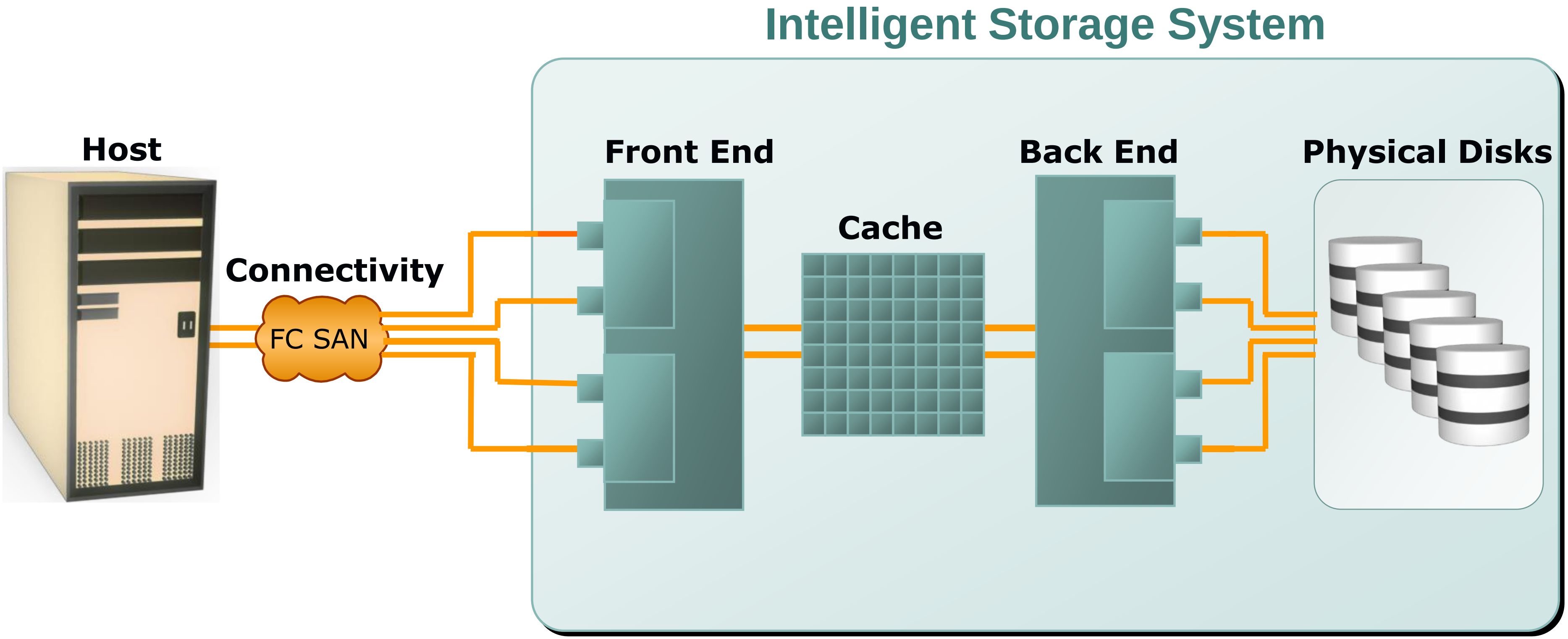
# Components of an Intelligent Storage System

An intelligent storage system consists of four key components:

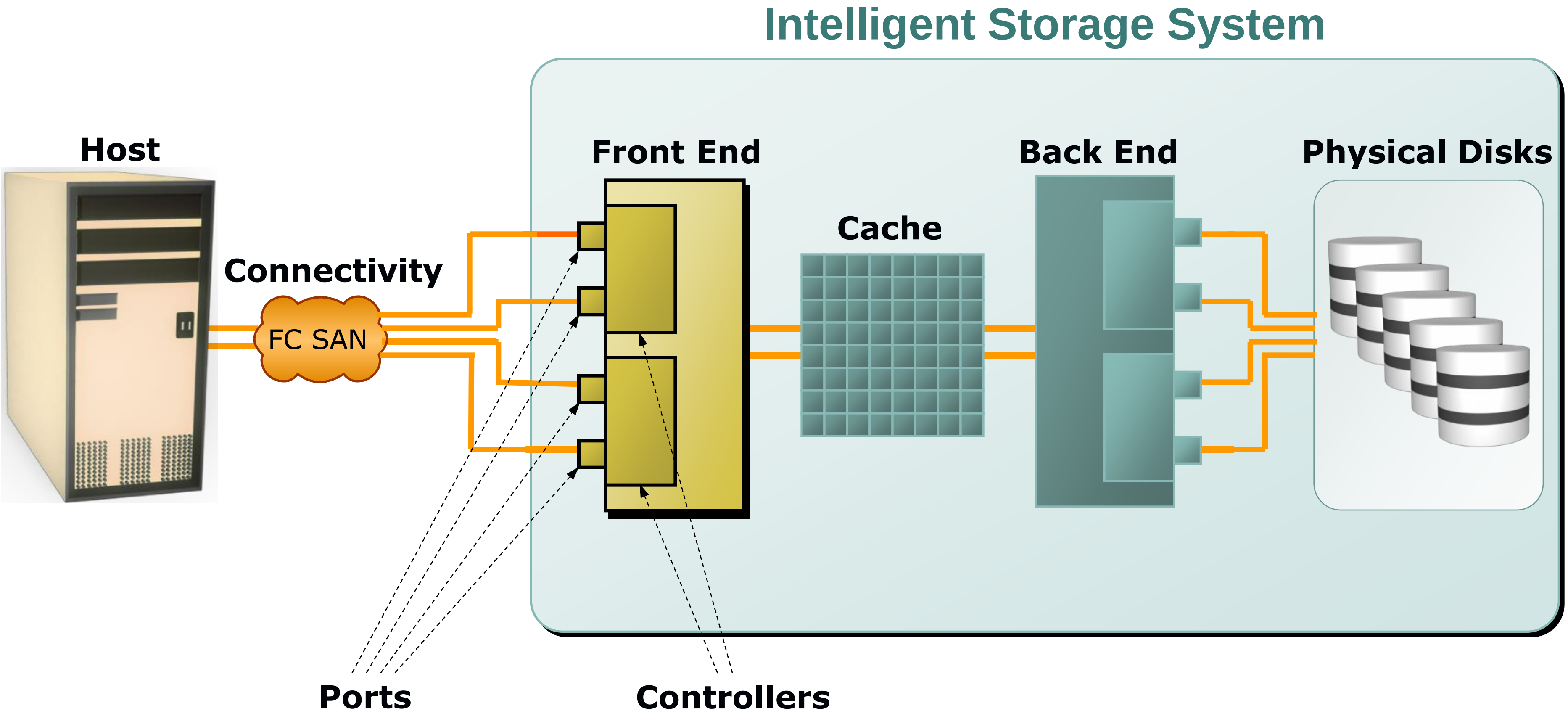
- **front end**
- **cache**
- **back end**
- **physical disks**

Figure 4-1 illustrates these components and their interconnections. An I/O request received from the host at the front-end port is processed through cache and the back end, to enable storage and retrieval of data from the physical disk. A read request can be serviced directly from cache if the requested data is found in cache.

# Components of an Intelligent Storage System



# Intelligent Storage System: Front End



# Front End

The front end provides the interface between the storage system and the host.

It consists of two components: front-end ports and front-end controllers.

The front-end ports enable hosts to connect to the intelligent storage system.

Each front-end port has processing logic that executes the appropriate transport protocol, such as SCSI, Fibre Channel, or iSCSI, for storage connections.

Redundant ports are provided on the front end for high availability.

**Front-end** controllers route data to and from cache via the internal data bus. When cache receives write data, the controller sends an acknowledgment message back to the host.

Controllers optimize I/O processing by using command queuing algorithms.

# Front-End Command Queuing

The most commonly used command queuing algorithms are as follows:

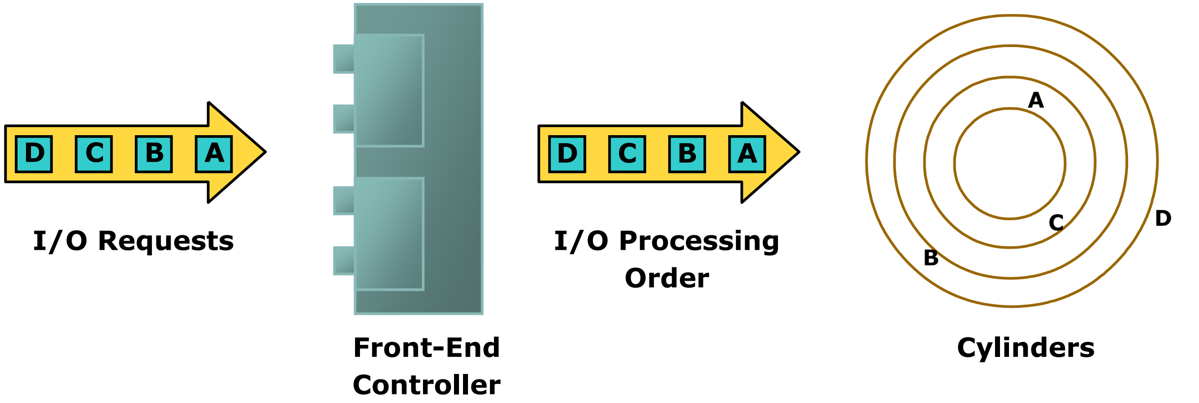
- **First In First Out (FIFO)**

This is the default algorithm where commands are executed in the order in which they are received. There is no reordering of requests for optimization; therefore, it is inefficient in terms of performance.

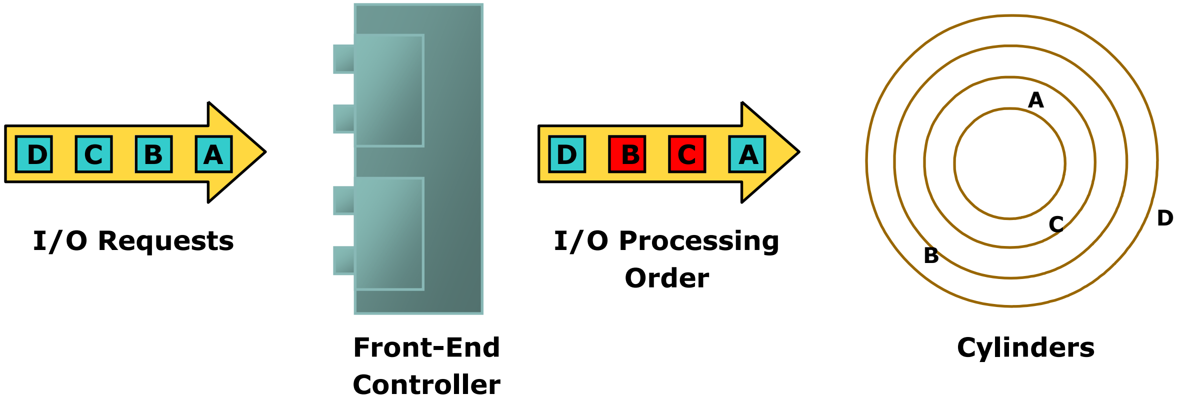
- **Seek Time Optimization**

Commands are executed based on optimizing read/write head movements, which may result in reordering of commands. Without seek time optimization, the commands are executed in the order they are received.

# Front End Command Queuing



Without Optimization (FIFO)



With command queuing

**For example**, as shown in Figure, the commands are executed in the order A, B, C and D.

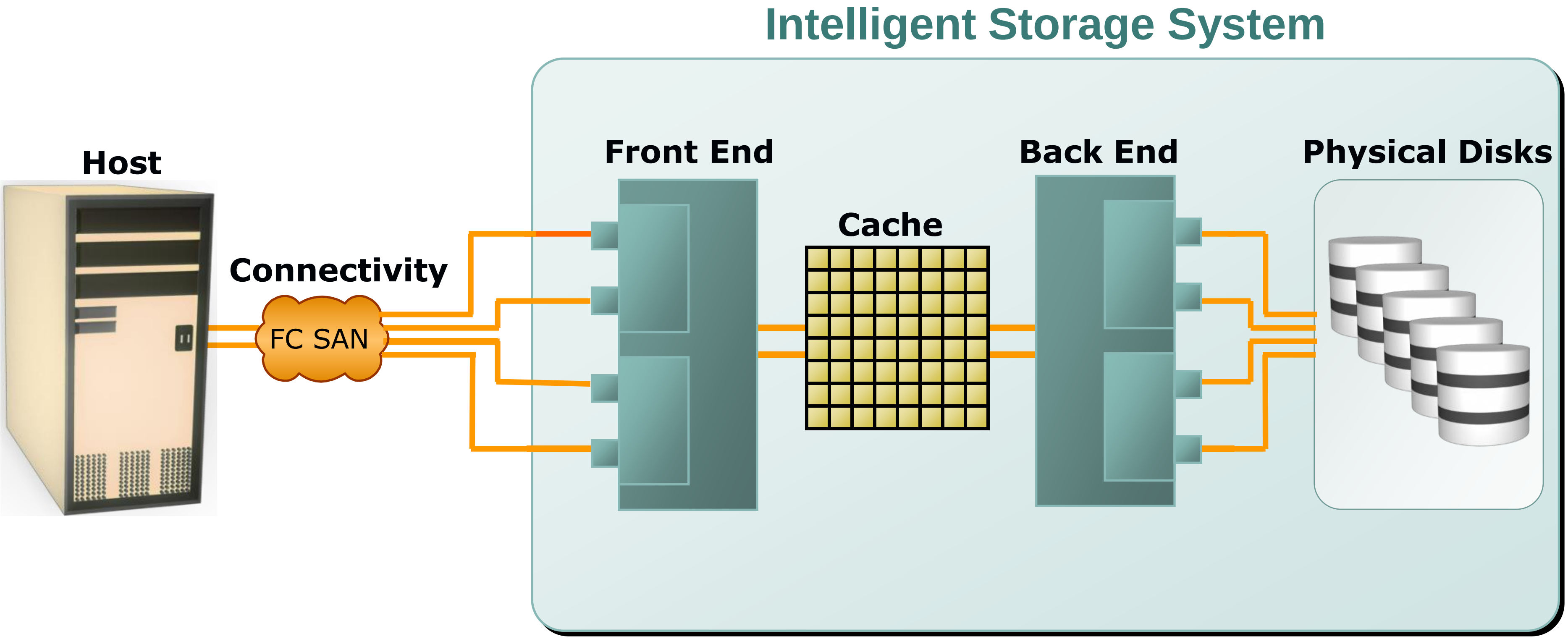
The radial movement required by the head to execute C immediately after A is less than what would be required to execute B.

With seek time optimization, the command execution sequence would be A, C, B and D as shown in the figure.

- **Access Time Optimization**

Commands are executed based on the combination of seek time optimization and an analysis of rotational latency for optimal performance.

# Intelligent Storage System: Cache



# Cache

**Cache** is an important component that enhances the I/O performance in an intelligent storage system.

Cache is semiconductor memory where data is placed temporarily to reduce the time required to service I/O requests from the host.

Cache improves storage system performance by isolating hosts from the mechanical delays associated with physical disks, which are the slowest components of an intelligent storage system.

Accessing data from a physical disk usually takes a few milliseconds because of seek times and rotational latency.

**If** a disk has to be accessed by the host for every I/O operation, requests are queued, which results in a delayed response.

## Structure of Cache

Cache is organized into pages or slots, which is the smallest unit of cache allocation.

The size of a cache page is configured according to the application I/O size.

Cache consists of the data store and tag RAM. The data store holds the data while tag RAM tracks the location of the data in the data store (Figure 4-3) and in disk.

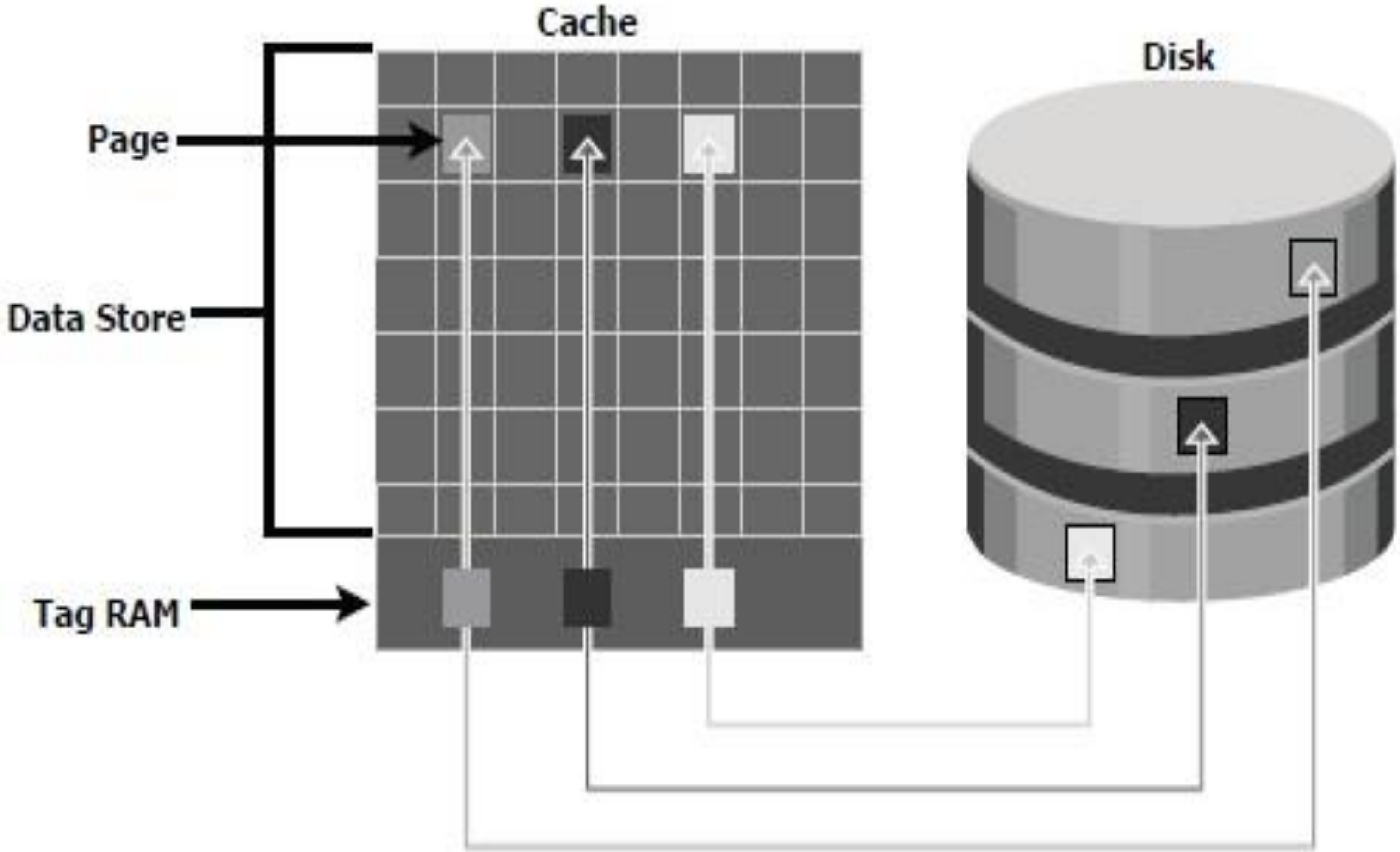


**Figure 4-3:** Structure of cache

# Read Operation with Cache

When a host issues a read request, the front-end controller accesses the tag RAM to determine whether the required data is available in cache.

If the requested data is found in the cache, it is called a read cache hit or read hit and data is sent directly to the host, without any disk operation (see Figure).

This provides a fast response time to the host (about a millisecond).

If the requested data is not found in cache, it is called a cache miss and the data must be read from the disk (see Figure).

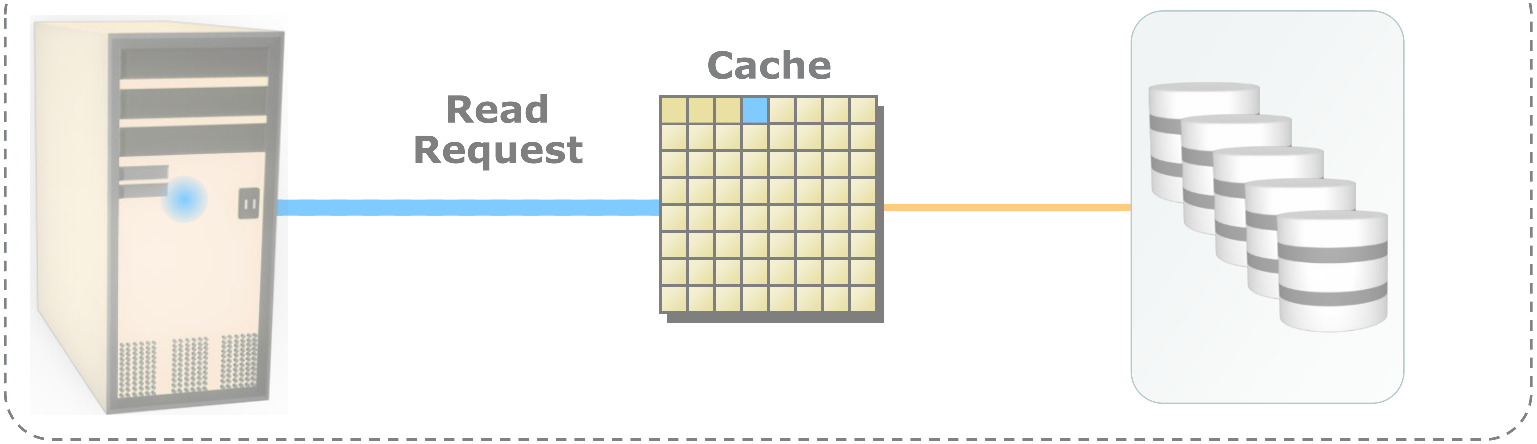
The back-end controller accesses the appropriate disk and retrieves the requested data.

Data is then placed in cache and is finally sent to the host through the front-end controller.

Cache misses increase I/O response time.

# Read Operation with Cache: ‘Hits’ and ‘Misses’

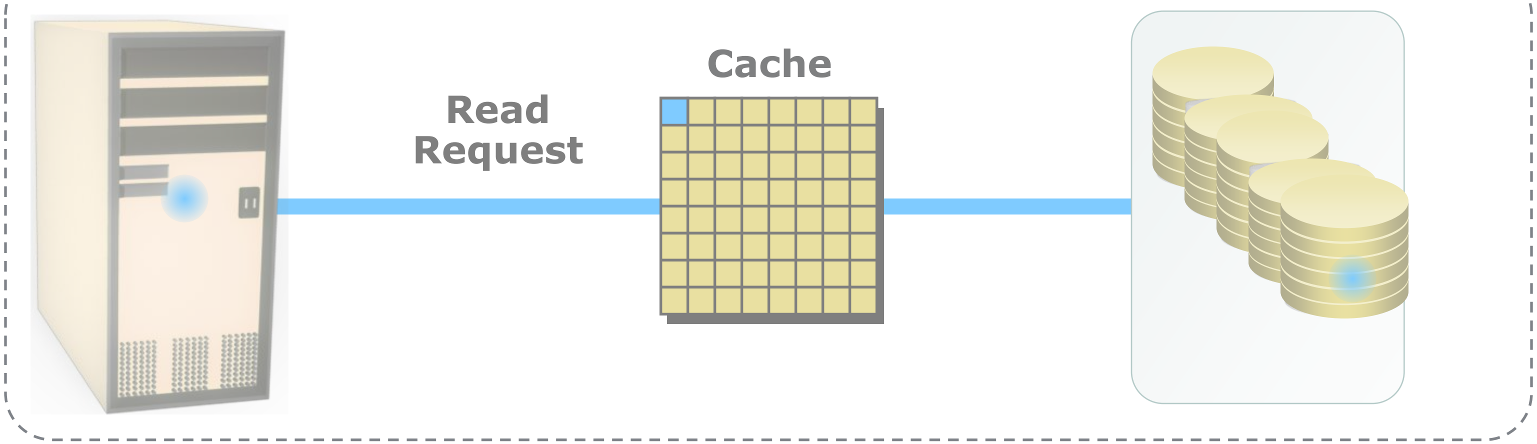
## Data found in cache = ‘Hit’



Read Request

Cache

## No data found = ‘Miss’



Read Request

Cache

# Write Operation with Cache

Write operations with cache provide performance advantages over writing directly to disks.

When an I/O is written to cache and acknowledged, it is completed in far less time (from the host’s perspective) than it would take to write directly to disk.

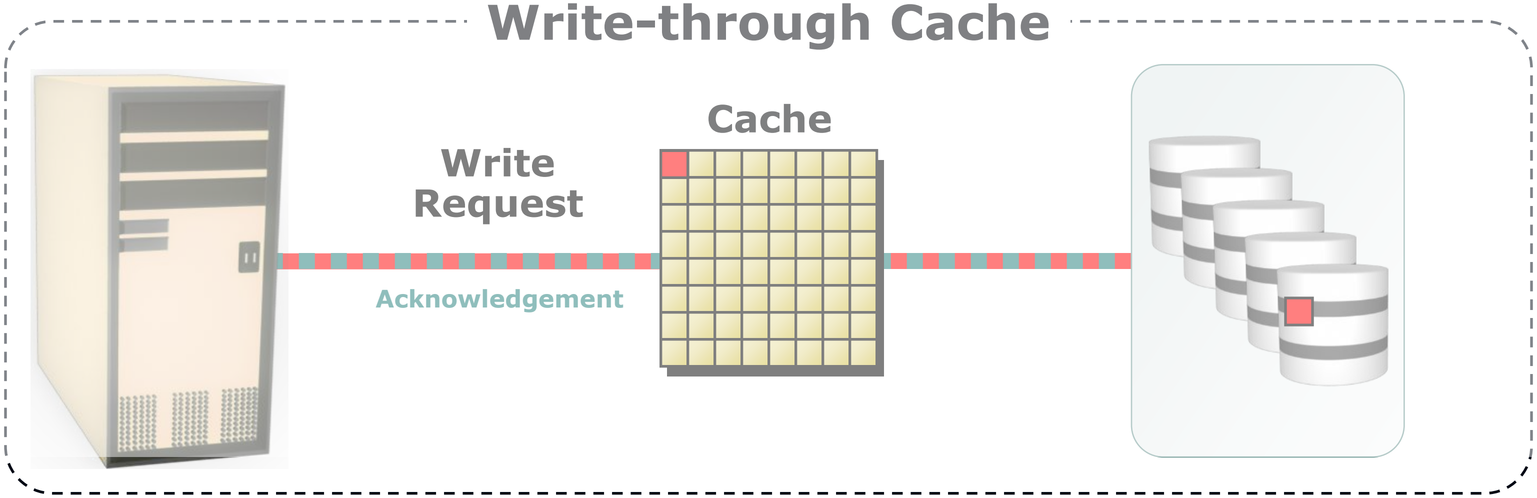
Sequential writes also offer opportunities for optimization because many smaller writes can be coalesced for larger transfers to disk drives with the use of cache.

A write operation with cache is implemented in the following ways:

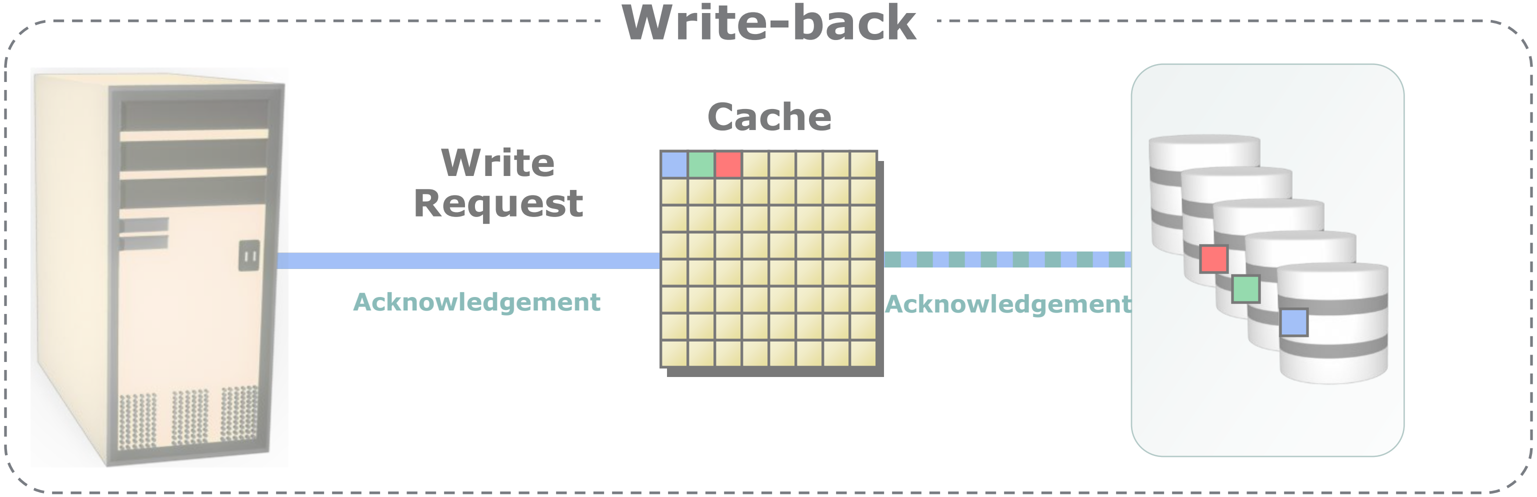
- **Write-back cache**
- **Write-through cache**

# Write Operation with Cache

## Write-through Cache



## Write-back



# Cache Implementation

Cache can be implemented as either dedicated cache or global cache.

With dedicated cache, separate sets of memory locations are reserved for reads and writes.

In global cache, both reads and writes can use any of the available memory addresses.

Cache management is more efficient in a global cache implementation, as only one global set of addresses has to be managed.

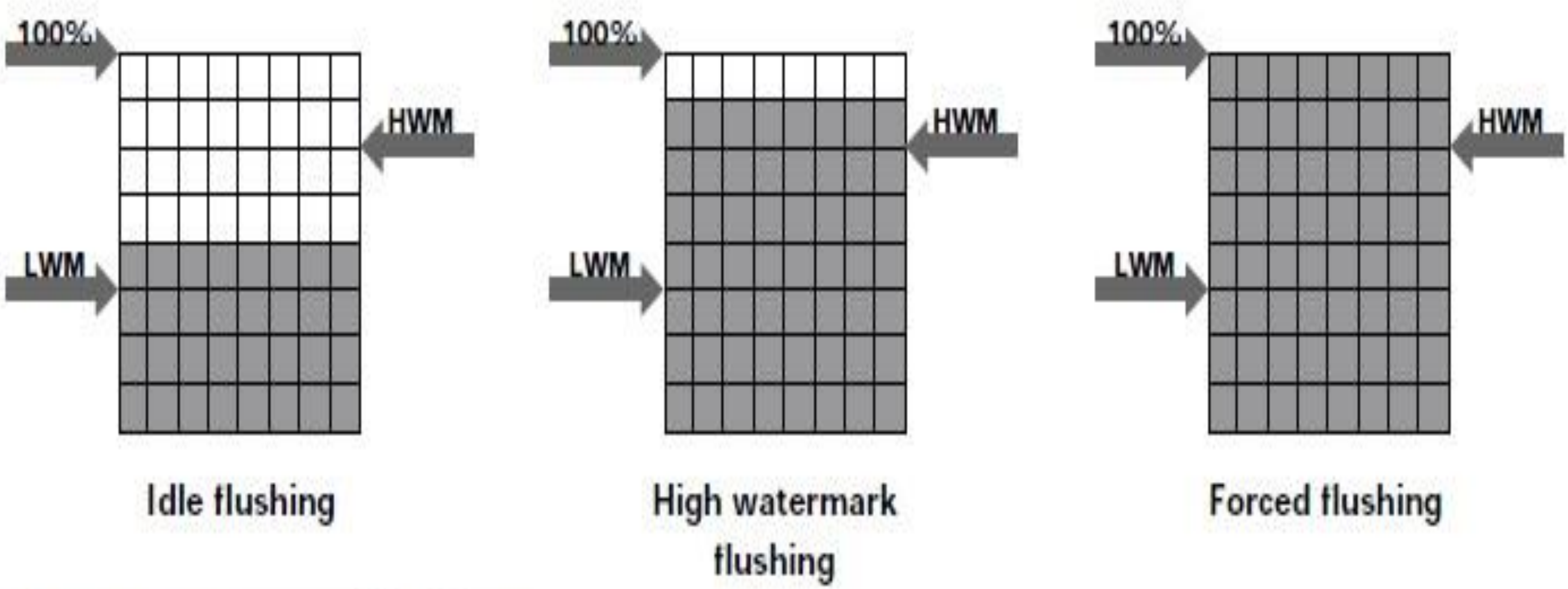
# Cache Management

Cache is a finite and expensive resource that needs proper management.

Even though intelligent storage systems can be configured with large amounts of cache, when all cache pages are filled, some pages have to be freed up to accommodate new data and avoid performance degradation.

Various cache management algorithms are implemented in intelligent storage systems to proactively maintain a set of free pages and a list of pages that can be potentially freed up whenever required:

- **Least Recently Used (LRU)**
- **Most Recently Used (MRU)**
- **Idle flushing**
- **High watermark flushing**
- **Forced flushing**



Idle flushing

High watermark flushing

Forced flushing

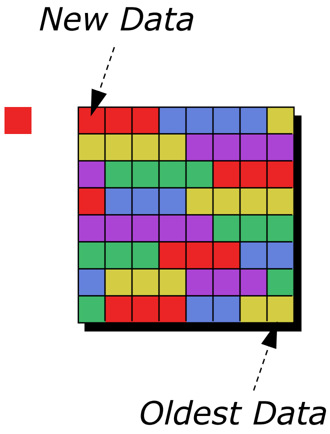
Figure 4-5: Types of flushing

## Cache Data Protection

Cache is volatile memory, so a power failure or any kind of cache failure will cause the loss of data not yet committed to the disk. This risk of losing uncommitted data held in cache can be mitigated using cache mirroring and cache vaulting:

- **Cache mirroring**
- **Cache vaulting**

# Cache Management: Algorithms



- Least Recently Used (LRU)
  - Discards least recently used data
- Most Recently Used (MRU)
  - Discards most recently used data

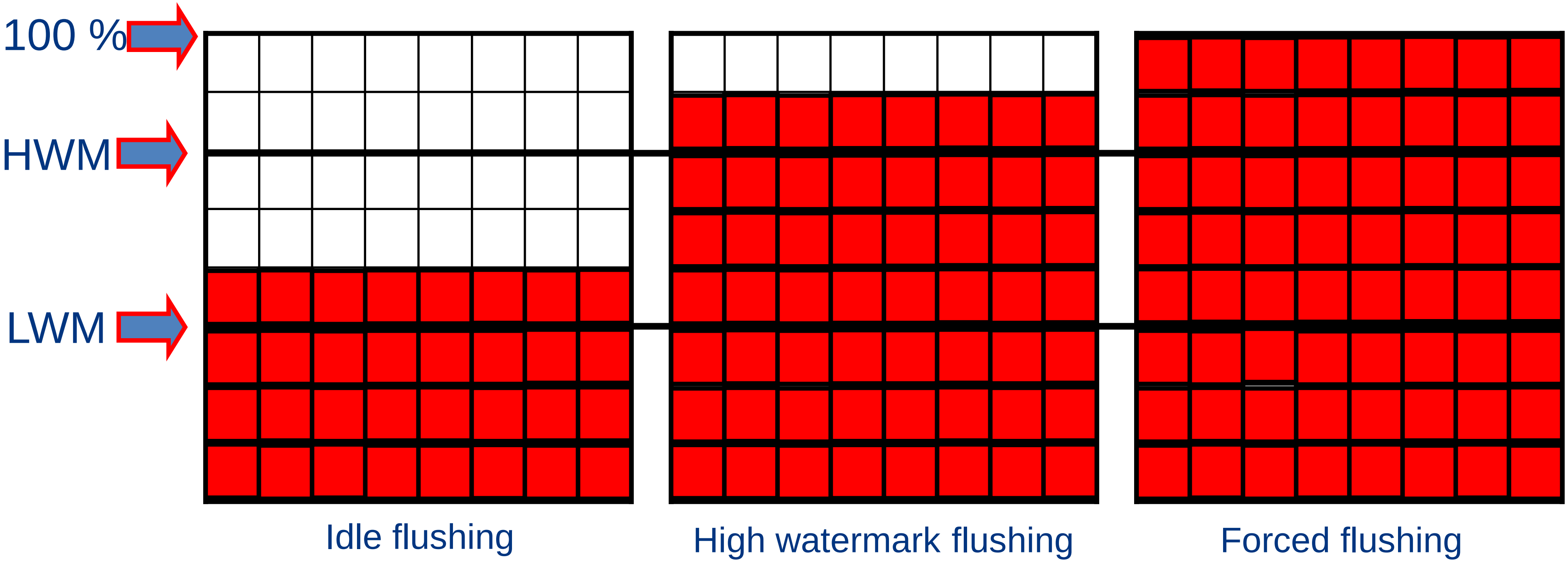
  ( assumed that recent data may not be required for a while)

## *Cache Implementation*

- Dedicated Cache
  - Separate memory sets reserved for read and write
- Global Cache
  - Both read and write operation use available memory.
  - More efficient

# Cache Management: Watermarking

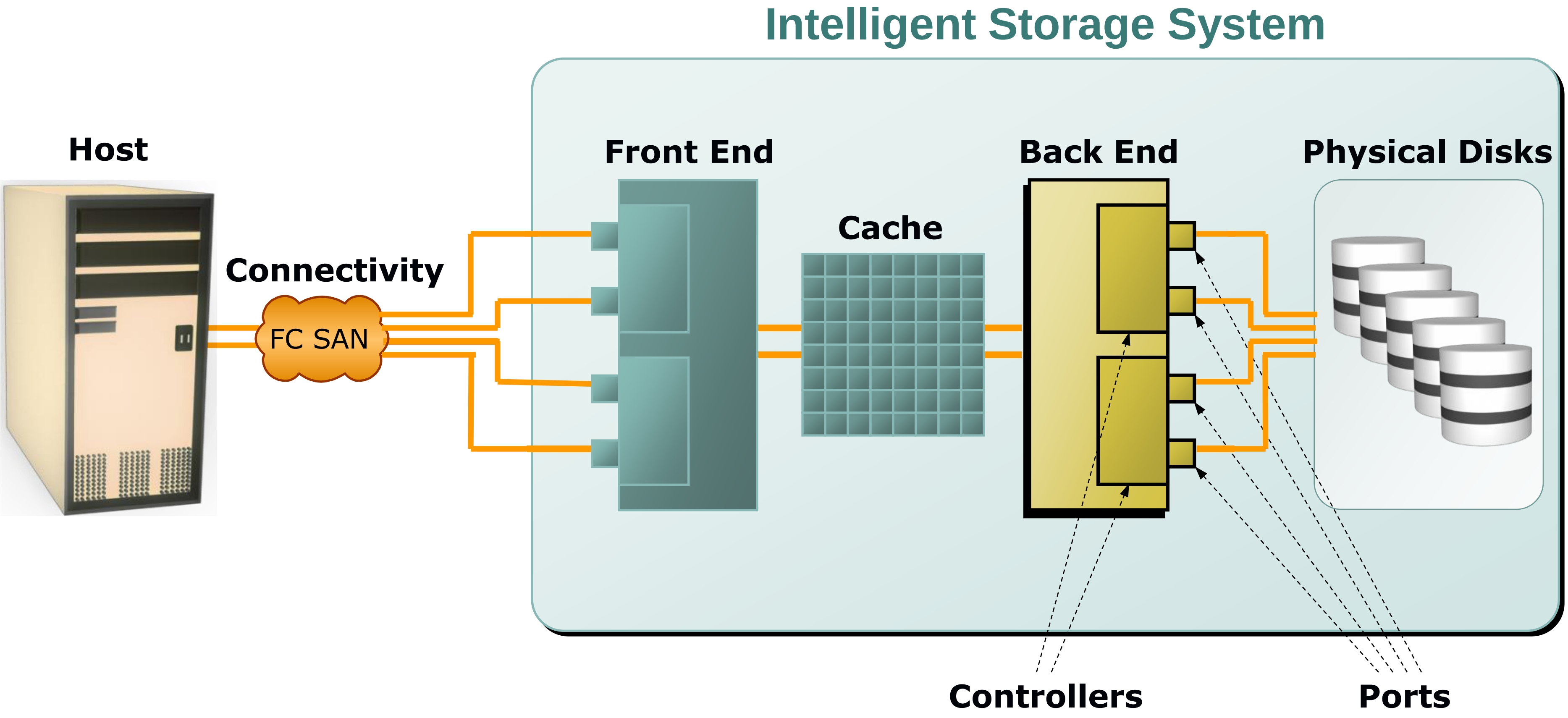
- Manage peak I/O requests "bursts" through flushing/de-staging
  - Idle flushing, High Watermark flushing and Forced flushing
- For maximum performance:
  - Provide headroom in write cache for I/O bursts



# Cache Data Protection

- Protecting cache data against failure: (power failure)
  - Cache mirroring
    - Each write to the cache is held in two different memory locations on two independent memory cards
    - Cache coherency(only writes r mirrored)
  - Cache vaulting
    - Cache is exposed to the risk of uncommitted data loss due to power failure
    - In the event of power failure, uncommitted data is dumped to a dedicated set of drives called vault drives

# Intelligent Storage System: Back End



# Back End

The back end provides an interface between cache and the **physical disks**.

It consists of two components: back-end ports and back-end controllers.

The back end controls data transfers between cache and the physical disks.

From cache, data is sent to the back end and then routed to the destination disk.

Physical disks are connected to ports on the back end.

The back end controller communicates with the disks when performing reads and writes and also provides additional, but limited, temporary data storage.

The algorithms implemented on back-end controllers provide error detection and correction, along with RAID functionality.

# Physical Disk

A physical disk stores data persistently. Disks are connected to the back-end with either SCSI or a Fibre Channel interface.

An intelligent storage system enables the use of a mixture of SCSI or Fibre Channel drives and IDE/ATA drives.

# Logical Unit Number

Physical drives or groups of RAID protected drives can be logically split into volumes known as logical volumes, commonly referred to as Logical Unit Numbers (LUNs).

The use of LUNs improves disk utilization.

For example, without the use of LUNs, a host requiring only 200 GB could be allocated an entire 1TB physical disk.

Using LUNs, only the required 200 GB would be allocated to the host, allowing the remaining 800 GB to be allocated to other hosts.

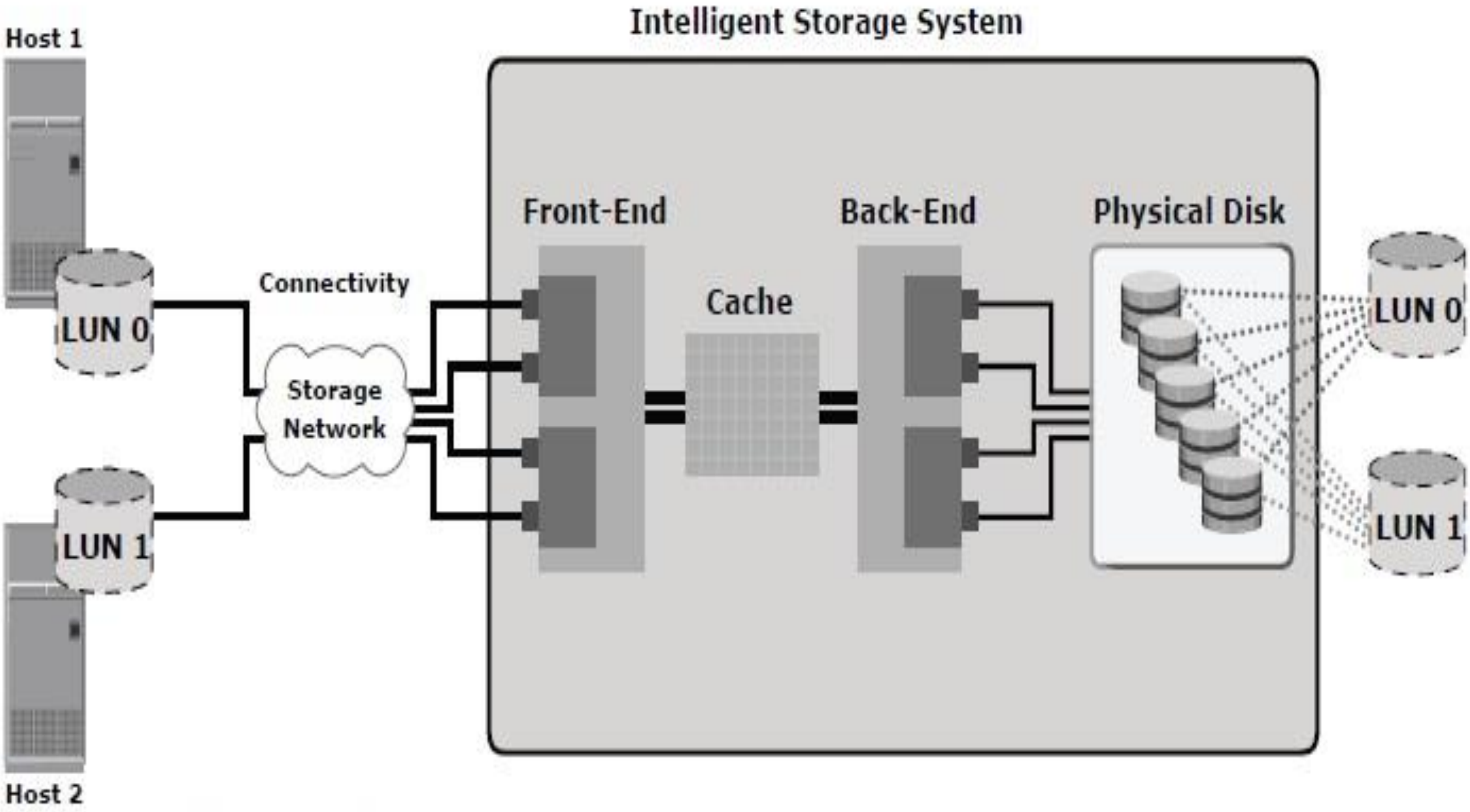


**Figure 4-6:** Logical unit number

# LUN Masking

LUN masking is a process that provides data access control by defining which LUNs a host can access. LUN masking function is typically implemented at the front end controller. This ensures that volume access by servers is controlled appropriately, preventing unauthorized or accidental use in a distributed environment.

**For example**, consider a storage array with two LUNs that store data of the sales and finance departments. Without LUN masking, both departments can easily see and modify each other’s data, posing a high risk to data integrity and security. With LUN masking, LUNs are accessible only to the designated hosts.

# Intelligent Storage Array

Intelligent storage systems generally fall into one of the following two categories:

- **High-end storage systems**
- **Midrange storage systems**

# High-end Storage Systems

High-end storage systems, referred to as active-active arrays, are generally aimed at large enterprises for centralizing corporate data.

These arrays are designed with a large number of controllers and cache memory.

An active-active array implies that the host can perform I/O s to its LUNs across any of the available paths (see Figure 4-7).

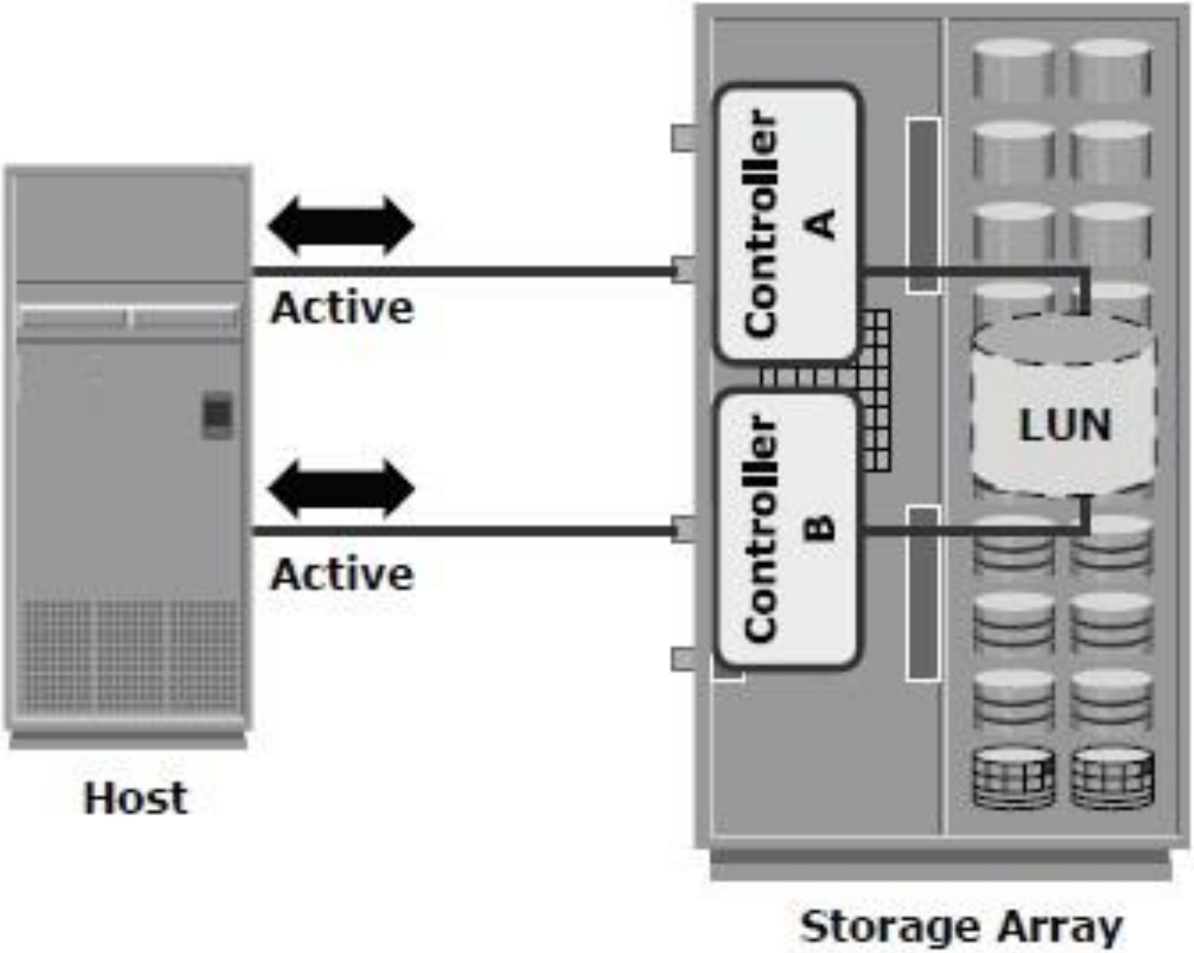


Figure 4-7: Active-active configuration

To address the enterprise storage needs, these arrays provide the following capabilities:

- **Large storage capacity**
- **Large amounts of cache to service host I/O s optimally**
- **Fault tolerance architecture to improve data availability**
- **Connectivity to mainframe computers and open systems hosts**

- Availability of multiple front-end ports and interface protocols to serve a large number of hosts
- Availability of multiple back-end Fibre Channel or SCSI RAID controllers to manage disk processing
- Scalability to support increased connectivity, performance, and storage capacity requirements
- Ability to handle large amounts of concurrent I/O s from a number of servers and applications
- Support for array-based local and remote replication

# Midrange Storage System

Midrange storage systems are also referred to as active-passive arrays and they are best suited for small- and medium-sized enterprises.

In an active-passive array, a host can perform I/Os to a LUN only through the paths to the owning controller of that LUN. These paths are called active paths.

The other paths are passive with respect to this LUN. As shown in Figure 4-8, the host can perform reads or writes to the LUN only through the path to controller A, as controller A is the owner of that LUN.

The path to controller B remains passive and no I/O activity is performed through this path.

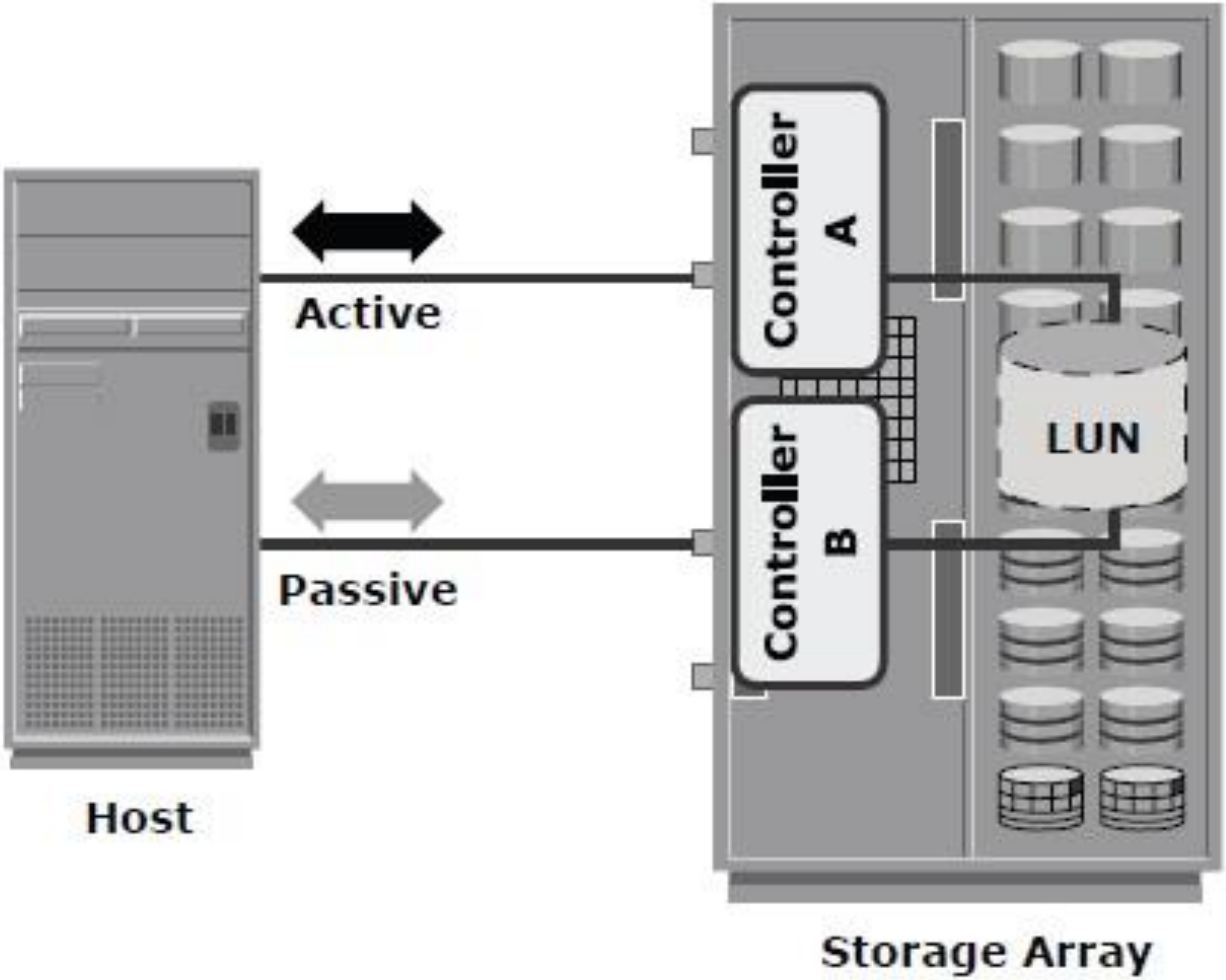


**Figure 4-8:** Active-passive configuration

# Symmetrix Storage Array

The EMC Symmetrix establishes the highest standards for performance and capacity for an enterprise information storage solution and is recognized as the industry’s most trusted storage platform. Figure 4-11 shows the EMC Symmetrix DMX-4 storage array. EMC Symmetrix uses the Direct Matrix Architecture and incorporates a fault tolerant design. Other features of the Symmetrix are as follows:

- **Incrementally scalable up to 2,400 disks**
- **Supports Flash-based solid-state drives**
- **Dynamic global cache memory (16 GB–512 GB)**
- **Advanced processing power (up to 130 PowerPC)**

- Large number of concurrent data paths available (32-128 data paths) for I/O processing
- High data processing bandwidth (up to 128 GB/s)
- Data protection with RAID 1, 1+0 (also known as 10 for mainframe), 5, and 6
- Storage-based local and remote replication for business continuity through Time Finder and SRDF software

# Symmetrix Component Overview

Figure 4-12 shows the basic block diagram of Symmetrix components.

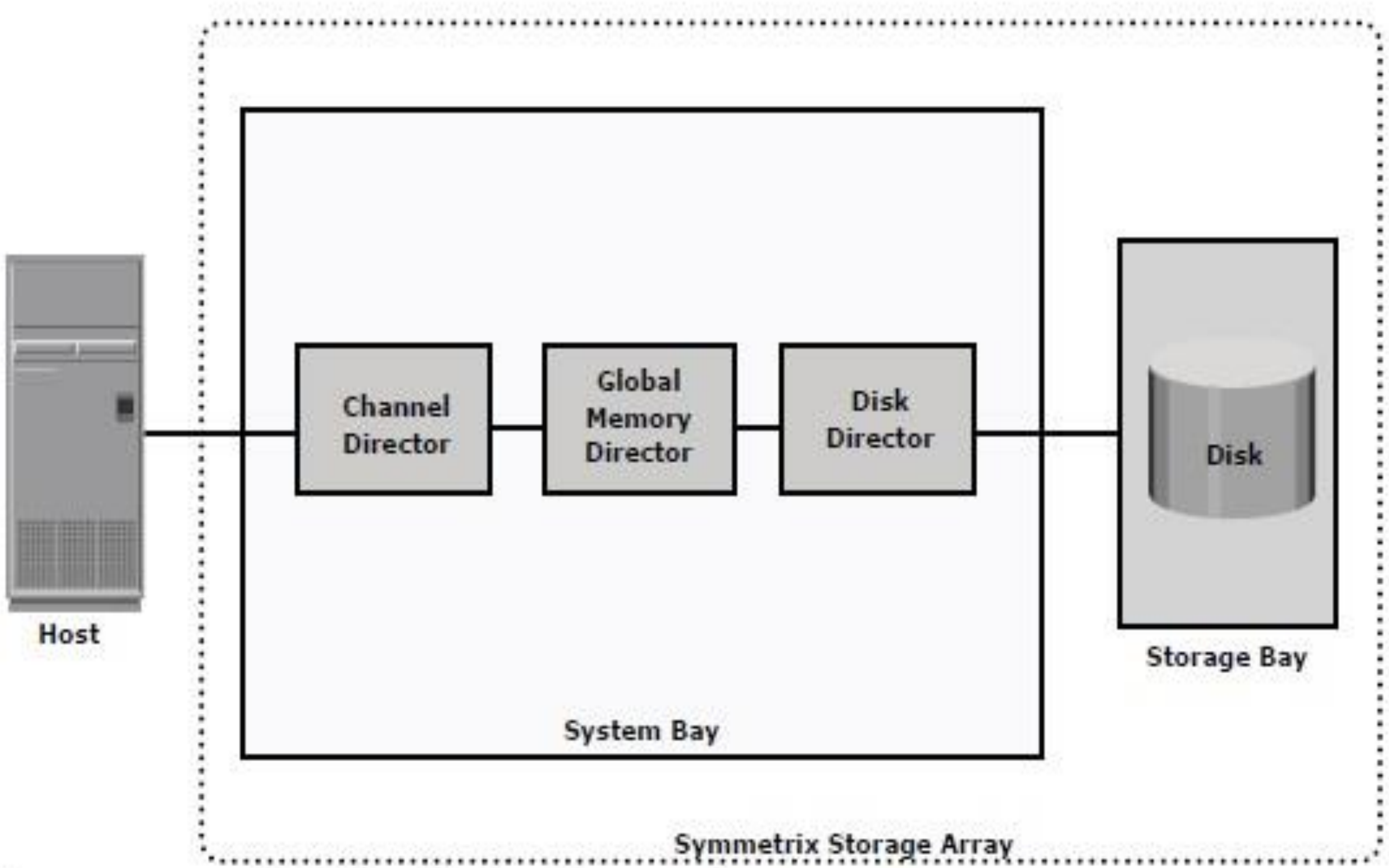


**Figure 4-12:** Basic building blocks of Symmetrix

The Symmetrix system bay consists of front-end controllers (called Channel Directors) for host connectivity, large amounts of global cache (called Global Memory Director [GMD]), and back-end controllers (called Disk Directors) for disk connectivity.

The storage bay is a disk enclosure that can house 240 drives in a cabinet.

Figure 4-13 shows the components of the Symmetrix system bay and storage bay.

# Direct Matrix Architecture

Symmetrix uses the Direct Matrix Architecture consisting of dedicated paths for data transfer between the front end, global memory, and the back end. Key components of Symmetrix DMX are as follows:

- **Front end**
- **Back end**
- **Global Memory**
- **XCM**
- **Symmetrix Enginuity**

# Summary

This chapter detailed the features and components of the intelligent storage system — front end, cache, back end, and physical disks. The active-active and active passive implementations of intelligent storage systems were also described. An intelligent storage system provides the following benefits to an organization:

- **Increased capacity**
- **Improved performance**
- **Easier storage management**

- **Improved data availability**
- **Improved scalability and flexibility**
- **Improved business continuity**
- **Improved security and access control**

An intelligent storage system is now an integral part of every mission-critical data center.

Although a high-end intelligent storage system addresses information storage requirements, it poses a challenge for administrators to share information easily and securely across the enterprise.

Storage networking is a flexible information-centric strategy that extends the reach of intelligent storage systems throughout an enterprise.

It provides a common way to manage, share, and protect information. Storage networking is detailed in the next section.

# Data Protection: RAID

A RAID is a Redundant Array of Inexpensive/ Independent Disks

# Data Protection: RAID

- "Cheap" SCSI hard drives are now big enough for most applications
- We use RAID today for
  - Increasing disk throughput by allowing parallel access
  - Eliminating the need to make disk backups
    - Disks are too big to be backed up in an efficient fashion

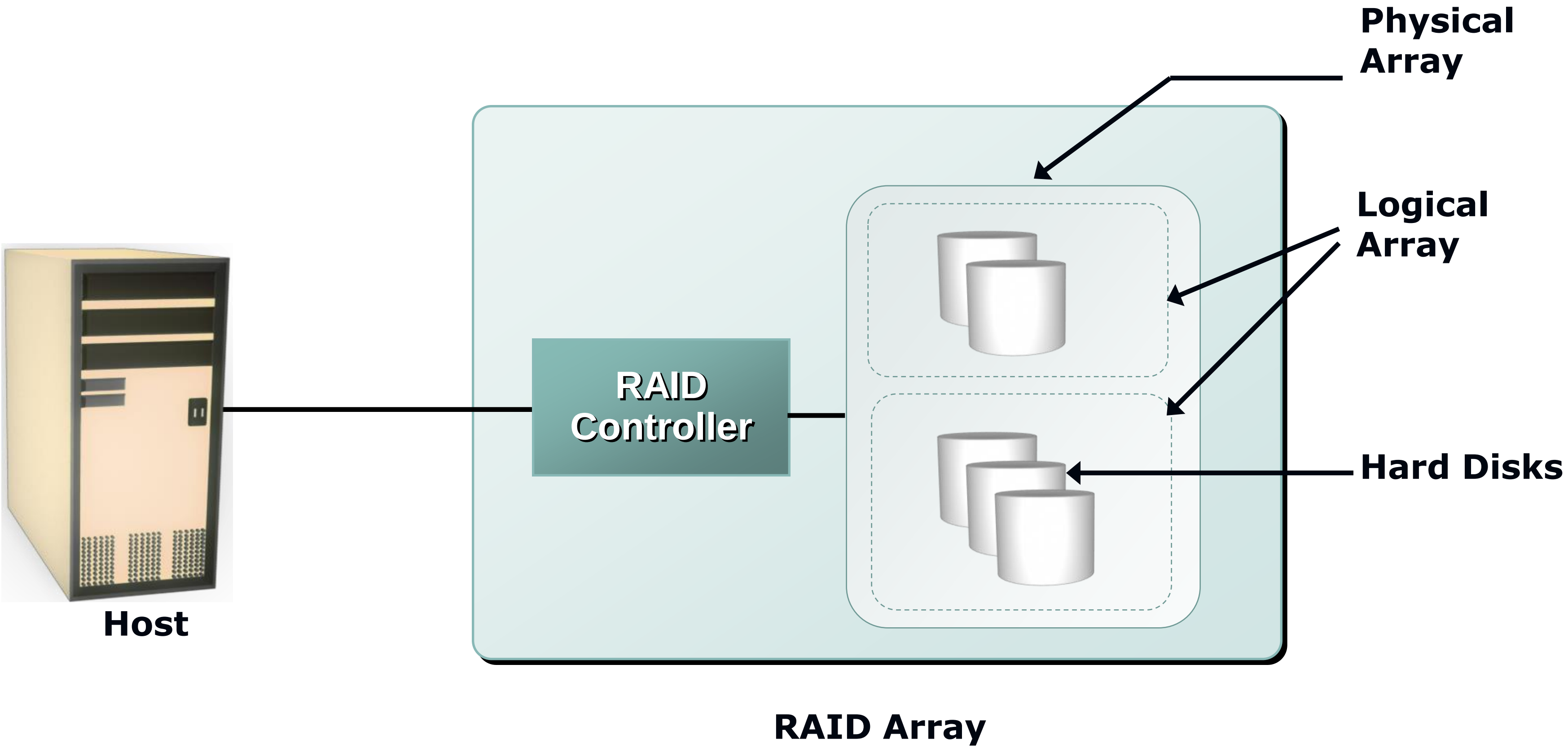
# *Disk Arrays*

- Disk arrays with **redundant disks** to tolerate faults
- If a single disk fails, the lost information is reconstructed from redundant information
- **Striping**: simply spreading data over multiple disks
- **RAID:** redundant array of inexpensive/independent disks

# Why RAID?

- Performance limitation of disk drive
- An individual drive has a certain life expectancy
  - Measured in MTBF (Mean Time Between Failure)
  - The more the number of HDDs in a storage array, the larger the probability for disk failure. For example:
    - If the MTBF of a drive is 750,000 hours, and there are 100 drives in the array, then the MTBF of the array becomes 750,000 / 100, or 7,500 hours
- RAID was introduced to mitigate this problem
- RAID provides:
  - Increase capacity
  - Higher availability
  - Increased performance

# RAID Array Components



# RAID Controller

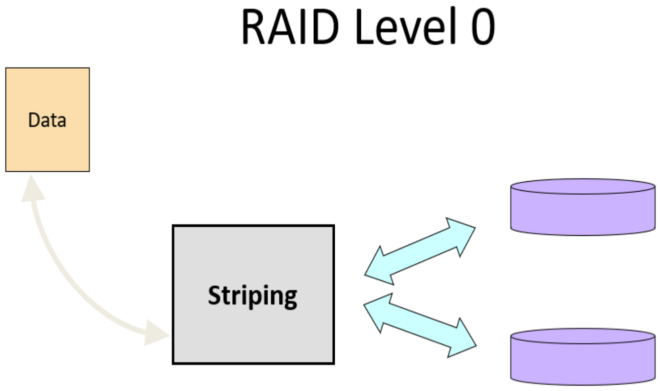
- Hardware (usually a specialized disk controller card)
  - Controls all drives attached to it
  - Array(s) appear to host operating system as a regular disk drive
  - Provided with administrative software
- Software
  - Runs as part of the operating system
  - Performance is dependent on CPU workload
  - Does not support all RAID levels

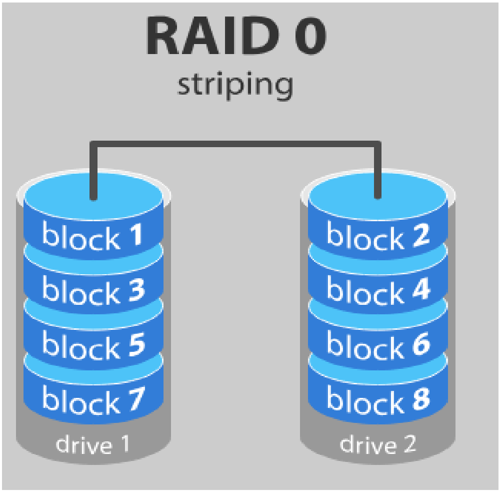
# RAID Levels

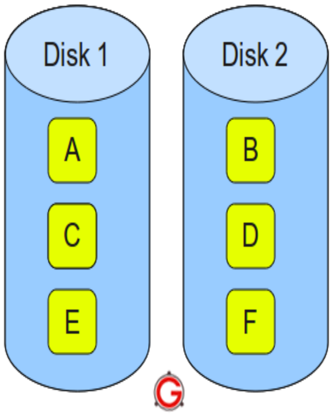
- RAID-0 Striped array with no fault tolerance
- RAID-1 Disk mirroring
- Nested RAID (i.e., 1 + 0, 0 + 1, etc.)
- RAID-3 Parallel access array with dedicated parity disk
- RAID-4 Striped array with independent disks and a  dedicated parity disk
- RAID-5 Striped array with independent disks and distributed parity
- RAID-6 Striped array with independent disks and dual distributed parity

# RAID Level 0

- RAID Level 0 is a non-redundant disk array
- Files are striped across disks, no redundant info
- High read throughput
- Best write throughput (no redundant info to write)
- Any disk failure results in data loss; sometimes a file, sometimes the entire *volume*





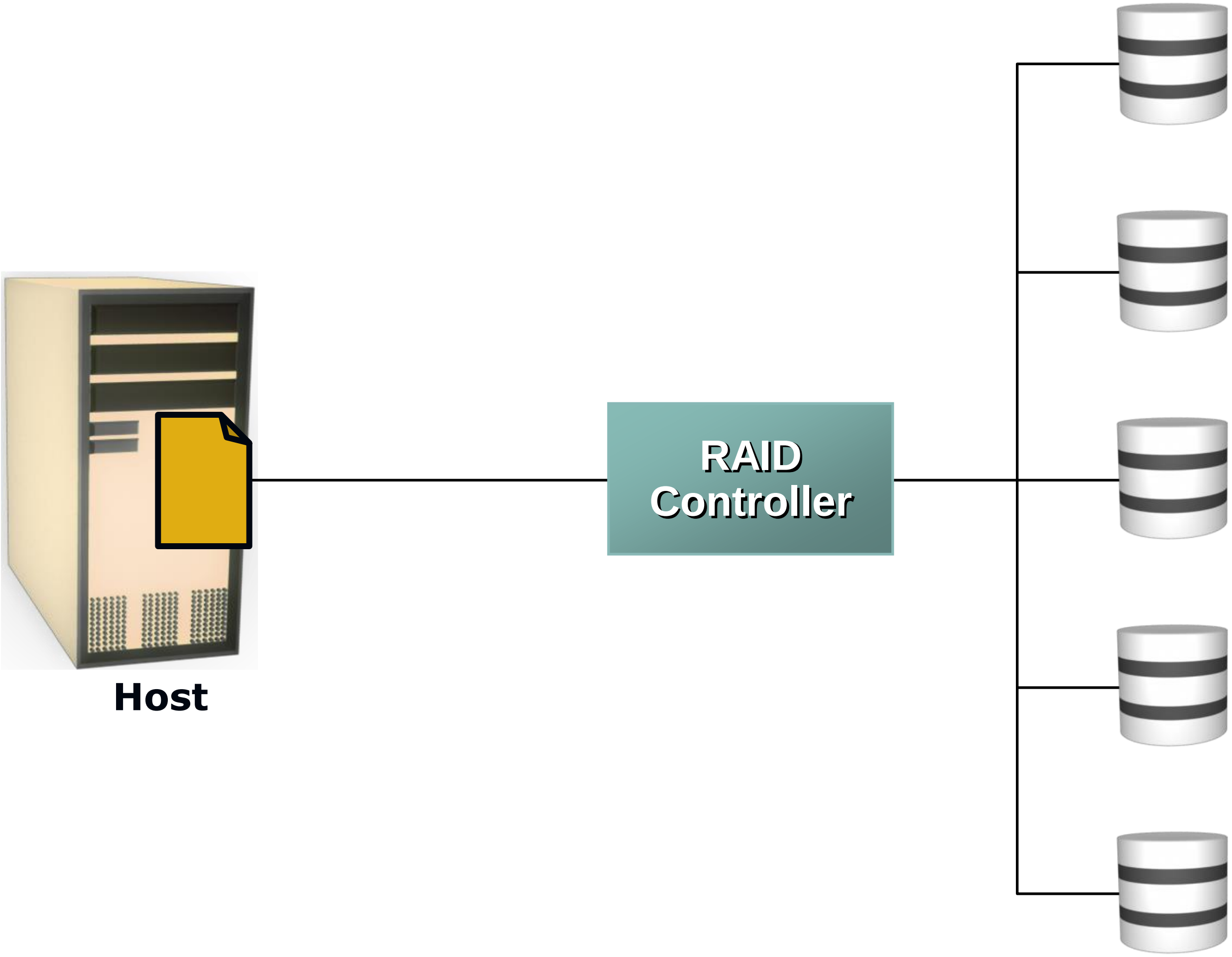


RAID 0 – Blocks Striped. No Mirror. No Parity.

# *RAID 0*

- Data is distributed across the HDDs in the RAID set.
- Allows multiple data to be read or written simultaneously/concurrently, and therefore improves performance.
- Does not provide data protection and availability in the event of disk failures.
  - No replication
  - ***Advantages:***
    - Simple to implement
    - No overhead
  - ***Disadvantage:***
    - If array has *n* disks failure rate is *n* times the failure rate of a single disk

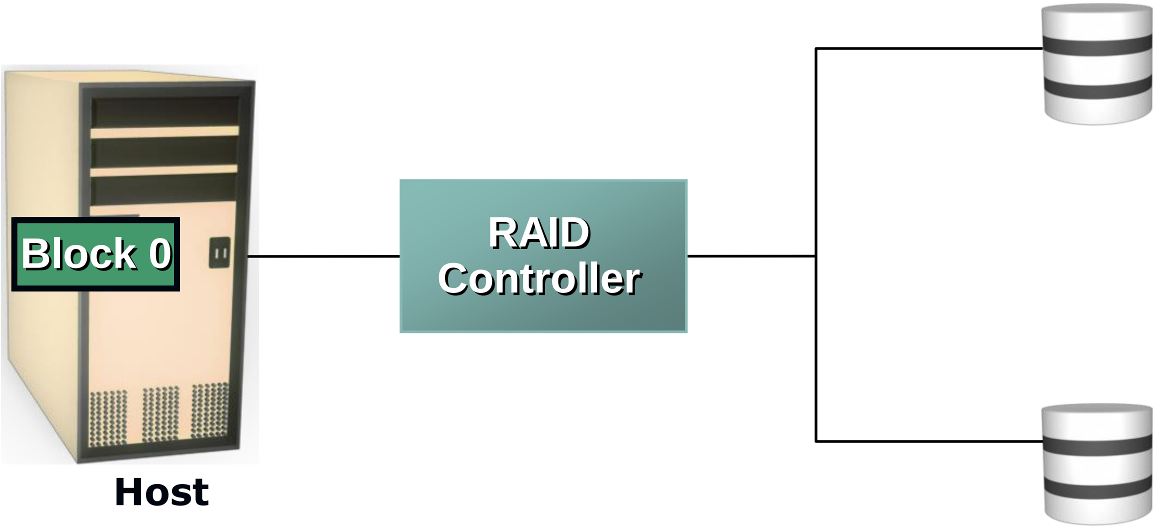
# RAID 0



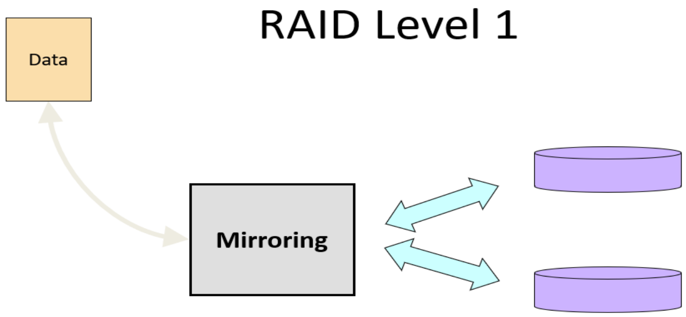
# RAID 1

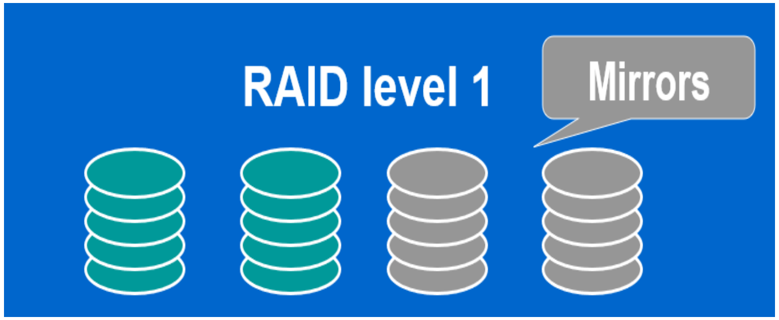
- Data is stored on two different HDDs, yielding two copies of the same data.
  - Provides availability.
- In the event of HDD failure, access to data is still available from the surviving HDD.
- When the failed disk is replaced with a new one, data is automatically copied from the surviving disk to the new disk.
  - Done automatically by RAID the controller.
- Disadvantage: The amount of storage capacity is twice the amount of data stored.
- Mirroring is NOT the same as doing backup!

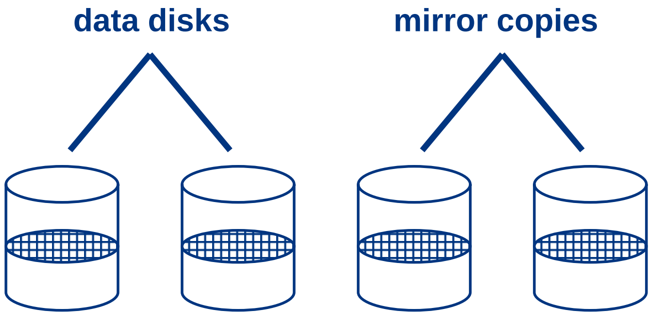
# RAID 1



# RAID Level 1



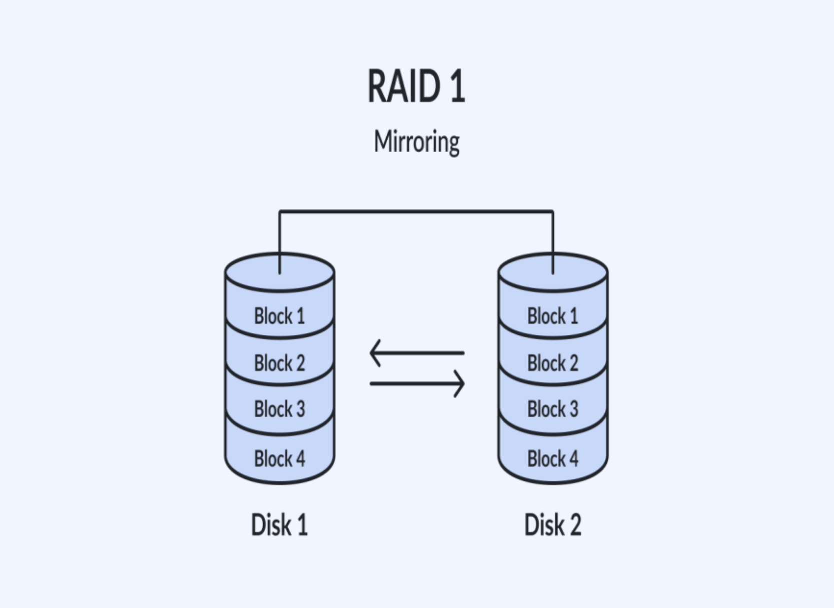




- Mirroring
  - Two copies of each disk block
- Advantages:
  - Simple to implement
  - Fault-tolerant
- Disadvantage:
  - Requires twice the disk capacity of normal file systems

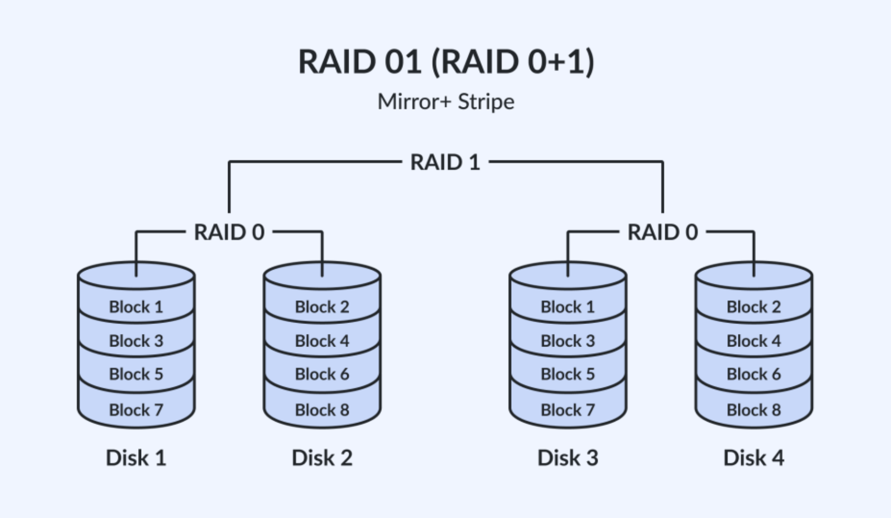
# RAID 1

- **Mirroring** or **Shadowing**
- Two copies for every piece of data
- one logical write = two physical writes
- 100% capacity/space overhead
- RAID Level 1 is mirrored disks
- Files are striped across (half) the disks
- Data is written to multiple (two) places – data disks and mirror disks
- On failure, just use the surviving disk(s)
- Factor of N (2x) space expansion

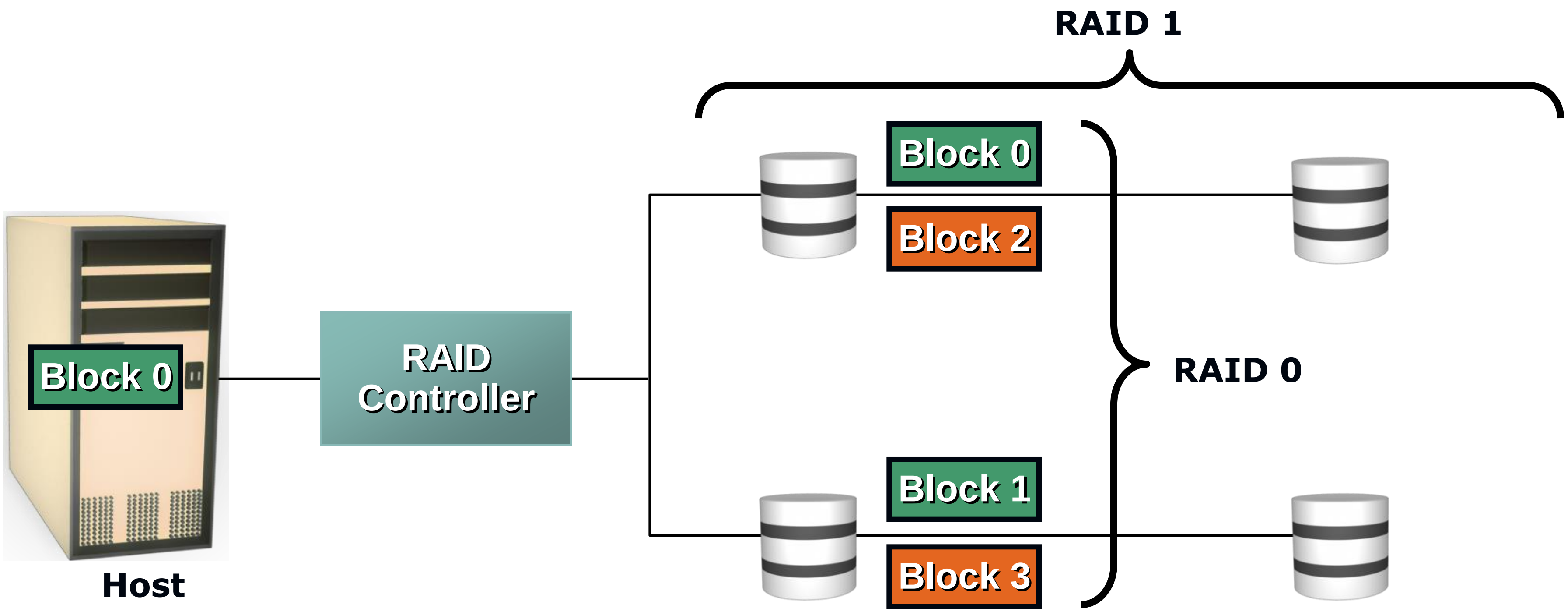


# Nested RAID

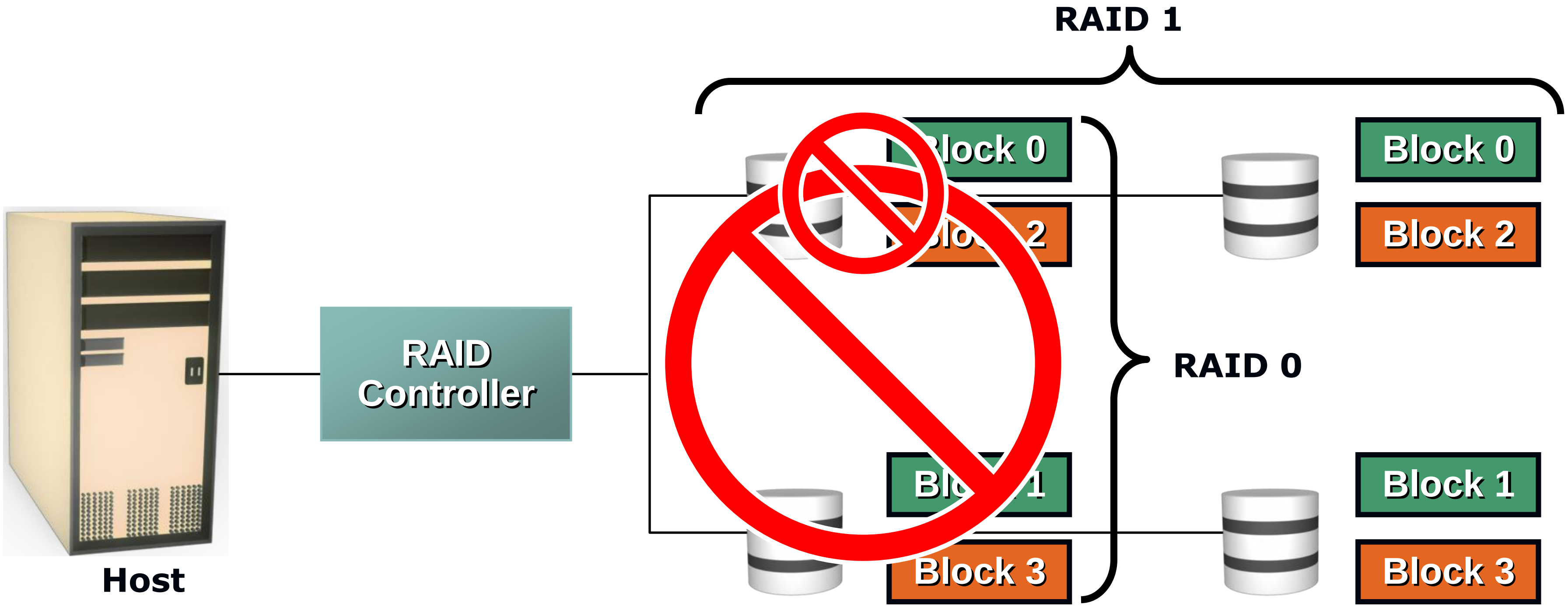
- Combines the performance benefits of RAID 0 with the redundancy benefit of RAID 1.
- RAID 0+1 – Mirrored Stripe
  - Data is striped across HDDs, then the entire stripe is mirrored.
  - If one drive fails, the entire stripe is faulted.
  - Rebuild operation requires data to be copied from each disk in the healthy stripe, causing increased load on the surviving disks.



# Nested RAID – 0+1 (Striping and Mirroring)

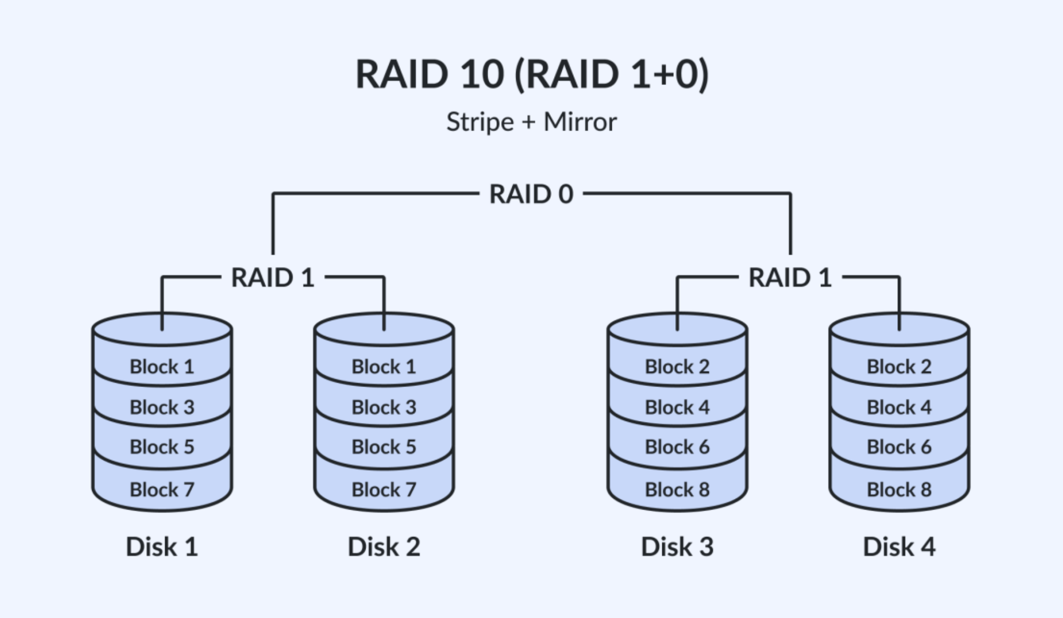


# Nested RAID – 0+1 (Striping and Mirroring)

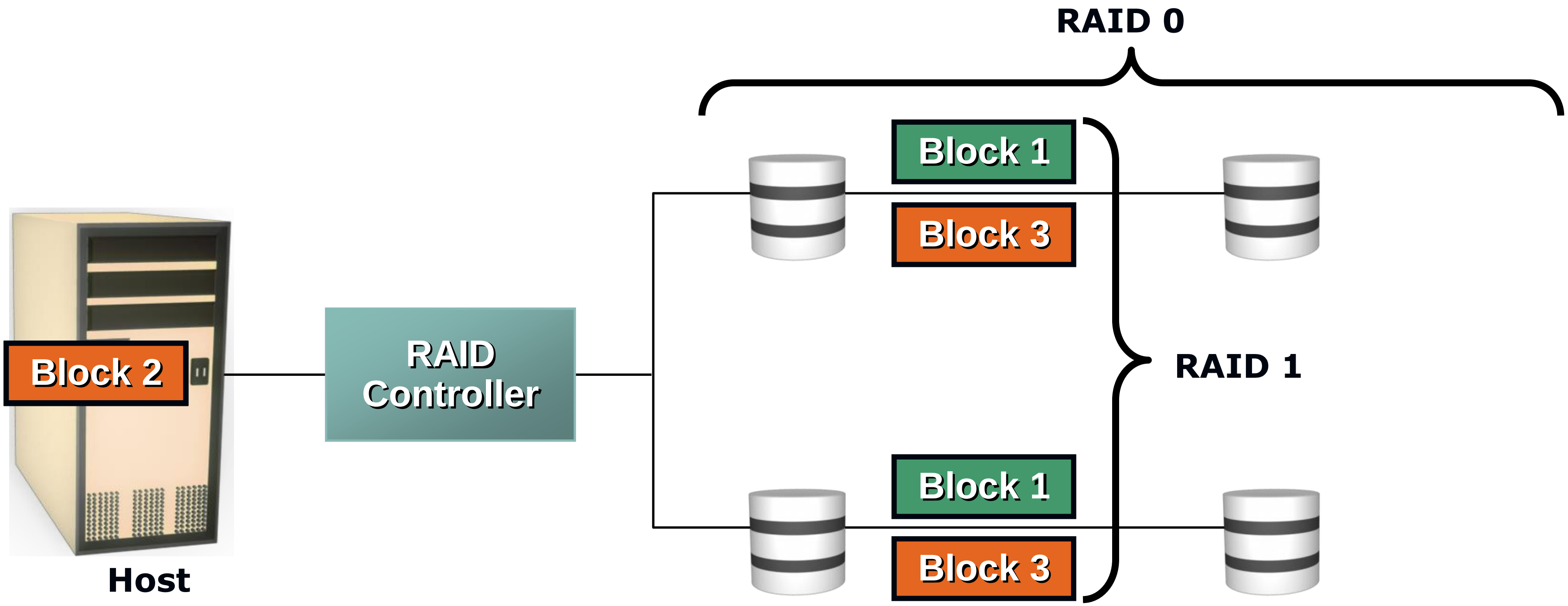


# Nested RAID

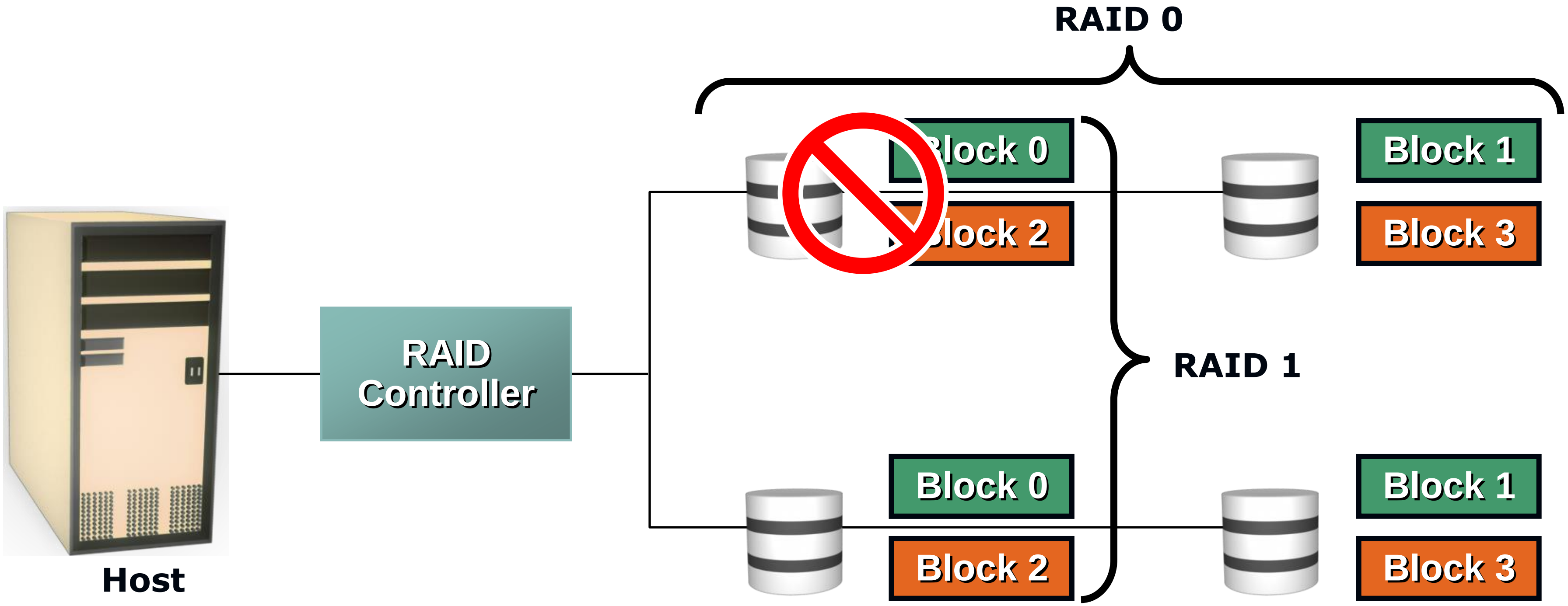
- RAID 1+0 – Striped Mirror
  - Data is first mirrored, and then both copies are striped across multiple HDDs.
  - When a drive fails, data is still accessible from its mirror.
  - Rebuild operation only requires data to be copied from the surviving disk into the replacement disk.



# Nested RAID – 1+0 (Mirroring and Striping)

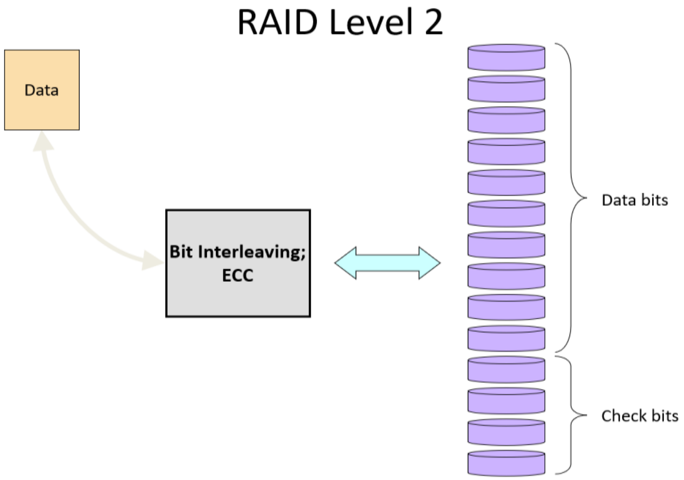


# Nested RAID – 1+0 (Mirroring and Striping)



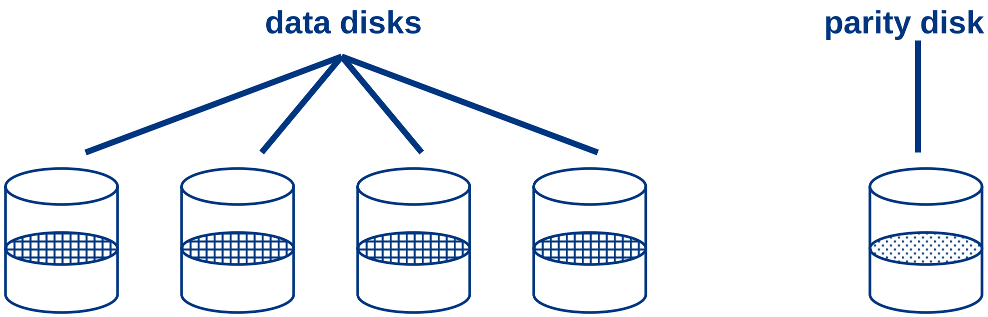
# RAID LEVEL 2

- Instead of duplicating the data blocks we use an **error correction** code
- ***Very bad idea*** because disk drives either work correctly or do not work at all
  - Only possible errors are ***omission errors***
  - We need an ***omission correction code***
    - A parity bit is enough to correct a single omission

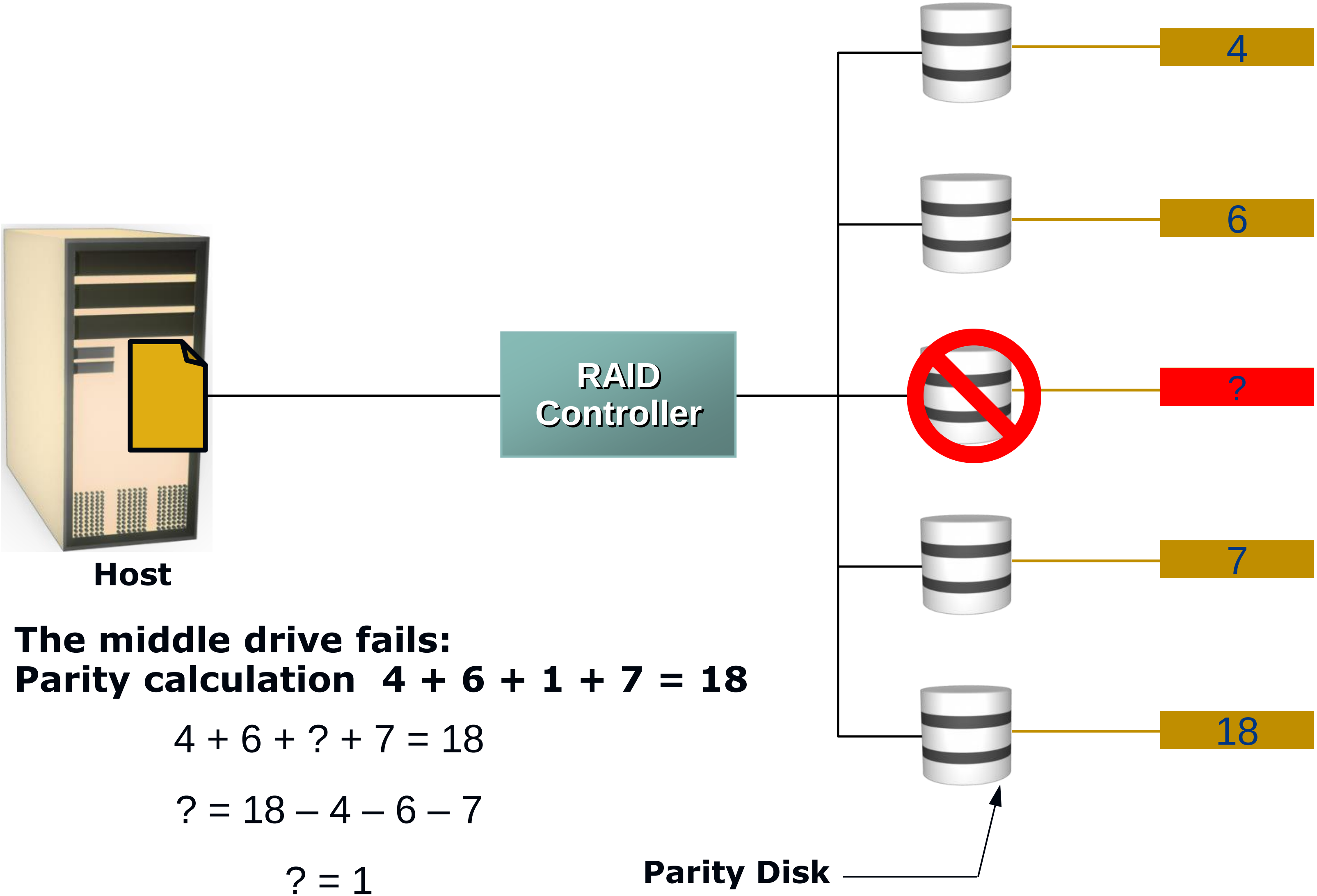


# RAID Levels 2, 3, and 4

- RAID levels 2, 3, and 4 use ECC (error correcting code) or parity disks
  - E.g., each byte on the parity disk is a parity function of the corresponding bytes on all the other disks
- A read accesses all the data disks
- A write accesses all the data disks plus the parity disk
- On disk failure, read the remaining disks plus the parity disk to compute the missing data



# RAID Redundancy: Parity



**The middle drive fails:**
**Parity calculation 4 + 6 + 1 + 7 = 18**

4 + 6 + ? + 7 = 18

? = 18 – 4 – 6 – 7

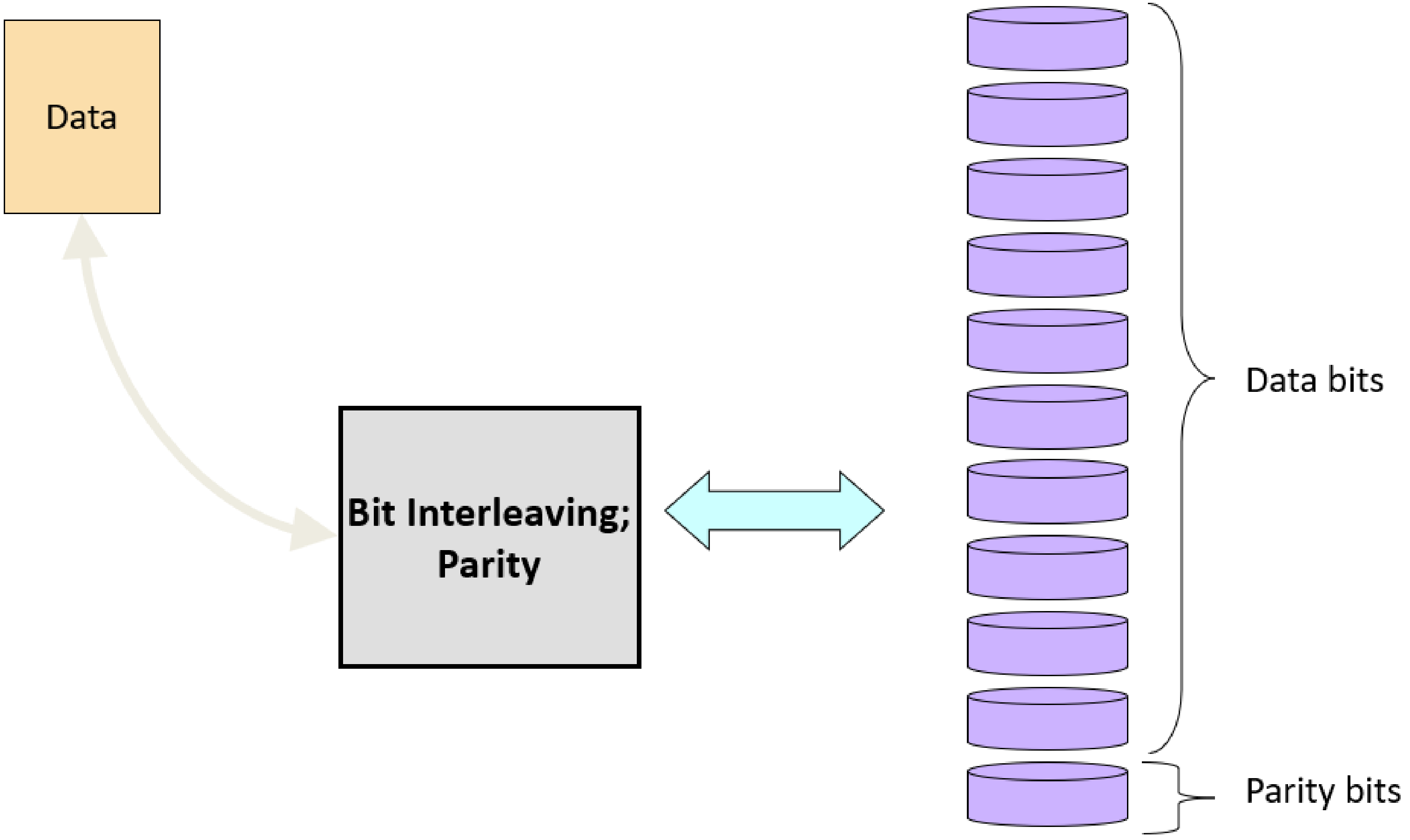
? = 1

# RAID LEVEL 3

- Requires N+1 disk drives
  - N drives contain data (1/N of each data block)
    - Block b[k] now partitioned into N fragments b[k,1], b[k,2], ... b[k,N]
  - Parity drive contains exclusive or of these N fragments

$$p[k] = b[k,1] \oplus b[k,2] \oplus \ldots \oplus b[k,N]$$

# RAID Level 3



# How parity works?

- Truth table for XOR (same as parity)

| A | B | $A \oplus B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Recovering from a disk failure

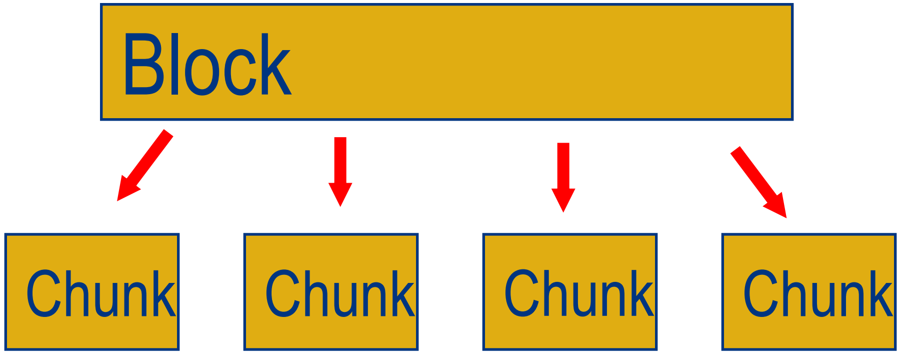
- Small RAID level 3 array with data disks D0 and D1 and parity disk P can tolerate failure of either D0 or D1

| D0 | D1 | P |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

| D1⊕P=D0 | D0⊕P=D1 |
|---|---|
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |

# How RAID level 3 works (I)

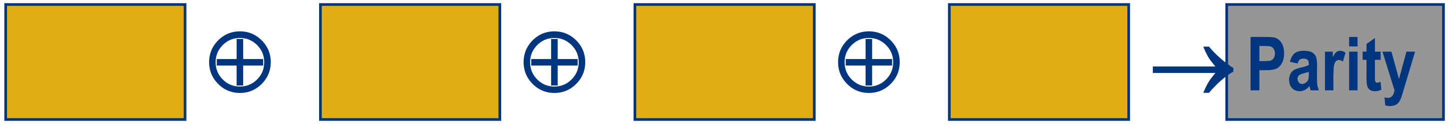
- Assume we have *N* + 1 disks
- Each block is partitioned into *N* equal chunks



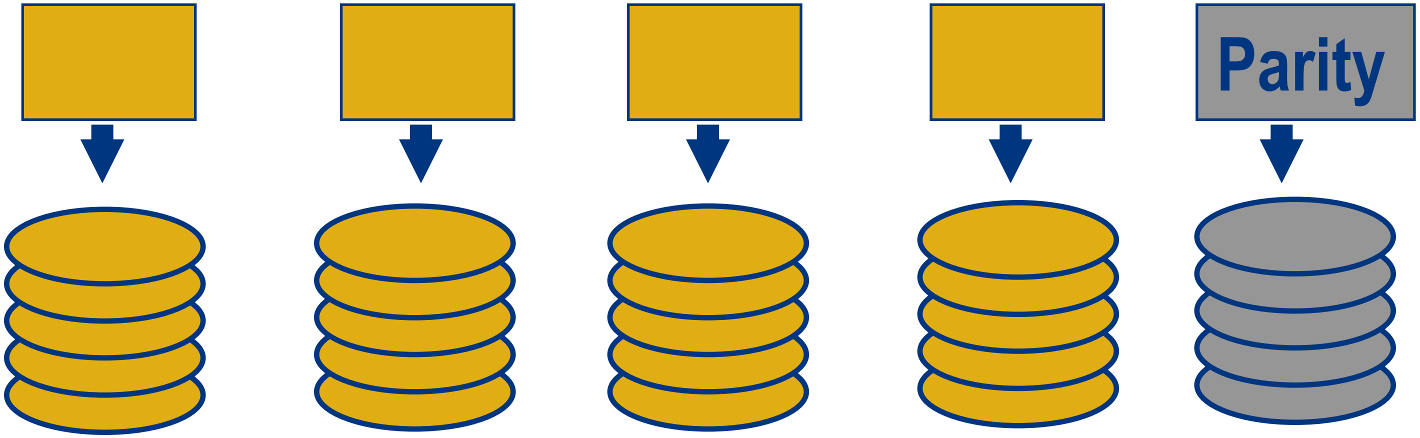
*N* = 4 in example

# *How RAID level 3 works (II)*

o XOR data chunks to compute the parity chunk



- Each chunk is written into a ***separate disk***

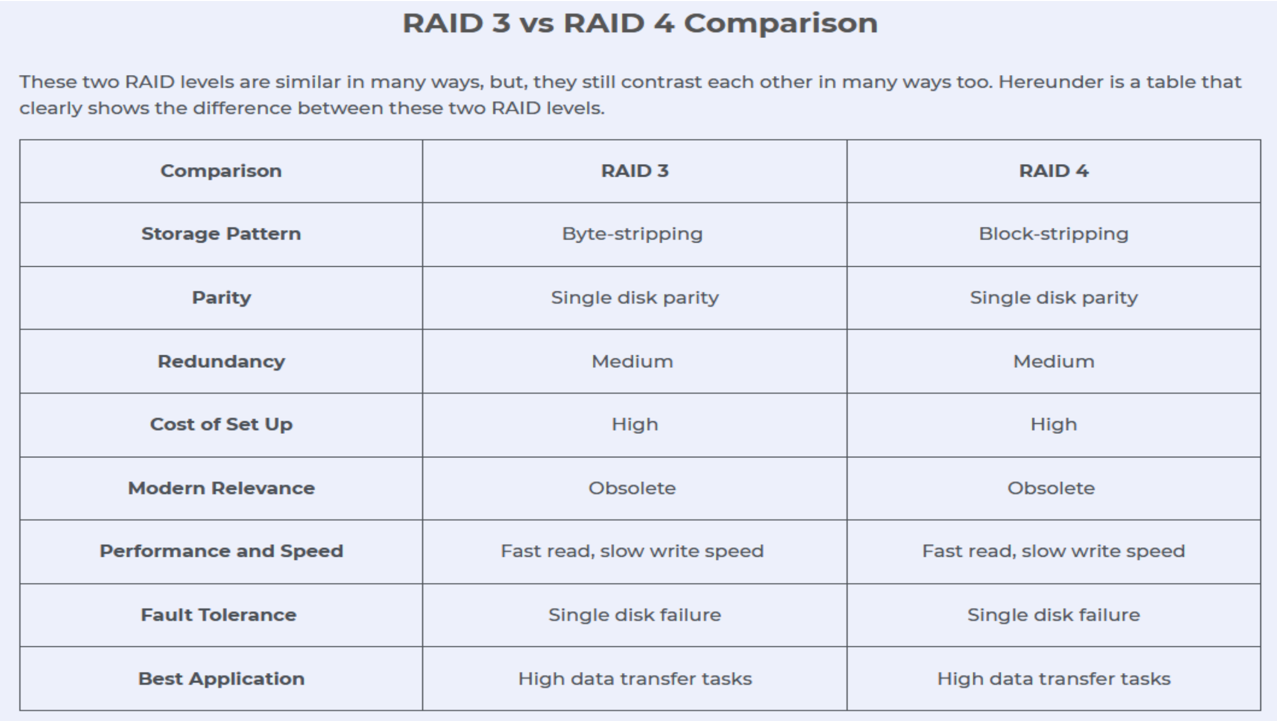


# How RAID level 3 works (III)

- Each read/write involves ***all disks*** in RAID array
  - Cannot do two or more reads/writes ***in parallel***
  - Performance of array not better than that of a ***single disk***

# RAID 3 and RAID 4

- Stripes data for high performance and uses parity for improved fault tolerance.
- One drive is dedicated for parity information.
- If a drive fails, data can be reconstructed using data in the parity drive.
- For RAID 3, data read / write is done across the entire stripe.
  - Provide good bandwidth for large sequential data access such as video streaming.
- For RAID 4, data read/write can be independently on single disk.

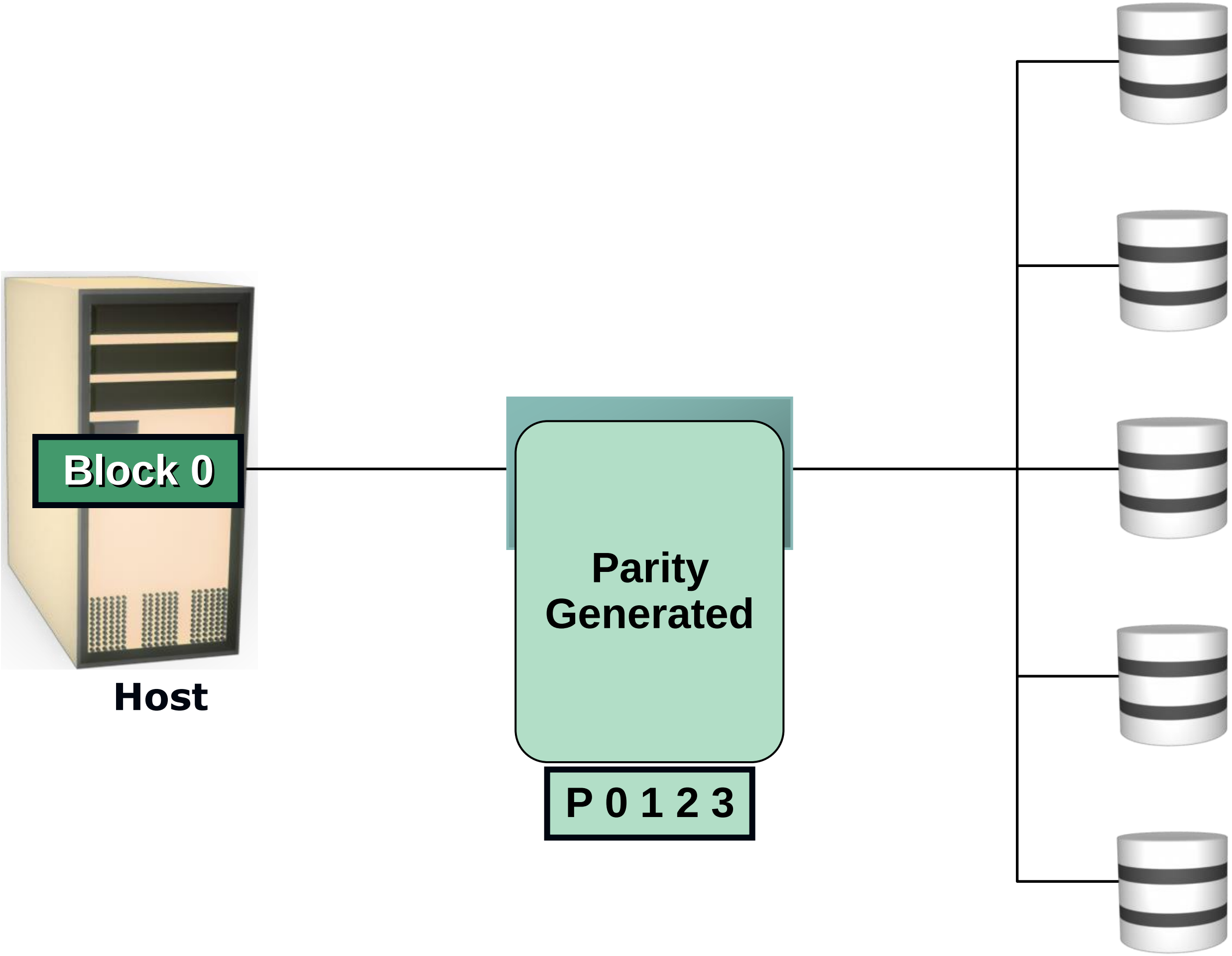


## RAID 3 vs RAID 4 Comparison

These two RAID levels are similar in many ways, but, they still contrast each other in many ways too. Hereunder is a table that clearly shows the difference between these two RAID levels.

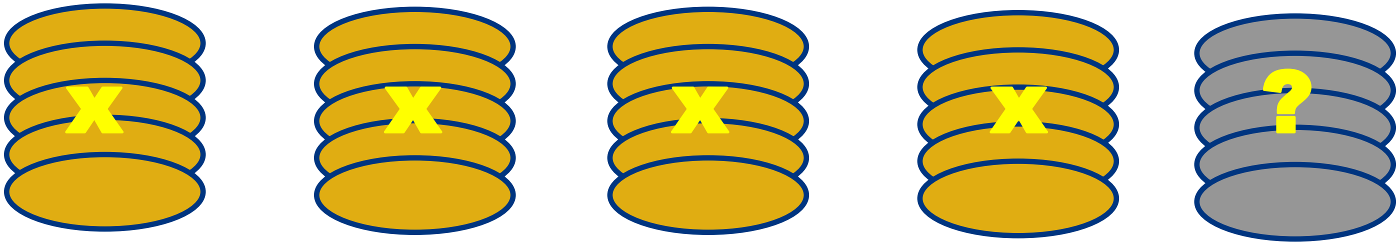
| Comparison | RAID 3 | RAID 4 |
|---|---|---|
| Storage Pattern | Byte-stripping | Block-stripping |
| Parity | Single disk parity | Single disk parity |
| Redundancy | Medium | Medium |
| Cost of Set Up | High | High |
| Modern Relevance | Obsolete | Obsolete |
| Performance and Speed | Fast read, slow write speed | Fast read, slow write speed |
| Fault Tolerance | Single disk failure | Single disk failure |
| Best Application | High data transfer tasks | High data transfer tasks |

# RAID 3



# RAID LEVEL 4 (I)

- Requires N+1 disk drives
  - N drives contain data
    - Individual blocks, not chunks
  - Blocks with same disk address form a ***stripe***



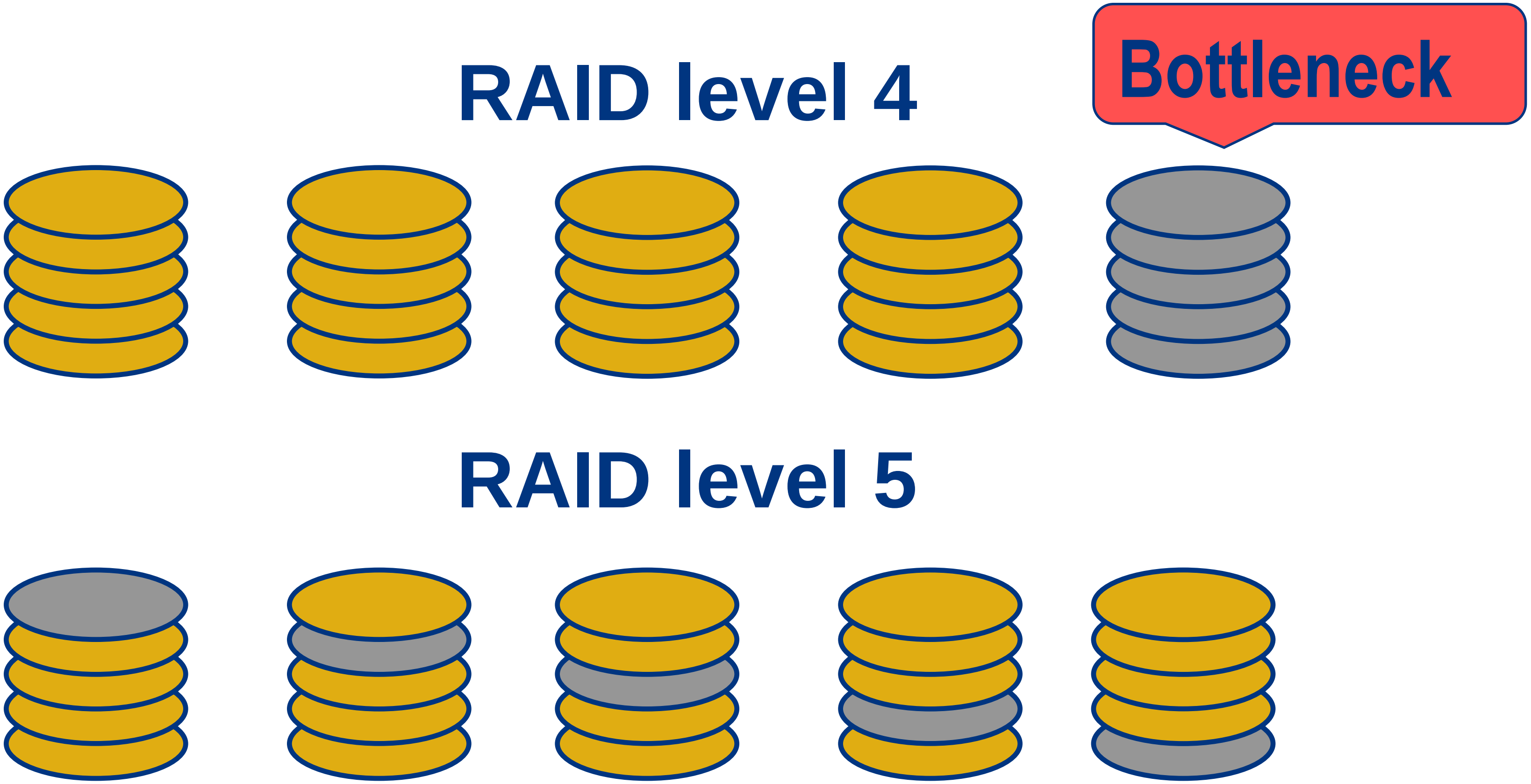
# *RAID LEVEL 4 (II)*

- Parity drive contains **exclusive or** of the *N* blocks in stripe

$$p[k] = b[k] \oplus b[k+1] \oplus \ldots \oplus b[k+N-1]$$

- Parity block now reflects contents of several blocks!
- Can now do parallel reads/writes

# RAID levels 4 and 5



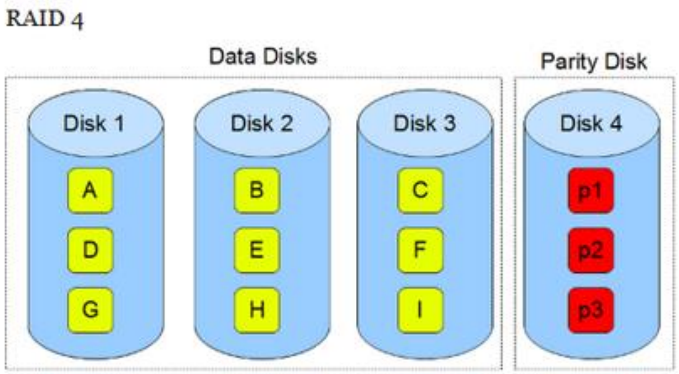
RAID level 4

Bottleneck

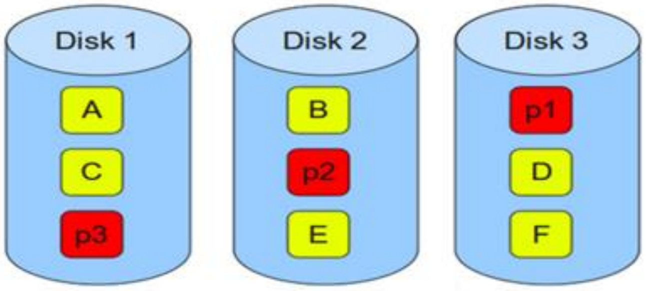
RAID level 5

# RAID LEVEL 5

- Single parity drive of RAID level 4 is involved in every write
  - ***Will limit parallelism***
- RAID-5 distribute the parity blocks among the  N+1 drives
  - ***Much better***



**RAID 4** – Blocks Striped. ( and Dedicated Parity Disk)

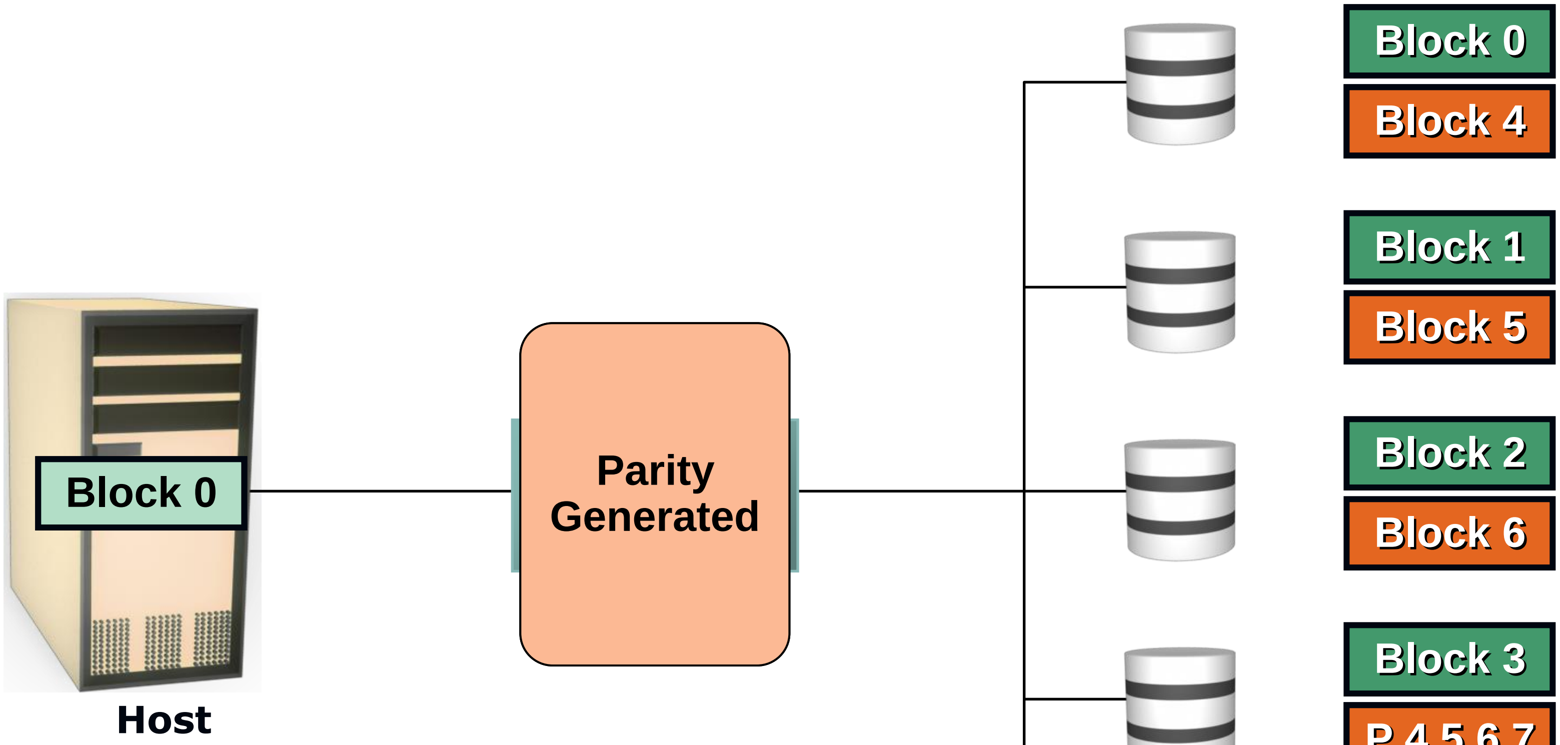


**RAID 5** – Blocks Striped. Distributed Parity.

# RAID 5 and RAID 6

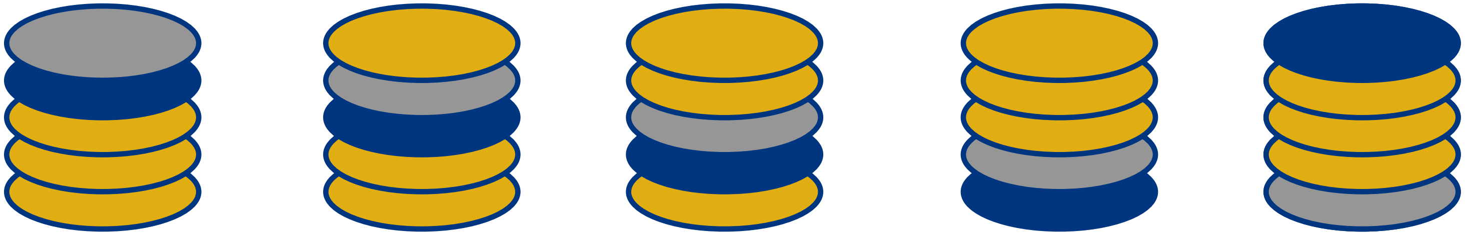
- RAID 5 is similar to RAID 4, except that the parity is distributed across all disks instead of stored on a dedicated disk.
  - This overcomes the write bottleneck on the parity disk.
- RAID 6 is similar to RAID 5, except that it includes a second parity element to allow survival in the event of two disk failures.
  - The probability for this to happen increases and the number of drives in the array increases.
  - Calculates both horizontal parity (as in RAID 5) and diagonal parity.
  - Has more write penalty than in RAID 5.
  - Rebuild operation may take longer than on RAID 5.

# RAID 5


Block 0
Host
Parity Generated
Block 0
Block 4
Block 1
Block 5
Block 2
Block 6
Block 3
P 4 5 6 7
P 0 1 2 3
Block 7

# Other RAID organizations (I)

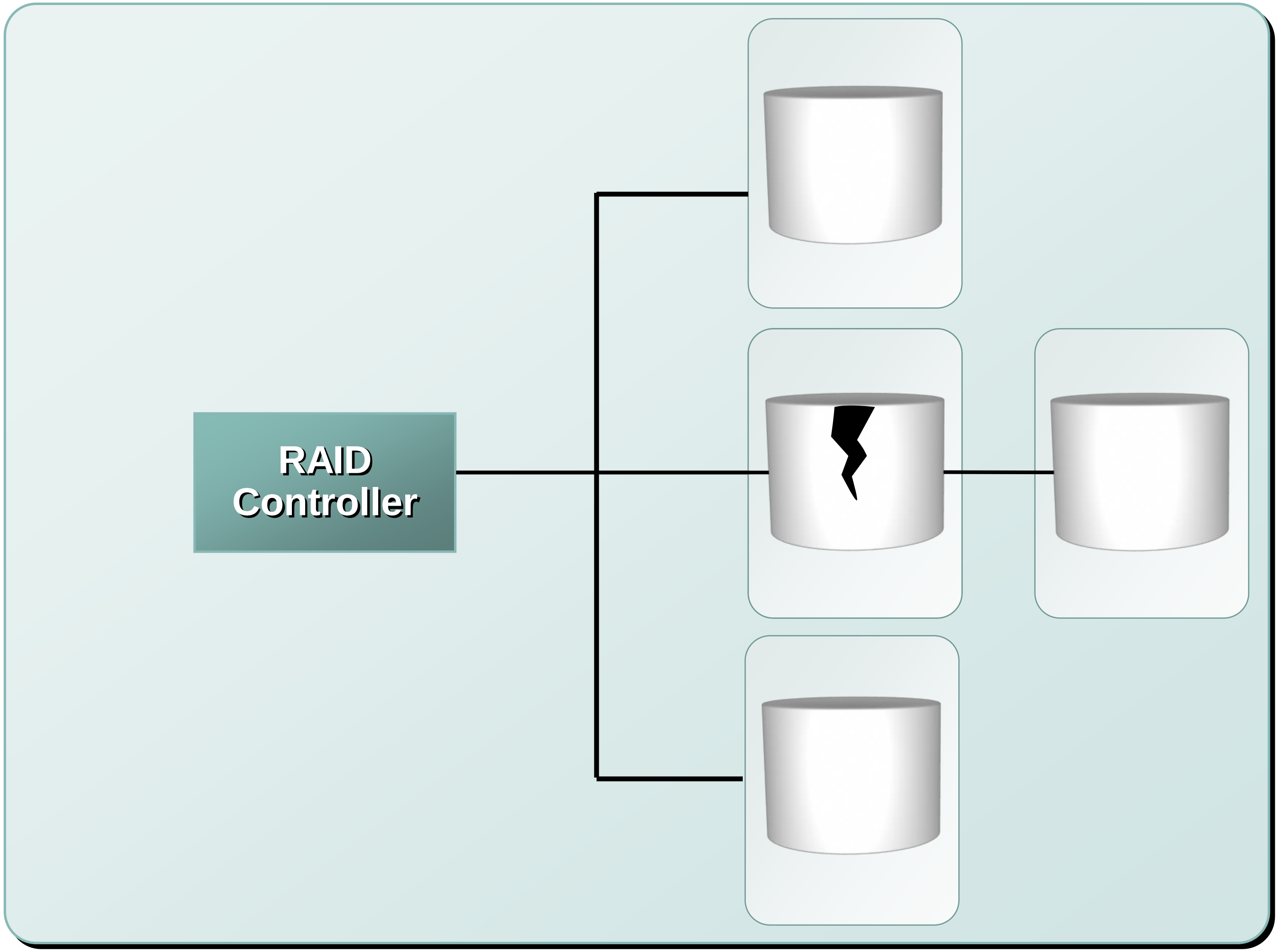
- **RAID 6:**
  - Two check disks
  - Tolerates two disk failures
  - More complex updates



# RAID Comparison

| RAID | Min Disks | Storage Efficiency % | Cost | Read Performance | Write Performance |
|---|---|---|---|---|---|
| 0 | 2 | 100 | Low | Very good for both random and sequential read | Very good |
| 1 | 2 | 50 | High | Good<br>Better than a single disk | Good<br>Slower than a single disk, as every write must be committed to two disks |
| 3 | 3 | (n-1)*100/n where n= number of disks | Moderate | Good for random reads and very good for sequential reads | Poor to fair for small random writes<br>Good for large, sequential writes |
| 5 | 3 | (n-1)*100/n where n= number of disks | Moderate | Very good for random reads<br>Good for sequential reads | Fair for random write<br>Slower due to parity overhead<br>Fair to good for sequential writes |
| 6 | 4 | (n-2)*100/n where n= number of disks | Moderate but more than RAID 5 | Very good for random reads<br>Good for sequential reads | Good for small, random writes<br>(has write penalty) |
| 1+0 and 0+1 | 4 | 50 | High | Very good | Good |

# Hot Spares



If one of the drives within the disk fail, hot spare is ready to go, seamlessly taking over the functions of the failed drive without any action or intervention by the server's administrator.

A Cold Spare is a spare part that will require physical intervention by the user to engage the device.

# *What about flash drives?*

- Having no moving parts should mean ***fewer failures***?
  - Failures still happen
  - Flash drives age as they are written to
  - Irrecoverable red errors occur (at least as frequently as in magnetic disks?)
- Pure Storage uses a proprietary 3D-RAID organization for their SSD stores

# *CONCLUSION (I)*

- RAID original purpose ***was*** to take advantage of Winchester drives that were smaller and cheaper than conventional disk drives
  - Replace a single drive by an array of smaller drives
- ***Current purpose*** is to build fault-tolerant file systems that do not need backups

# *CONCLUSION (II)*

- Low cost of disk drives made RAID level 1 attractive for small installations
- Otherwise pick
  - RAID level 6 for ***higher protection***
    - Can tolerate ***one disk failure and irrecoverable read errors***

# A review question

- Consider an array consisting of four 750 GB disks
- What is the storage capacity of the array if we organize it
  - As a RAID level 0 array?
  - As a RAID level 1 array?
  - As a RAID level 5 array?

# The answers

- Consider an array consisting of four 750 GB disks
- What is the storage capacity of the array if we organize it

| | |
|---|---|
| As a RAID level 0 array? | 3 TB |
| As a RAID level 1 array? | 1.5 TB |
| As a RAID level 5 array? | 2.25 TB |

# Summary

- Disks can cause reliability and performance problems
- To mitigate such problems we can adopt “multiple disks” and accordingly gain:
  1. More capacity
  2. Redundancy
  3. Concurrency
- To achieve only redundancy we can apply mirroring
- To achieve only concurrency we can apply striping
- To achieve redundancy *and* concurrency we can apply RAID (e.g., levels 2, 3, 4 or 5)

# QUIZ

## RAID Levels

1. **RAID 0 (Striping):**
    - **Purpose:** High performance.
    - **No Redundancy:** If one drive fails, data is lost.
    - **Use Case:** Non-critical systems requiring high speed.
2. **RAID 1 (Mirroring):**
    - **Purpose:** Redundancy.
    - **Data Safety:** Copies data to multiple drives.
    - **Use Case:** Critical systems with moderate performance needs.
3. **RAID 5 (Striping with Parity):**
    - **Purpose:** Balance of redundancy and performance.
    - **Fault Tolerance:** Can sustain one drive failure.
    - **Use Case:** Databases or file servers.

# QUIZ

1. **RAID 6 (Striping with Dual Parity):**
   - **Purpose:** Enhanced redundancy.
   - **Fault Tolerance:** Can sustain two drive failures.
   - **Use Case:** Large storage systems with high fault tolerance.
2. **RAID 10 (1+0):**
   - **Purpose:** Combines RAID 1 (mirroring) and RAID 0 (striping).
   - **High Cost:** Requires at least four drives.
   - **Use Case:** High-performance systems needing redundancy.
3. **RAID 50 and 60:**
   - **Purpose:** Nested RAID levels offering more performance and redundancy.
   - **Use Case:** Enterprise-level storage.

# Solution

- **How many drives are required for each RAID level?**

- **Solution:**
  - RAID 0: Minimum 2 drives.
  - RAID 1: Minimum 2 drives.
  - RAID 5: Minimum 3 drives.
  - RAID 6: Minimum 4 drives.
  - RAID 10: Minimum 4 drives.

- **5. What is the usable storage capacity of RAID arrays?**

- **Solution:**
  - **RAID 0:** Total capacity = Sum of all drives.
  - **RAID 1:** Total capacity = Capacity of one drive.
  - **RAID 5:** Total capacity = (N-1) x Drive Capacity (where N = number of drives).
  - **RAID 6:** Total capacity = (N-2) x Drive Capacity.
  - **RAID 10:** Total capacity = 50% of total drive capacity.

**1. Problem:**

A RAID 10 array has 8 disks of 1 TB each. What is the usable storage capacity and fault tolerance?

**2. Problem:**

A RAID 6 array consists of 6 disks, each 10 TB. What is the total usable storage, and how many disk failures can it tolerate?

**3. Problem:**

A RAID 10 array has 8 disks of 1 TB each. What is the usable storage capacity and fault tolerance?

**4. Problem:**

Explain why RAID is not a substitute for backups.

**5. Problem**

Calculate the probability of failure in a RAID 6 array with 5 disks, each with a 2% failure rate over a year.

**6. Rebuilding a RAID 5 Array**

- RAID 5 setup with 4 disks: Disk A, Disk B, Disk C, and Disk D (parity disk rotates).
- Each disk contains 4 blocks of data.
- Disk A fails, and we need to rebuild its data using the parity information.

**Data Structure Before Failure:**

| Block | Disk A | Disk B | Disk C | Disk D (Parity) |
|---|---|---|---|---|
| 1 | 5 | 7 | 3 | P1 |
| 2 | 2 | 4 | 6 | P2 |
| 3 | 1 | 8 | 2 | P3 |
| 4 | 9 | 3 | 7 | P4 |

# Solutions

**1. Problem:**

A RAID 10 array has 8 disks of 1 TB each. What is the usable storage capacity and fault tolerance?

**Solution:**

Usable storage = Half the total disk size = (8 / 2) × 1 TB = 4 TB.

Fault tolerance: Can tolerate multiple disk failures as long as no mirrored pair is lost.

**2. Problem:**

A RAID 6 array consists of 6 disks, each 10 TB. What is the total usable storage, and how many disk failures can it tolerate?

**Solution:**

- Usable storage = (Number of disks - 2) × Disk size = (6 - 2) × 10 TB = 40 **TB**.
- Tolerates up to **2 disk failures**.

**3. Problem:**

A RAID 10 array has 8 disks of 1 TB each. What is the usable storage capacity and fault tolerance?

**Solution:**

- Usable storage = Half the total disk size = (8 / 2) × 1 TB = **4 TB**.
- Fault tolerance: Can tolerate multiple disk failures as long as no mirrored pair is lost.

**4. Problem:**

Explain why RAID is not a substitute for backups.

**Solution:**

- RAID protects against hardware failures but not against data corruption, accidental deletion, or malware. Backups provide an independent data copy for recovery.

**5. Problem:**

Calculate the probability of failure in a RAID 6 array with 5 disks, each with a 2% failure rate over a year.

**Solution:**

- Probability of a single disk failing = 0.02.
- Probability of two simultaneous failures ≈ (0.02 × 0.02) = **0.0004 or 0.04%**.

# Solutions

- Parity Calculation (XOR for each block):
  $P1 = A1 \oplus B1 \oplus C1$
  $P2 = A2 \oplus B2 \oplus C2$, and so on.

- Rebuilding Disk A (Example for Block 1):

- Known values:
  $B1=7, C1=3, P1=1$

- Calculate missing A1:
  $A1=P1 \oplus B1 \oplus C1$
  $A1=1 \oplus 7 \oplus 3$
  $A1=5$.

- Repeat this calculation for all blocks to restore Disk A.