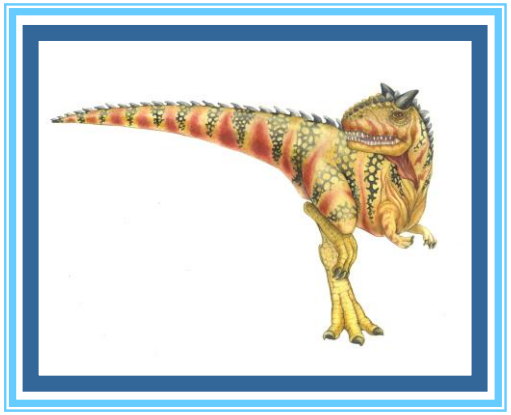
Chapter 11: Mass-Storage Systems



Chapter 11: Mass-Storage Systems

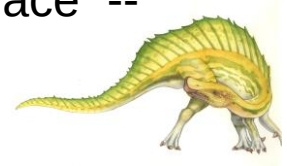
- Overview of Mass Storage Structure
- Disk Structure
- Disk Attachment
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure
- Stable-Storage Implementation

Objectives

- To describe the physical structure of secondary storage devices and its effects on the uses of the devices
- To explain the performance characteristics of mass-storage devices
- To evaluate disk scheduling algorithms
- To discuss operating-system services provided for mass storage, including RAID

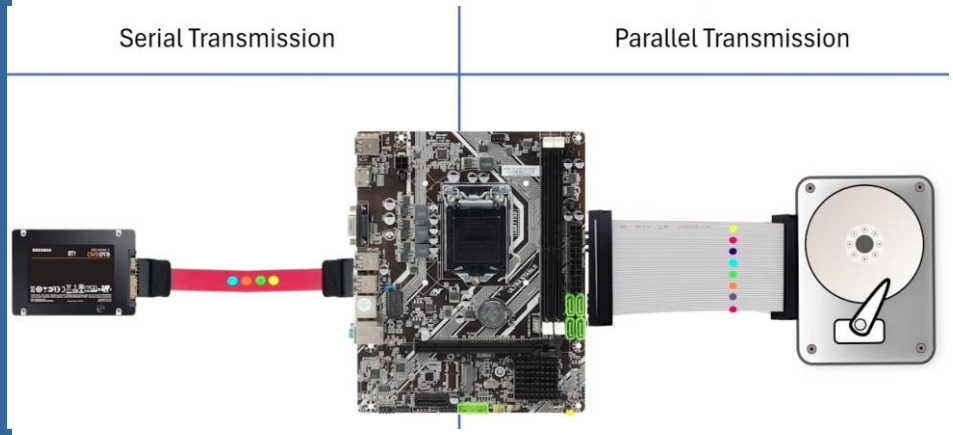
Overview of Mass Storage Structure

- **Disk Structure and Operation**
  - **Magnetic Disks:** **These disks store data magnetically on rotating platters. They provide most of the secondary storage (i.e., long-term storage) in computers.**
  - **Rotation Speed:** **Disks typically rotate between 60 and 250 times per second (equivalent to 3,600 to 15,000 RPM).**
  - **Data Transfer Rate:** **This is the rate at which data flows between the disk drive and the computer. Faster transfer rates mean quicker access to data.**
- **Positioning Time (Random-Access Time):** **Positioning Time is the total time required to access a specific piece of data on the disk:**
  - **Seek Time:** **The time to move the disk arm (which holds the read/write head) to the correct cylinder (a set of tracks across platters).**
  - **Rotational Latency:** **The time for the desired sector to rotate under the read/write head once the arm is positioned.**
- **Head crash** results from disk head making contact with the disk surface -- That's bad

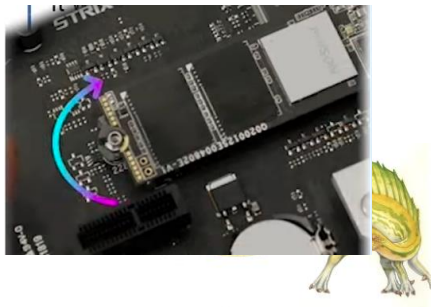


Overview of Mass Storage Structure

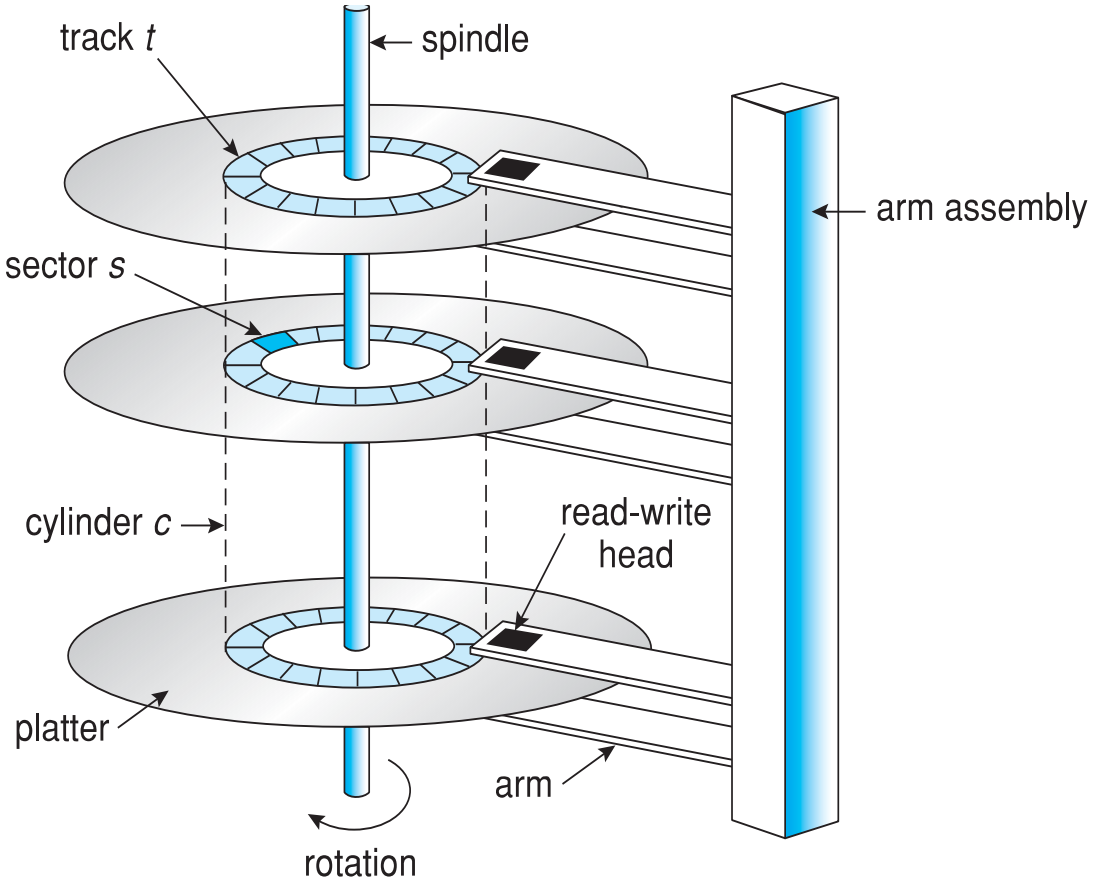
- Disks can be removable, ex: **External hard drives**
- Disk Connection and **I/O Buses**: Disks connect to the computer via I/O buses, which transfer data between the disk drive and the computer's memory and processors. Different types of buses include:
  - **EIDE/ATA, SATA:** Common in consumer desktop and laptop hard drives.
  - **USB:** Common for external hard drives.
  - **Fibre Channel, SCSI, SAS:** High-speed interfaces often used in servers and enterprise storage solutions.
  - **Firewire:** Less common but sometimes used for external drives.
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array



Small Computer System Interface (SCSI)	Parallel Advanced Technology Attachment (PATA)	Serial Advanced Technology Attachment (SATA)	NVMe (Non-volatile Memory Express)
It was developed in the mid-70s as a common interface to connect scanners, printers, and hard disks to the motherboard. It can transfer data up to 80 Mbps. It is no longer used as an interface to connect a hard disk to the motherboard.	It was launched in 1986 and was widely used in computers until the early 2000s. It is also known as IDE. It uses parallel data transmission technology. It offers a maximum data transfer speed of 133 Mbps. It is also no longer used in personal computers.	It was developed as a replacement for PATA. It was launched in 2003. It uses serial transmission. All modern computers use it to connect HDD and SATA SSD to the motherboard. It can offer a maximum data transfer rate of 6 Gbps.	It is explicitly developed for SSD drives. It does not use a cable. It attaches the SSD drive directly to the motherboard on a M dot 2 slot. It can transfer data at a speed of 32 Gbps. It does not support HDD.



Moving-head Disk Mechanism



Hard Disks

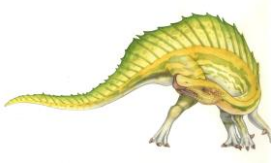
- Platters range from .85” to 14” (historically)
  - Commonly 3.5”, 2.5”, and 1.8”
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency
    - The data will be about halfway around the disk, not a full revolution away.

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

(From Wikipedia)

Hard Disk Performance

- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- **Average I/O time** = average access time + (amount to transfer / transfer rate) + controller overhead
- For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead :
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms



Calculate the **average I/O time** for transferring a **4KB block** of data

- **Given Information**
  - **Disk Speed**: 7200 RPM (Revolutions Per Minute)
  - **Average Seek Time**: 5 milliseconds (ms)
  - **Data Transfer Rate**: 1 Gb/sec (1 gigabit per second)
  - **Controller Overhead**: 0.1 ms

$$\text{Average Latency} = \frac{60}{2 \times \text{RPM}}$$

Plugging in the RPM:

$$\text{Average Latency} = \frac{60}{2 \times 7200} = 4.17 \text{ ms}$$

- Transfer Time Calculation

1. **Convert 4KB to bits** since the transfer rate is given in **bits per second**:

$$4\ \text{KB} = 4 \times 1024\ \text{bytes} = 4096\ \text{bytes}$$

$$4096\ \text{bytes} = 4096 \times 8\ \text{bits} = 32,768\ \text{bits}$$

2. **Calculate Transfer Time**: Using the transfer rate of **1 Gb/sec** (1,000,000,000 bits/sec):

$$\text{Transfer Time} = \frac{32,768\ \text{bits}}{1,000,000,000\ \text{bits/sec}}$$

$$\text{Transfer Time} = 0.0328\ \text{ms}$$

$$\text{Average I/O Time} = 5\ \text{ms (Seek Time)} + 4.17\ \text{ms (Latency)} + 0.1\ \text{ms (Controller Overhead)} + 0.0328\ \text{ms (Transfer Time)}$$

$$\text{Average I/O Time} \approx 9.3028\ \text{ms}$$

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Solid-State Disks

- Nonvolatile memory used like a hard drive
  - Many technology variations
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity
- But much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Why traditional HHDs are still widely used?

1. **Cost-Effectiveness**: Applications requiring substantial storage, such as data centers or for personal use in storing large multimedia libraries.

2. **Higher Storage Capacities**: vast amounts of data need to be stored economically.

3. **Lifespan for Write-Intensive Applications**: SSDs have limited write cycles due to the nature of flash memory, which wears down over time. HDDs, though they can also wear out, generally have a longer lifespan in write-intensive environments, making them suitable for specific workloads.

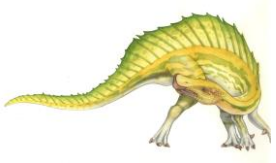
4. **Compatibility and Legacy Systems**: Many older systems are built for HDDs, and upgrading to SSDs may not be feasible or cost-effective for some organizations, especially when performance needs are moderate.

5. **Reliability for Sequential Data Access**: HDDs are still effective for sequential data access (e.g., reading or writing large files in one go). Some systems, like video surveillance, still use HDDs for this reason, as they handle continuous, large data writes without the high cost of SSDs.

6. **Backup and Archiving**: For long-term storage where access speed is less critical, HDDs are often preferred. They offer affordable, high-capacity storage for backup, archiving, and data recovery solutions.

Magnetic Tape

- Was early secondary-storage medium
  - Evolved from open spools to cartridges
- Relatively permanent and holds large quantities of data
- Access time slow
- Random access ~1000 times slower than disk
- Mainly used for backup, storage of infrequently-used data, transfer medium between systems
- Kept in spool and wound or rewound past read-write head
- Once data under head, transfer rates comparable to disk
  - 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-{3,4,5} and T10000



Disk Structure

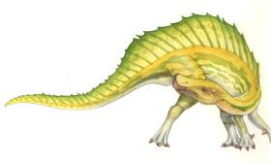
- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
  - Low-level formatting creates **logical blocks** on physical media
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost cylinder
  - Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost
  - Logical to physical address should be easy
    - Except for bad sectors
    - Non-constant # of sectors per track via constant angular velocity

Disk Attachment

- Host-attached storage accessed through I/O ports talking to I/O busses
- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC (Fiber Channel) is high-speed serial architecture
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SANs)** in which many hosts attach to many storage units
- I/O directed to bus ID, device ID, logical unit (LUN)

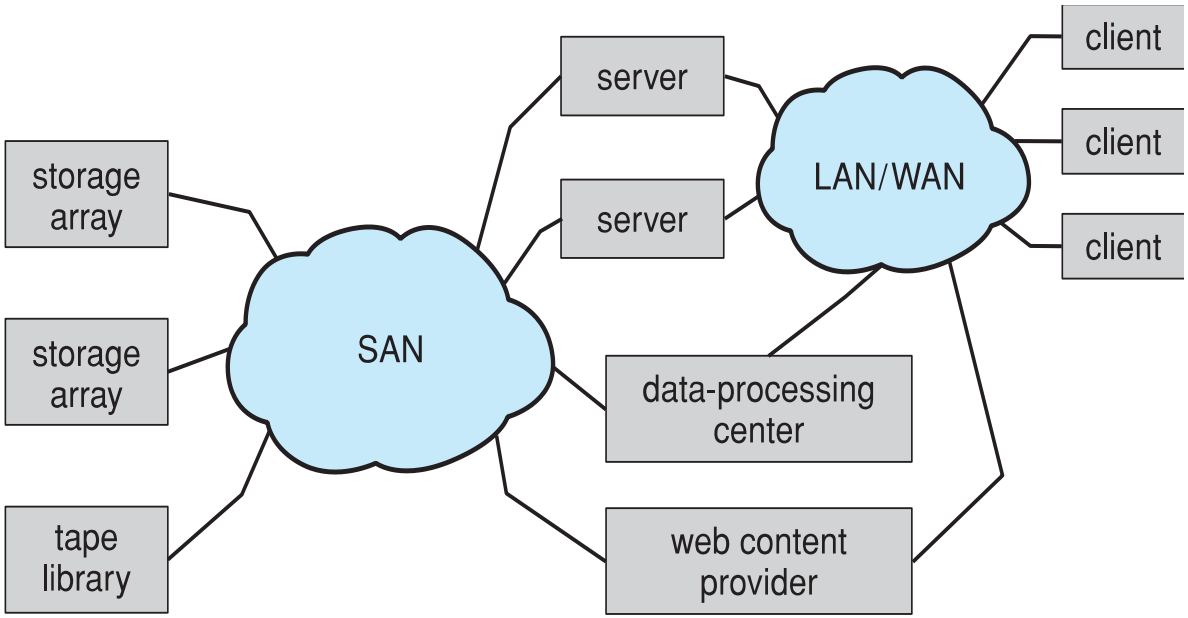
Storage Array

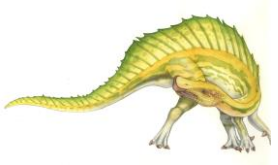
- Can just attach disks, or arrays of disks
- Storage Array has controller(s), provides features to attached host(s)
  - Ports to connect hosts to array
  - Memory, controlling software (sometimes NVRAM, etc)
  - A few to thousands of disks
  - RAID, hot spares, hot swap (discussed later)
  - Shared storage -> more efficiency
  - Features found in some file systems
    - Snapshots, clones, thin provisioning, replication, deduplication, etc



Storage Area Network

- Common in large storage environments
- Multiple hosts attached to multiple storage arrays - flexible



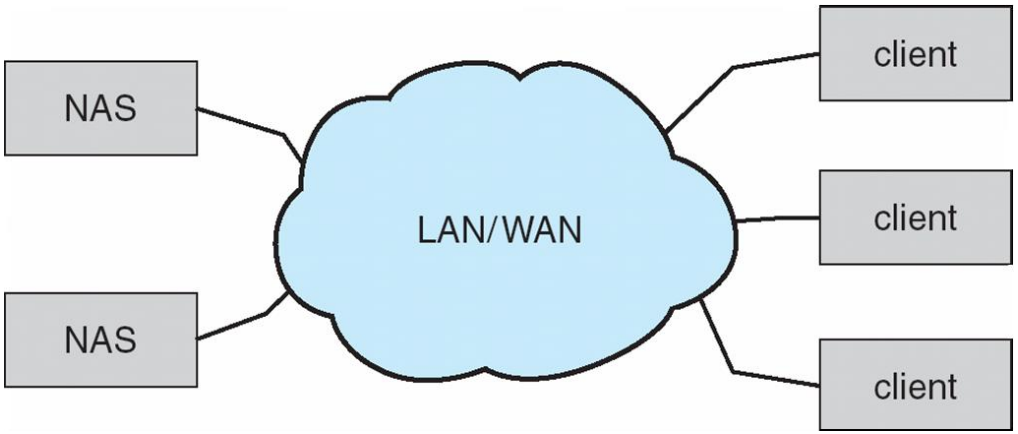


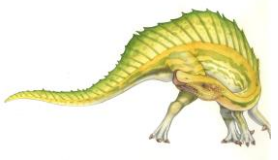
Storage Area Network (Cont.)

- SAN is one or more storage arrays
  - Connected to one or more Fibre Channel switches
- Hosts also attach to the switches
- Storage made available via **LUN Masking (Loagical Unit Number Masking)** from specific arrays to specific servers
- Easy to add or remove storage, add new host and allocate it storage
  - Over low-latency Fibre Channel fabric: The Fibre Channel network operates at high speeds with very low delays (**low latency**)
- Why have separate storage networks and communications networks?
  - iSCSI: This allows SANs to run over standard networks (like Ethernet) - a cheaper option.
  - FCoE (Fibre Channel over Ethernet): combines Fibre Channel and Ethernet, allowing both storage and regular data to share the same network but still use high-speed Fibre Channel for storage tasks..

Network-Attached Storage

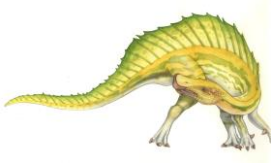
- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
  - Remotely attaching to file systems
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network
- **iSCSI** protocol uses IP network to carry the SCSI protocol
  - Remotely attaching to devices (blocks)





Disk Scheduling

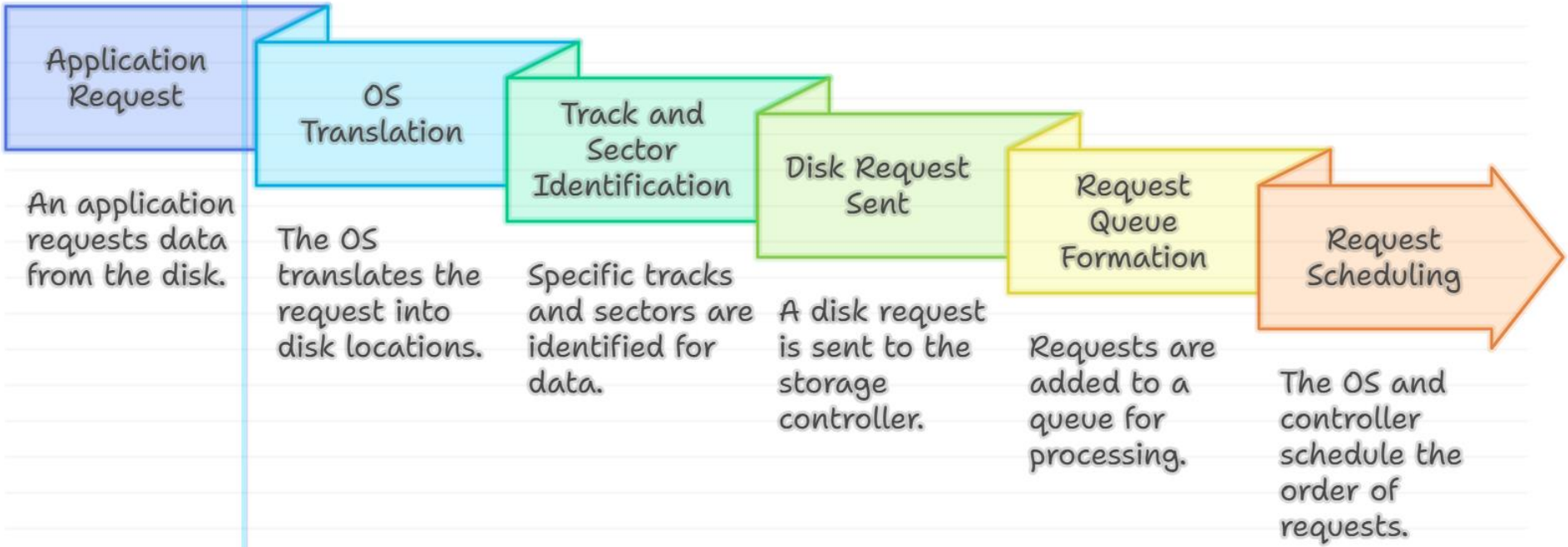
- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- Minimize seek time
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

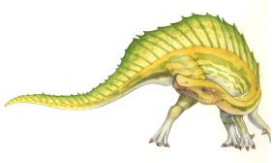


Disk Scheduling (Cont.)

- There are many sources of disk I/O request
  - OS
  - System processes
  - Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer
- OS maintains queue of requests, per disk or device
- Idle disk can immediately work on I/O request, busy disk means work must queue
  - Optimization algorithms only make sense when a queue exists

Track Number Generation Process





Disk Scheduling (Cont.)

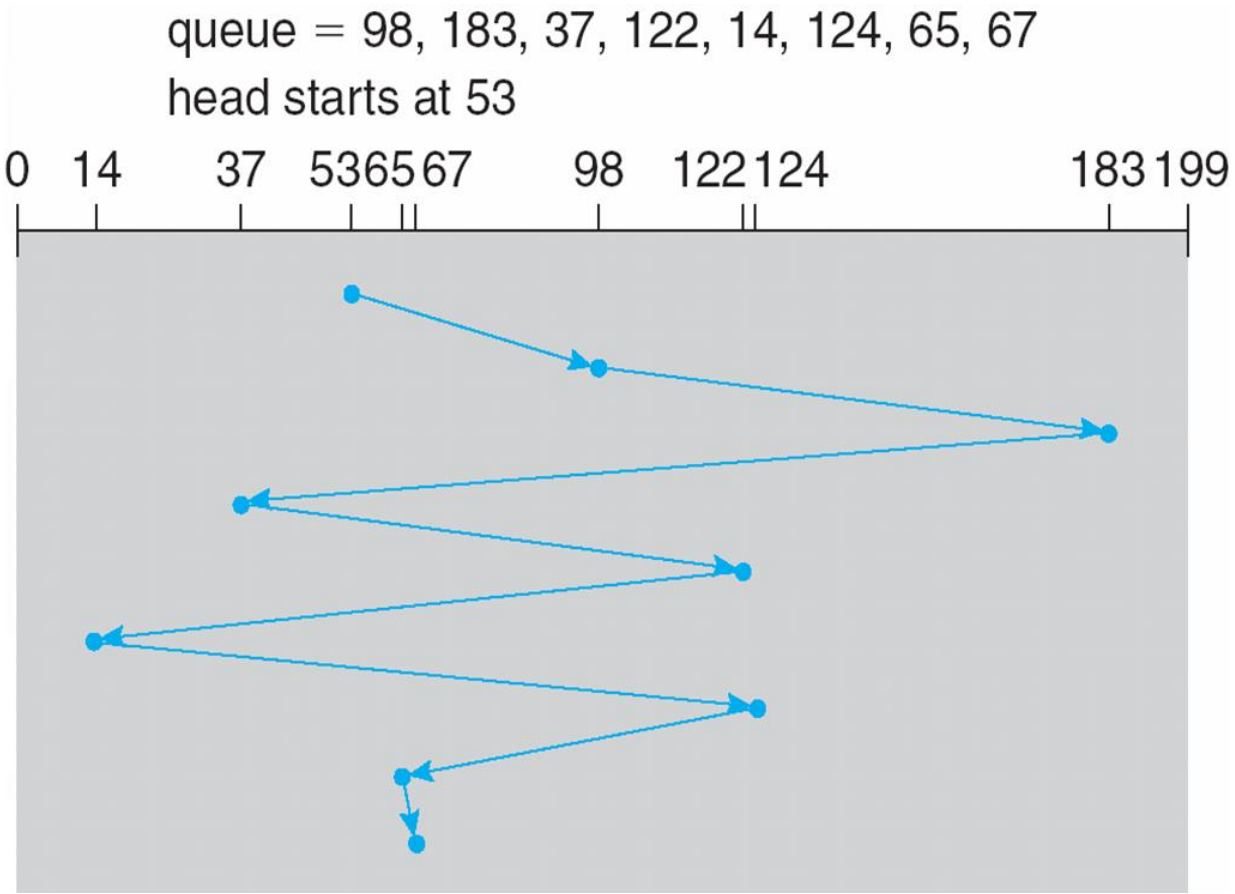
- Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

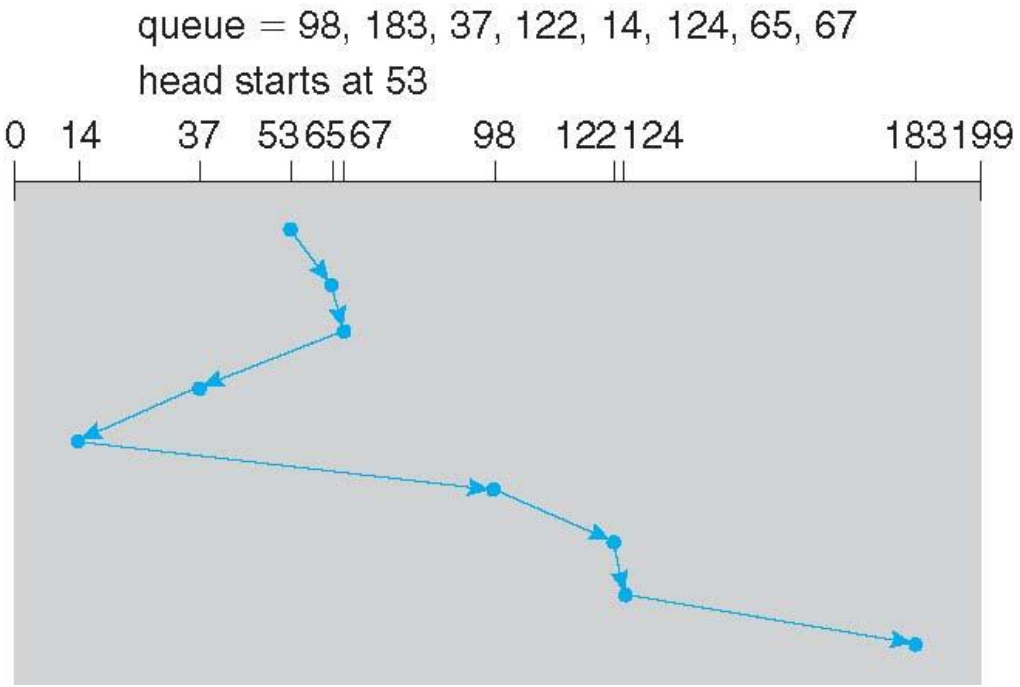
FCFS

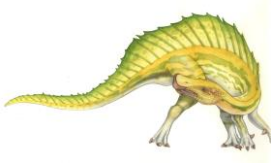
Illustration shows total head movement of 640 cylinders



SSTF

- Shortest Seek Time First selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders





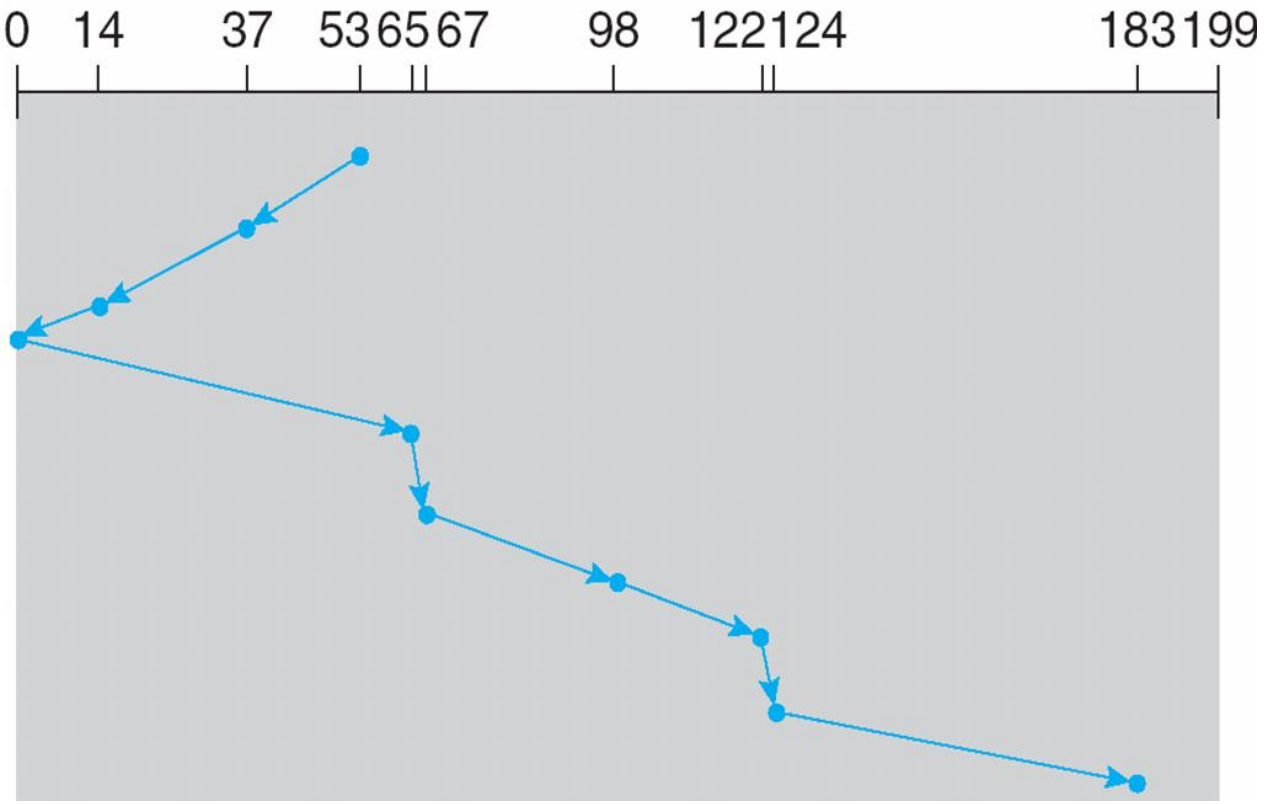
SCAN

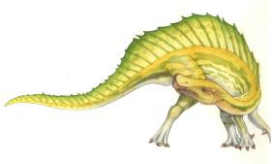
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 236 cylinders
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

SCAN (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53





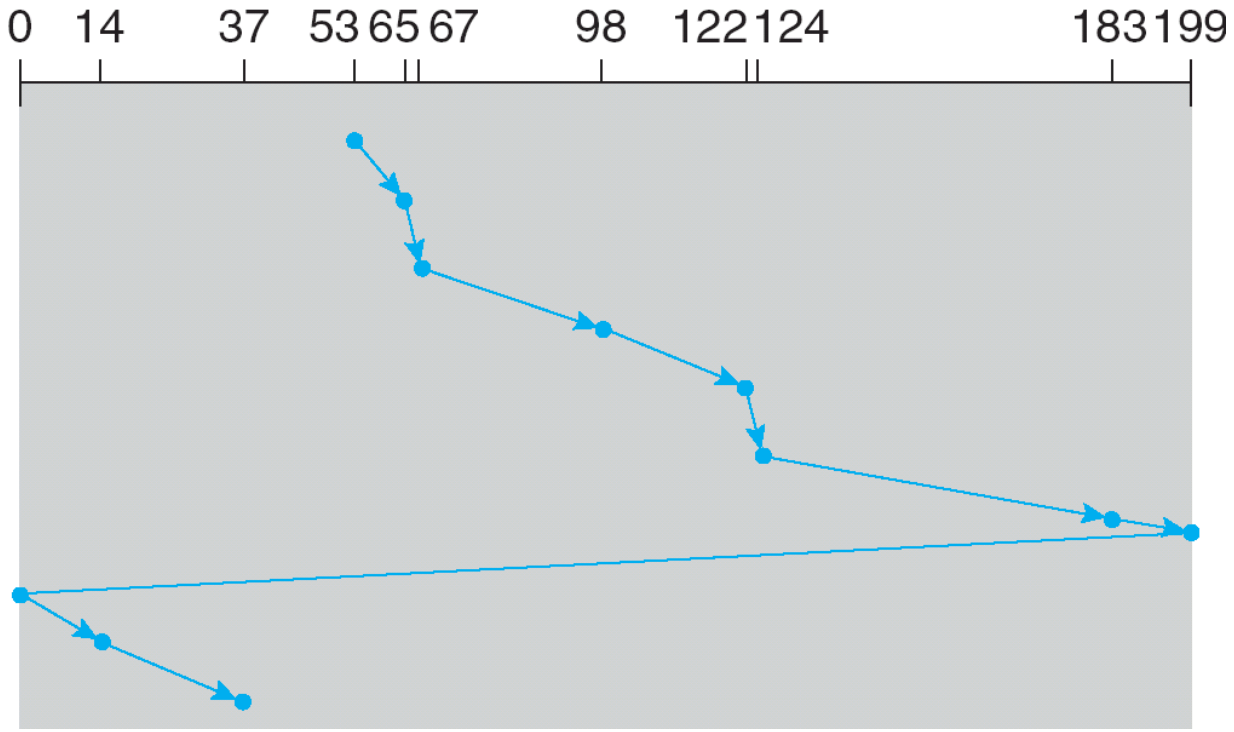
C-SCAN

- Provides a more uniform wait time than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one
- Total number of cylinders?

C-SCAN (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53



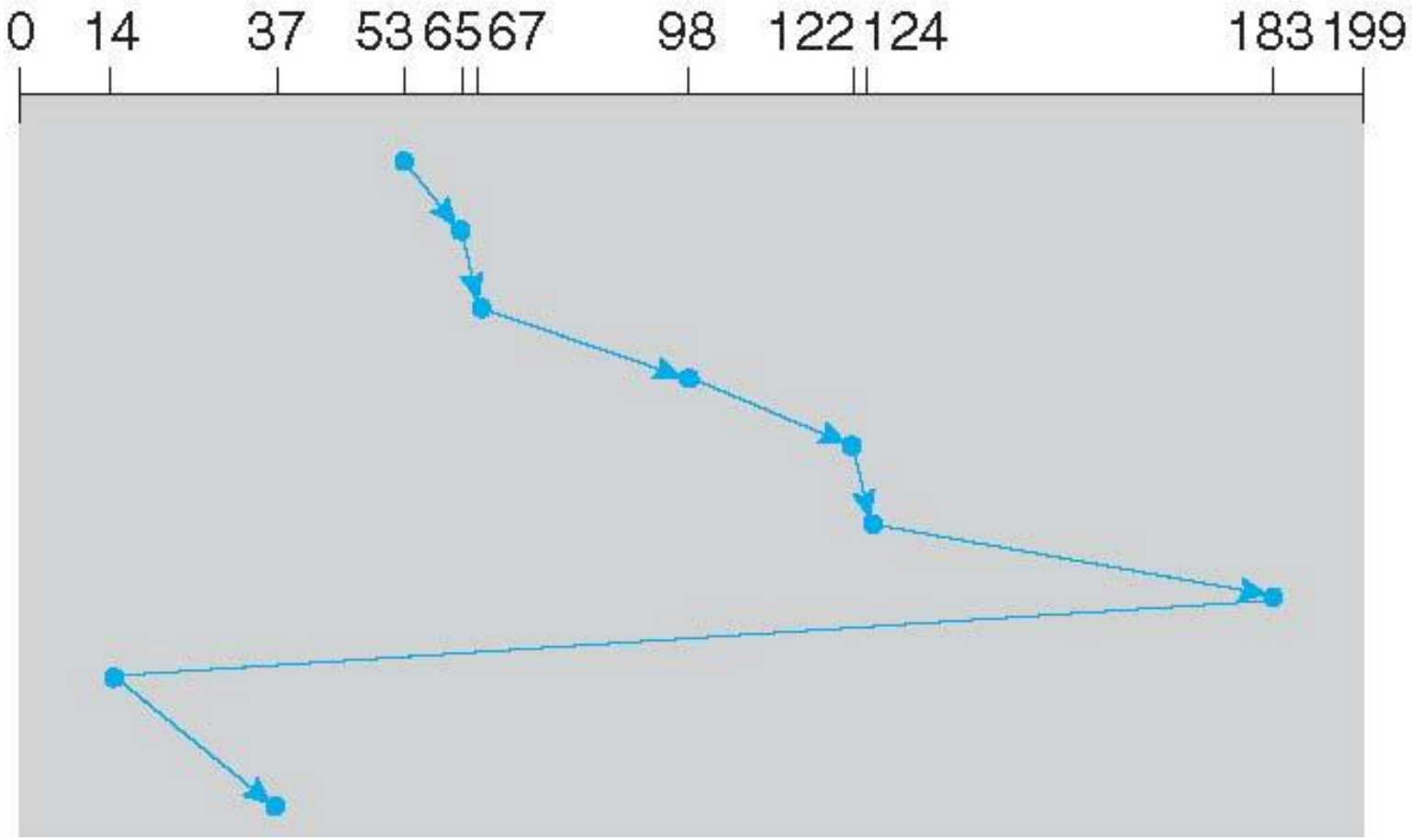
C-LOOK

- LOOK a version of SCAN, C-LOOK a version of C-SCAN
- Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk
- Total number of cylinders?

C-LOOK (Cont.)

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

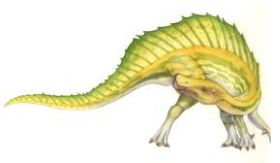


Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a heavy load on the disk
  - Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
  - And metadata layout
- The disk-scheduling algorithm should be **written as a separate module** of the operating system, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- What about rotational latency?
  - Difficult for OS to calculate
- How does disk-based queueing effect OS queue ordering efforts?
  - Reduce seek times and increase throughput, limit the OS's precise control over request order

Disk Management

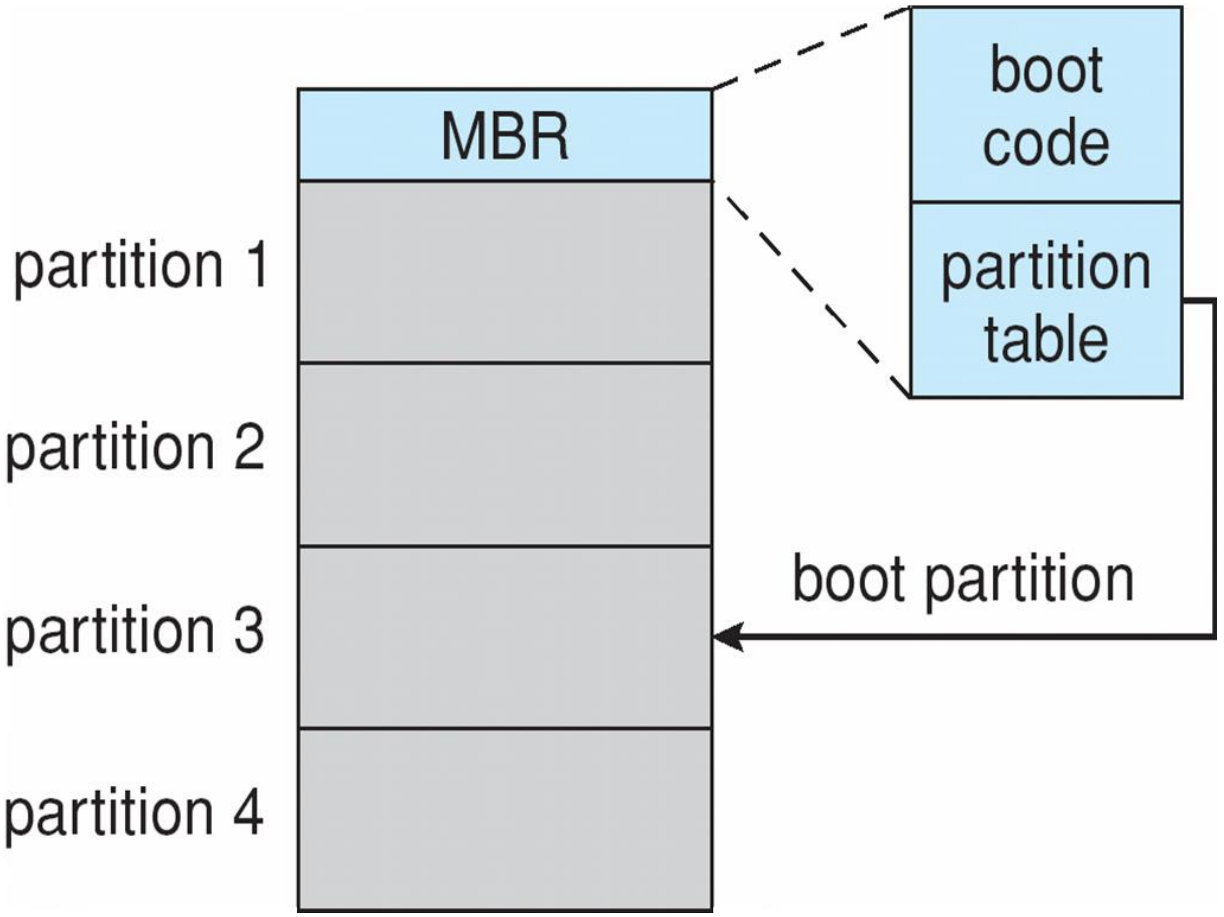
- **Low-level formatting**, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write
  - Each sector can hold header information, plus data, plus error correction code (**ECC**)
  - Usually 512 bytes of data but can be selectable
- To use a disk to hold files, the operating system still needs to record its own data structures on the disk
  - **Partition** the disk into one or more groups of cylinders, each treated as a logical disk
  - **Logical formatting** or "making a file system"
  - To increase efficiency most file systems group blocks into **clusters**
    - Disk I/O done in blocks: operates in small blocks, suitable for low-level operations directly with the hardware.
    - File I/O done in clusters: operates in larger clusters, which are more efficient for reading and writing entire files.

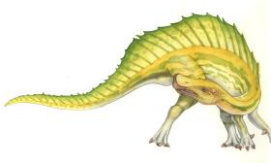


Disk Management (Cont.)

- Raw disk access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM
  - **Bootstrap loader** program stored in boot blocks of boot partition
- Methods such as **sector sparing** used to handle bad blocks

Booting from a Disk in Windows

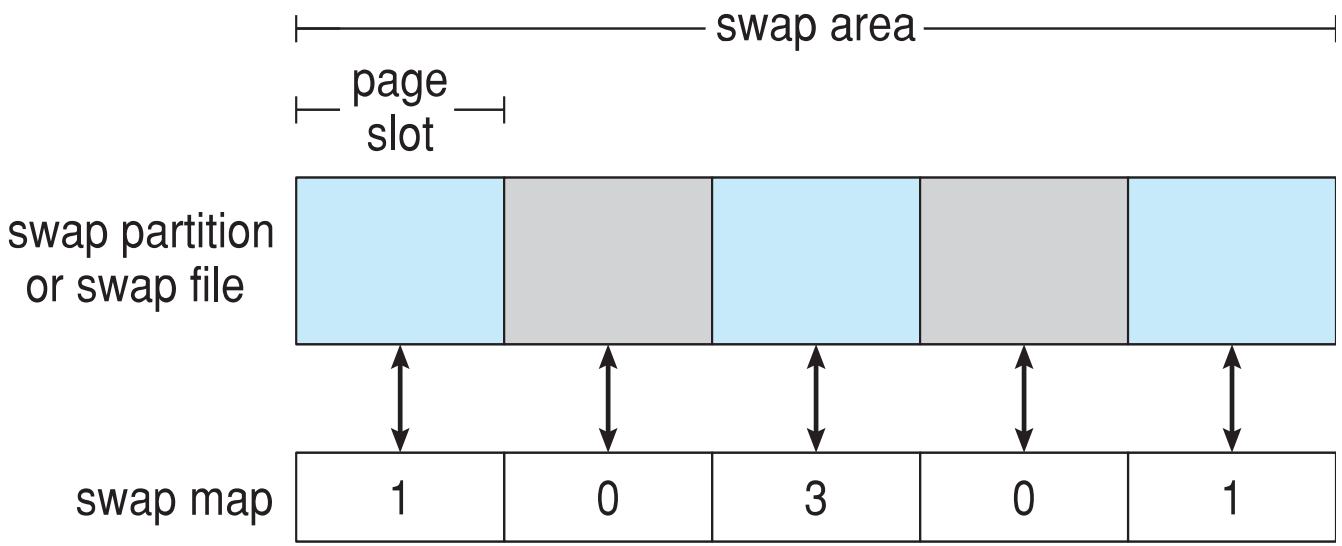




Swap-Space Management

- Swap-space — Virtual memory uses disk space as an extension of main memory
  - Less common now due to memory capacity increases
- Swap-space can be carved out of the normal file system, or, more commonly, it can be in a separate disk partition (raw)
  - part of the disk purely for swap OR just another file in your main storage
- Swap-space management
  - 4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment
  - Kernel uses **swap maps** to track swap-space use
  - Solaris 2 allocates swap space only when a dirty page is forced out of physical memory, not when the virtual memory page is first created
    - File data written to swap space until write to file system requested
    - Other dirty pages go to swap space due to no other home
    - Text segment pages thrown out and reread from the file system as needed
- What if a system runs out of swap space?
  - Some systems allow multiple swap spaces

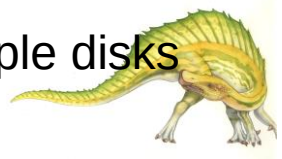
Data Structures for Swapping on Linux Systems



A counter of **3**, meaning this page slot contains data that three different processes are using (perhaps a shared library)

RAID Structure

- RAID – redundant array of independent disks
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure**
- **Mean time to repair –** exposure time when another failure could cause data loss
- **Mean time to data loss** based on above factors
- The ***mean time between failures*** (*MTBF*) of a single disk is 100,000 hours. Then the MTBF of some disk in an array of 100 disks will be 100,000/100 = 1,000 hours, or 41.66 days,
- If mirrored disks fail independently, consider disk with 100,000 mean time to failure and 10 hour mean time to repair
  - Mean time to data loss is $100,000^2 / (2 * 10) = 500 * 10^6$ hours, or 57,000 years!
- Frequently combined with **NVRAM** to improve write performance
  - temporarily storing data in the NVRAM, the controller can gather enough data to write an entire block (called a "stripe") across all drives in one go
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

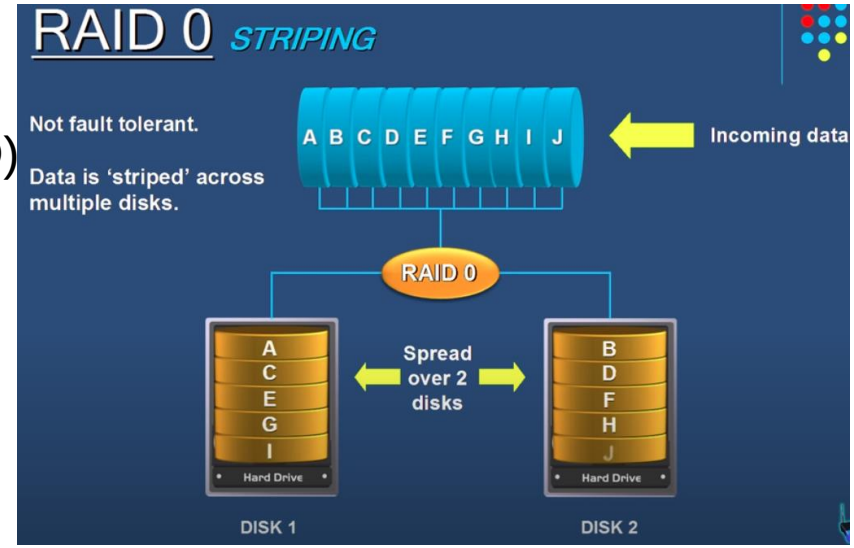


RAID (Cont.)

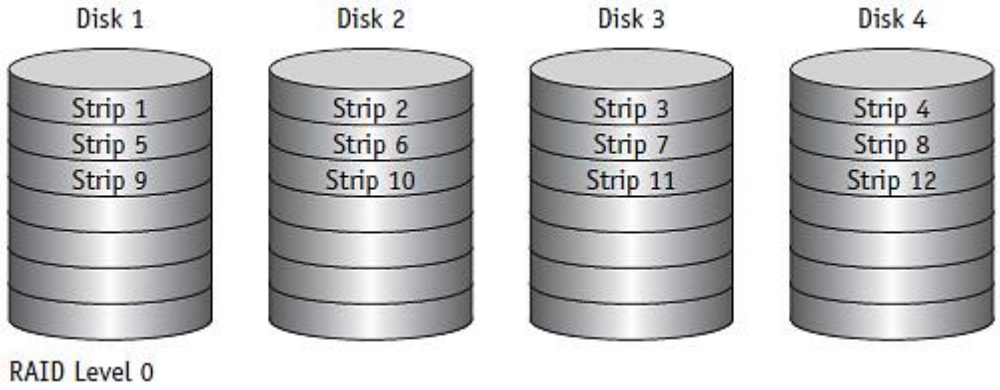
- Disk **striping** uses a group of disks as one storage unit
- RAID is arranged into six different levels
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common
- Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

Level Zero

- Uses data striping (not considered true RAID)
  - No parity and error corrections
  - No error correction/redundancy/recovery
- Benefits
  - Devices appear as one logical unit
  - Best for large data quantity: non-critical data



- RAID Level 0 with four disks in the array. Strips 1, 2, 3, and 4 make up a stripe. Strips 5, 6, 7, and 8 make up another stripe, and so on.

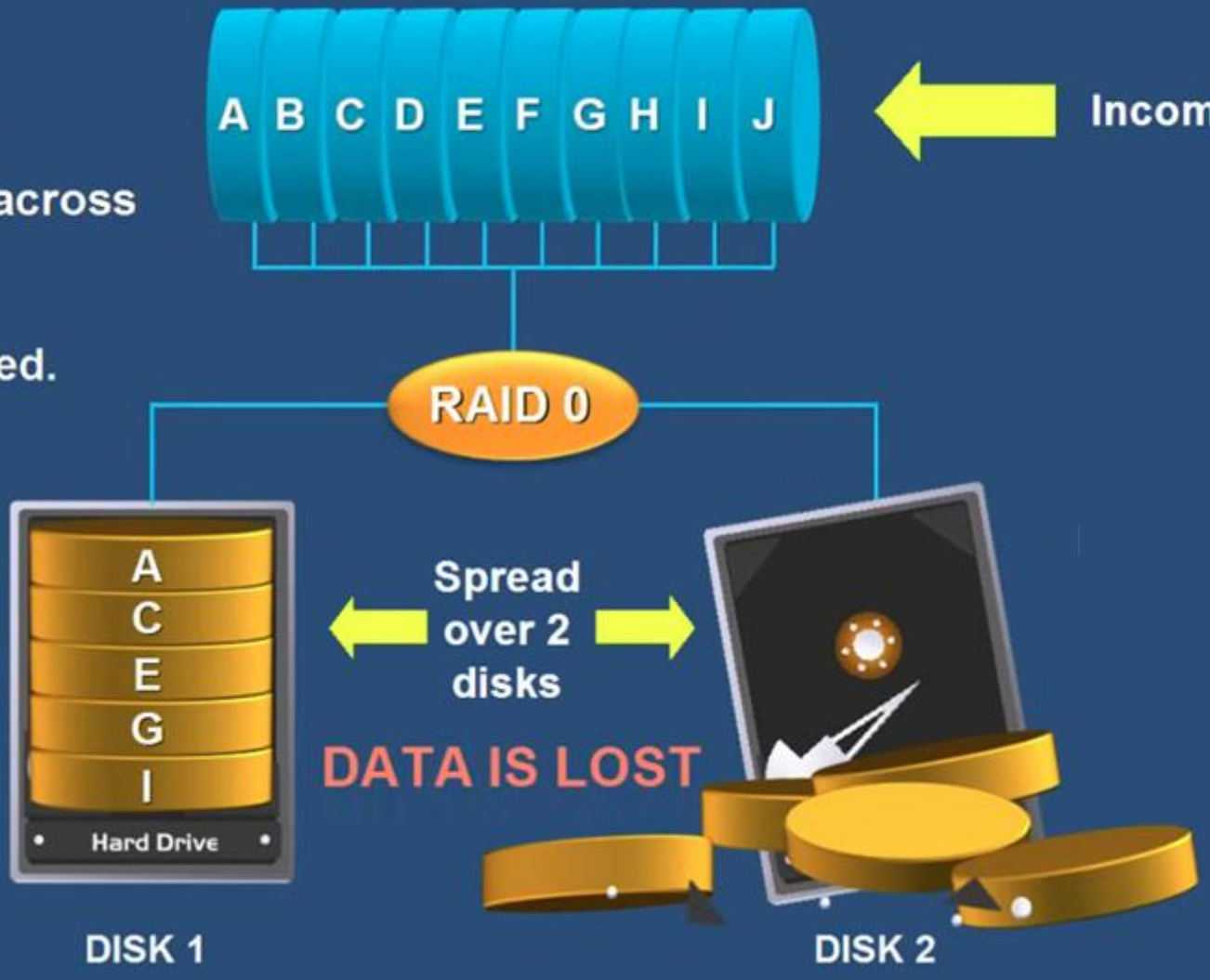


RAID 0 STRIPING

Not fault tolerant.

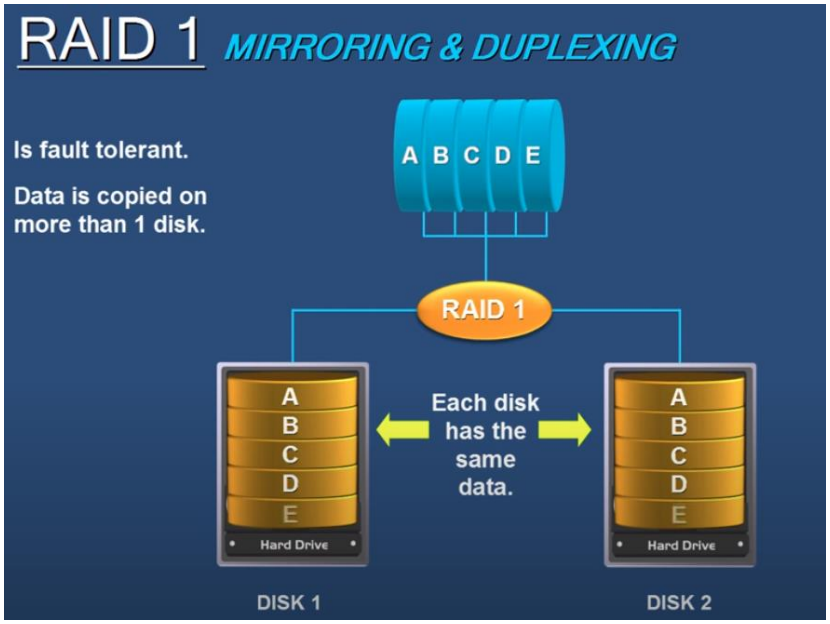
Data is 'striped' across multiple disks.

Advantage is speed.

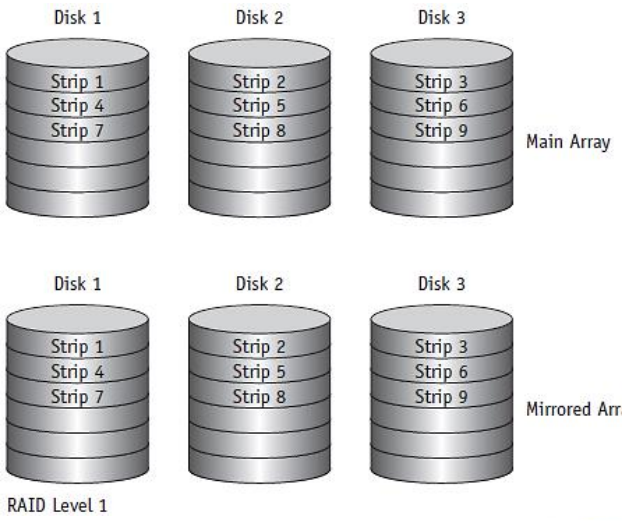


Level One

- Uses data striping (considered true RAID)
  - Mirrored configuration (backup)
    - Duplicate set of all data (expensive)
  - Provides redundancy and improved reliability



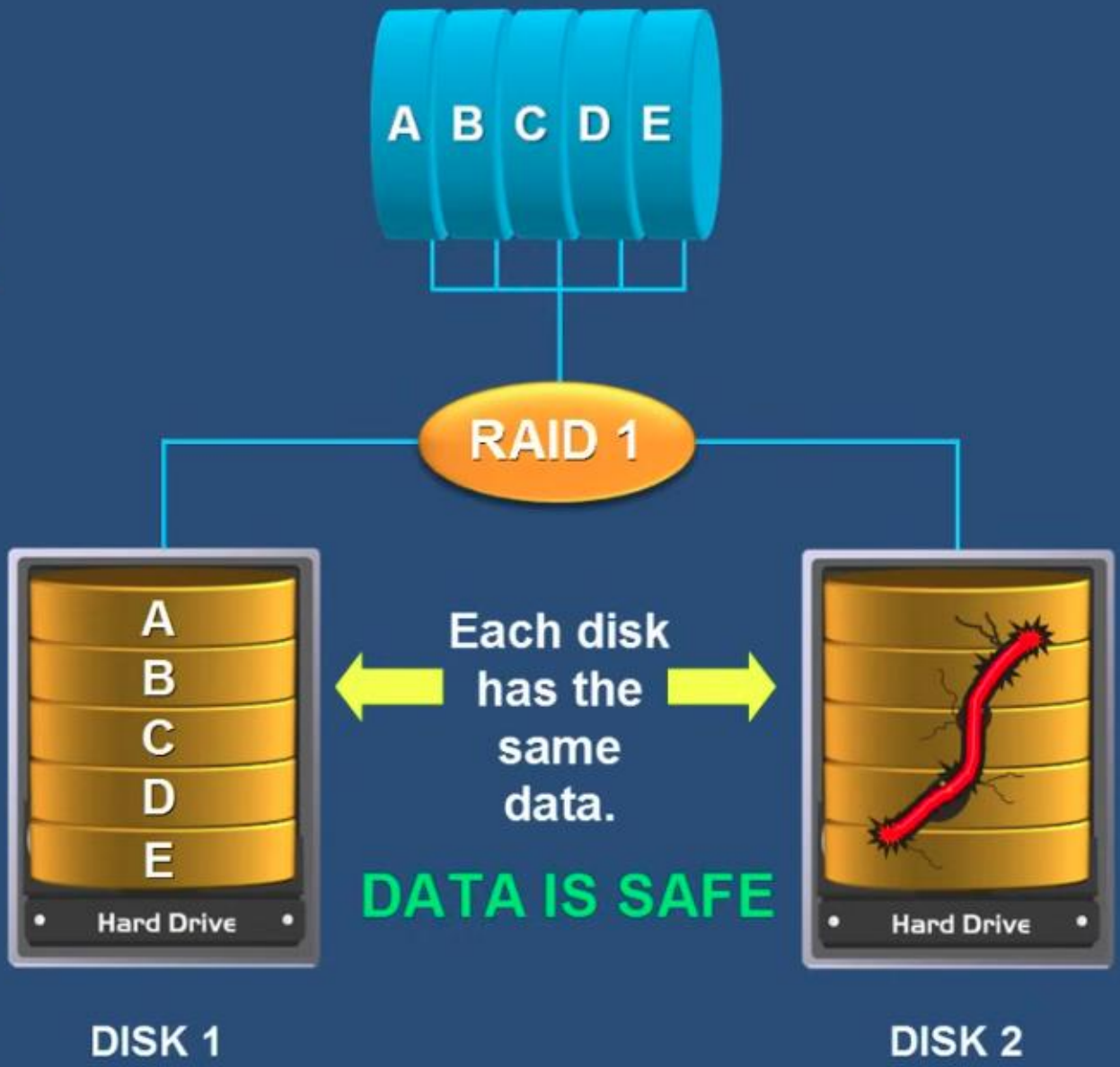
- RAID Level 1 with three disks in the main array and three corresponding disks in the backup array—the mirrored array.



RAID 1 MIRRORING & DUPLEXING

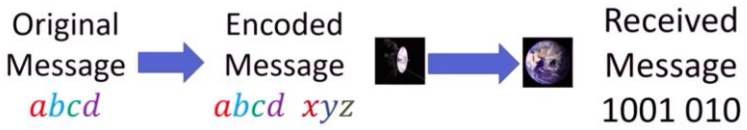
Is fault tolerant.

Data is copied on more than 1 disk.



Level Two

- Uses small strips (considered true RAID)
- Hamming code: error detection and correction
- Expensive and complex
  - Size of strip determines number of array disks



$$x = a \oplus b \oplus d$$
$$y = a \oplus c \oplus d$$
$$z = b \oplus c \oplus d$$

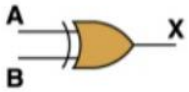
1001 010

1011 010

Boolean Expression

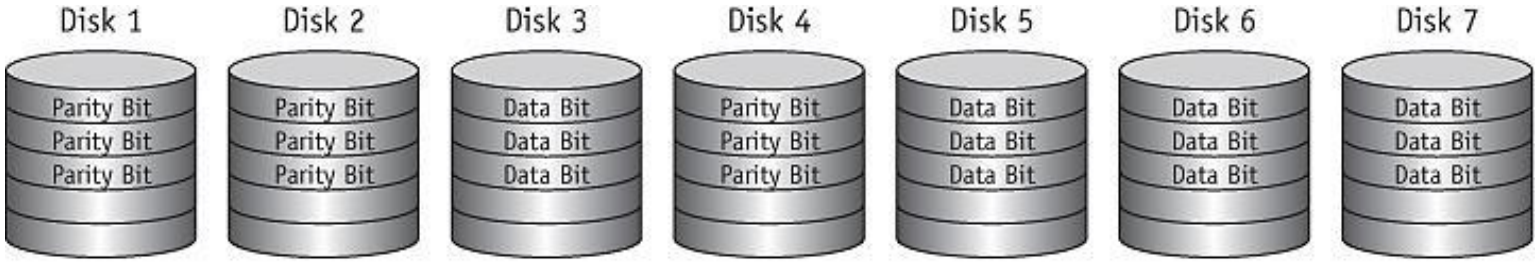
$X = A \oplus B$

Logic Diagram Symbol



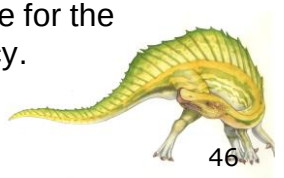
Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



RAID Level 2 - Hamming Code

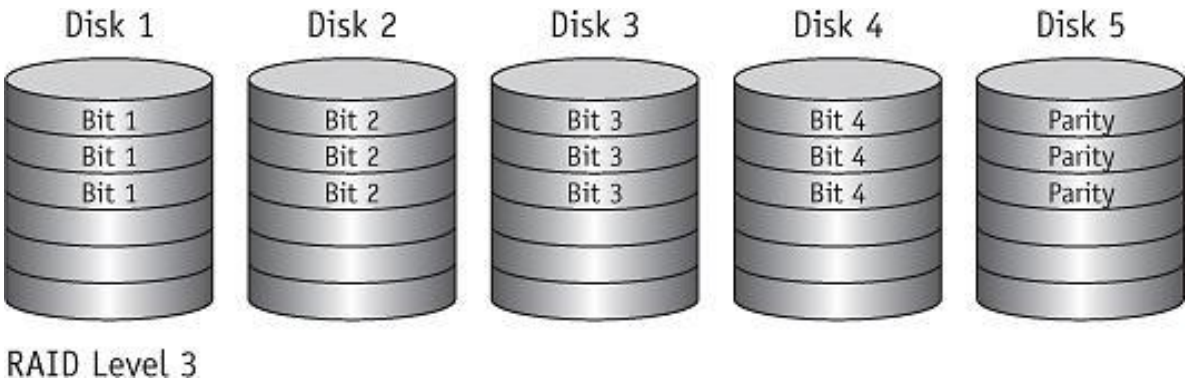
- RAID Level 2. Seven disks are needed in the array to store a 4-bit data item, one for each bit and three for the parity bits. Each disk stores either a bit or a parity bit based on the Hamming code used for redundancy.

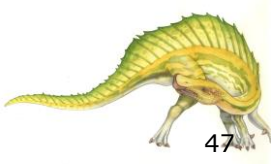


Level Three

- Modification of Level 2
  - Requires one disk for redundancy
    - One parity bit for each strip

- RAID Level 3. A 4-bit data item is stored in the first four disks of the array. The fifth disk is used to store the parity for the stored data item.

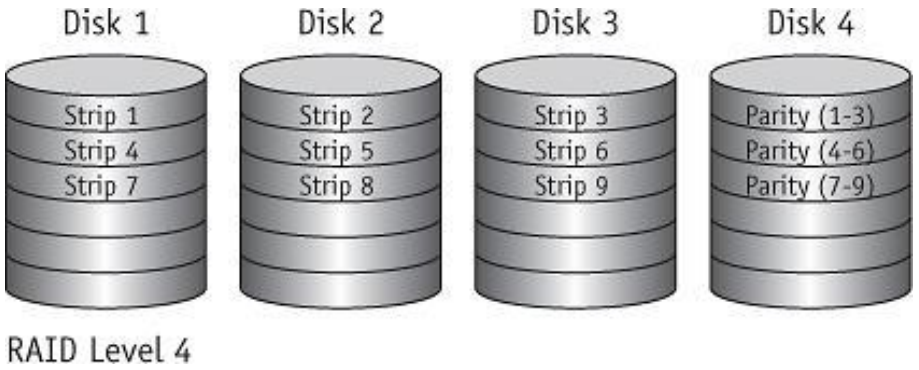




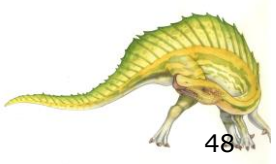
Level Four

- Same strip scheme as Levels 0 and 1
  - Computes parity for each strip
  - Stores parities in corresponding strip
    - Has designated parity disk

- **(**RAID Level 4. The array contains four disks: the first three are used to store data strips, and the fourth is used to store the parity of those strips.

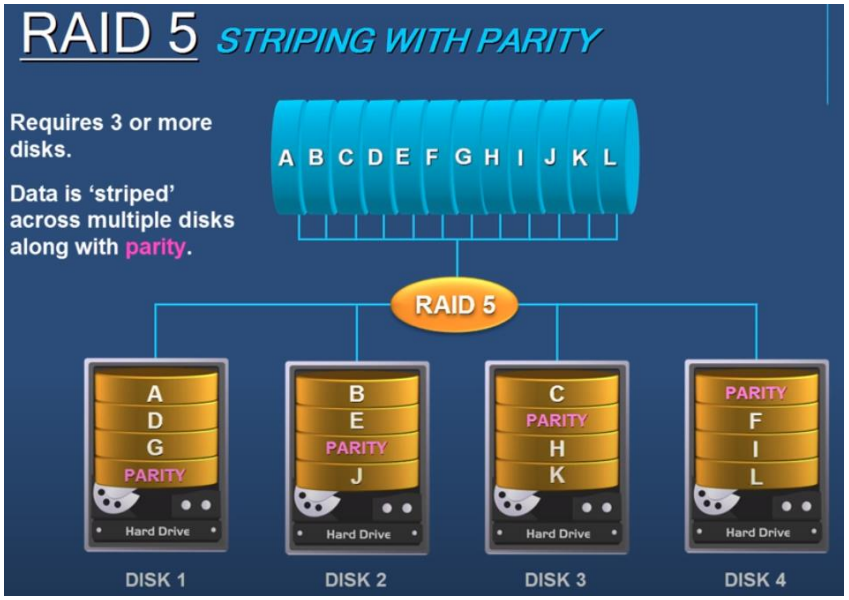


RAID Level 4

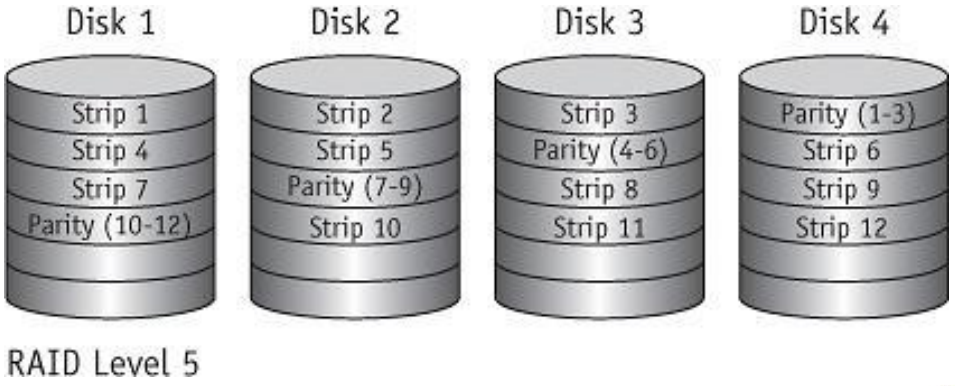


Level Five

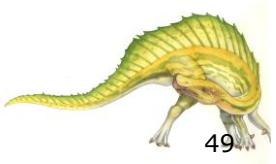
- Modification of Level 4
- Distributes parity strips across disks
  - Avoids Level 4 bottleneck
- Disadvantage
  - Complicated to regenerate data from failed device



- RAID Level 5 with four disks. Notice how the parity strips are distributed among the disks.

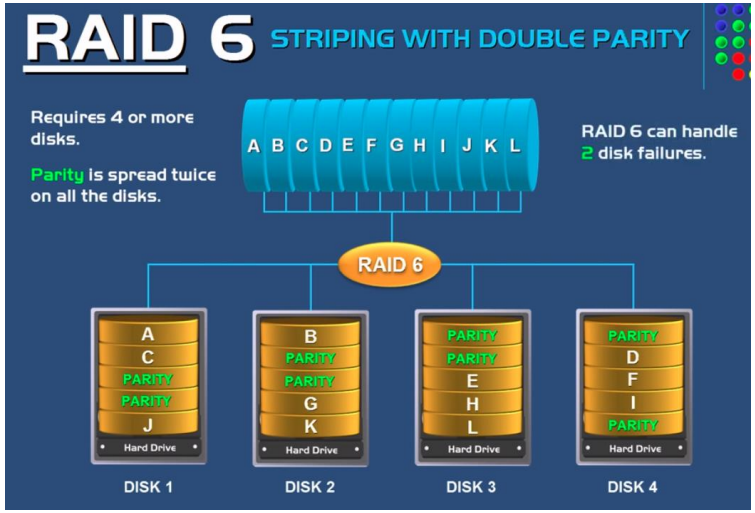


RAID Level 5

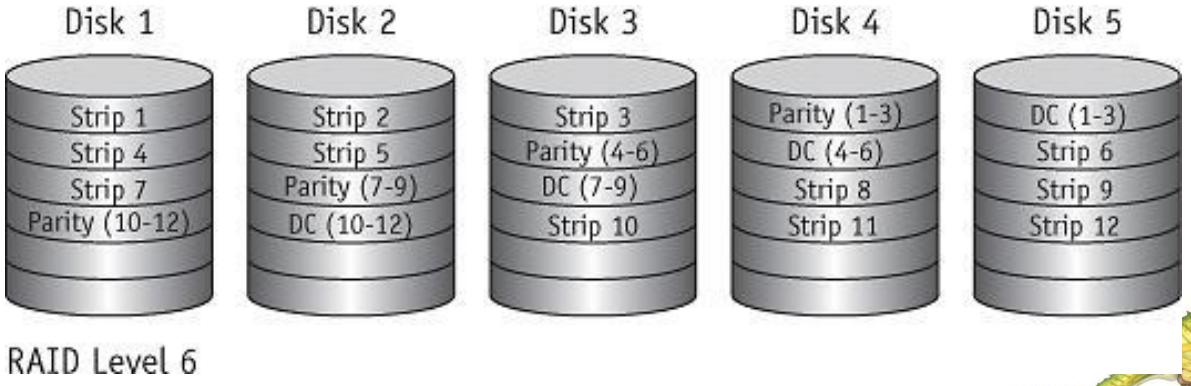


Level Six

- Provides extra degree of error protection/correction
  - Two different parity calculations (double parity)
    - Same as level four/five and independent algorithm
  - Parities stored on separate disk across array
    - Stored in corresponding data strip
- Advantage: data restoration even if two disks fail



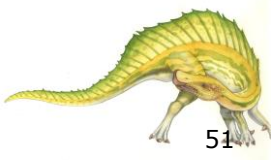
- RAID Level 6. Notice how parity strips and data check (DC) strips are distributed across the disks.



RAID Level 6

- Level 0: organization writes consecutive strips over the drivers in round-robin fashion.
- Level 1: duplicate all the disks, there are same backup disks as primary disks.
- Level 2: works on a word basis, possible even a byte basis. Hamming coded word with parity bits. (losing one drive did not cause problems)
- Level 3: a simplified version of RAID level 2. a single parity bit is computed for each data word and written to a parity drive.
- Level 4 and 5 work with strips again, not individual words with parity.
- Level 4 is like level 0, with a strip-for-strip parity written onto an extra drive. (heavy load on the parity drive)
- Level 5: distribute the parity bits uniformly over all the drives, round robin fashion.

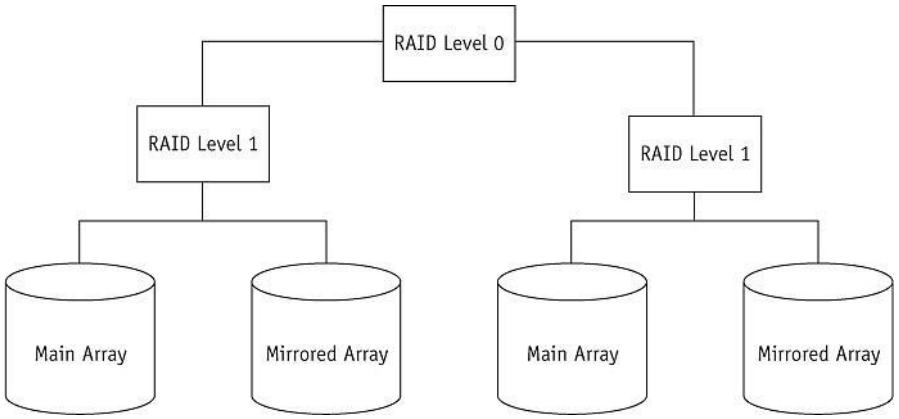
- RAID 1+0 first mirrors and then partitions the data, and then divides all hard disks into two groups as the lowest combination of RAID 0, and then treats these two groups as RAID 1 operation.
- RAID50 is also called mirror array stripe, which is composed of at least six hard drives. Like RAID0, data is partitioned into stripes and written to multiple disks at the same time; like RAID5, it is also based on data parity To ensure data security, and check strips are evenly distributed on each disk. Its purpose is to improve the read and write performance of RAID5.
- RAID 0/1/5/10/50 is used more in practical applications, and RAID 2, 3, and 4 are less practical applications, because RAID 5 already covers the required functions.

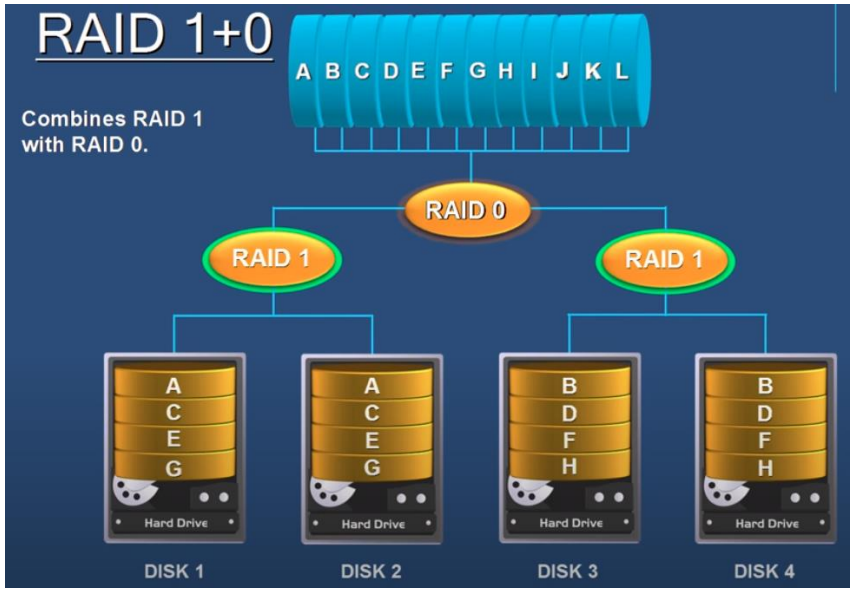


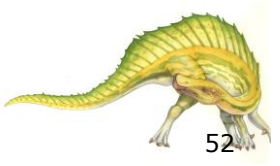
Nested RAID Levels

- Combines multiple RAID levels (complex)

- This is a simple nested RAID Level 10 system, which is a Level 0 system consisting of two Level 1 systems.



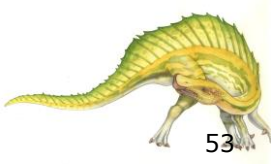




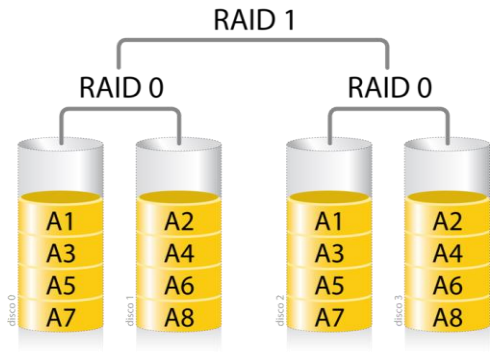
Nested RAID Levels (2 of 2)

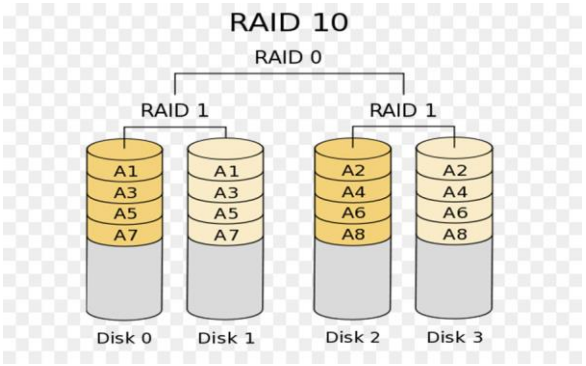
- Some common nested RAID configurations, always indicated with two numbers that signify the combination of levels. For example, neither Level 01 nor Level 10 is the same as Level 1.

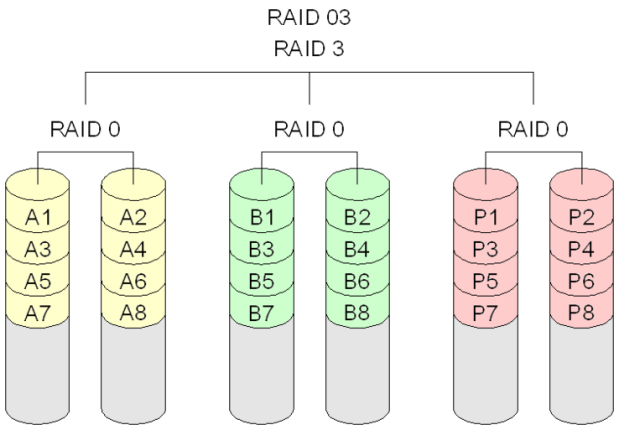
Nested Level	Combinations
01 (or 0+1)	A Level 1 system consisting of multiple Level 0 systems
10 (or 1+0)	A Level 0 system consisting of multiple Level 1 systems
03 (or 0+3)	A Level 3 system consisting of multiple Level 0 systems
30 (or 3+0)	A Level 0 system consisting of multiple Level 3 systems
50 (or 5+0)	A Level 0 system consisting of multiple Level 5 systems
60 (or 6+0)	A Level 0 system consisting of multiple Level 6 systems

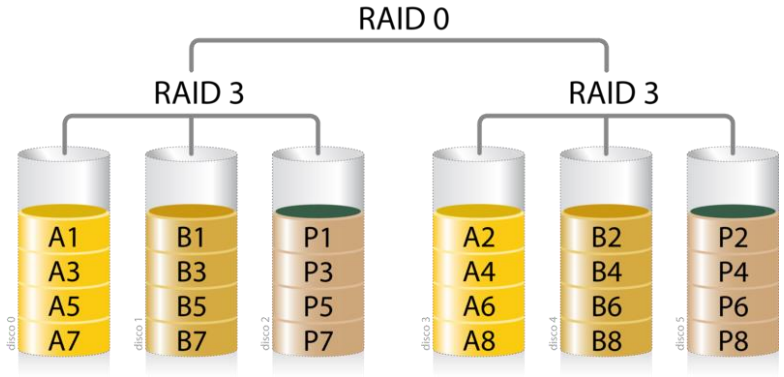


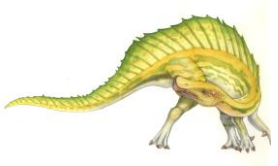
RAID 0+1

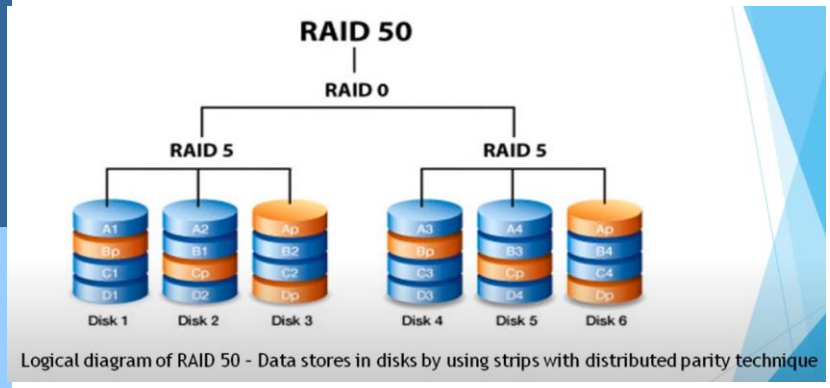

RAID 1
RAID 0
RAID 0
A1
A3
A5
A7
A2
A4
A6
A8
A1
A3
A5
A7
A2
A4
A6
A8


RAID 10
RAID 0
RAID 1
RAID 1
A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8
Disk 0
Disk 1
Disk 2
Disk 3

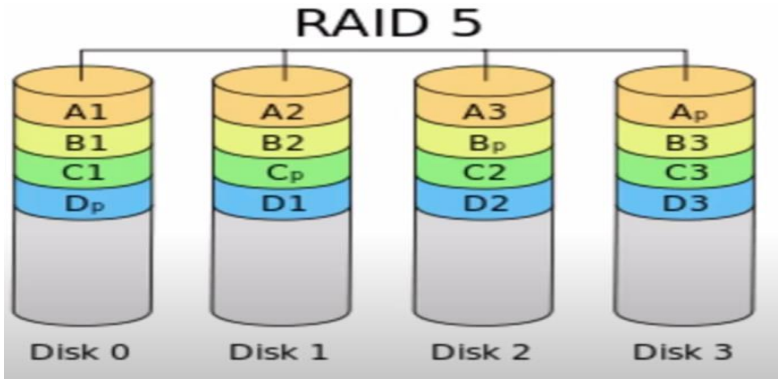

RAID 03
RAID 3
RAID 0
RAID 0
RAID 0
A1
A3
A5
A7
A2
A4
A6
A8
B1
B3
B5
B7
B2
B4
B6
B8
P1
P3
P5
P7
P2
P4
P6
P8

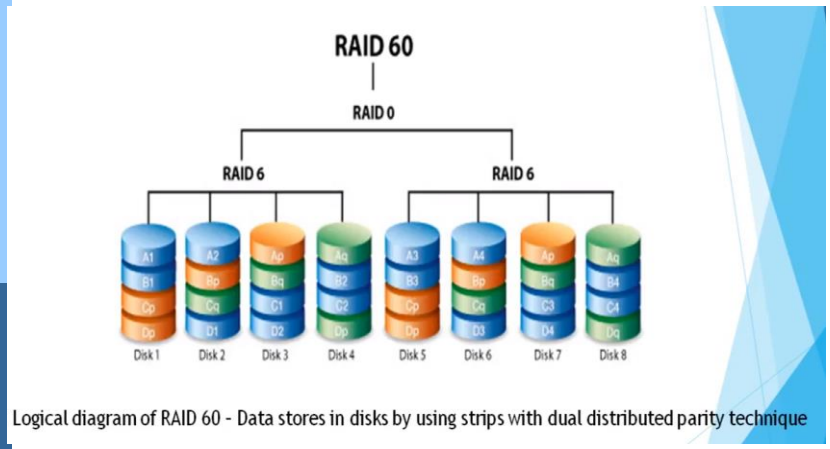

RAID 30
RAID 0
RAID 3
RAID 3
A1
A3
A5
A7
B1
B3
B5
B7
P1
P3
P5
P7
A2
A4
A6
A8
B2
B4
B6
B8
P2
P4
P6
P8



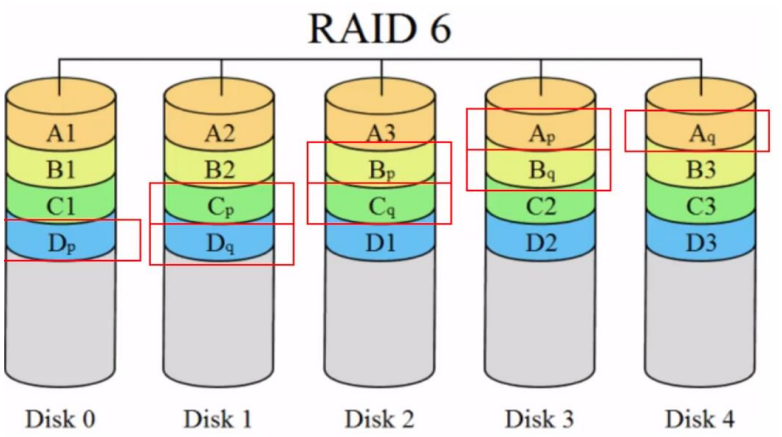


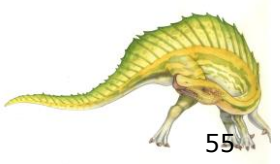
Logical diagram of RAID 50 - Data stores in disks by using strips with distributed parity technique





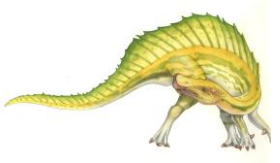
Logical diagram of RAID 60 - Data stores in disks by using strips with dual distributed parity technique





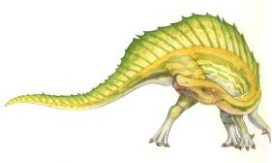
Other Features

- Regardless of where RAID implemented, other useful features can be added
- **Snapshot** is a view of file system before a set of changes take place (i.e. at a point in time)
  - More in Ch 12
- **Replication** is automatic duplication of writes between separate **sites**
  - For redundancy and disaster recovery
  - Can be synchronous or asynchronous
- **Hot spare** disk is unused, automatically used by RAID production if a disk fails to replace the failed disk and rebuild the RAID set if possible
  - Decreases mean time to repair

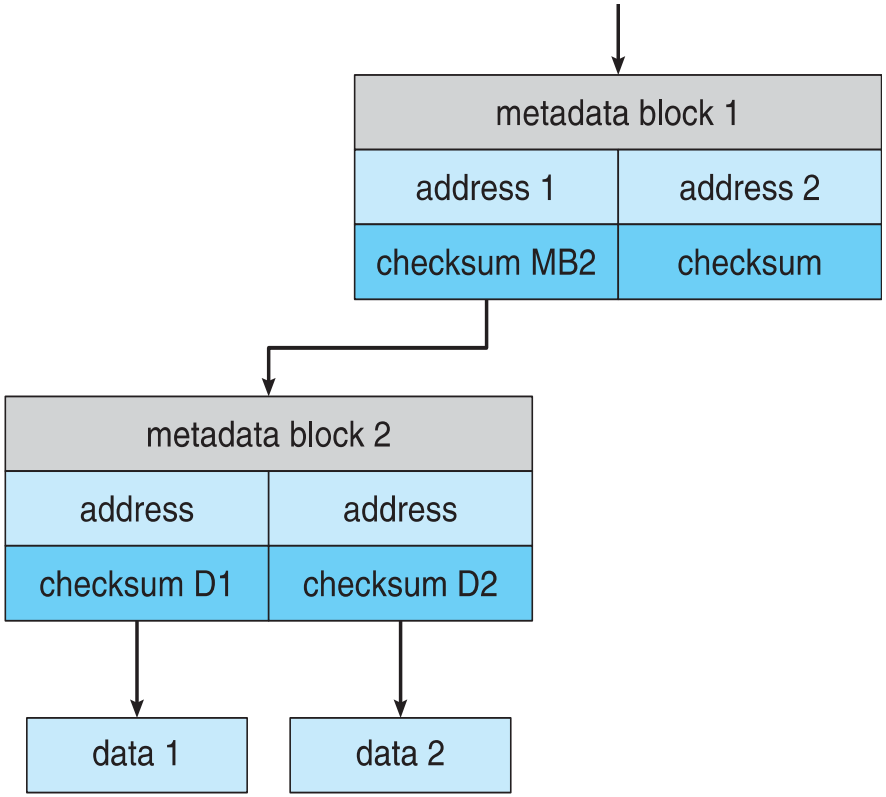


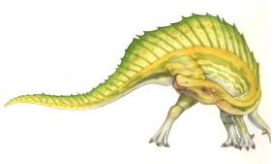
Extensions

- RAID alone does not prevent or detect data corruption or other errors, just disk failures
- Solaris ZFS adds **checksums** of all data and metadata
- Checksums kept with pointer to object, to detect if object is the right one and whether it changed
- Can detect and correct data and metadata corruption
- ZFS also removes volumes, partitions
  - Disks allocated in **pools**
  - Filesystems with a pool share that pool, use and release space like **`malloc()`** and **`free()`** memory allocate / release calls

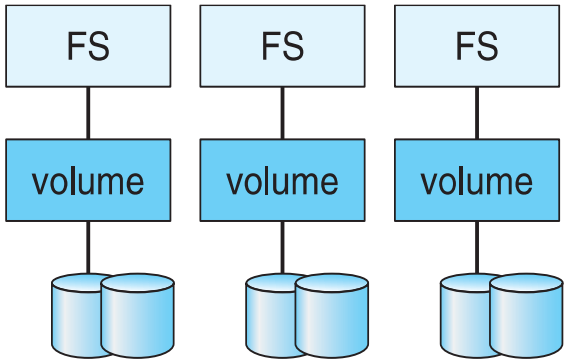


ZFS Checksums All Metadata and Data

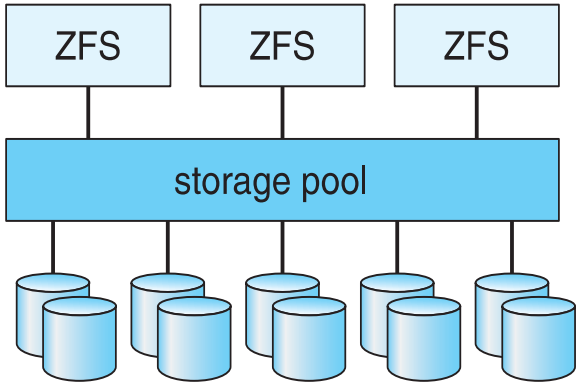




Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Stable-Storage Implementation

- Write-ahead log scheme requires stable storage
- Stable storage means data is never lost (due to failure, etc)
- To implement stable storage:
  - Replicate information on more than one nonvolatile storage media with independent failure modes
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery
- Disk write has 1 of 3 outcomes
  1. **Successful completion -** The data were written correctly on disk
  2. **Partial failure -** A failure occurred in the midst of transfer, so only some of the sectors were written with the new data, and the sector being written during the failure may have been corrupted
  3. **Total failure -** The failure occurred before the disk write started, so the previous data values on the disk remain intact

Stable-Storage Implementation (Cont.)

- If failure occurs during block write, recovery procedure restores block to consistent state
  - System maintains 2 physical blocks per logical block and does the following:
    1. Write to 1st physical
    2. When successful, write to 2nd physical
    3. Declare complete only after second write completes successfully

  Systems frequently use NVRAM as one physical to accelerate

End of Chapter 11