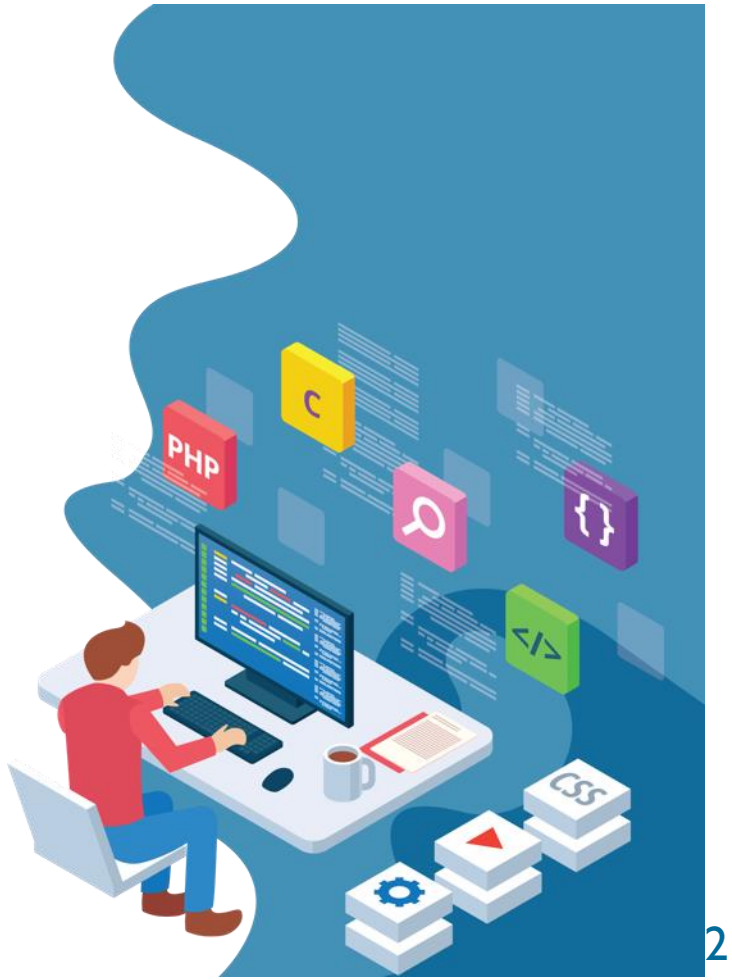
NATIONAL ECONOMICS UNIVERSITY
SCHOOL OF INFORMATION TECHNOLOGY AND DIGITAL ECONOMICS

CHAPTER 4

MASS-STORAGE SYSTEMS

OUTLINE

- Overview of Mass Storage Structure
- I/O Systems
- Magnetic Disks
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure



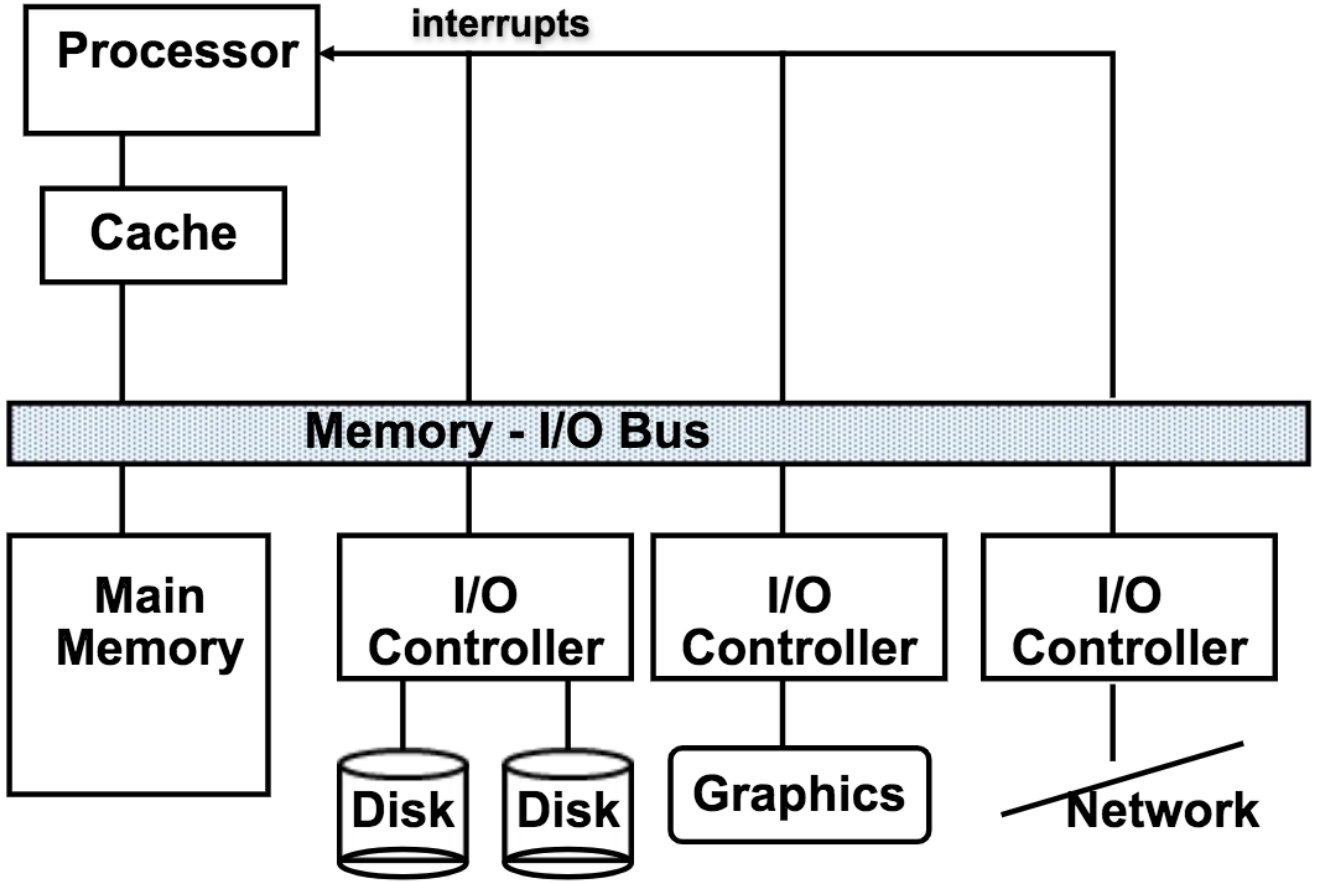
OBJECTIVES

- Describe the physical structure of secondary storage devices and the effect of a device's structure on its uses
- Explain the performance characteristics of mass-storage devices
- Evaluate I/O scheduling algorithms
- Discuss operating-system services provided for mass storage, including RAID

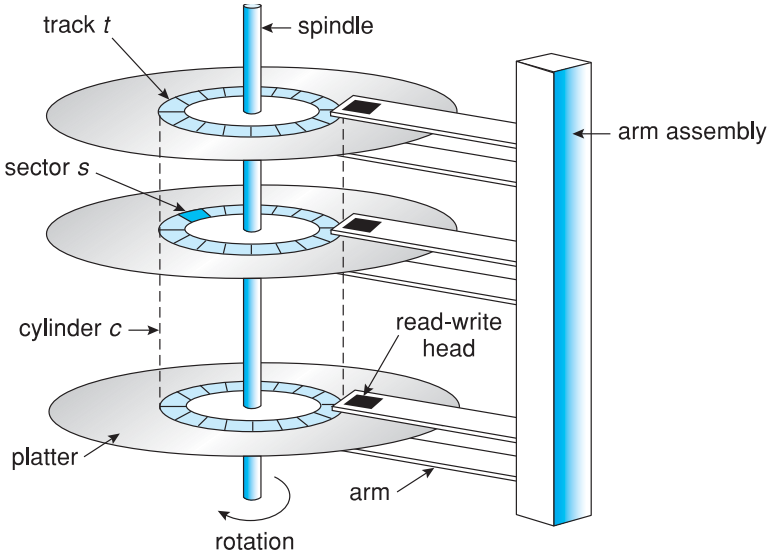
OVERVIEW OF MASS STORAGE STRUCTURE

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second. Common drives spin at 5,400, 7,200, 10,000, and 15,000 RPM.
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) + time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface
- Disks can be removable
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI, SAS, Firewire**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

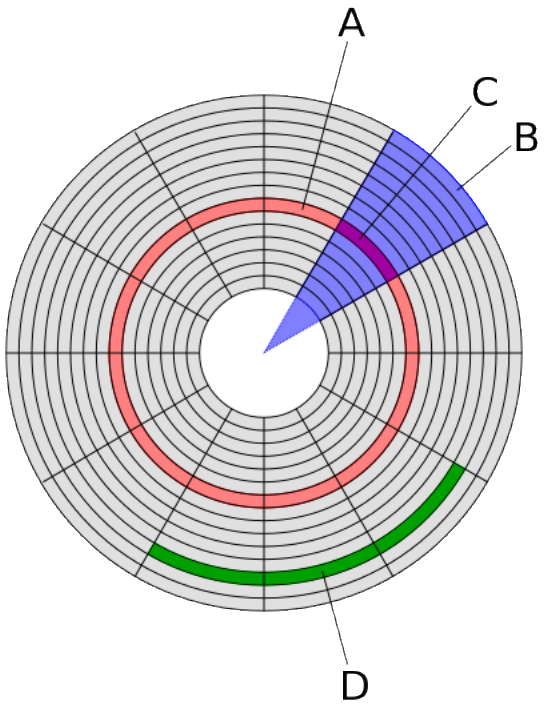
I/O SYSTEMS

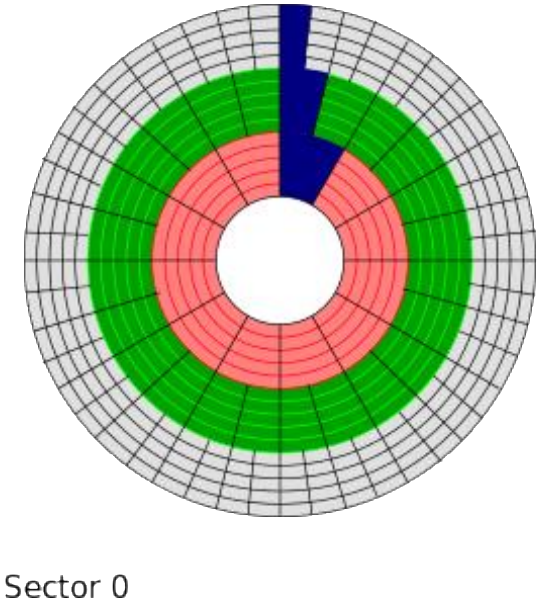

Processor
interrupts
Cache
Memory - I/O Bus
Main Memory
I/O Controller
I/O Controller
I/O Controller
Disk
Disk
Graphics
Network

MAGNETIC DISKS



DISK SECTOR





SOLID-STATE DISKS

- SSDs: **Old technologies** are used in new ways as economics change or the technologies evolve.
- Same characteristics as hard disks but can be more reliable because they have no moving parts => **no seek time** or **latency**
- They consume **less power**, **more expensive** per megabyte, **less capacity**, **shorter life spans**
- Some systems use them as a direct **replacement** for disk drives, while others use them as a new **cache** tier

MAGNETIC TAPE

- 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
- Moving to the correct spot on a tape can take minutes
- 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

MAGNETIC DISKS

- Platters range from .85” to 14” (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)
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency

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

MAGNETIC DISK PERFORMANCE

- **Access Latency** = **Average access time** = average seek time + average latency
  - For fastest disk: 3ms + 2ms = 5ms
  - For common 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 =
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = 4KB / 1Gb/s = 32 / 1024^2 = 0.031 ms
  - Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

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
- 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

DISK SCHEDULING (CONT.)

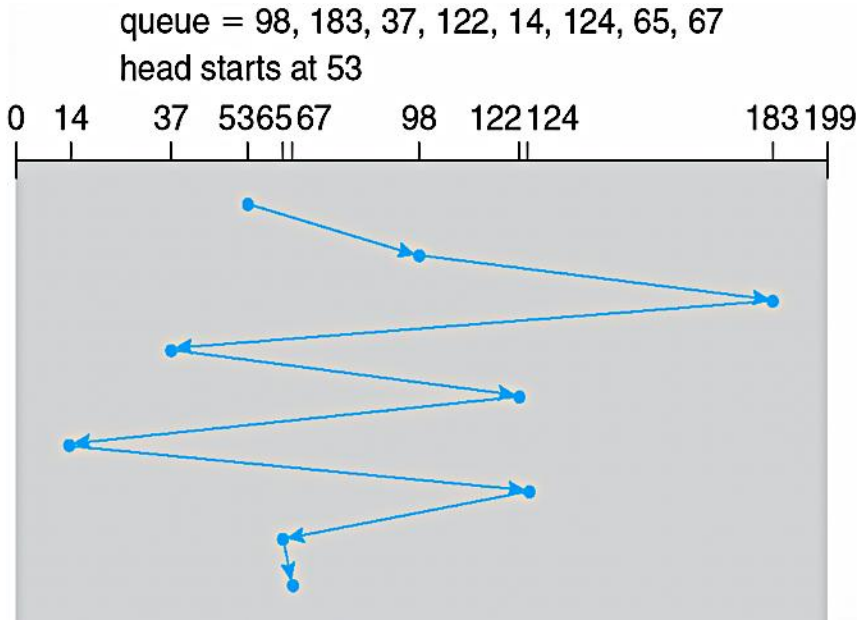
- Several algorithms exist to schedule the servicing of disk I/O requests
- The analysis is true for one or many platters
- We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

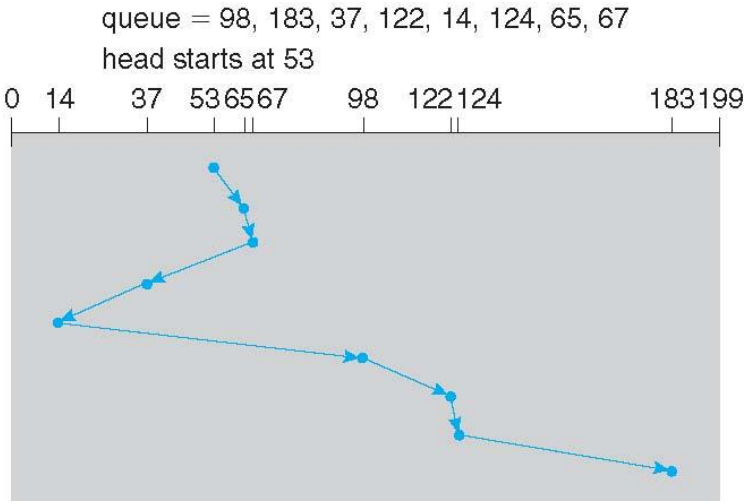
1. FCFS SCHEDULING

- **F**irst-**C**ome, **F**irst-**S**erved (**FCFS**) algorithm
- Illustration shows total head movement of 640 cylinders



2. SSTF SCHEDULING

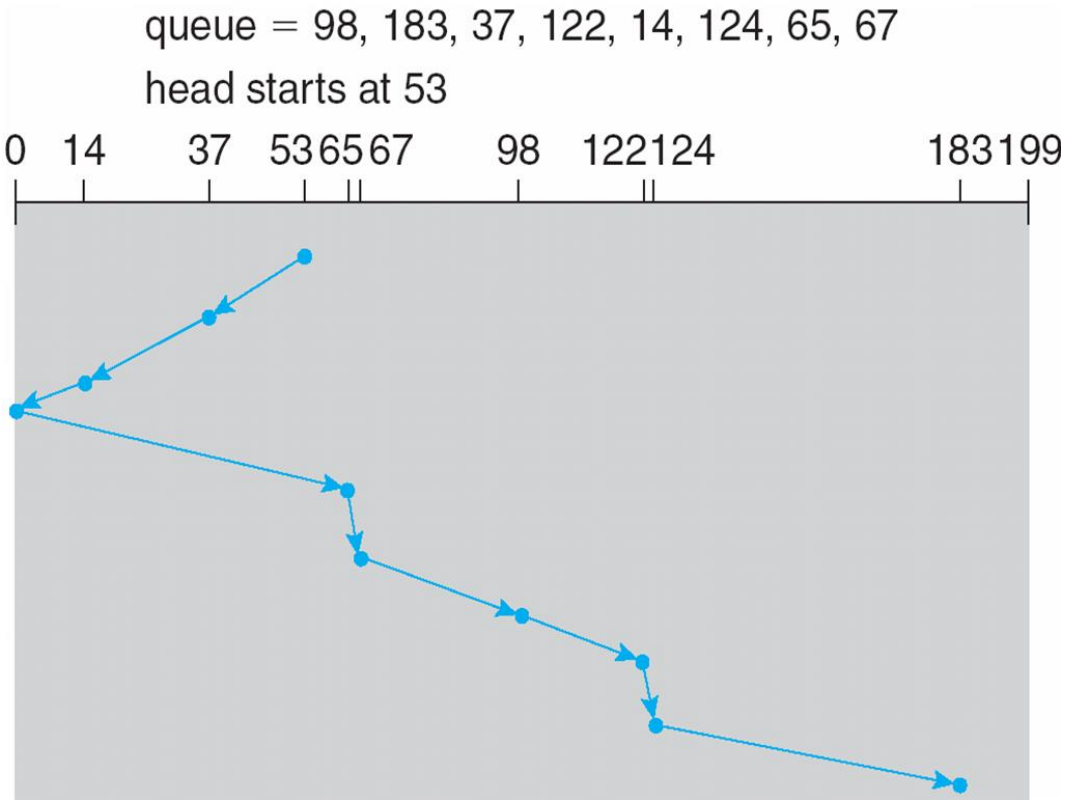
- **S**hortest **S**eek **T**ime **F**irst (**SSTF**) selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests
- Illustration shows total head movement of 236 cylinders



3. SCAN SCHEDULING

- 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**
- But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

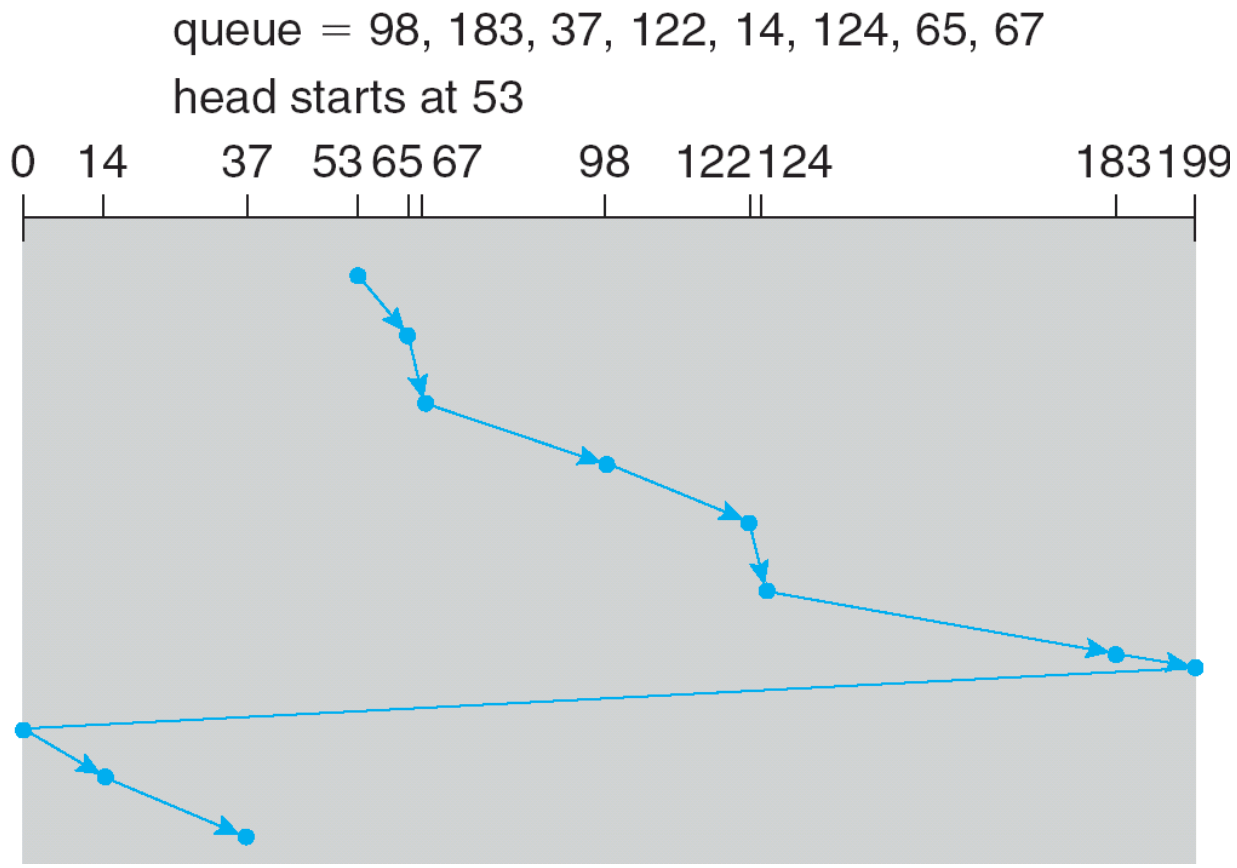
3. SCAN SCHEDULING (CONT.)



4. C-SCAN SCHEDULING

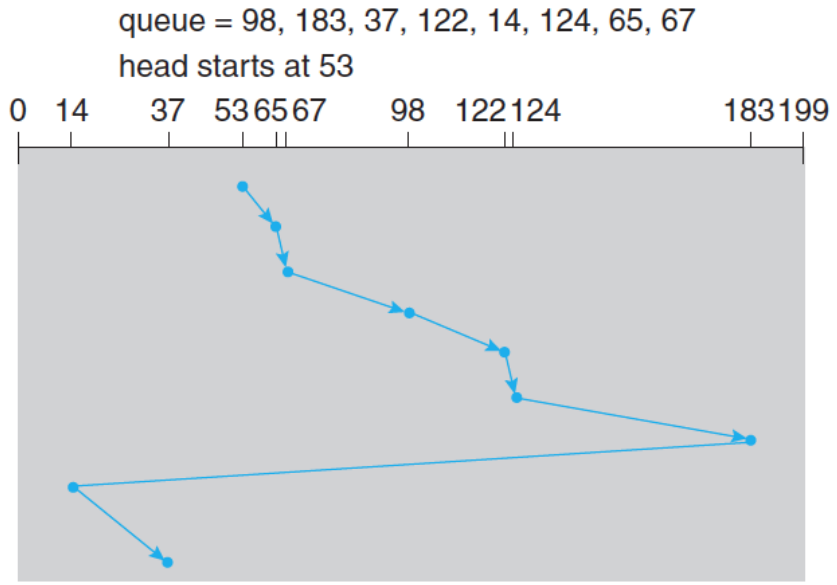
- Provides a more uniform wait time than SCAN
- The head moves from one end of the disk to the other, servicing requests as it goes
  - When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip
- Treats the cylinders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

4. C-SCAN SCHEDULING (CONT.)



5. LOOK, CLOOK SCHEDULING

- 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



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
- Either SSTF or LOOK is a reasonable choice for the default algorithm
- What about rotational latency? => Difficult for OS to calculate

DISK MANAGEMENT

- Before a disk can store data, it must be divided into sectors that the disk controller can read and write. This process is called low **Low-level formatting**, or **physical formatting**
  - Each sector can hold header information, plus data (Usually 512 bytes of data), plus error correction code (**ECC**)
- 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":
    - OS stores the initial file-system data structures
    - Include maps of free and allocated space
  - Increase efficiency: group blocks into **clusters**

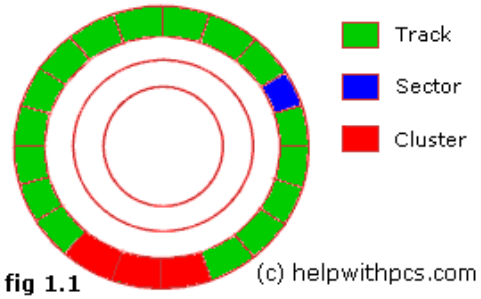


fig 1.1

DISK MANAGEMENT (CONT.)

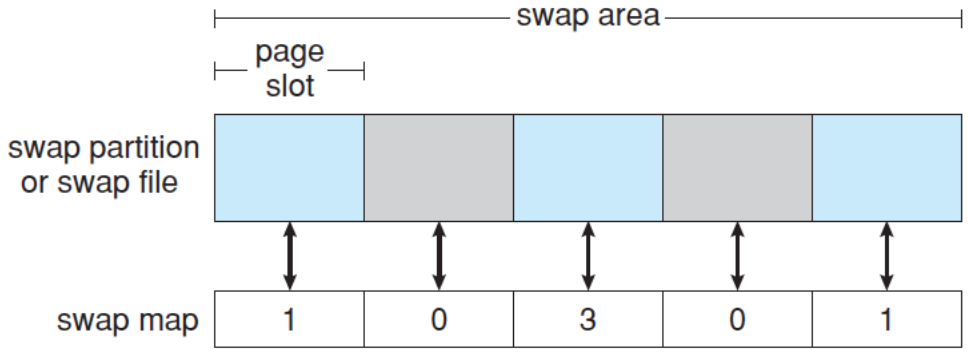
- **Raw disk** access for apps that want to do their own block management, keep OS out of the way (databases for example)
- Boot block initializes system
  - The bootstrap is stored in ROM
  - The problem is that changing this bootstrap code requires changing the ROM hardware chips.
  - For this reason, most systems store a tiny bootstrap loader program in the boot
  - ROM whose only job is to bring in a full bootstrap program from disk
  - Disk that has a boot partition is called a **boot disk** or **system disk.**

BAD BLOCKS

- Because disks have moving parts and small tolerances (recall that the disk head flies just above the disk surface), they are prone to failure
- More frequently, one or more sectors become defective. Most disks even come from the factory with bad blocks
- Depending on the disk and controller in use, these blocks are handled in a variety of ways
- One strategy is to scan the disk to find bad blocks while the disk is being formatted.
- Any bad blocks that are discovered are flagged as unusable so that the file system does not allocate them
- A special program (such as the Linux badblocks command) must be run to search for the bad blocks and to lock them away
- Data that resided on the bad blocks usually are lost

SWAP-SPACE MANAGEMENT

- Swap-space - Virtual memory uses disk space as an extension of main memory
- Less common now due to memory capacity increases
- Kernel uses **swap maps** to track swap-space use



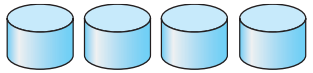
RAID STRUCTURE

- **RAID** – **R**edundant **A**rray of **I**nexpensive **D**isks
  - multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure** (single disk = 100,000 hours)
- **Mean time to repair –** exposure time when another failure could cause data loss (10 hours)
- **Mean time to data loss** based on above factors = $MTF^2/(2*MTR)$
- 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
- 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

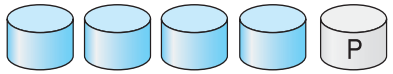
- RAID Levels



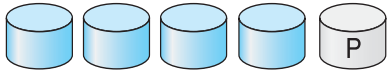
(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

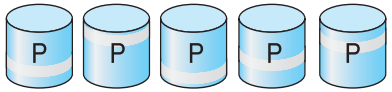
(c) RAID 2: memory-style error-correcting codes.



(d) RAID 3: bit-interleaved parity.



(e) RAID 4: block-interleaved parity.

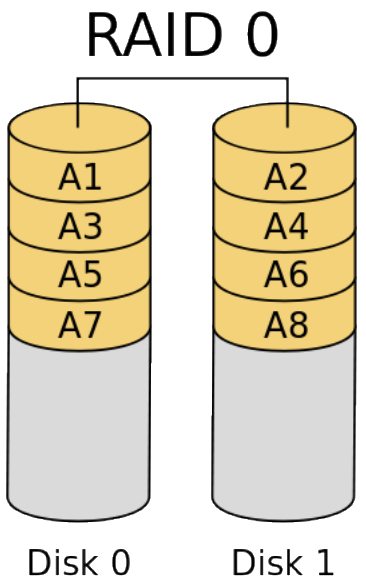


(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

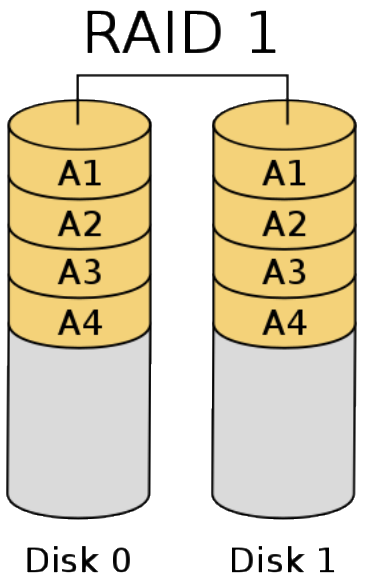
RAID 0

- **RAID 0** splits data evenly across two or more disks, without parity information, redundancy
- This configuration is typically implemented having speed as the intended goal
- Since RAID 0 provides no fault tolerance or redundancy, the failure of one drive will cause the entire array to fail;
- A RAID 0 setup can be created with disks of differing sizes,



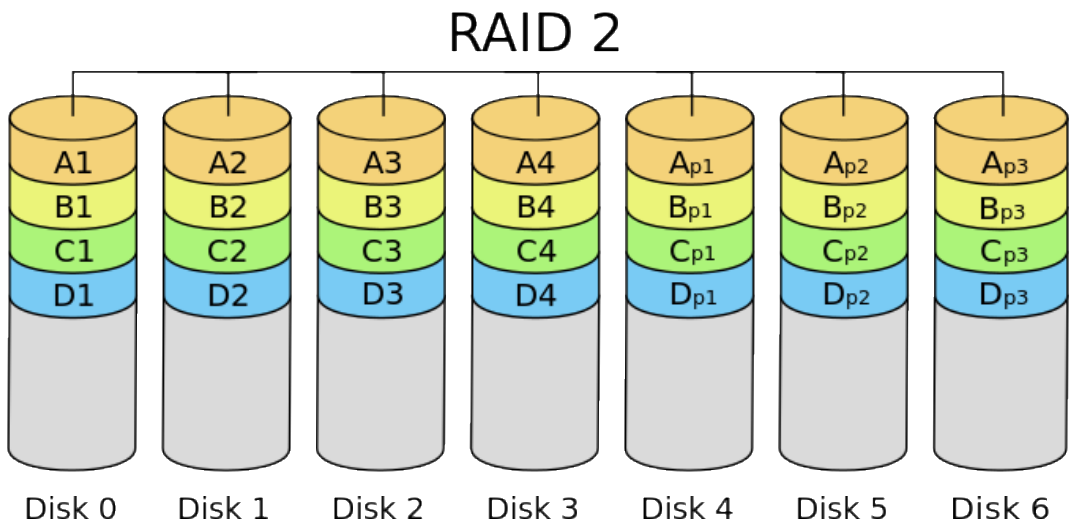
RAID 1

- **RAID 1** consists of an exact copy (or *mirror*) of a set of data on two or more disks; a classic RAID 1 mirrored pair contains two disks.
- However, if disks with different speeds are used in a RAID 1 array, overall write performance is equal to the speed of the slowest disk.



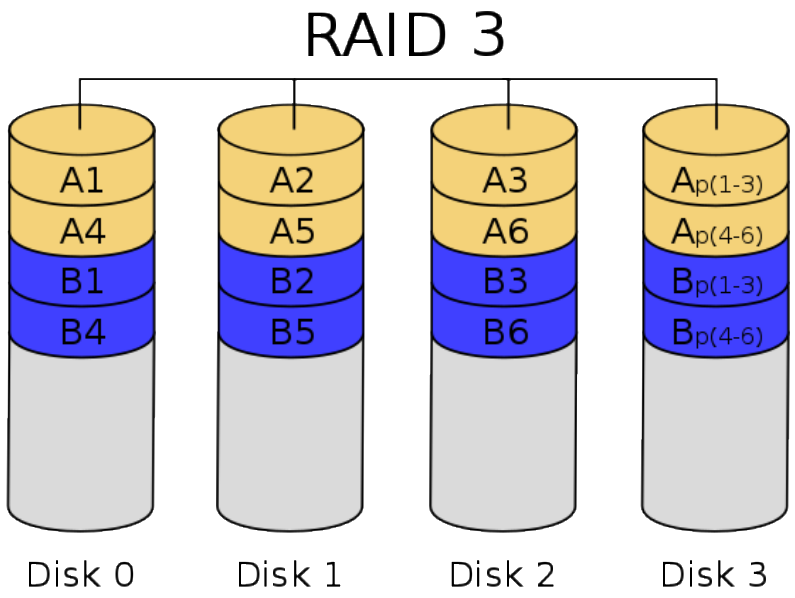
RAID 2

- RAID 2, which is rarely used in practice, stripes data at the bit
- uses a Hamming code for error correction.



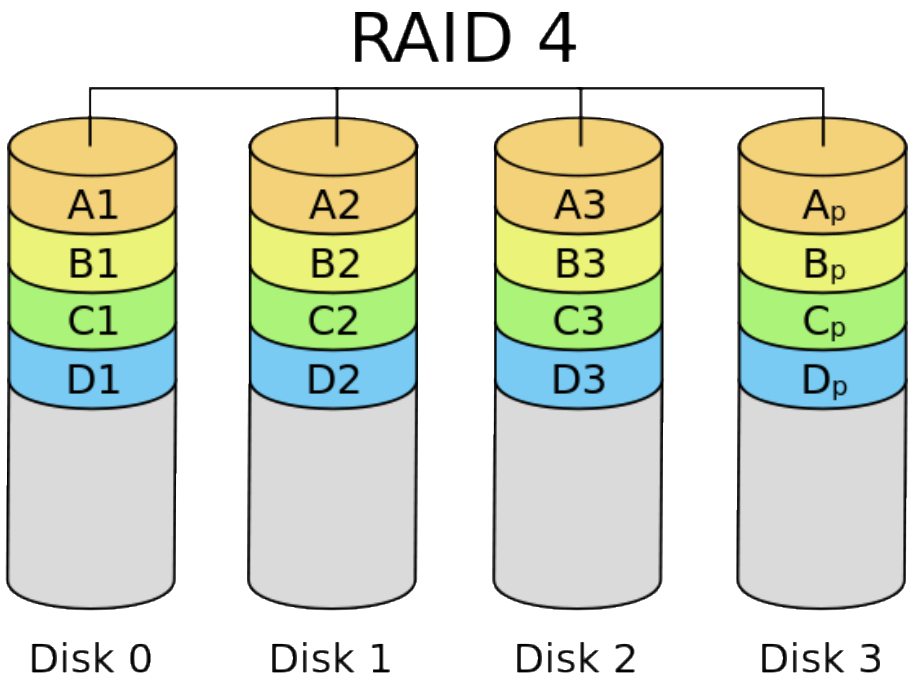
RAID 3

- **RAID 3**, which is rarely used in practice, consists of byte-level striping with a dedicated parity disk



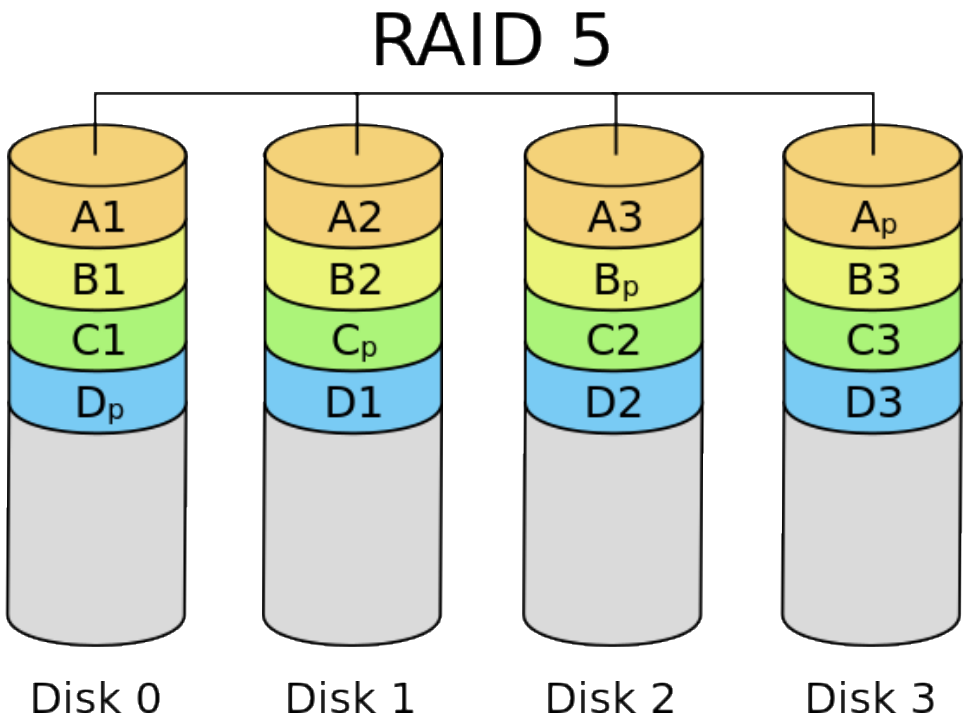
RAID 4

- **RAID 4** consists of block-level striping with a dedicated parity disk.

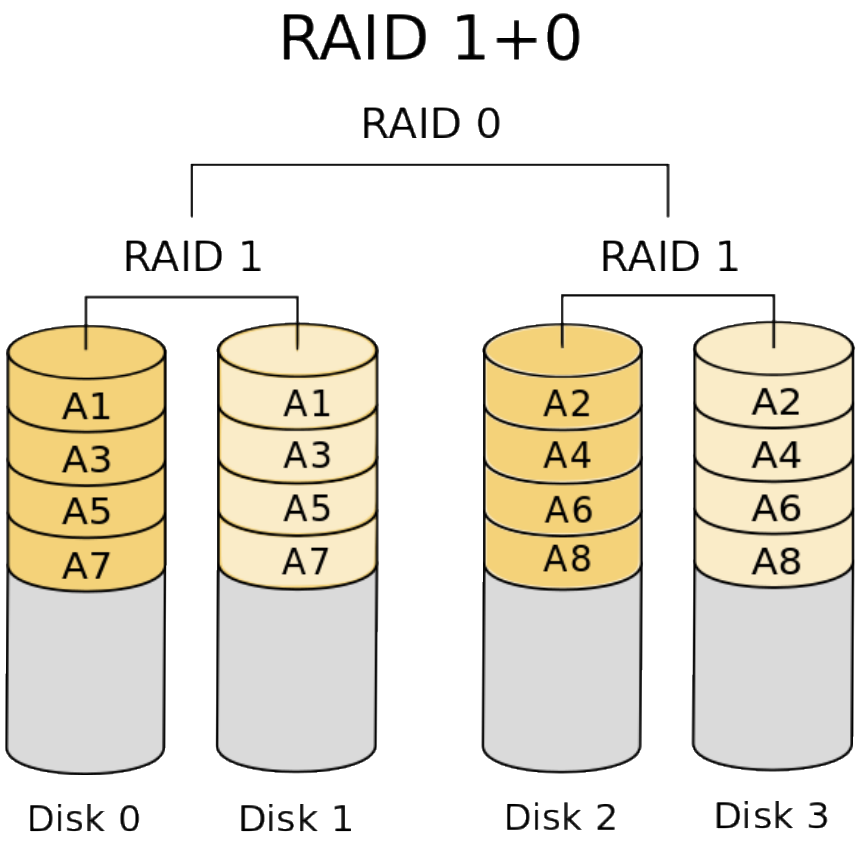


RAID 5

- Unlike in RAID 4, parity information is distributed among the drives.



RAID 10


RAID 1+0
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