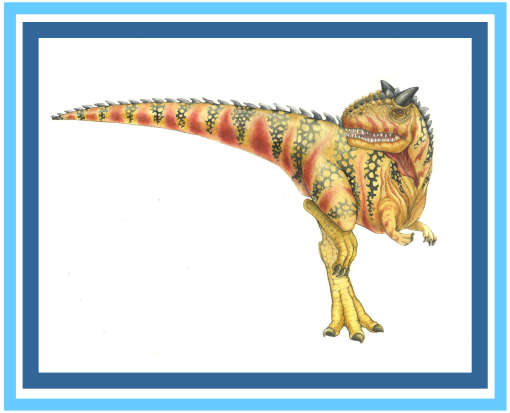
Module 7- L1

Mass-Storage Systems

Dr. Rishikeshan C A

Asst. Professor (Sr.)

SCOPE, VIT Chennai



Outline

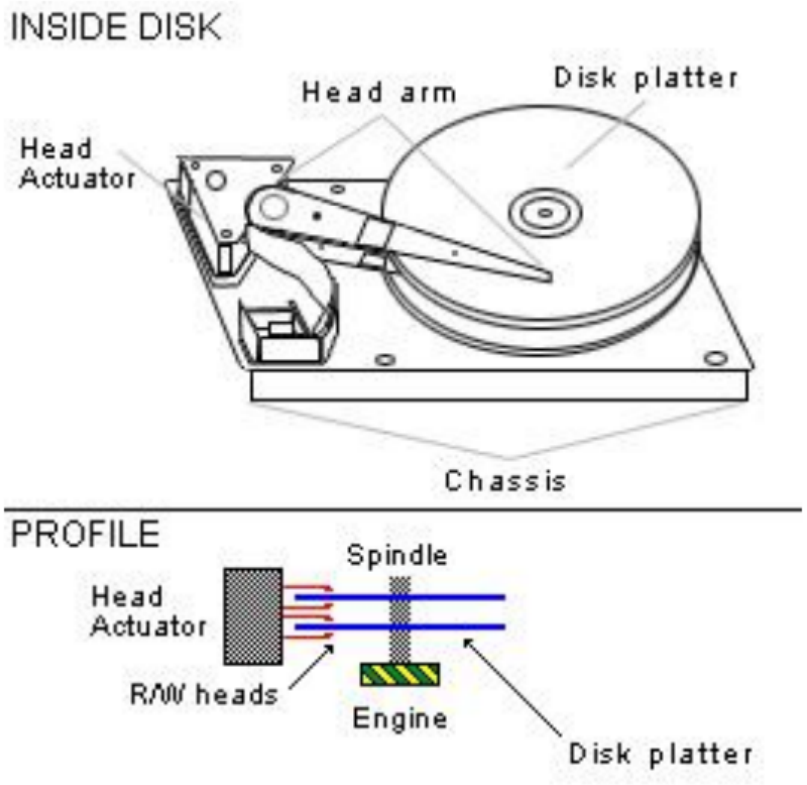
- Overview of Mass Storage Structure
- Disk Scheduling

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

Mass-Storage Systems

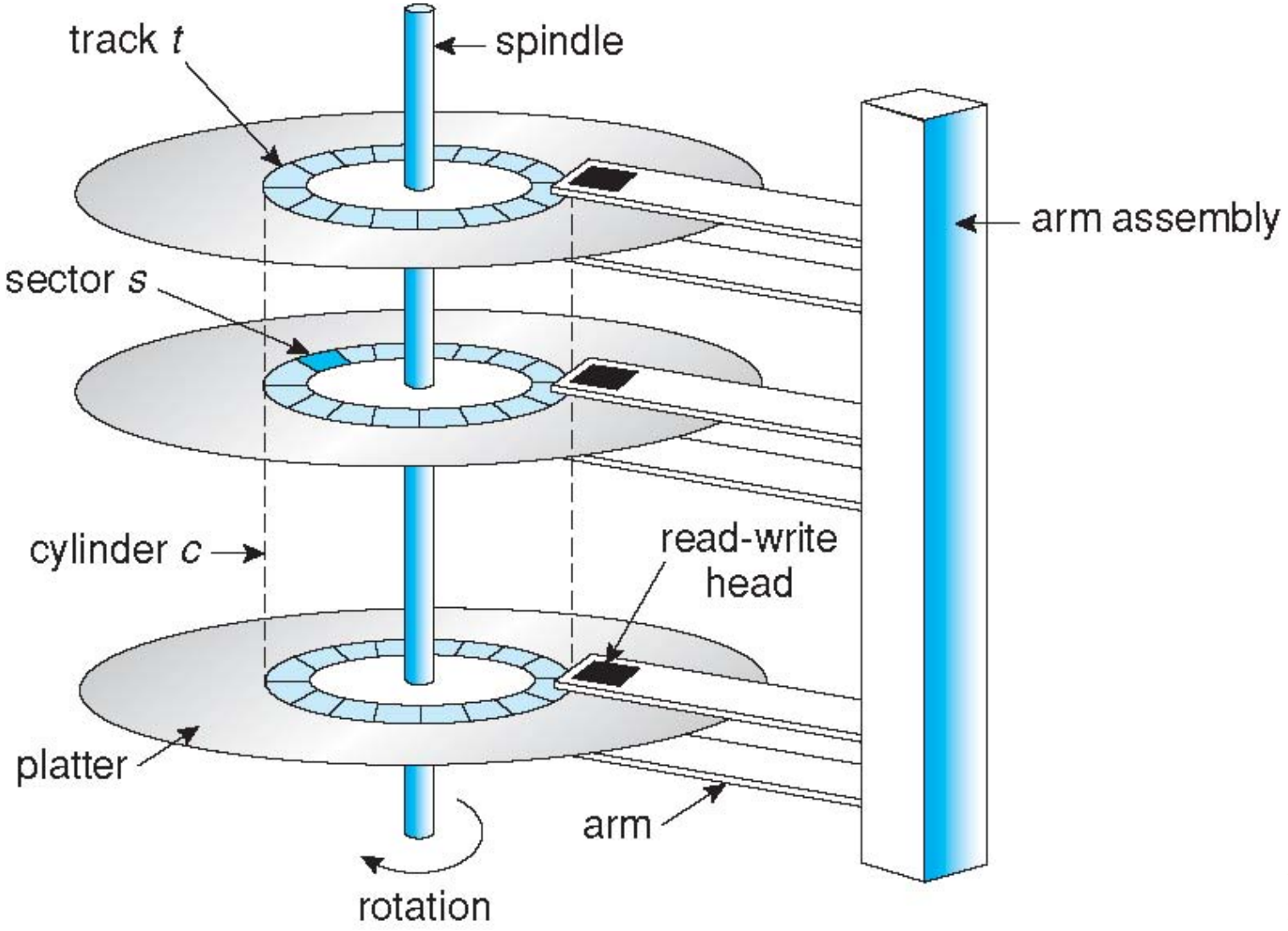
Magnetic disks



Overview of Mass Storage Structure

- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 250 times per second
- **Transfer rate** is rate at which data flow between drive and computer
- **Positioning time** (**access time**) = **seek time + rotational latency**
  - **seek time:** time to move disk arm to desired **cylinder**
  - **rotational latency:** time for desired **sector** to rotate under the disk head
- **Head crash** results from disk head making contact with the disk surface
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI, SAS, Firewire**

Moving-head Disk Mechanism



Hard Disks

- Platters
  - Commonly 3.5”, 2.5”, and 1.8”
- Size
  - 30GB to 3TB per drive
- Performance
  - Transfer Rate
    - 6 GB/sec in Theory and 1GB/sec in practice
  - Seek time
    - 3ms to 12ms – 9ms common for desktop drives
  - Rotation Latency based on spindle speed

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

(Source: Wikipedia)

Solid-State Disks

- **Nonvolatile** memory used like a hard drive
- Can be **more reliable** than HDDs
- **More expensive** per MB
- Maybe have **shorter life** span
- **Less capacity**
- **Much faster**
  - No moving parts, so no seek time or rotational latency

Magnetic Tape

- Mainly used for **backup**
- Holds **large quantities** of data
  - 200GB to 1.5TB typical storage
- Access time **slow**
  - Random access ~1000 times slower than disk
- Once data under head, transfer rates **comparable to disk**
  - 140MB/sec and greater

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
- Access time has two major components
  - **Seek time** is the time for the disk are to move the heads to the cylinder containing the desired sector
  - **Rotational latency** is the additional time waiting for the disk to rotate the desired sector to the disk head
- 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.)

- **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
  - **must queue** requests
  - Optimization algorithms apply

Disk Scheduling (Cont.)

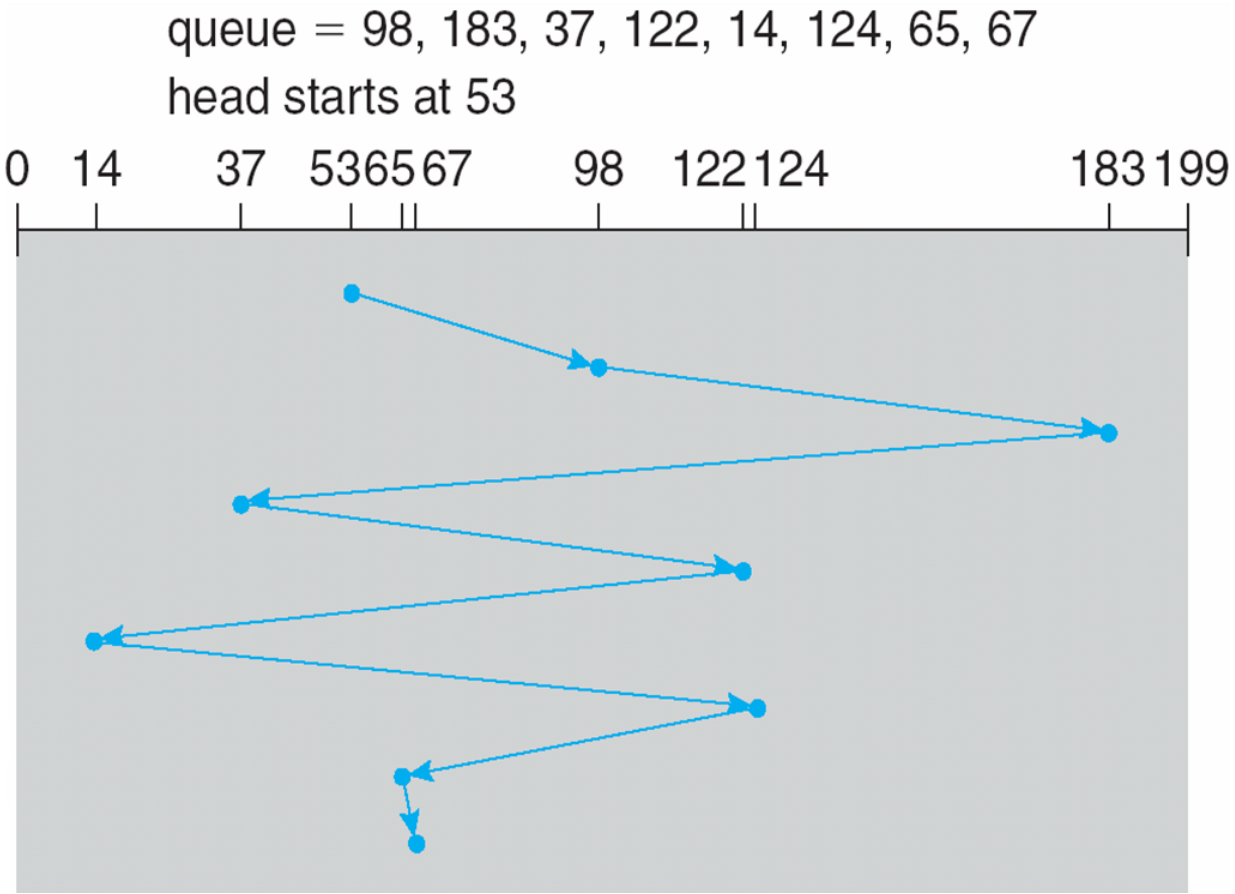
- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

FCFS

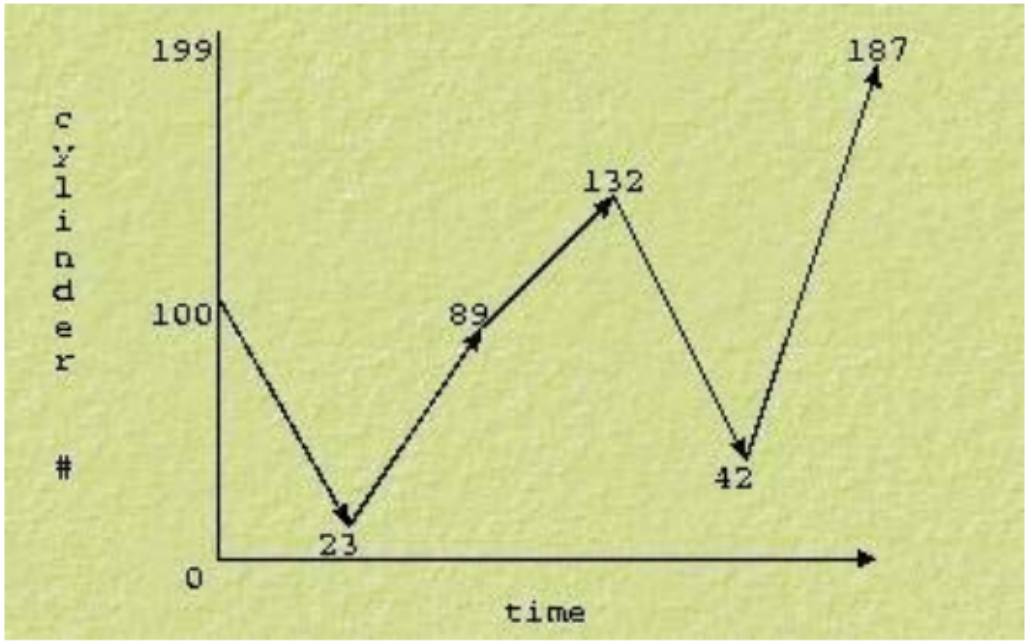
Illustration shows total head movement of **640** cylinders



FCFS

FCFS

23, 89, 132, 42, 187

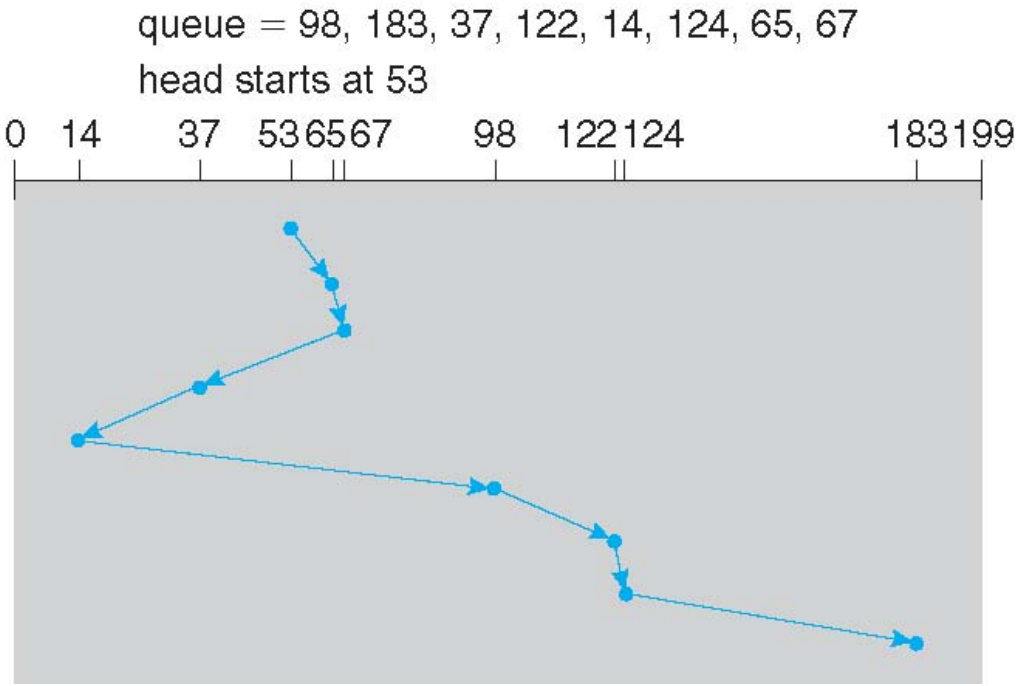


77+66+43+90+145=421

If the requests for cylinders 23 and 42 could be serviced together, total head movement could be decreased substantially.

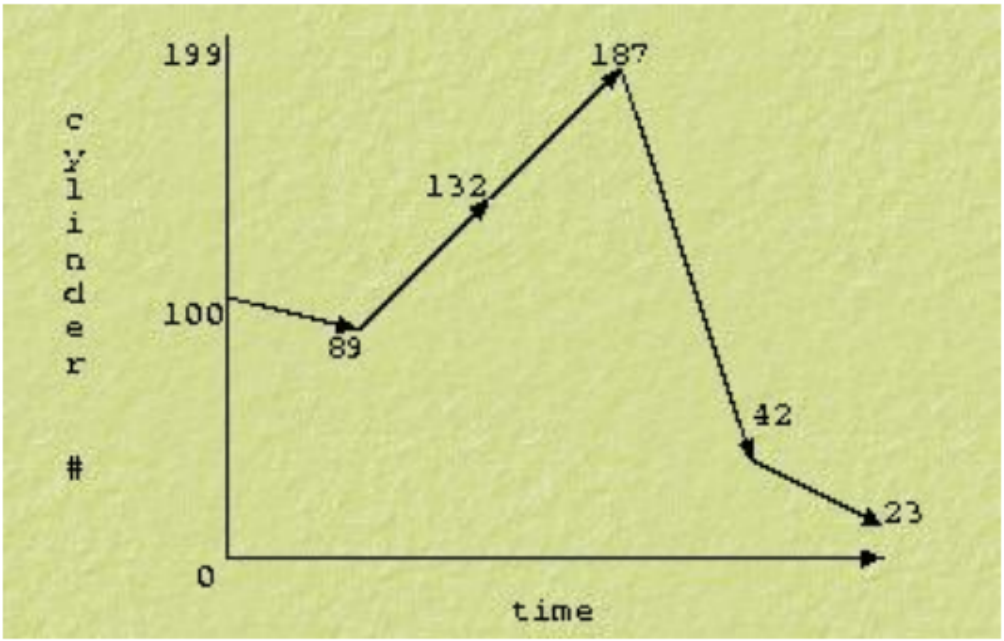
SSTF

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



SSTF

23, 89, 132, 42, 187

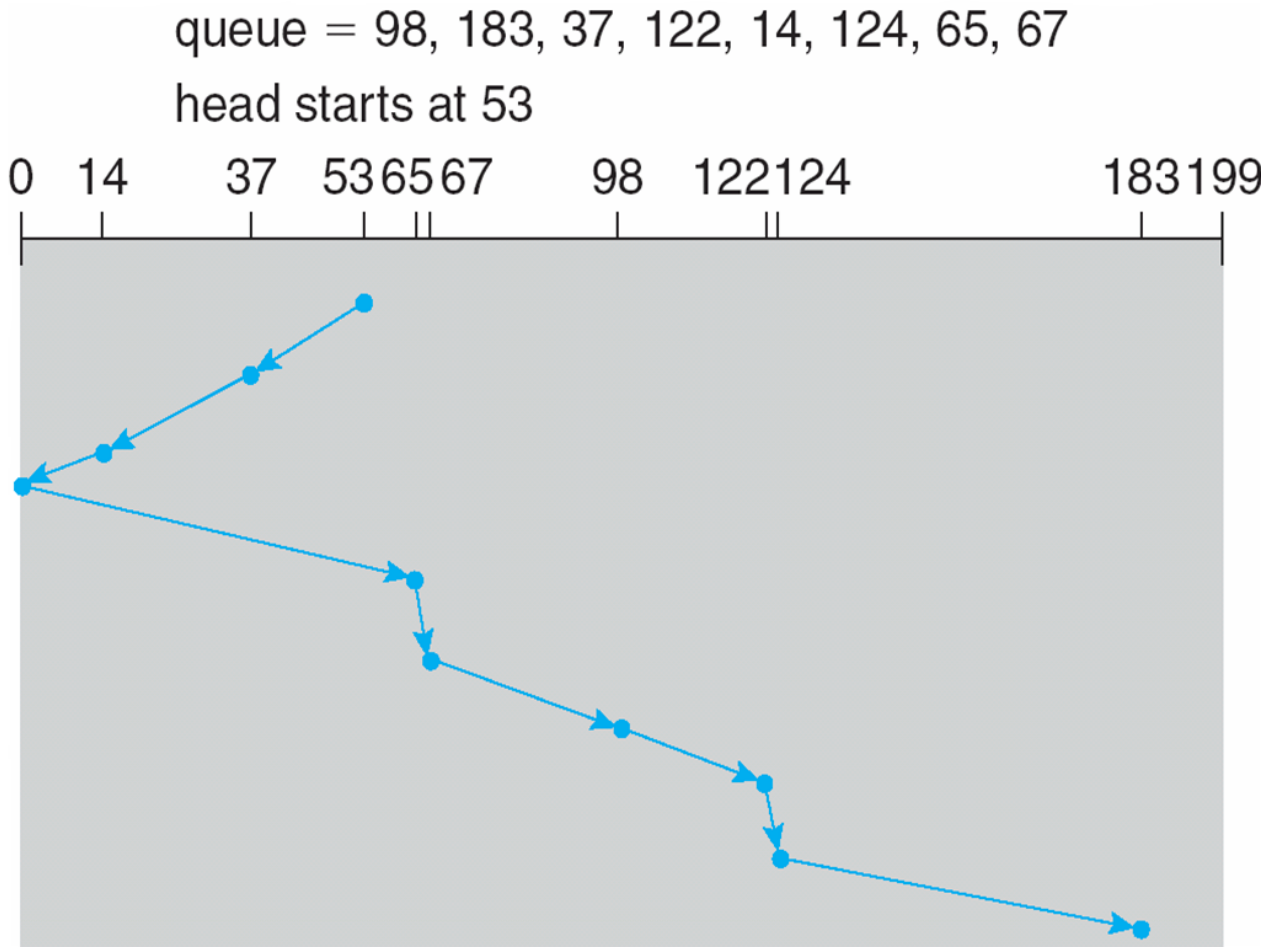


11+43+55+145+19=273

SCAN

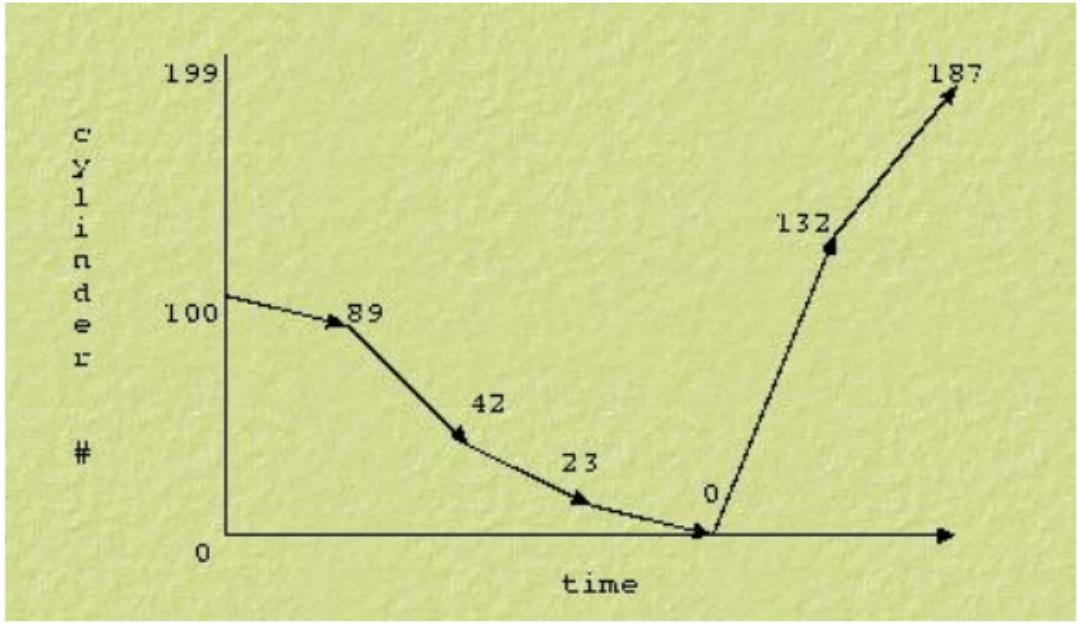
- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
- **SCAN algorithm** Sometimes called the **elevator algorithm**
- Illustration shows total head movement of 208 cylinders
- ***But note that if requests are uniformly dense, the largest density is at other end of disk and thus they will wait the longest***
  - ***Sol: C-SCAN***

SCAN (Cont.)


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

SCAN

23, 89, 132, 42, 187



11+47+19+23+132+55=287

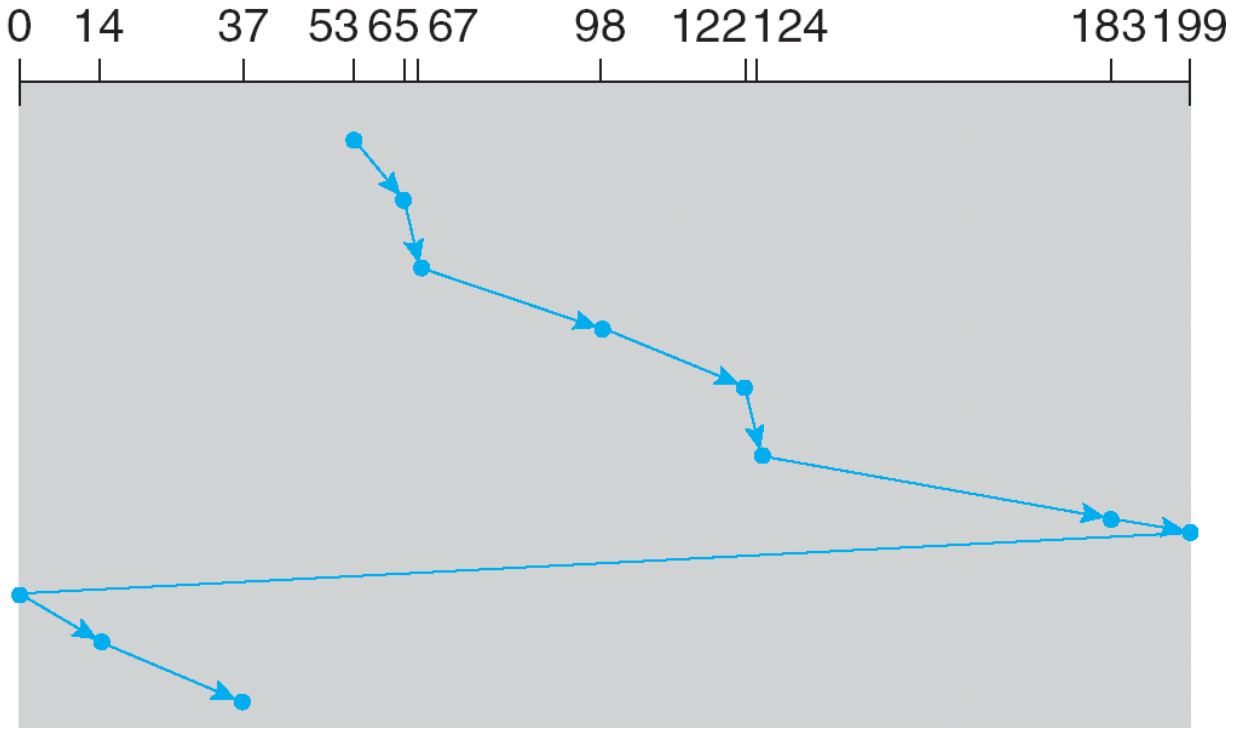
C-SCAN

- 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

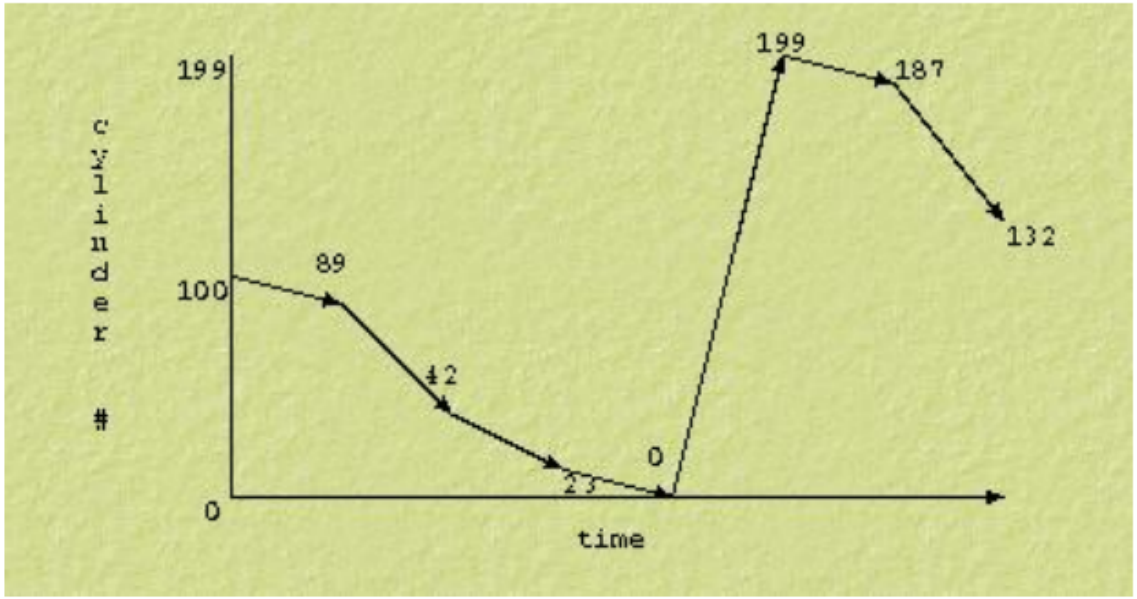
head starts at 53



C-SCAN

C-SCAN

23, 89, 132, 42, 187



11+47+19+23+199+12+55=366

Head movement can be reduced if the request for cylinder 187 is serviced directly after request at 23 without going to the disk 0

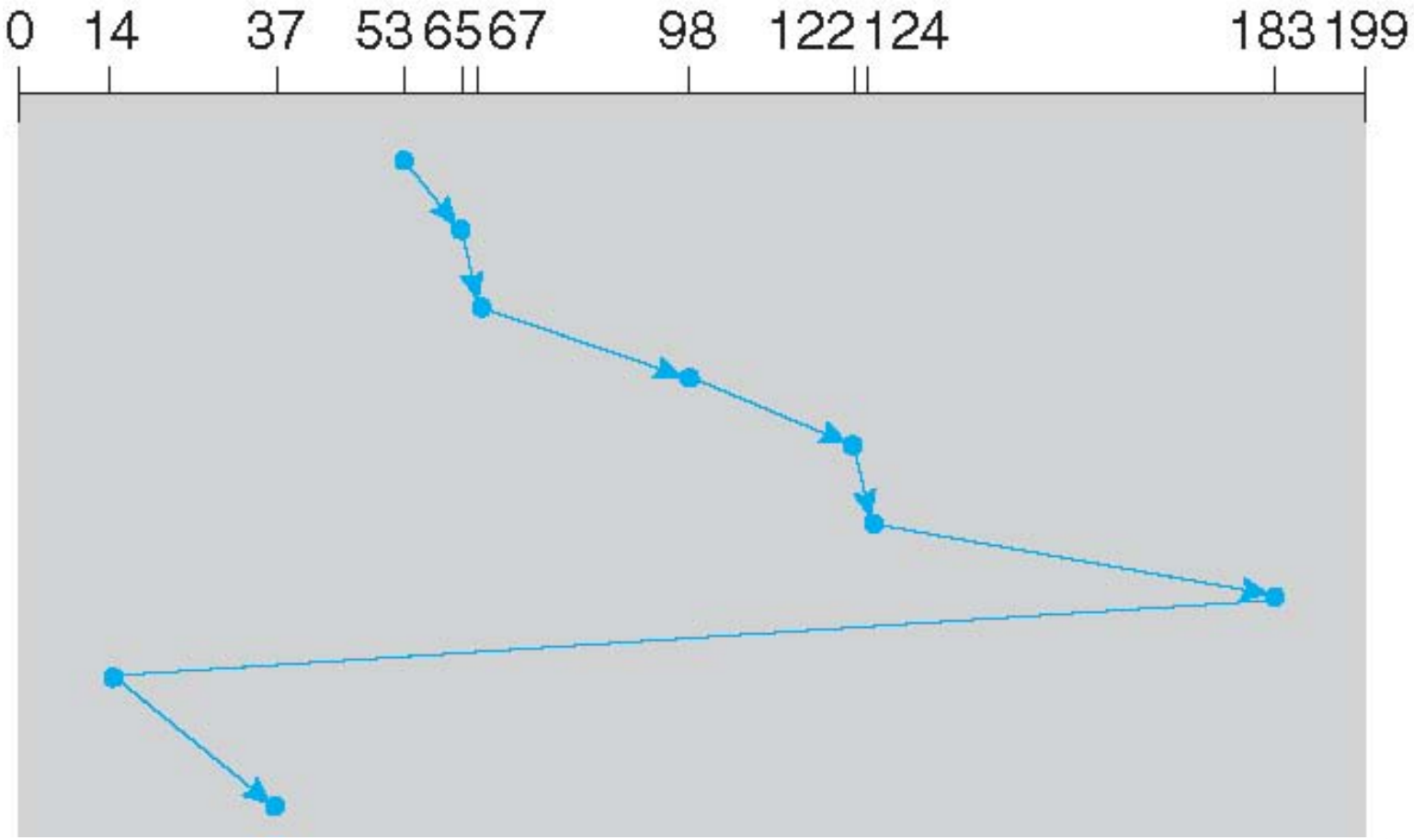
C-LOOK

- 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

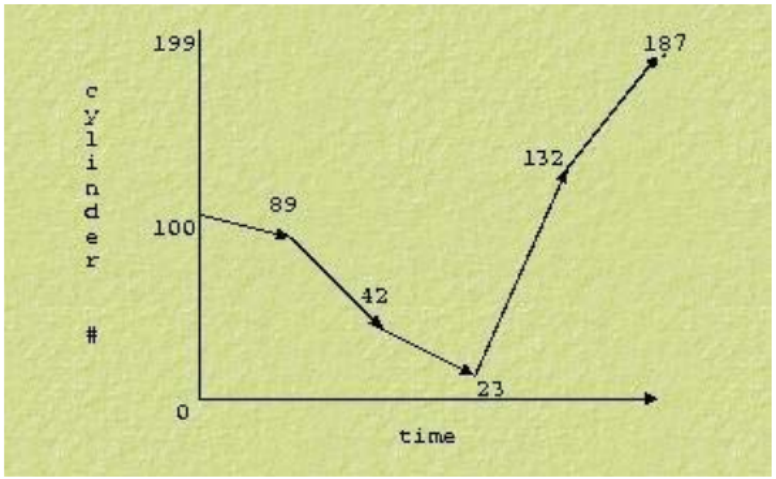


LOOK

- like SCAN but stops moving inwards (or outwards) when no more requests in that direction exist

LOOK

23, 89, 132, 42, 187



11+47+19+109+55=241

Compared to SCAN, LOOK saves going from 23 to 0 and then back.

Most efficient for this sequence of requests

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
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method
- 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

Selecting a Disk-Scheduling Algorithm

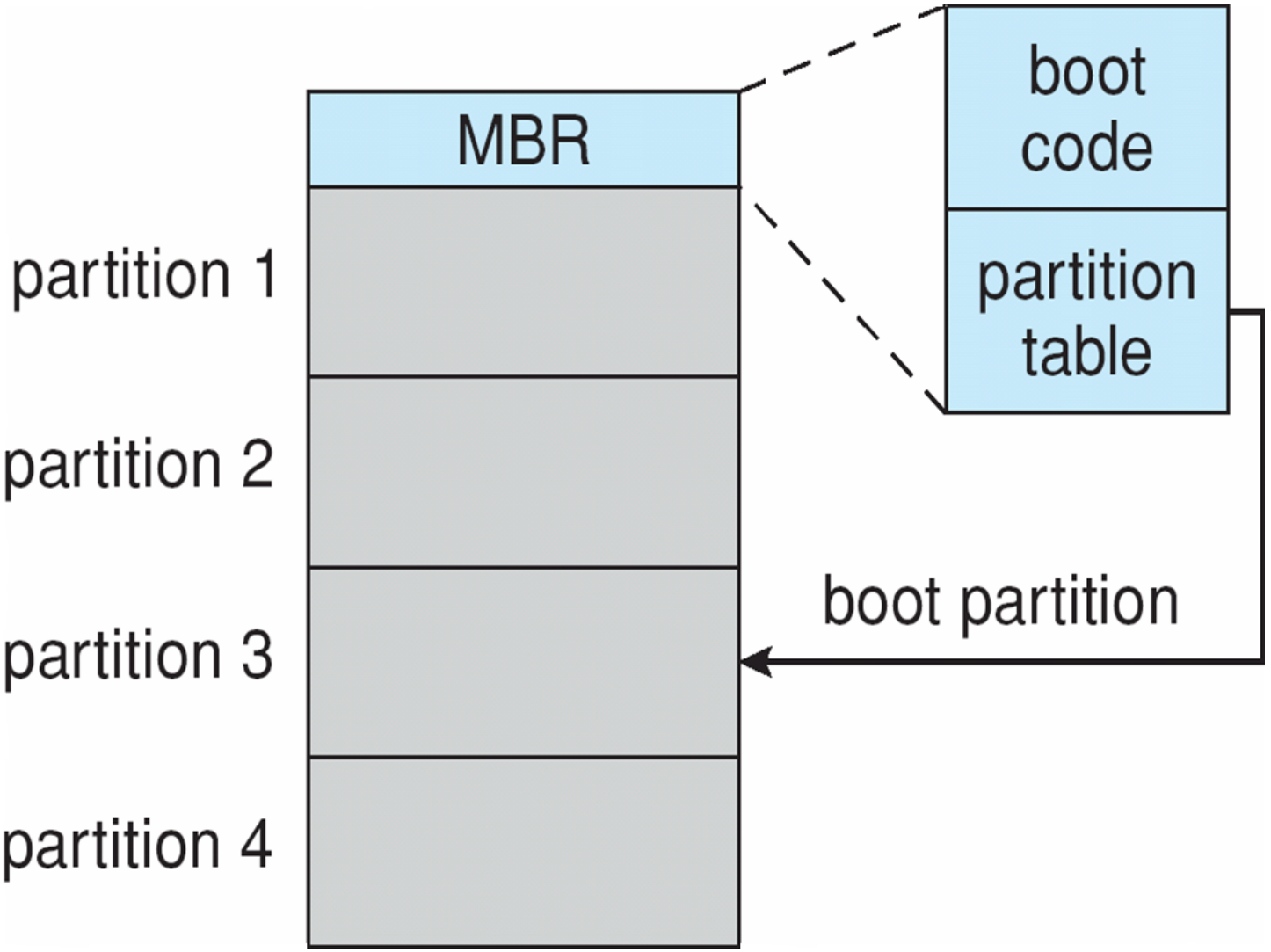
- **SSTF is common** and has a natural appeal
- SCAN and C-SCAN perform better for systems that place a **heavy load** on the disk
  - **Less starvation**
- The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary

Thank You!

Disk Management

- **Low-level formatting**, or **physical formatting** — Dividing a disk into sectors that the disk controller can read and write
- 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
  - **Logical formatting** or “making a file system”
  - To increase efficiency most file systems group blocks into **clusters**
    - Disk I/O done in blocks
    - File I/O done in clusters
- Boot block initializes system
  - The bootstrap is stored in ROM
  - **Bootstrap loader** program
- Methods such as **sector sparing** used to handle bad blocks

Booting from a Disk in Windows 2000



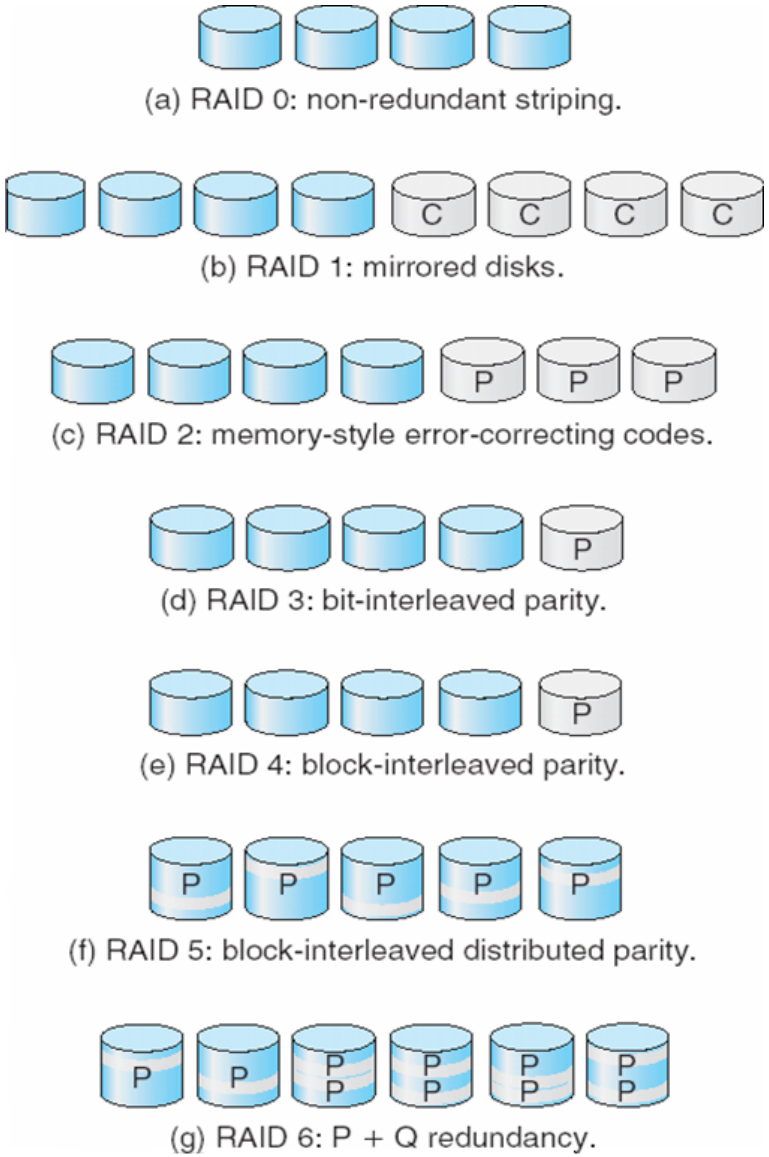
RAID Structure

- RAID – multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure**
- Frequently combined with **NVRAM** to improve write performance
- RAID is arranged into six different levels

RAID (Cont.)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively
- Disk **striping** uses a group of disks as one storage unit
- 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.

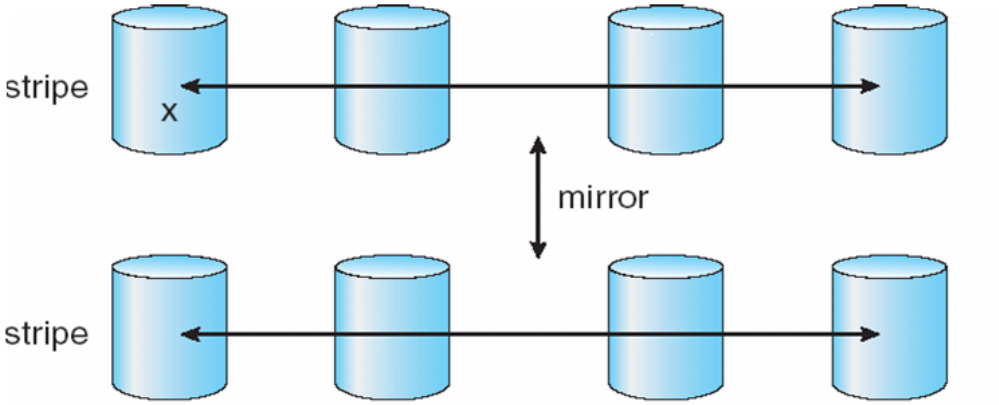
(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

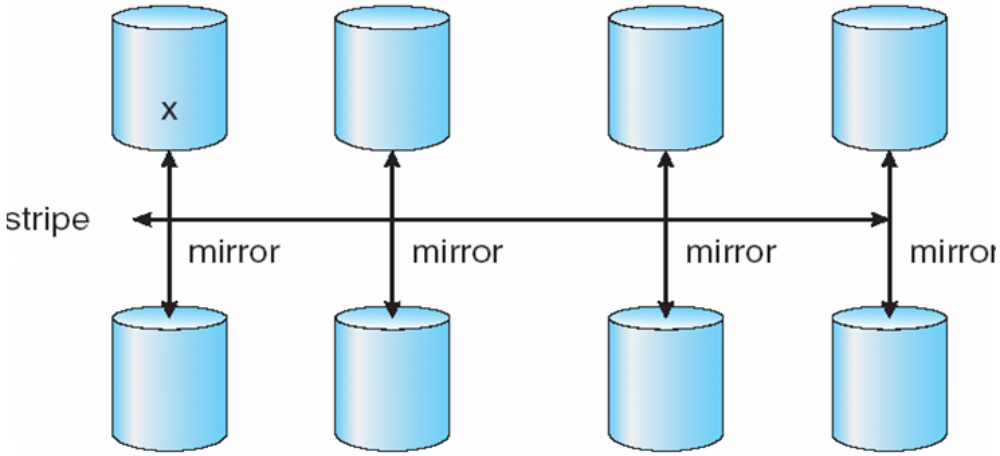
(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.