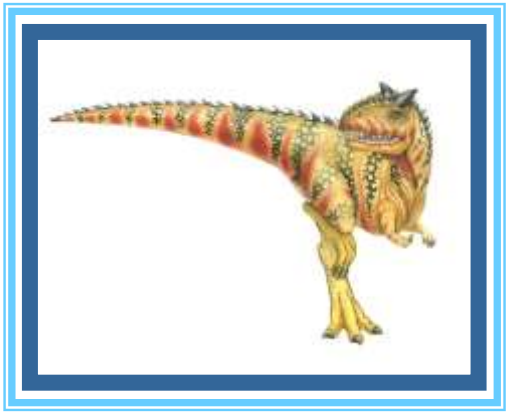
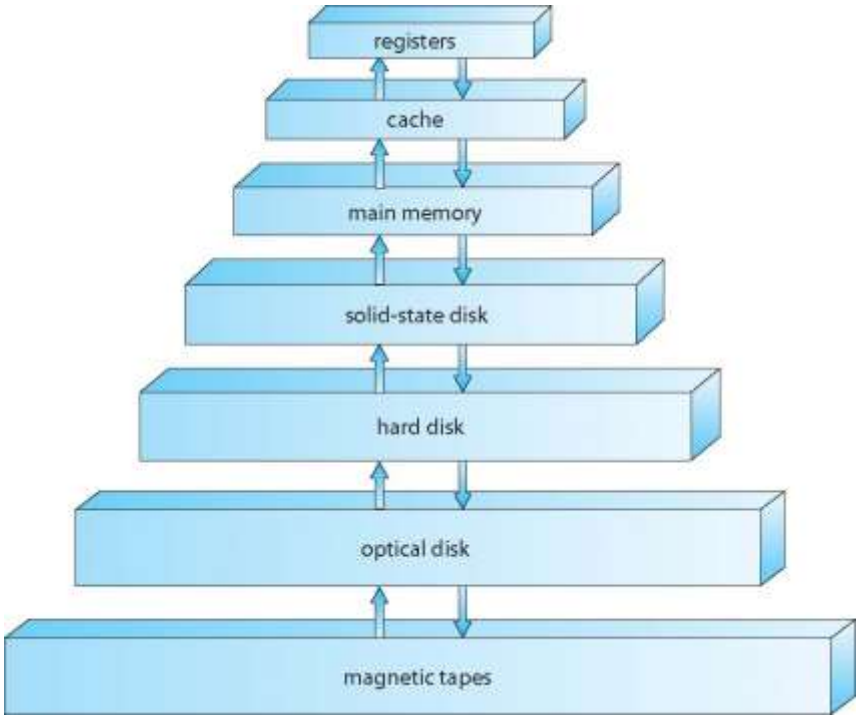
Chapter 10: Mass-Storage Systems

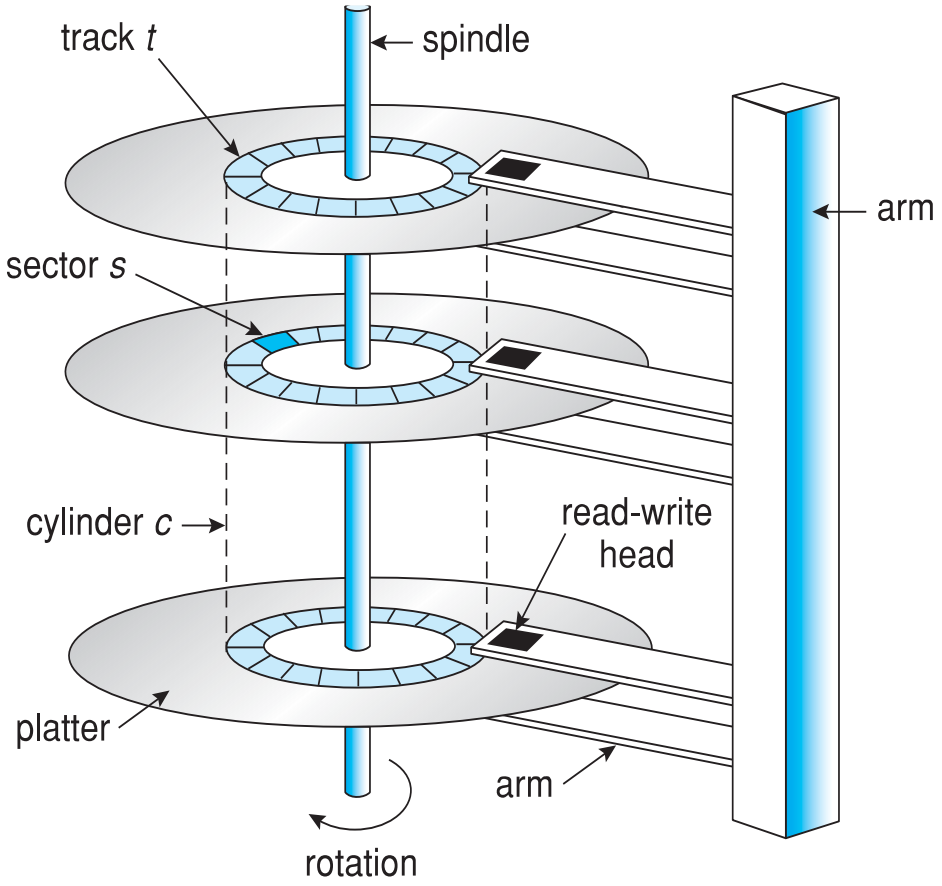


Silberschatz, Galvin and Gagne ©2013



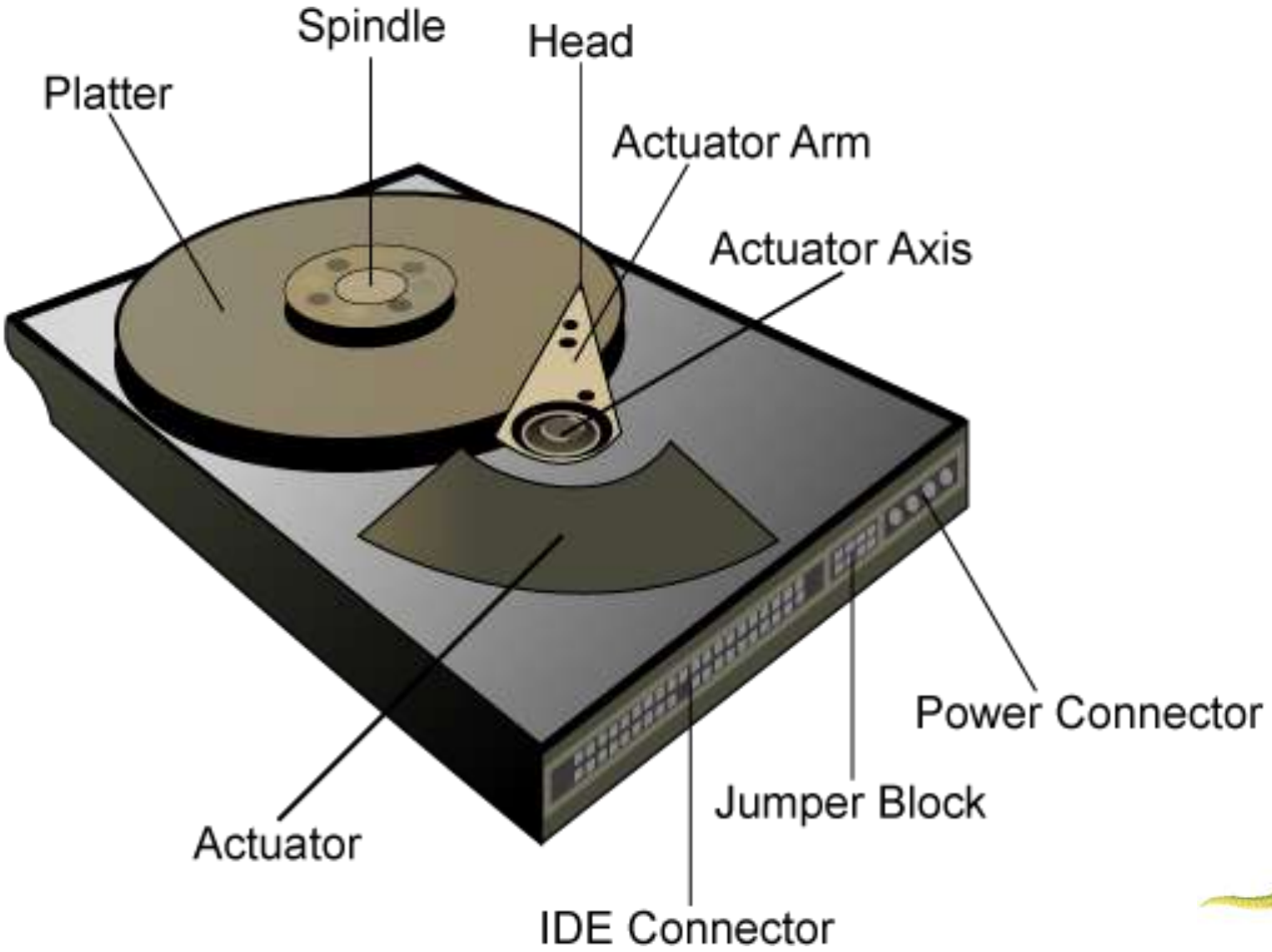
Since main memory is usually too small to accommodate all the data and programs permanently, the computer system must provide secondary storage to back up main memory. Modern computer systems use disks as the primary on-line storage medium for information (both programs and data).

Overview of Magnetic Disks



- **Magnetic disks** provide bulk of secondary storage of modern computers
  - Each disk **platter** has a flat circular shape, like a CD
  - We store information by recording it magnetically on the platters
  - A read–write head "flies" just above each surface of every platter
  - The surface of a platter is logically divided into circular **tracks**, which are subdivided into **sectors**. The set of tracks that are at one arm position makes up a **cylinder**.

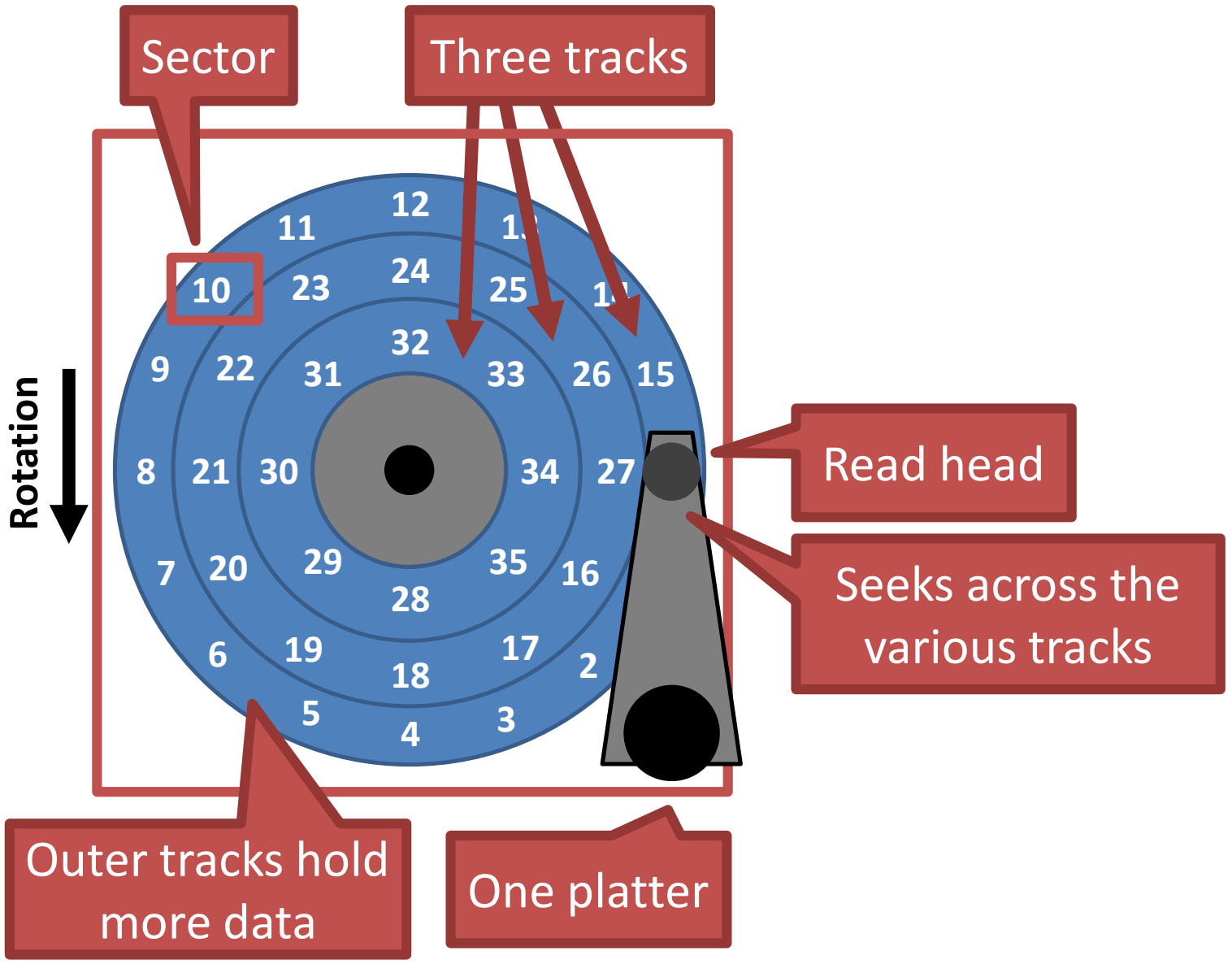
Hard Drive Hardware



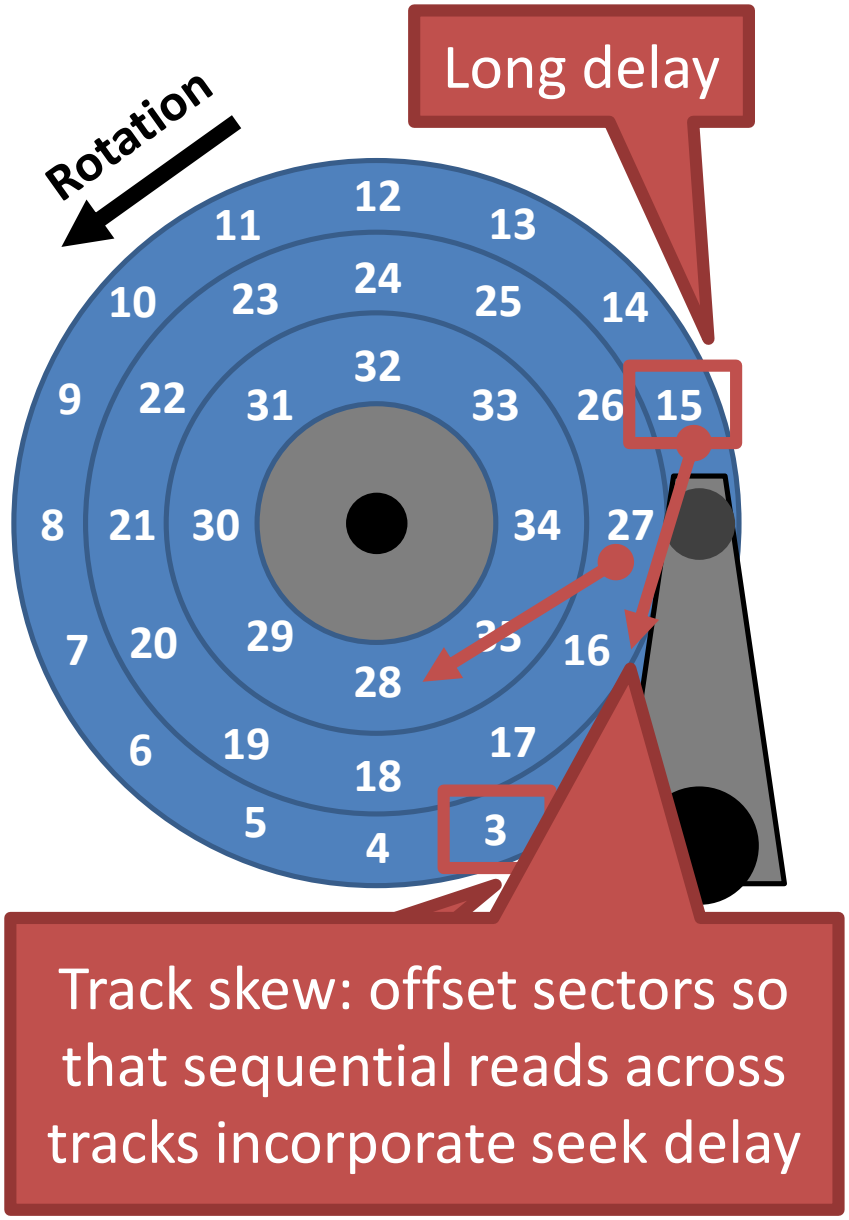
Addressing and Geometry

- Externally, hard drives expose a large number of sectors (blocks)
  - Typically 512 or 4096 bytes
  - Individual sector writes are atomic
  - Multiple sectors writes may be interrupted
- Drive geometry
  - Sectors arranged into tracks
  - A cylinder is a particular track on multiple platters
  - Tracks arranged in concentric circles on platters
  - A disk may have multiple, double-sided platters
- Drive motor spins the platters at a constant rate
  - Measured in revolutions per minute (RPM)

Geometry Example



Types of Delay With Disks

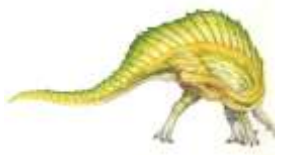


Three types of delay

1. Rotational Delay
   - Time to rotate the desired sector to the read head
   - Related to RPM
2. Seek delay
   - Time to move the read head to a different track
3. Transfer time
   - Time to read or write bytes

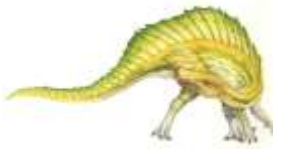
Overview of Magnetic Disks

- When the disk is in use, a drive motor spins it at high speed
- Disk speed has two parts
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**)
    - **Seek time** is the time for the disk arm to move the heads to the **cylinder** containing the desired sector
    - **Rotational latency** is the additional time for the disk to rotate the desired **sector** to the disk head



Hard Disk Performance

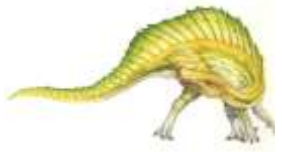
- Positioning time = seek time + rotational latency
  - For fastest disk 3ms + 2ms = 5ms
  - For slow disk 9ms + 5.56ms = 14.56ms
- Average I/O time = Positioning time + (amount to transfer / transfer rate) + controller overhead
- For example, to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with 0.1ms controller overhead
  - Rotational latency = (1/120)/2 = 4.17ms
  - 5ms + 4.17ms + 0.1ms + transfer time =
  - Transfer time = $$\frac{4KB}{1Gb/s} = \frac{4 \times 1024 \times 8bit}{1024^3 bit/s} = \frac{32}{1024^2} = 0.031ms$$
  - Average I/O time for 4KB block = 9.27ms + 0.031ms = 9.301ms



Disk Scheduling

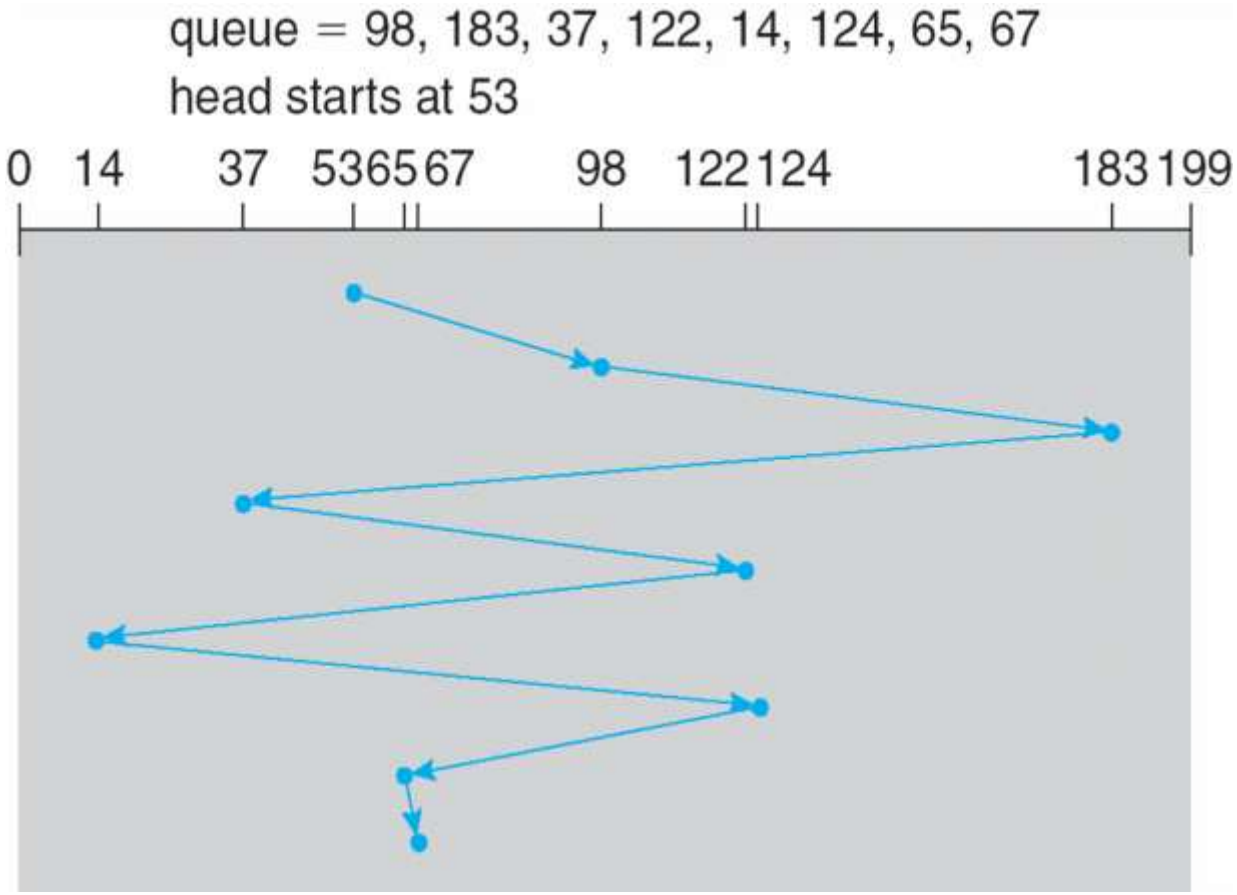
- The OS is responsible for using hardware efficiently — for the disk drives, this means having a fast access time (Positioning time) and large disk bandwidth
- **Minimize seek time**
- Seek time ≈ seek distance
- Disk **bandwidth** is the total number of bytes transferred, divided by the total time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

We can improve both the access time and the bandwidth by managing the order in which disk I/O requests are serviced



FCFS

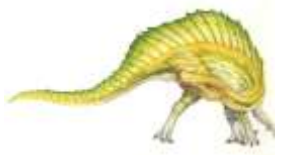
A disk queue with requests for I/O to blocks on cylinders



This algorithm is intrinsically fair, but it generally does not provide the fastest service

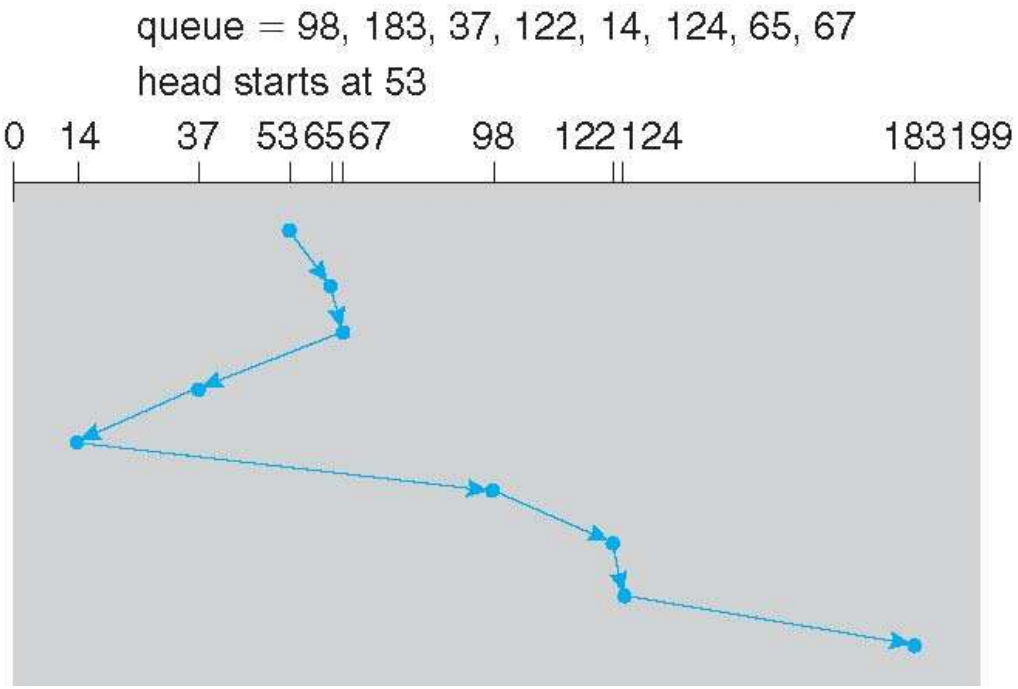
The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule.

Illustration shows total head movement of 640 cylinders

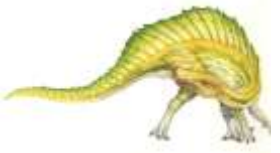


SSTF

- **Shortest Seek Time First** selects the request with the minimum seek time from the **current head position**
  - Service all the requests **close to** the current head position before moving the head far away to service other requests

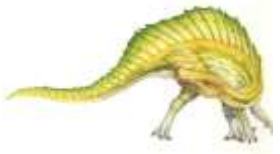


- Illustration shows total head movement of 236 cylinders



SSTF (cont.)

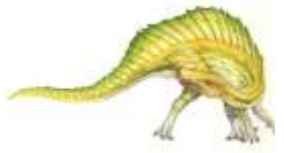
- Think about one downside of SSTF
- SSTF may cause starvation of some requests
- Although the SSTF algorithm is a substantial improvement over the FCFS algorithm, it is **not optimal**
  - See next algorithm



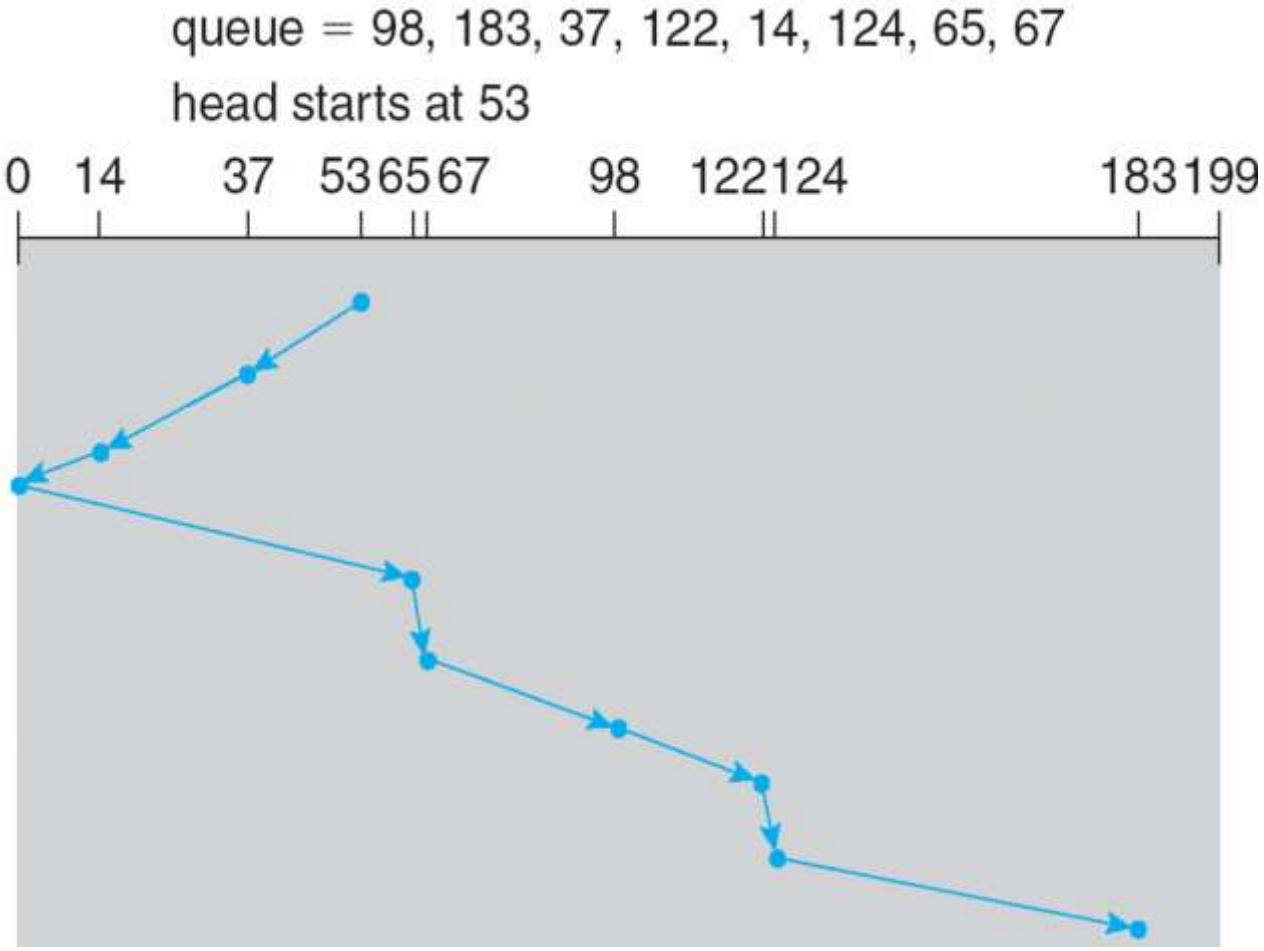
SCAN

- The disk arm starts at one end of the disk, and moves toward the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk.
- At the other end, the direction of head movement is reversed, and servicing continues

- The head continuously scans back and forth across the disk
- **SCAN algorithm** Sometimes called the **elevator algorithm**



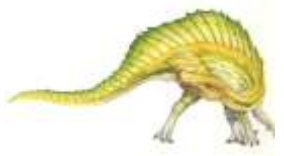
SCAN (Cont.)



If a request arrives in the queue just in front of the head, it will be serviced almost immediately;

A request arriving just behind the head will have to wait until the arm moves to the end of the disk, reverses direction, and comes back.

- Illustration shows total head movement of 208 cylinders

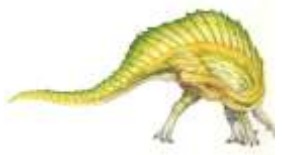


C-SCAN

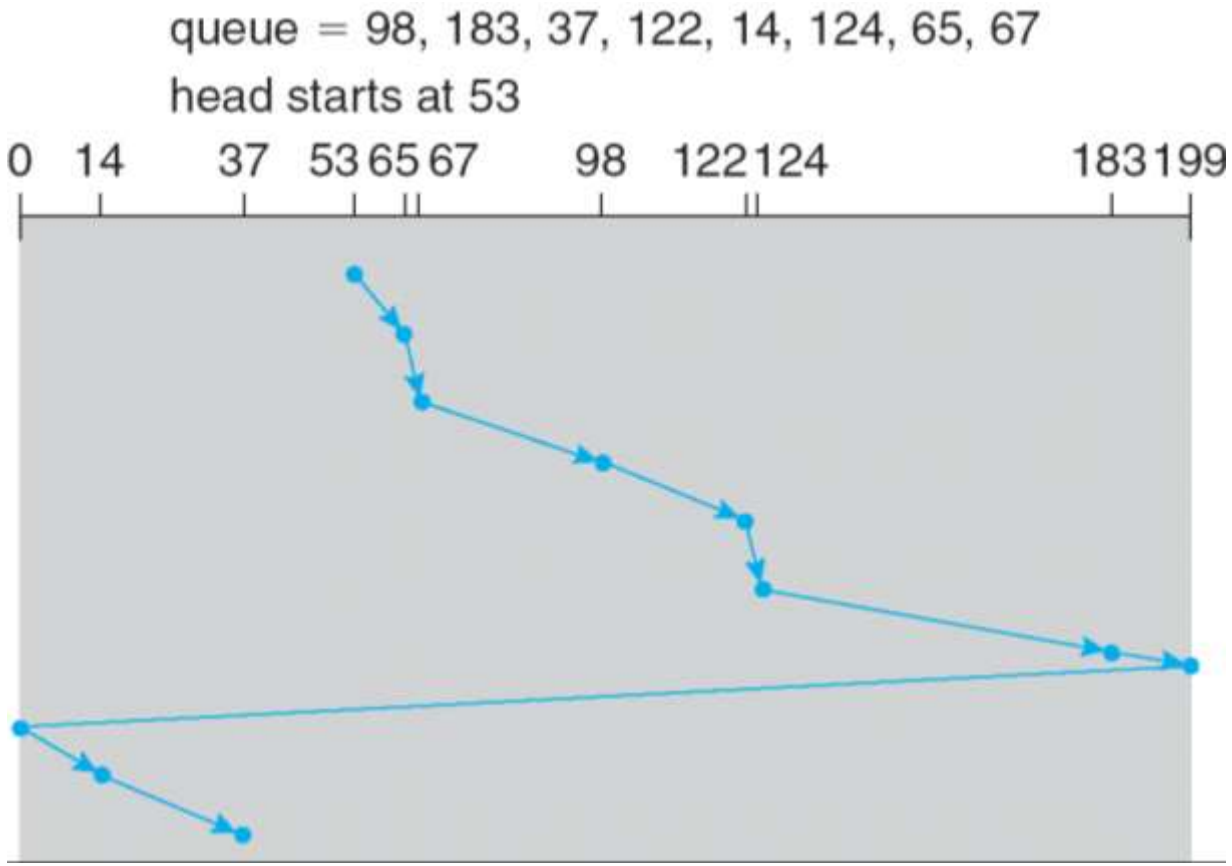
- **Circular SCAN** (**C-SCAN**) is a variant of SCAN
- Provides a more uniform wait time than SCAN

- 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

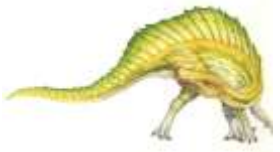
- Treats the cylinders as a circular list that wraps around from the last cylinder to the first one



C-SCAN (Cont.)

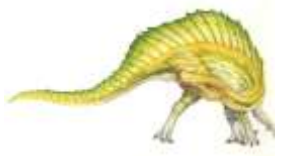


Total number of cylinders?



C-LOOK

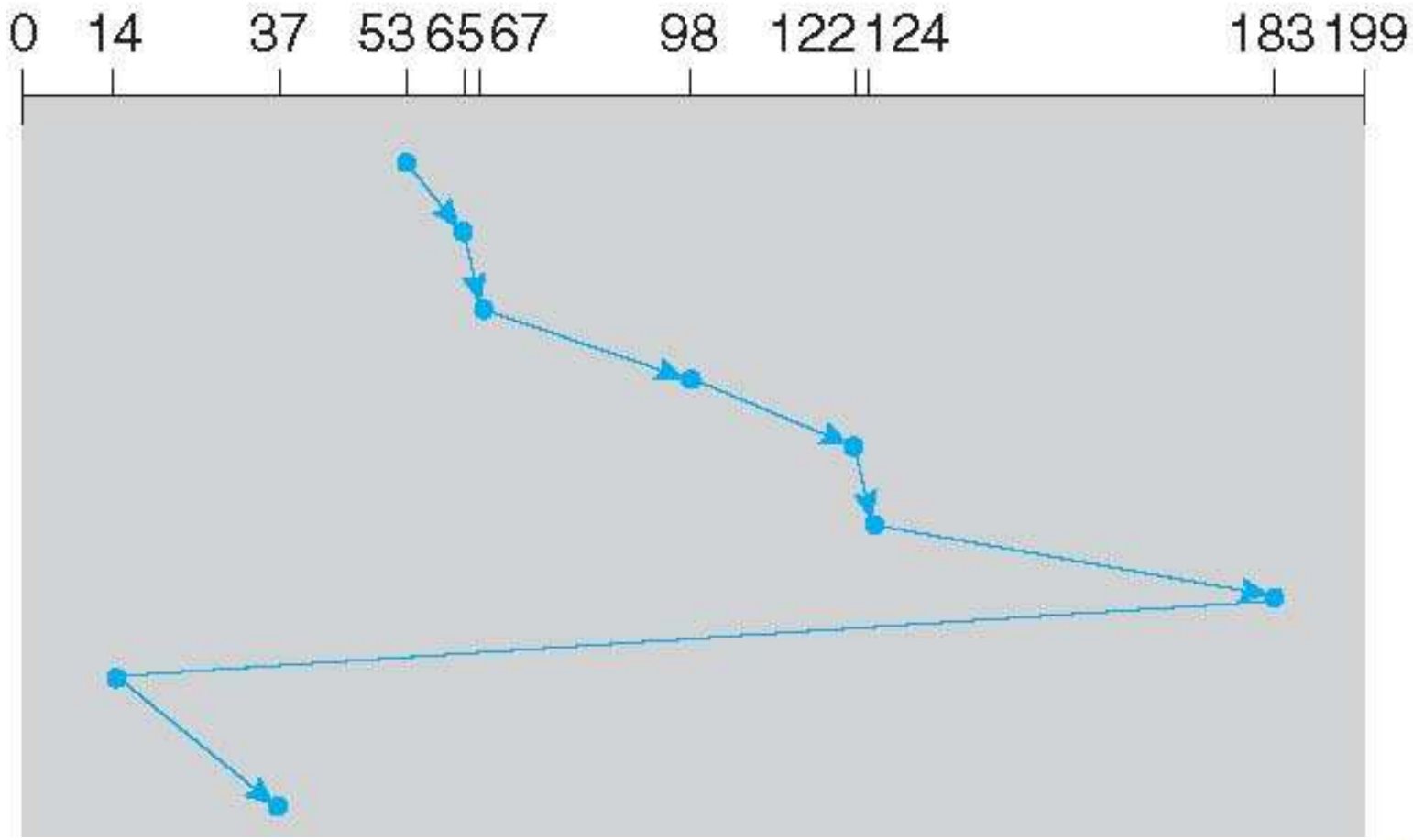
- Both SCAN and C-SCAN move the disk arm across the full width of the disk.
- In C-LOOK, arm only **goes as far as the final request** in each direction, then **reverses** direction immediately, without first going all the way to the end of the disk
- **LOOK** a version of SCAN, **C-LOOK** a version of C-SCAN



C-LOOK (Cont.)

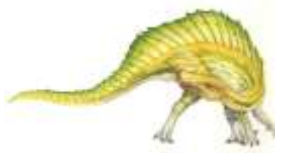
queue = 98, 183, 37, 122, 14, 124, 65, 67

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
- Requests for disk service can be influenced by the file-allocation method
  - e.g., a contiguously allocated file resulting in limited head movement
  - e.g., a linked or indexed file, blocks that are widely scattered on the disk, resulting in greater head movement
- What about rotational latency?
  - Difficult for OS to schedule for improved rotational latency



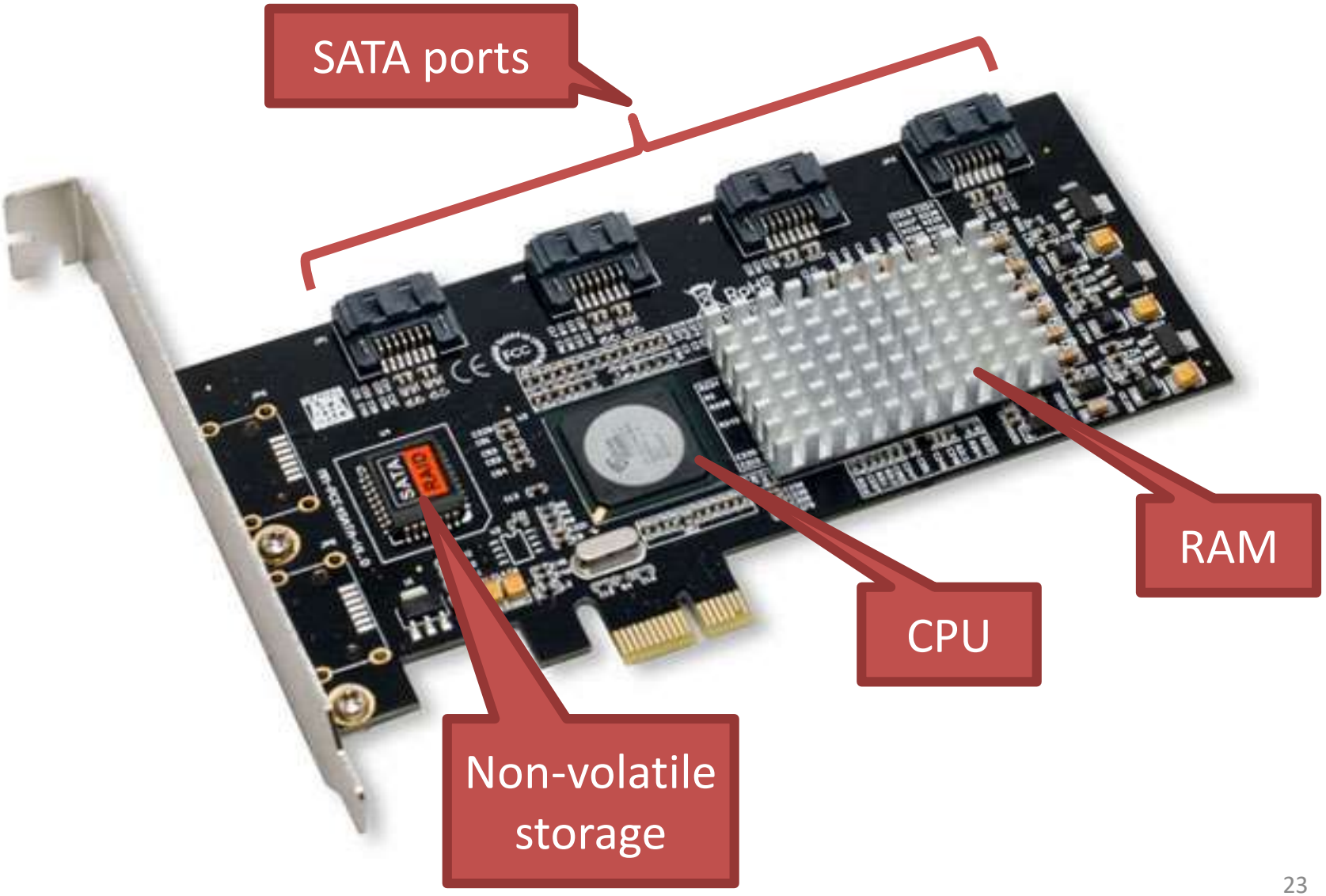
Beyond Single Disks

- Hard drives are great devices
  - Relatively fast, persistent storage
- Shortcomings:
  - How to cope with disk failure?
    - Mechanical parts break over time
    - Sectors may become silently corrupted
  - Capacity is limited
    - Managing files across multiple physical devices is cumbersome
    - Can we make 10x 1 TB drives look like a 10 TB drive?

Redundant Array of Inexpensive Disks

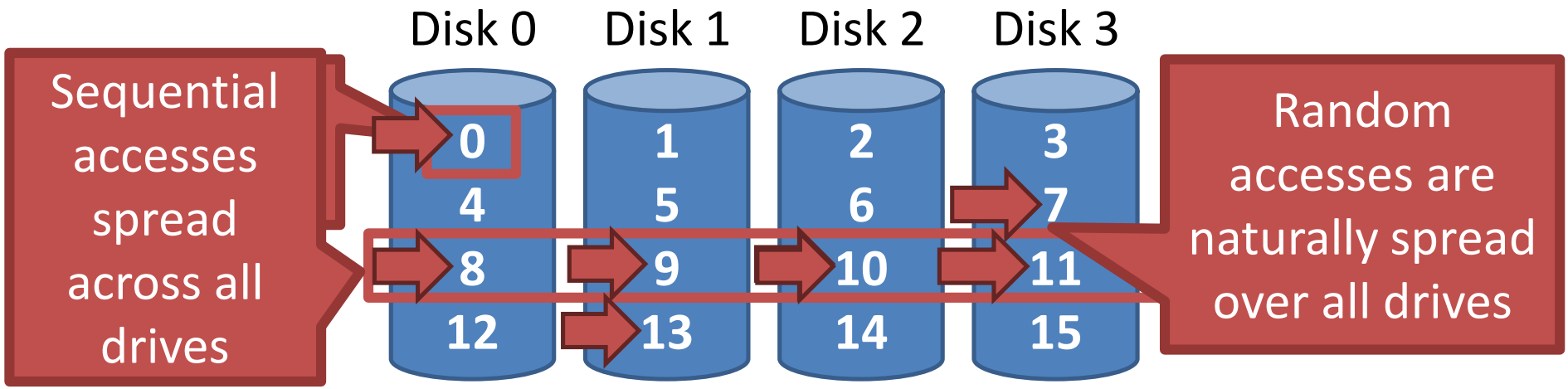
- RAID: use multiple disks to create the illusion of a large, faster, more reliable disk
- Externally, RAID looks like a single disk
  - i.e. RAID is transparent
  - Data blocks are read/written as usual
  - No need for software to explicitly manage multiple disks or perform error checking/recovery
- Internally, RAID is a complex computer system
  - Disks managed by a dedicated CPU + software
  - RAM and non-volatile memory
  - Many different configuration options (RAID levels)

Example RAID Controller



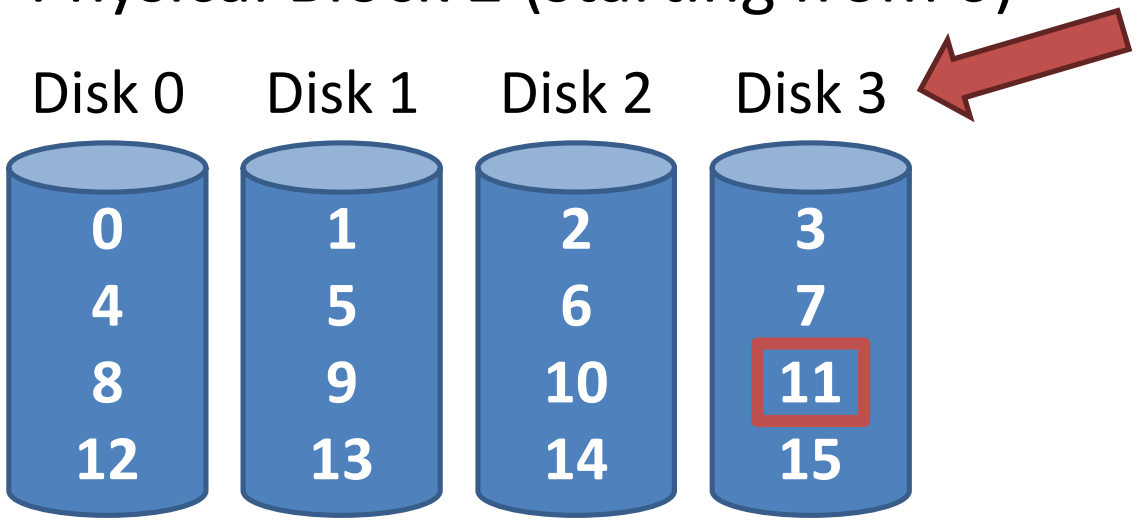
RAID 0: Striping

- Key idea: present an array of disks as a single large disk
- Maximize parallelism by striping data cross all *N* disks



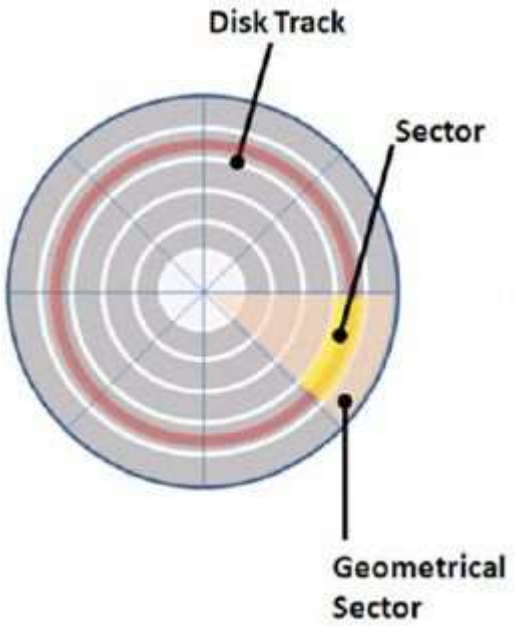
Addressing Blocks

- How do you access specific data blocks?
  - Disk = logical_block_number % number_of_disks
  - Offset = logical_block_number / number_of_disks
- Example: read block 11
  - 11 % 4 = Disk 3
  - 11 / 4 = Physical Block 2 (starting from 0)



Old type

- Are there same number of sectors in different tracks?
  - There are two types of disk
    - **old type**: the same number of sectors in different tracks
    - **new type**: the different number of sectors in different tracks

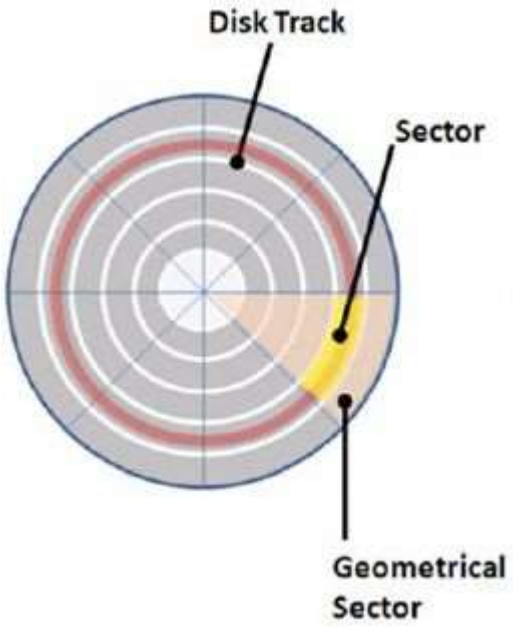


old type

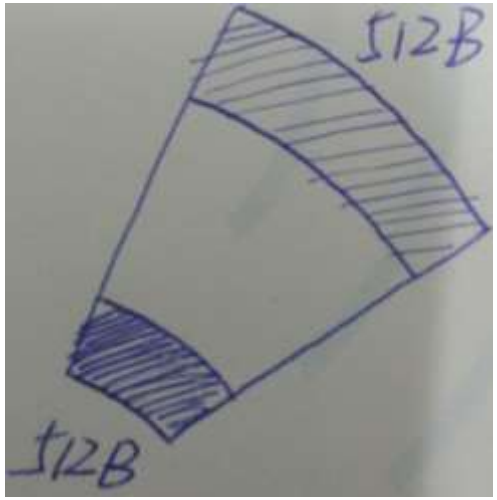
We can observe:

1. sectors increase in size farther away from the center.
2. the rotation speed increases farther away from the center.

Old type



old type

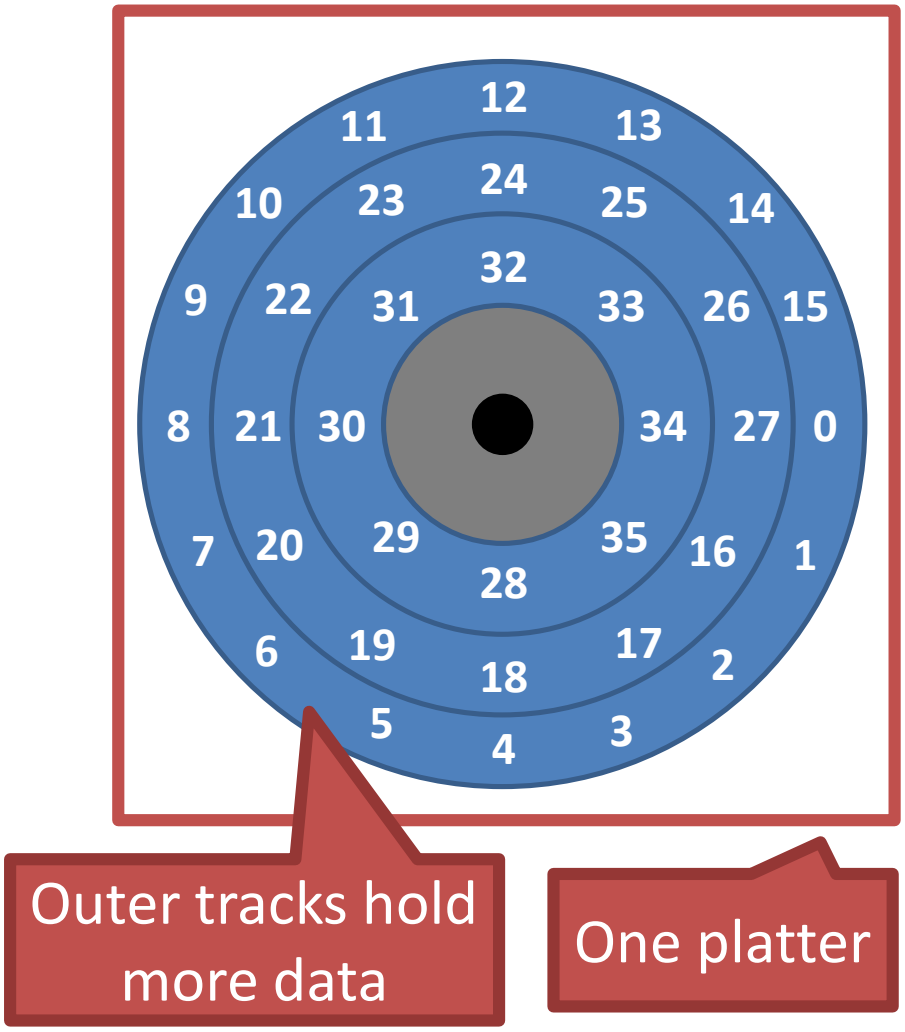


We can observe:

3. Since each sector can hold the same amount of data (512 bytes), so the data density of the outer sector is low, and the velocity density of the inner sector is high

4. Then a large area of the outer sector is wasted

New type



Since tracks increase in size further away from the center, the more sectors should be divided, each with the same area and the same amount of data

Review: old type

- If you have an empty hard disk and now write to a file, do you write to all sectors of the same track first and then write to another track?
  - Yes, write the file from outside to inside, first filling the outermost track, then the inner track.

- Why after one disk is used for a long time, its read and write speed will be slow?
  - Because when the outer track is used up, start to use the inner track
  - Tracks decrease in read/write speed further close to the center
  - Then disk organization or formatting is needed, which allows the outer track to be used

Transfer Time

- The time it takes to transmit or move data from one place to another.
- It is the time interval between the start of the transfer and the completion of the transfer.
- Pretty fast — depends on RPM and sector density.

Transfer Time

- T = b/rN
  - T is Transfer time
  - b is number of bytes to be transferred
  - N is number of bytes on a track
  - r is rotational speed in RPM

Workload Performance

- So…
  - seeks are slow
  - rotations are slow
  - transfers are fast
- What kind of workload is faster for disks?
  - ***Sequential***: access sectors in order (transfer dominated)
  - ***Random***: access sectors arbitrarily (seek+rotation dominated)

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

- Sequential workload: what is throughput for each?
  - Cheetah: 125 MB/s
  - Barracuda: 105 MB/s

Disk Spec

	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

- Random workload: what is throughput for each?
- Assume size of each random read is 16KB

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

Time = seek + rotate + transfer

Seek = 4 ms

Full rotation = 60 / (15,000) = 4 ms Half rotation = 2 ms

Transfer = 16 KB / 125 MBps = 125 us

Throughput = 16 KB / (6.125 ms) = **2.5 MBps**

	Cheetah	Barracuda
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s

Time = seek + rotate + transfer

Seek = 9 ms

Full rotation = 60 / (7,200) = 8.3 ms Half rotation = 4.1 ms

Transfer = 16 KB / 100 MBps = 160 us

Throughput = 16 KB / (13.260 ms) = **1.2 MBps**

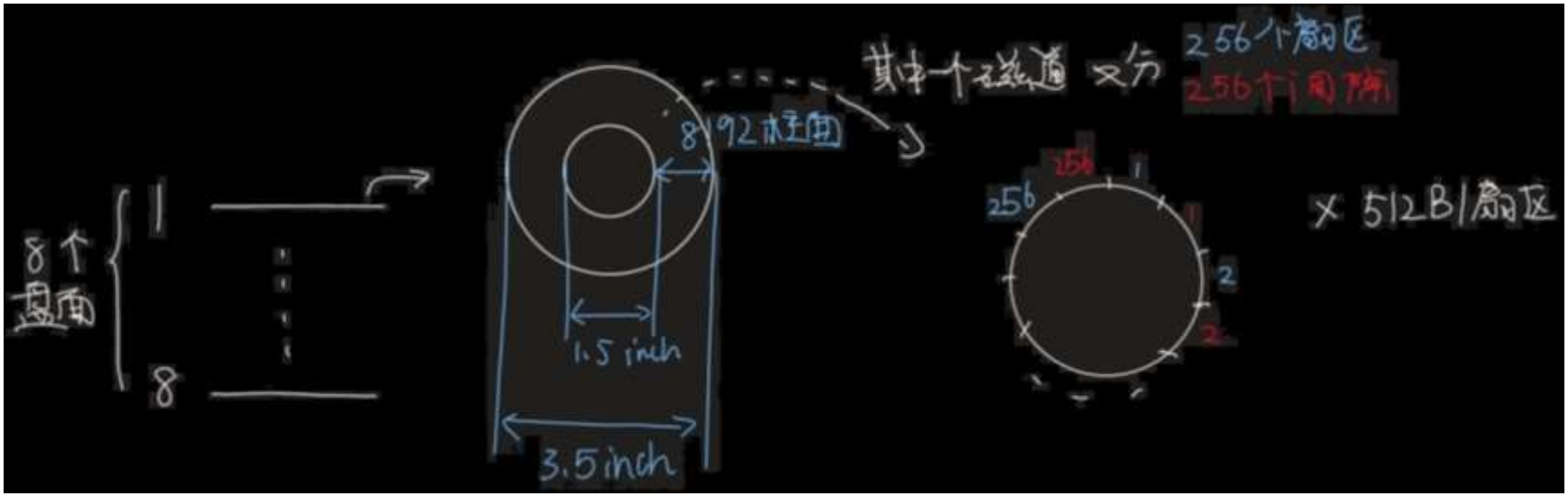
	Cheetah	Barracuda
Capacity	300 GB	1 TB
RPM	15,000	7,200
Avg Seek	4 ms	9 ms
Max Transfer	125 MB/s	105 MB/s
Platters	4	4
Cache	16 MB	32 MB

	Cheetah	Barracuda
Sequential	125 MB/s	105 MB/s
Random	2.5 MB/s	1.2 MB/s

This shows the importance of proper ***disk scheduling*** to achieve good disk performance.

Exercise 1

- Assume that a disk has the following characteristics:
  - There are 8 platters and 8192 cylinders
  - The disk diameter is 3.5 inches, in which the inner ring **does not** store data, and the inner ring diameter is 1.5 inches
  - Each track has an average of 256 sectors, each sector size is 512 bytes
  - 10% of each track are gaps
  - The disk speed is 7200 RPM
  - It takes 1ms to start and stop the magnetic head, and 1ms is needed for every 500 cylinders moved



Exercise 1

- What's the disk size?

Answer:

C = 8 platters * 8192 cylinders * 256 sectors * 512bytes/sector = 8GB

Exercise 1

- If all the tracks have the same number of sectors, what is the density (bits per inch) of the innermost track?

Answer:

Track density = track capacity ***c*** / track length *l*

Because all tracks have the same number of sectors, all tracks have the same capacity: c = 256 sectors * 512 Bytes/sector * 8 bits

Since for each track, gap part occupies 10%, so efficient track part occupy 90%.

Then the length of the innermost track is: $l = 1.5 \text{ inch} * \pi * 90\%$

So the density of the innermost track is c/l = 247,238.598 bpi

Exercise 1

- What is the transfer time for a block with 8KB?

Answer:

- One track capacity c = 256 sectors * 512 Bytes/sector = 2^7 KB, so this 8KB block is in one track.
- Transfer time T_t = the ratio of a block to a track * the time it takes to make one disk revolution
- 8KB / (512B/sector) = 16 sectors. There are 15 gaps between sector 1 and 16. A track has 256 sectors accounting for 90%, 256 gaps accounting for 10%
- Therefore, the ratio of a block to a track is:

  (16/256) * 90% + (15/256) * 10% = 0.0617
- One disk revolution takes: 1 / (7200 RPM) = 8.333 ms
- Therefore, T_t = 0.0617 * 8.333 ms = 0.514 ms

Exercise 1

- What's the average seek time?

Answer:

Average seek time T_f= head start/stop time + average track number to move * movement time between adjacent tracks

Average track number to move = 8192 tracks / 3 = 2730 tracks

T_f = 1 ms + 2730 tracks * (1ms / 500 tracks) = 6.5 ms

Exercise 1

- What's the average rotation time?

Answer:

One disk revolution takes: 1 / (7200 RPM) = 8.333 ms

Average rotation time T_r= half of 1 round time

Then T_f = 8.333ms / 2 = 4.17ms

Exercise 2

- Assume that a disk has the following characteristics:
  - The capacity is 36.7GB
  - the transfer rate is 45MB/s
  - the rotation time is 4ms
  - the average seek time is 5ms,
  - the minimum seek time is 0.65ms (refers to the time it takes the head moves to next track)
  - the track size is 180KB
  - one disk block is 4KB.

Exercise 2

How long does it take to randomly read 1000 disk blocks?

- Random reads: reading each block needs to seek, rotate and transfer
- Therefore, the time to read a block at random: t = average seek time t_1 + average rotation time t_2 + transfer time of a block with 4KB t_3
- t_1 = 5 ms, t_2 = 4 ms / 2 = 2 ms
- The transfer time of a block = the ratio of a block to a track * the time it takes to make one revolution
- So t_3 = (4KB / 180KB) * 4ms = 0.09ms
- t= 4 ms + 2 ms + 0.09 ms
- Randomly read 1000 blocks time = 1000 * t = 7090 ms

Exercise 2

Assuming the 1000 blocks are stored contiguously on a single track and all the blocks are stored on adjacent tracks, how long does it take to read the 1000 blocks?

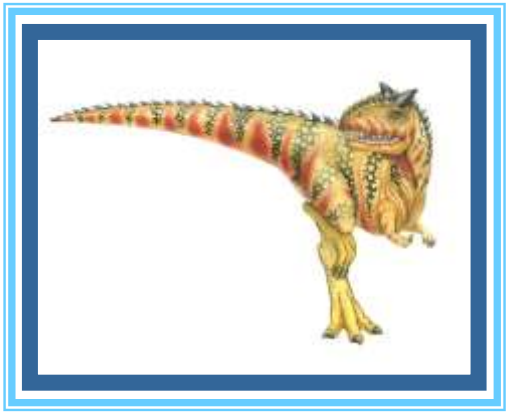
- The number of tracks required for 1000 4KB blocks = (1000 * 4KB) / (180 KB/ track) = 23 tracks, so the read will cross adjacent tracks 22 times.

- Since blocks are stored contiguously, as long as the first track seek t_1 and
- Rotational delay (for sector) t_2
- plus the transfer time t_3 and
- the time across adjacent tracks t_4

- t_1 = 5ms; t_2 = 2ms, t_3 = 0.09ms; t_4 = 0.65ms

- Therefore, the time to read 1000 consecutive blocks = $t_1 + t_2 + 1000* t_3 + 22* t_4$ = 111.3ms

End of Chapter 10



Silberschatz, Galvin and Gagne ©2013