

# CS161: Operating Systems

Matt Welsh  
mdw@eecs.harvard.edu



Lecture 13: Disks and Disk I/O Scheduling  
April 3, 2007

# Today: Disks

Physical operation of modern disk drives

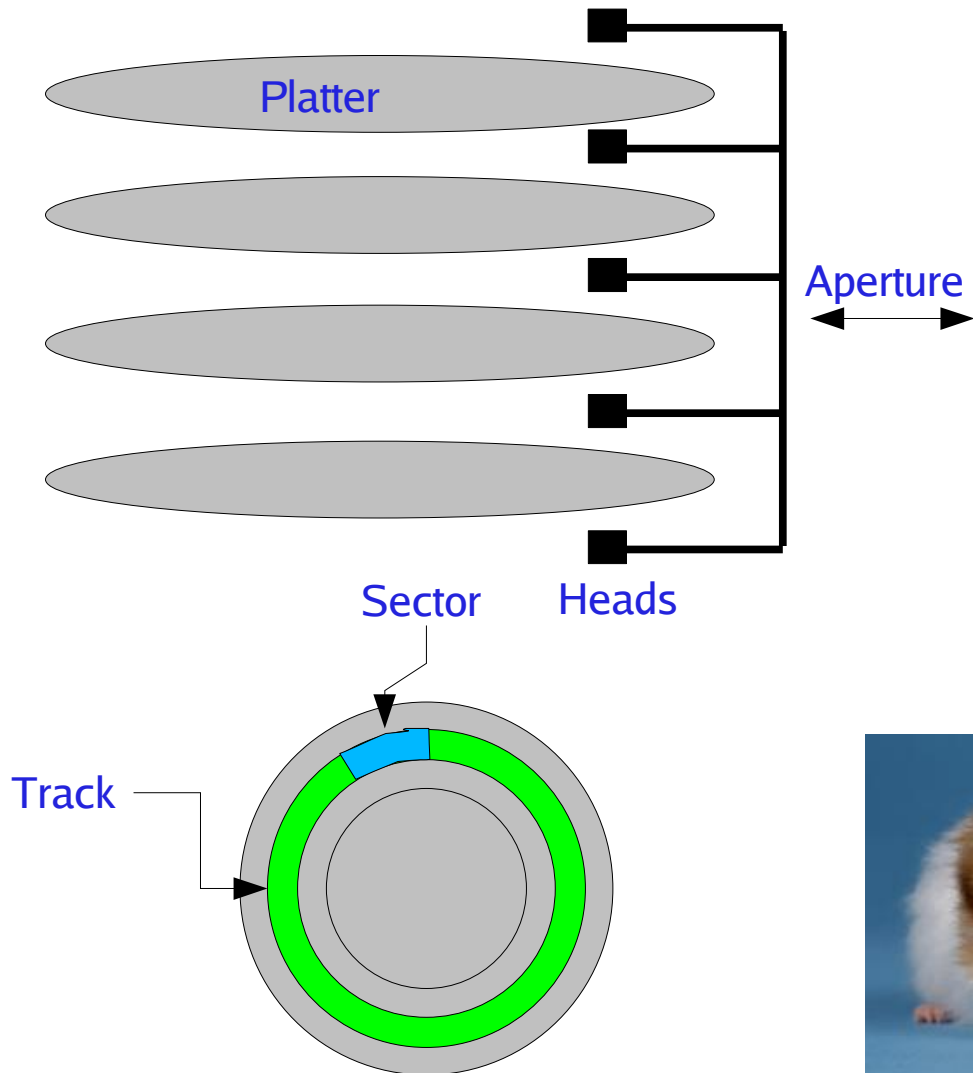
Operating system access to raw disk

Scheduling disk I/O

# A Disk Primer

Disks consist of one or more *platters* divided into *tracks*

- Each platter may have one or two *heads* that perform read/write operations
- Each track consists of multiple *sectors*
- The set of sectors across all platters is a *cylinder*



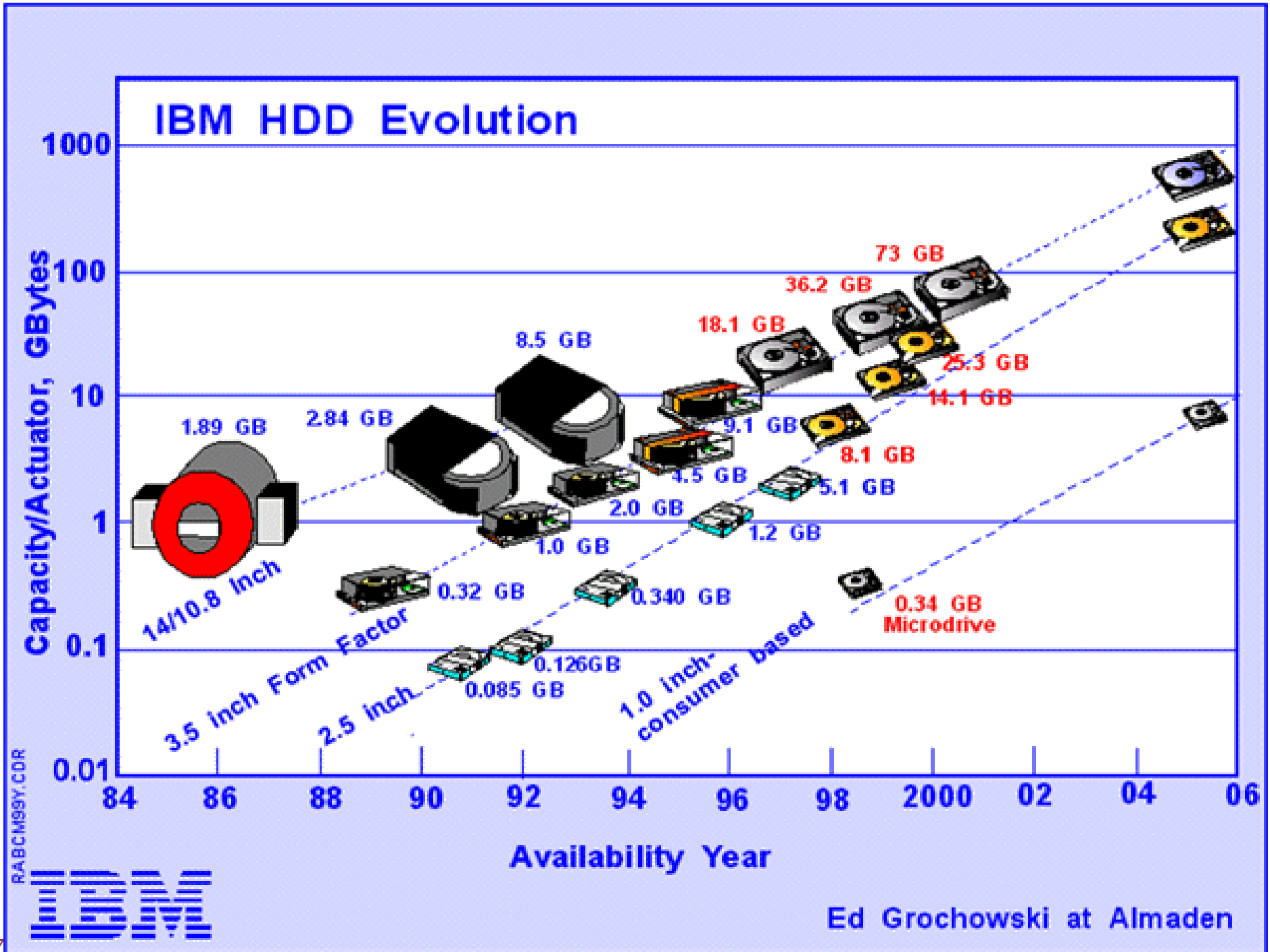
# Hard Disk Evolution

## IBM 305 RAMAC (1956)

- First commercially produced hard drive
- 5 Mbyte capacity, 50 platters each 24" in diameter!



# Hard Disk Evolution



# Disk access time

## Command overhead:

- Time to issue I/O, get the HDD to start responding, select appropriate head

## Seek time:

- Time to move disk arm to the appropriate track
- Depends on how fast you can physically move the disk arm
  - *These times are not improving rapidly!*

## Settle time:

- Time for head position to stabilize on the selected track

## Rotational latency:

- Time for the appropriate sector to move under the disk arm
- Depends on the rotation speed of the disk (e.g., 7200 RPM)

## Transfer time

- Time to transfer a sector to/from the disk controller
- Depends on density of bits on disk and RPM of disk rotation
- Faster for tracks near the outer edge of the disk – why?
  - *Modern drives have more sectors on the outer tracks!*

# Example disk characteristics

## IBM Ultrastar 36XP drive

- form factor: 3.5"
- capacity: 36.4 GB
- rotation rate: 7,200 RPM (120 RPS, musical note C3)
- platters: 10
- surfaces: 20
- sector size: 512-732 bytes
- cylinders: 11,494
- cache: 4MB
- transfer rate: 17.9 MB/s (inner) – 28.9 MB/s (outer)
- full seek: 14.5 ms
- head switch: 0.3 ms

## Disk interface speeds

- SCSI: From 5 MB/sec to 320 MB/sec
- ATA: from 33 MB/sec to 100 MB/sec
- Serial ATA (single wire): Starting at 150 MB/sec
- Firewire: 50 MB/sec

# Disks are messy and slow

## Low-level interface for reading and writing sectors

- Generally allow OS to read/write an entire sector at a time
- No notion of “files” or “directories” -- just raw sectors
- So, what do you do if you need to write a single byte to a file?
- Disk may have numerous bad blocks – OS may need to mask this from filesystem

## Access times are still very slow

- Disk seek times are around 10 ms
  - *Although raw throughput has increased dramatically*
- Compare to several nanosec to access main memory
- Requires careful *scheduling* of I/O requests

## [Tom's Hardware Disk Comparisons](#)

# ATA and IDE Interfaces

## IDE stands for Integrated (or “Intelligent”) Drive Electronics

- Same as “ATA” (Advanced Technology Attachment)
- Standard interface to hard drives that integrate a drive controller in the drive itself
- 1 or 2 drives on a chain

## Enhanced IDE (EIDE) and ATA-2

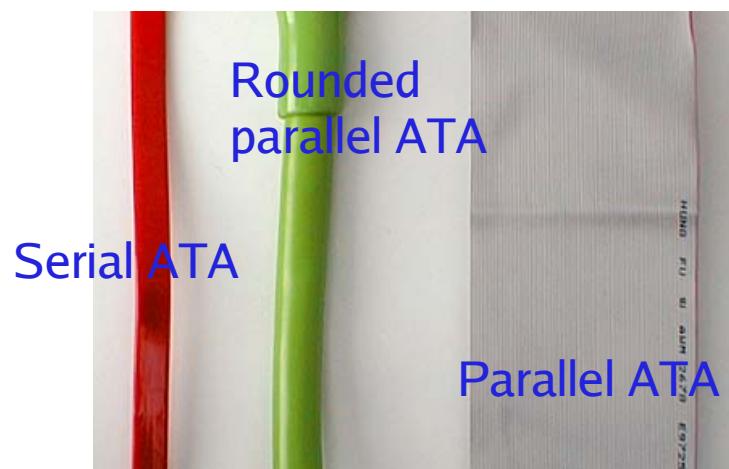
- Faster version of ATA/IDE that supports Direct Memory Access (DMA) transfers

## Ultra ATA: Speed enhancements to ATA standard

- Versions running at 33, 66, and 100 Mbytes/sec

## Serial ATA: Emerging standard using a serial (not parallel) interface

- Speeds starting at 150 Mbyte/sec
- Can drive longer cables at much higher clock speeds than parallel cable



# SCSI Interface

Standard hardware interface to wide range of I/O devices

- Disks, CDs, DVDs, tapes, etc.
- Bus-based design: single shared set of I/O lines that all devices connect to

Access model using logical blocks on disk

- On-disk controller maps logical block # to sector/track/head combination

SCSI-1: 8-bit bus, 5 Mhz = 5 Mbytes/sec max speed

- Supported up to 8 devices on a single bus
- Lots of problems with termination: required physical connector on end of cable to avoid signal refraction!

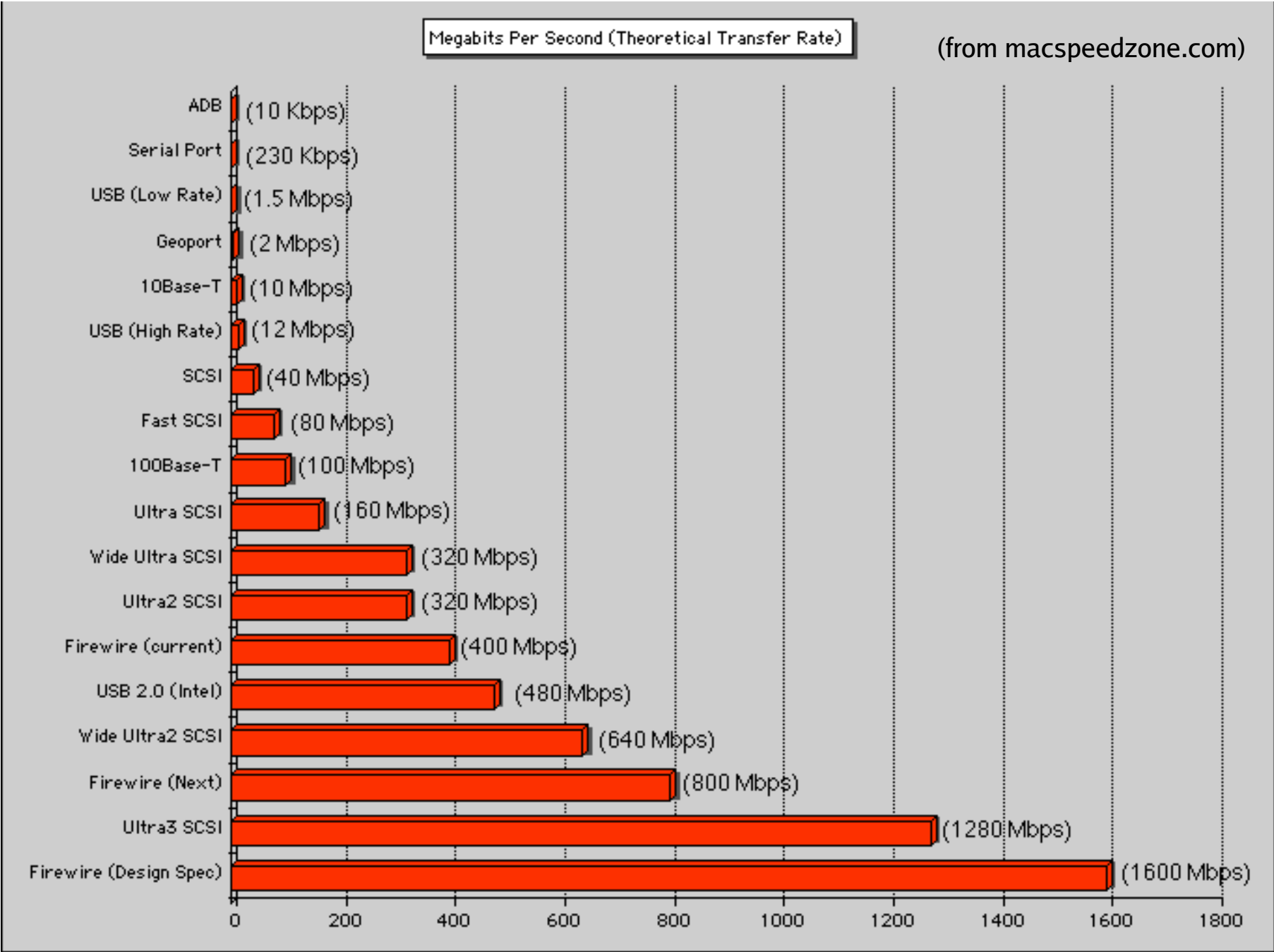
SCSI-2: The next generation

- Fast SCSI: 10 Mhz clock speed
- Wide SCSI: 16 bit bus width
- Fast wide SCSI: 10 Mhz + 16 bit bus = 20 MB/sec throughput

SCSI-3: Ramping up on speed and bus width

- Highest speed now is “Ultra320 SCSI”: 160 Mhz x 16 bits = 320 MB/sec max speed

# Relative Interconnect Speeds



# Disk I/O Scheduling

Given multiple outstanding I/O requests, what order to issue them?

Why does it matter?

Major goals of disk scheduling:

# Disk I/O Scheduling

Given multiple outstanding I/O requests, what order to issue them?

Why does it matter?

Major goals of disk scheduling:

1) Minimize **latency** for small transfers

- Primarily: Avoid long seeks by ordering accesses according to disk head locality

2) Maximize **throughput** for large transfers

- Large databases and scientific workloads often involve enormous files and datasets

Note that disk block layout also has a large impact on performance

- Where we place file blocks, directories, file system metadata, etc.
- This will be covered in future lectures

# Disk I/O Scheduling

Given multiple outstanding I/O requests, what order to issue them?

FIFO: Just schedule each I/O in the order it arrives

- What's wrong with this?

# Disk I/O Scheduling

Given multiple outstanding I/O requests, what order to issue them?

FIFO: Just schedule each I/O in the order it arrives

- What's wrong with this? *Potentially lots of seek time!*

SSTF: Shortest seek time first

- Issue I/O with the nearest cylinder to the current one
- Why might this not work so well???

# Disk I/O Scheduling

Given multiple outstanding I/O requests, what order to issue them?

FIFO: Just schedule each I/O in the order it arrives

- What's wrong with this? *Potentially lots of seek time!*

SSTF: Shortest seek time first

- Issue I/O with the nearest cylinder to the current one
- *Favors middle tracks: Head rarely moves to edges of disk*

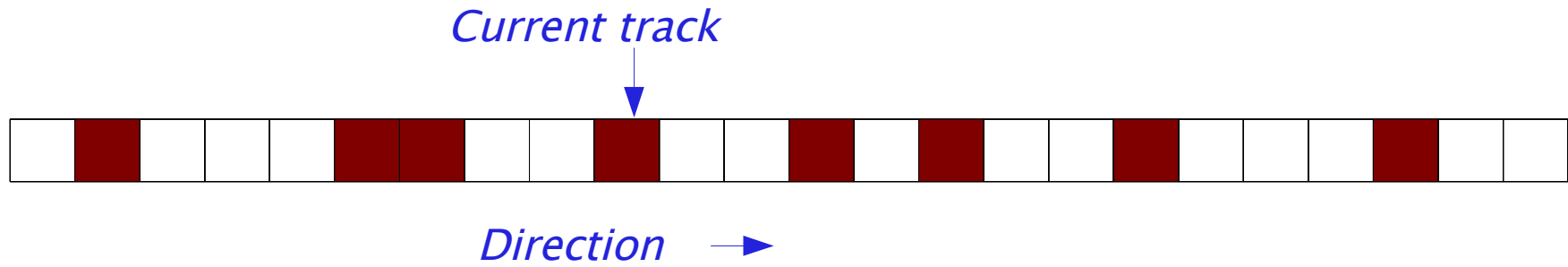
SCAN (or Elevator) Algorithm:

- Head has a current direction and current cylinder
- Sort I/Os according to the track # in the current direction of the head
- If no more I/Os in the current direction, reverse direction

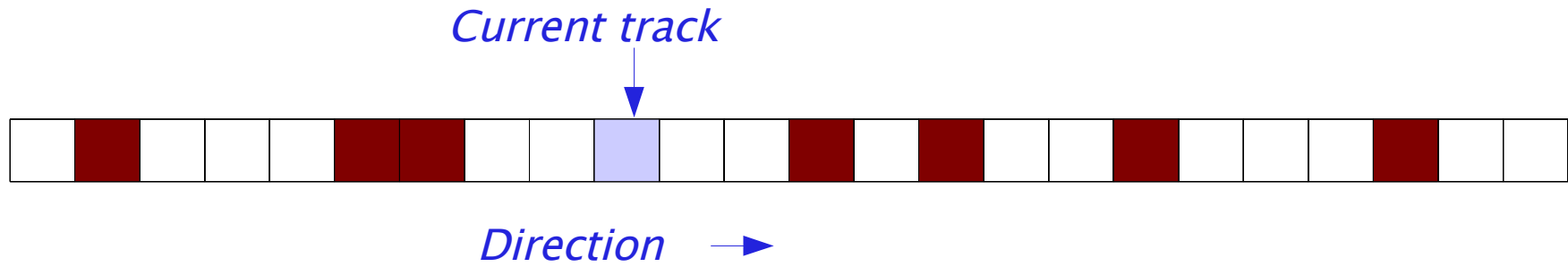
CSCAN Algorithm:

- Always move in one direction, “wrap around” to beginning of disk when moving off the end
- Idea: Reduce variance in seek times, avoid discriminating against the highest and lowest tracks

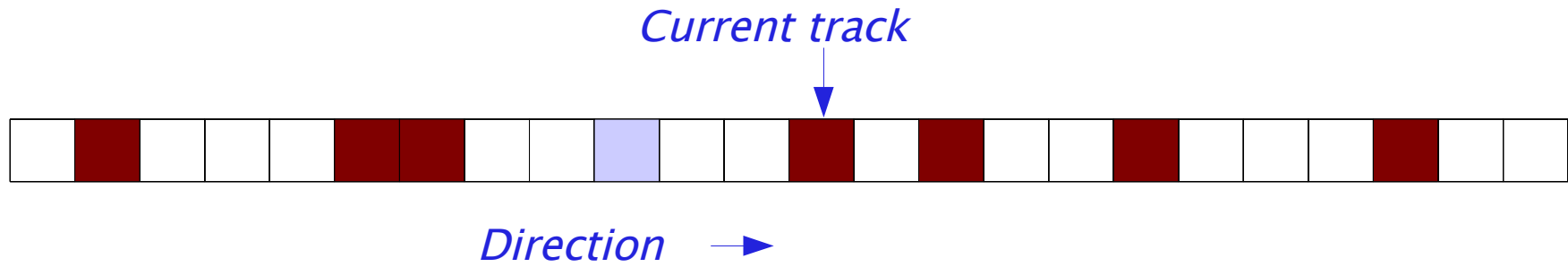
# SCAN example



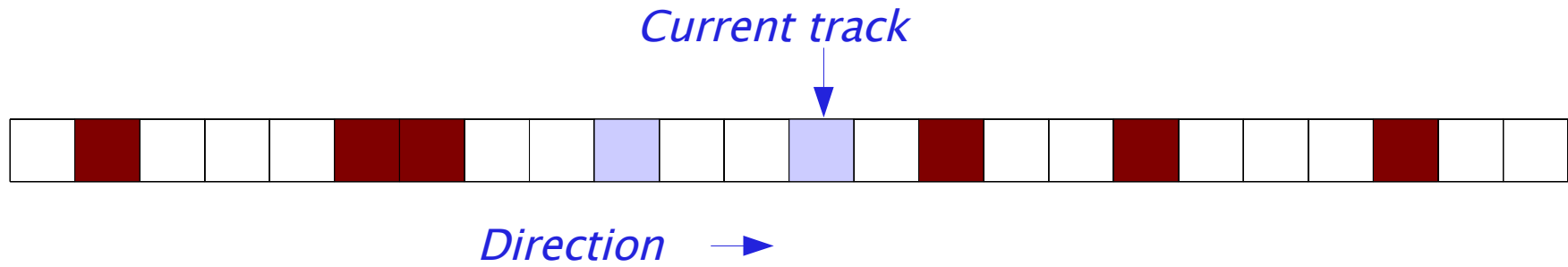
# SCAN example



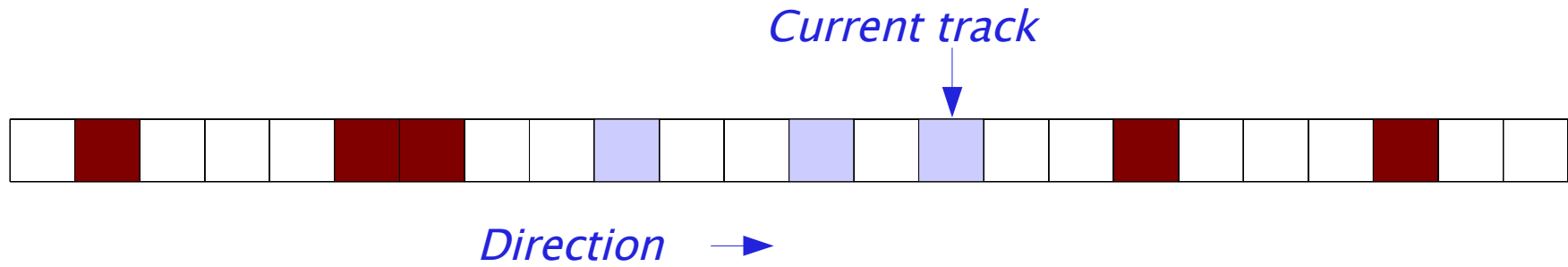
# SCAN example



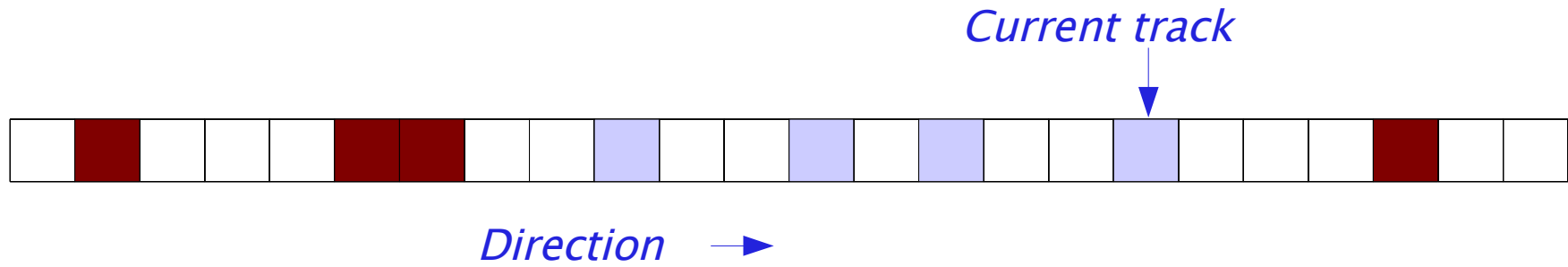
# SCAN example



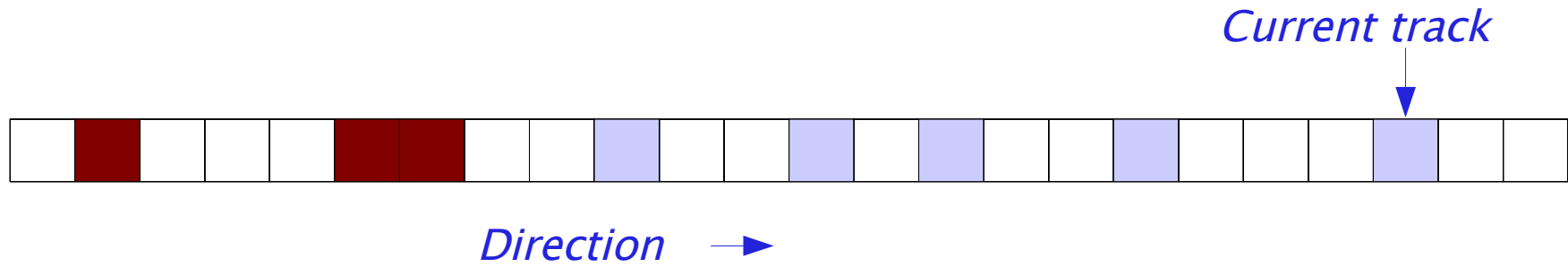
# SCAN example



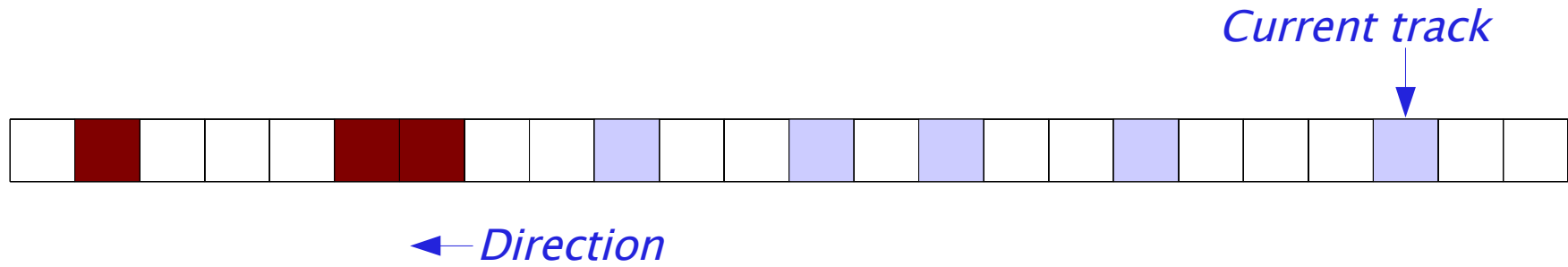
# SCAN example



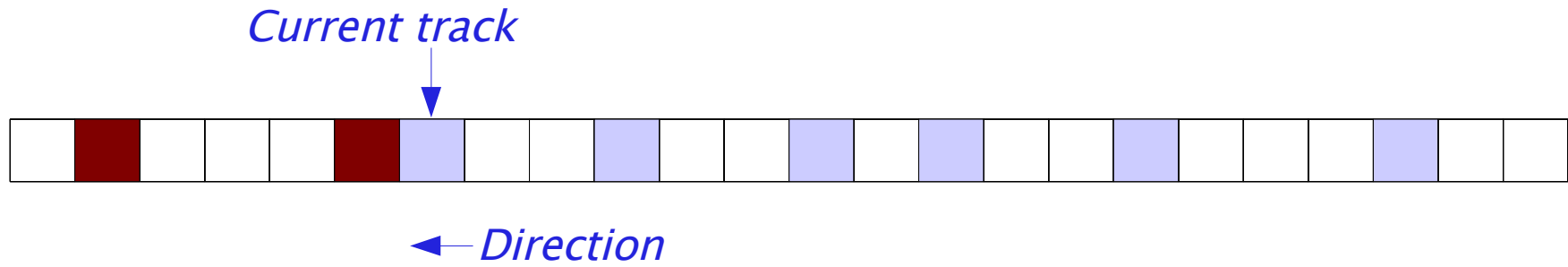
# SCAN example



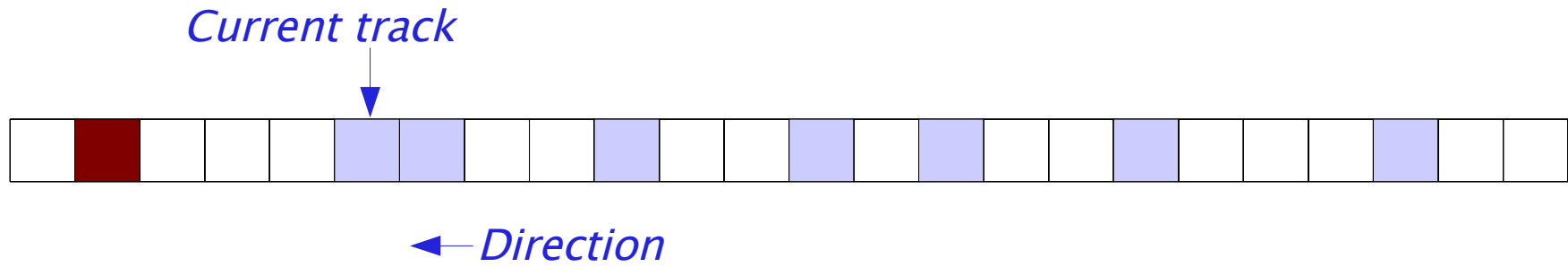
# SCAN example



# SCAN example

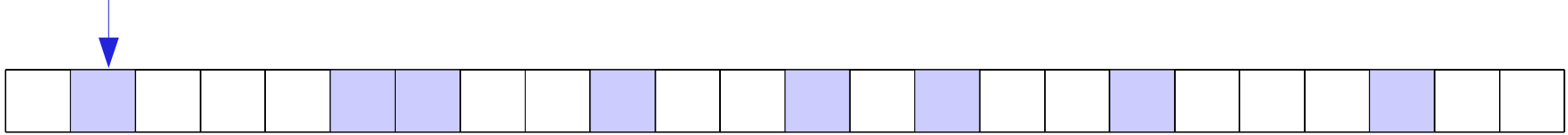


# SCAN example



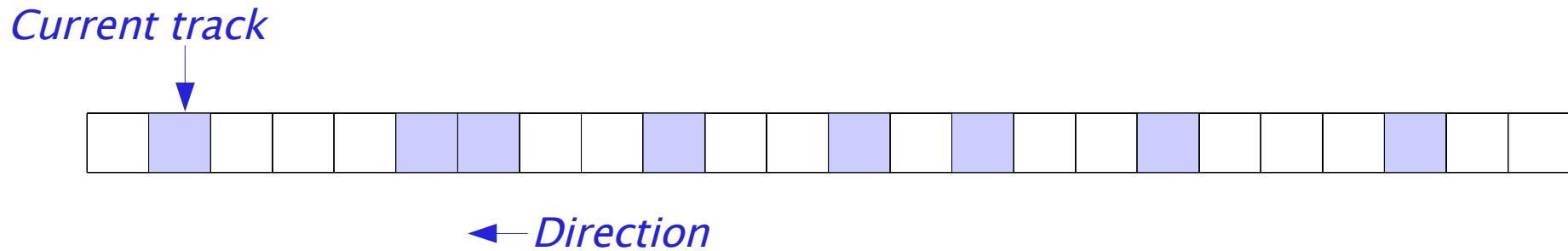
# SCAN example

*Current track*



*← Direction*

# SCAN example



What is the overhead of the SCAN algorithm?

- Count the *total amount of seek time* to service all I/O requests
- In this case, 12 tracks in --> direction
- 15 tracks for long seek back
- 5 tracks in <-- direction
  - *Total: 12+15+5 = 32 tracks*

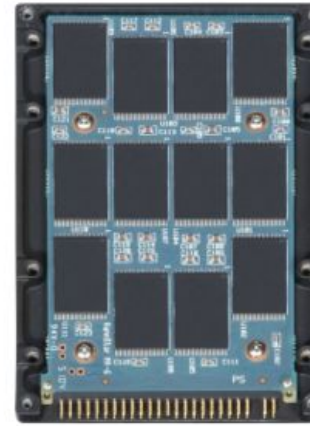
# What about flash?

## Non-volatile, solid state storage

- No moving parts!
- Fast access times (about 0.1 msec)

## Can read and write individual bytes at a time

- However, must erase a whole “block” before writing to it



## Limited number of erase/write cycles

- Most flash on the market today can withstand up to 1 million erase/write cycles

## How does this affect how we design filesystems???

# Next Lecture

## Filesystem introduction

- How do we go from low-level disk blocks to files and directories?

## Berkeley FFS Filesystem

- One of the most successful filesystem designs

Read Tanenbaum 6.3-6.6