UNIVERSITY OF OREGON

CIS 415: Operating Systems

Storage

Spring 2014
Prof. Kevin Butler

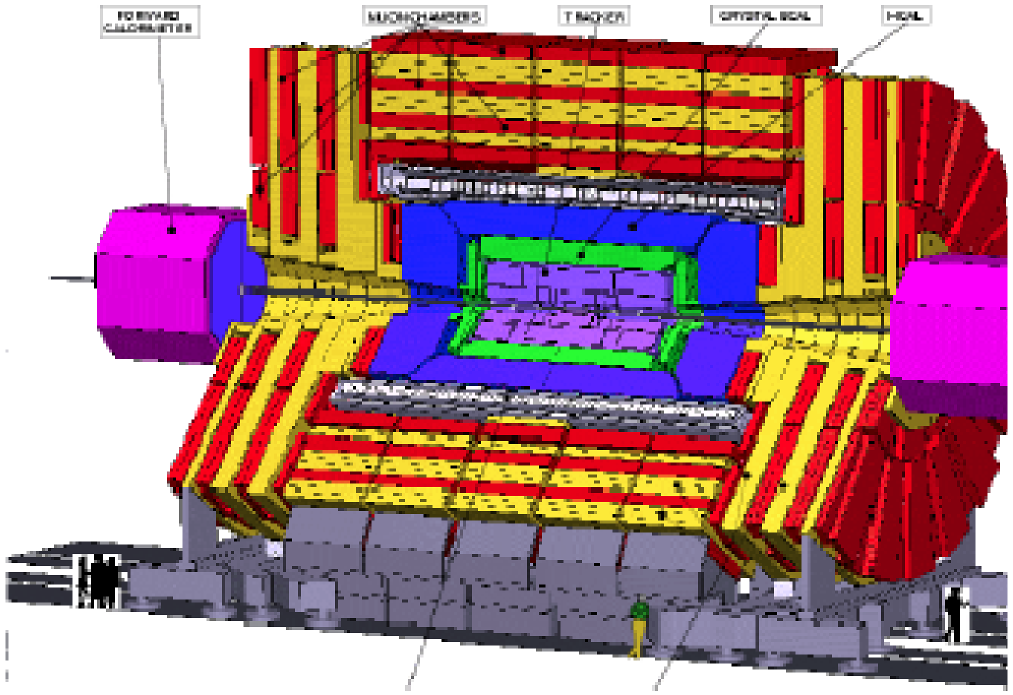
Outline

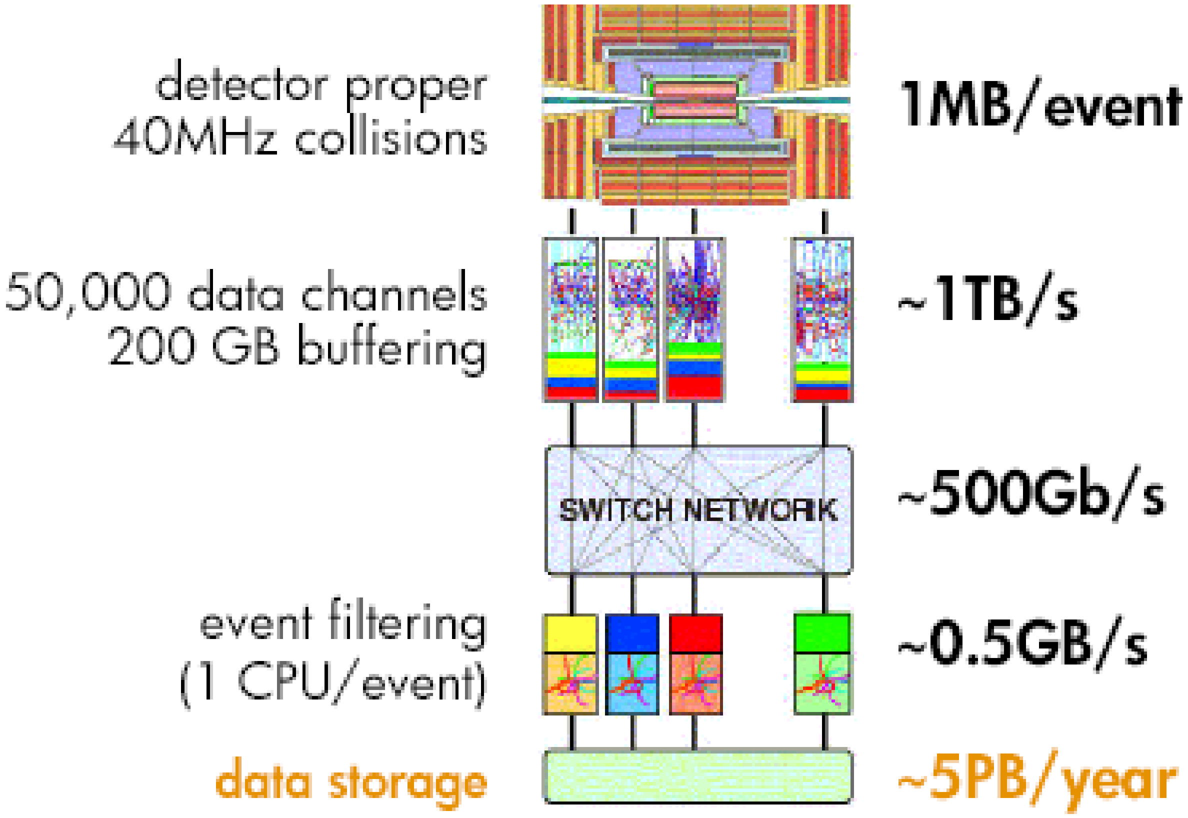
- Disk structure: physical and logical
- Disk addressing
- Disk scheduling
- Management

Need for Storage

- Memory is:
  - volatile: persistence is required
  - insufficient: large capacity is required
  - not portable: how can we take information with us?
- Long-lasting backup data is needed:
  - scientific applications
  - industry and finance

Mass Storage Application





detector proper
40MHz collisions

1MB/event

50,000 data channels
200 GB buffering

~1TB/s

~500Gb/s

event filtering
(1 CPU/event)

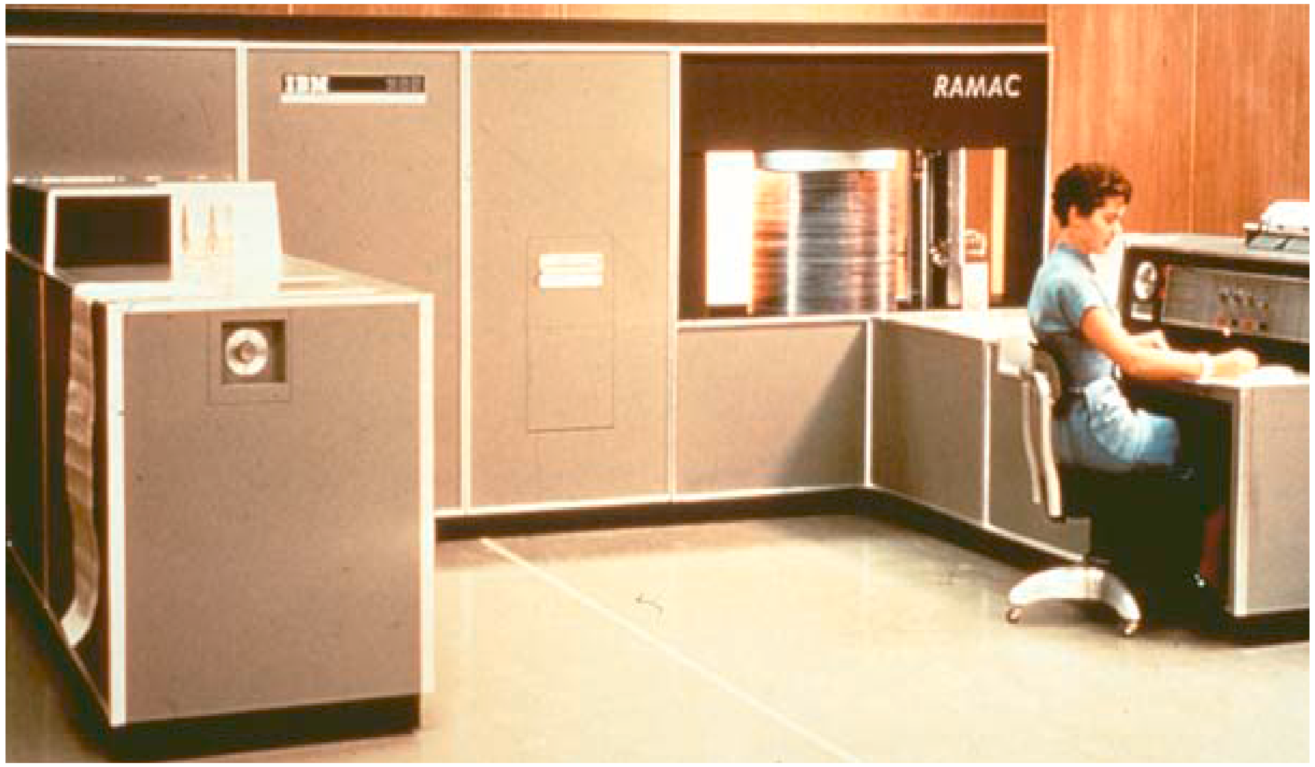
~0.5GB/s

data storage

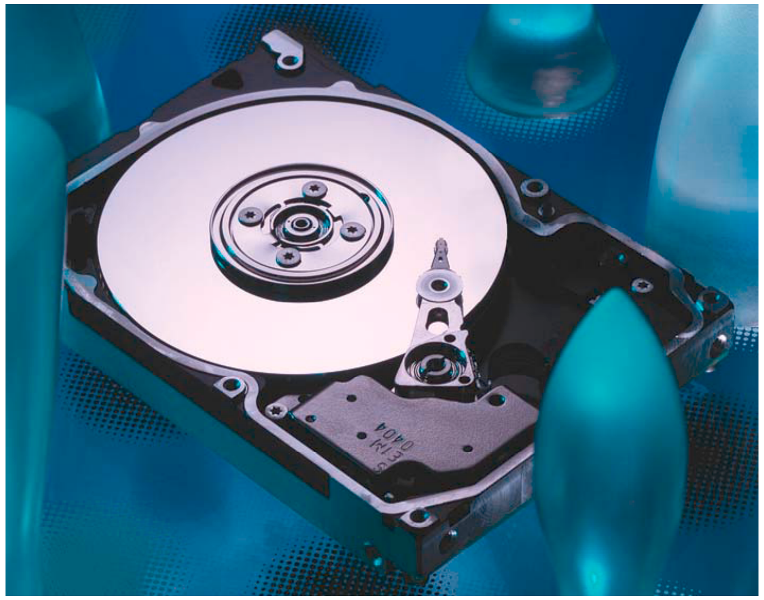
~5PB/year

CERN Particle Collider

Past & Present in

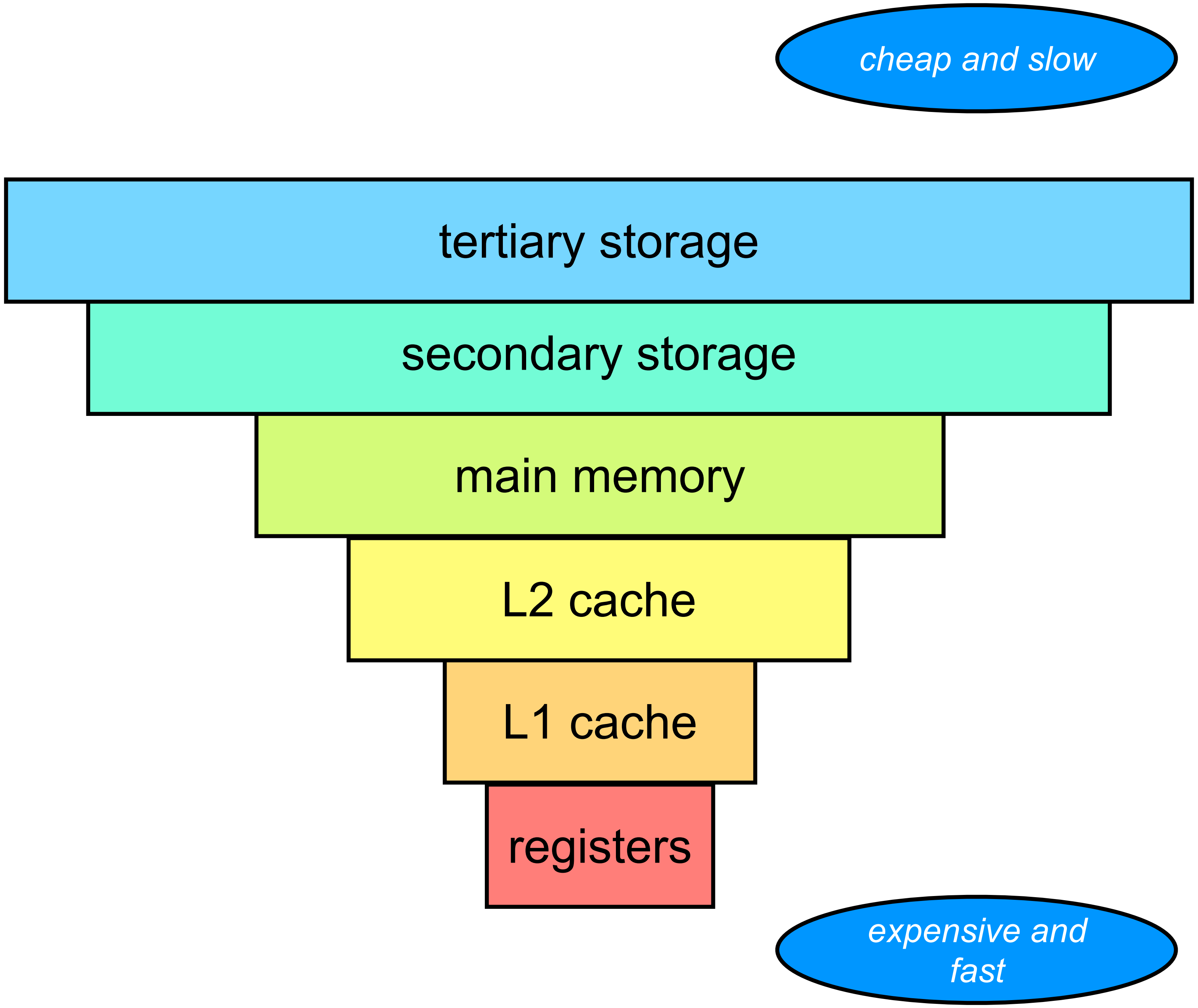


1956: IBM 305 RAMAC - 5 MB capacity (50 disks, each 24” in diameter)



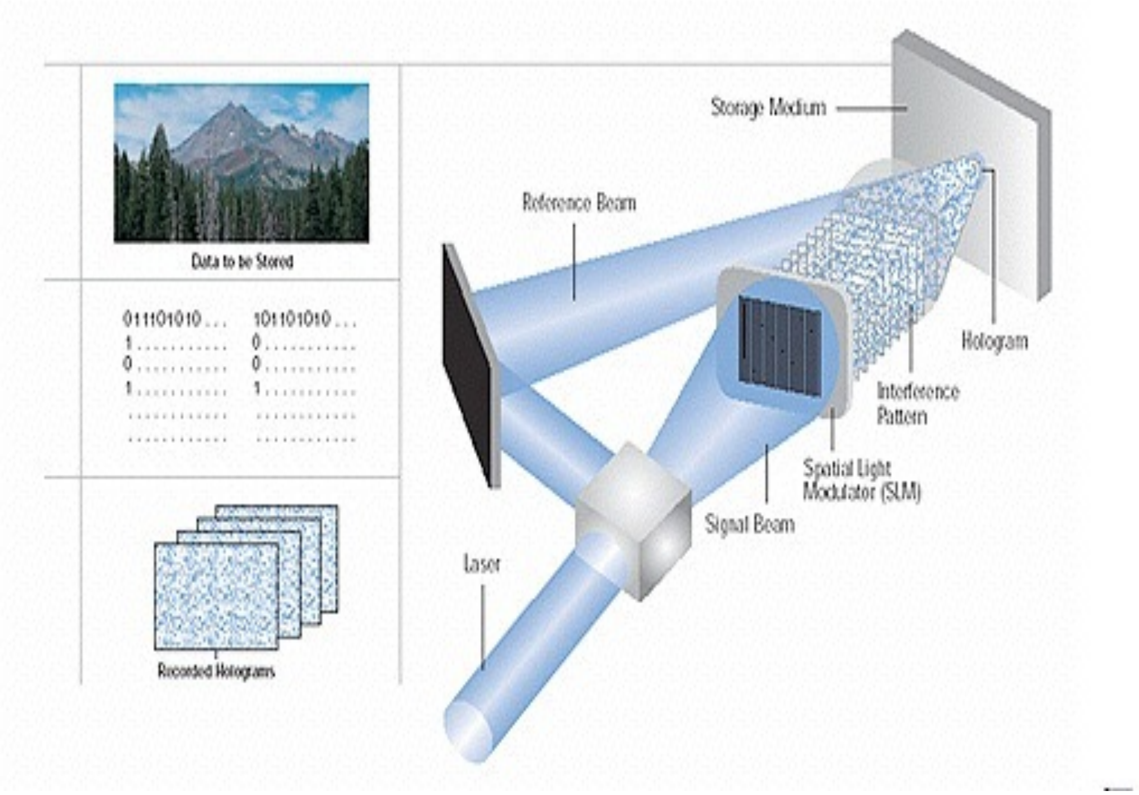
2008: Seagate Savvio 15K - 73.4 GB capacity, 2.5” diameter
 - can read/write complete works of Shakespeare 15 times per second

Storage Hierarchy



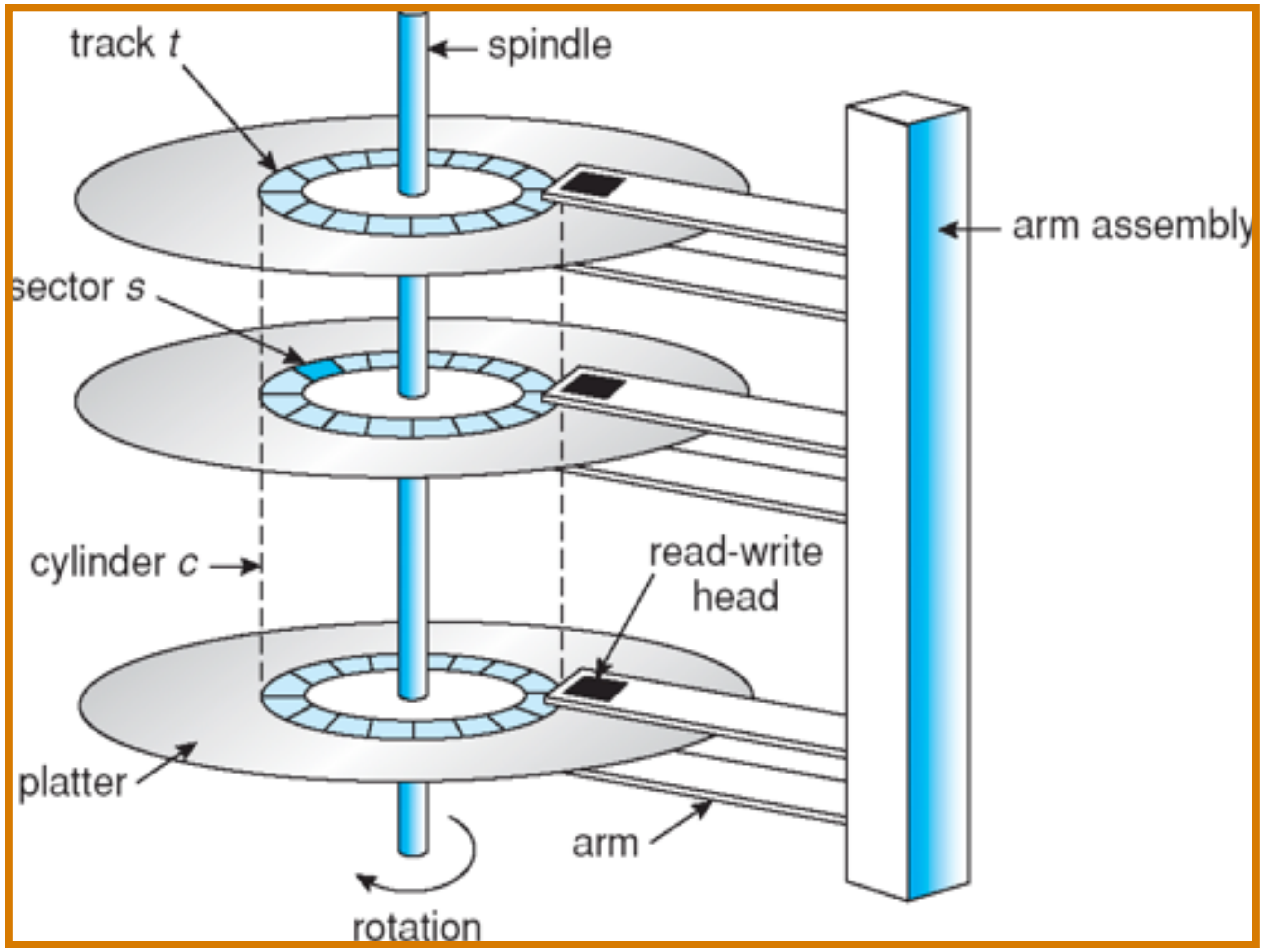
Secondary Storage

- Generally, magnetic disks provide the bulk of secondary storage in systems
  - future alternative: solid-state drives?
    - e.g. MacBook Air
  - MEMS and NEMS(nanotech)
  - holographic storage
    - data read from intersecting laser beams



www.inphase-technologies.com

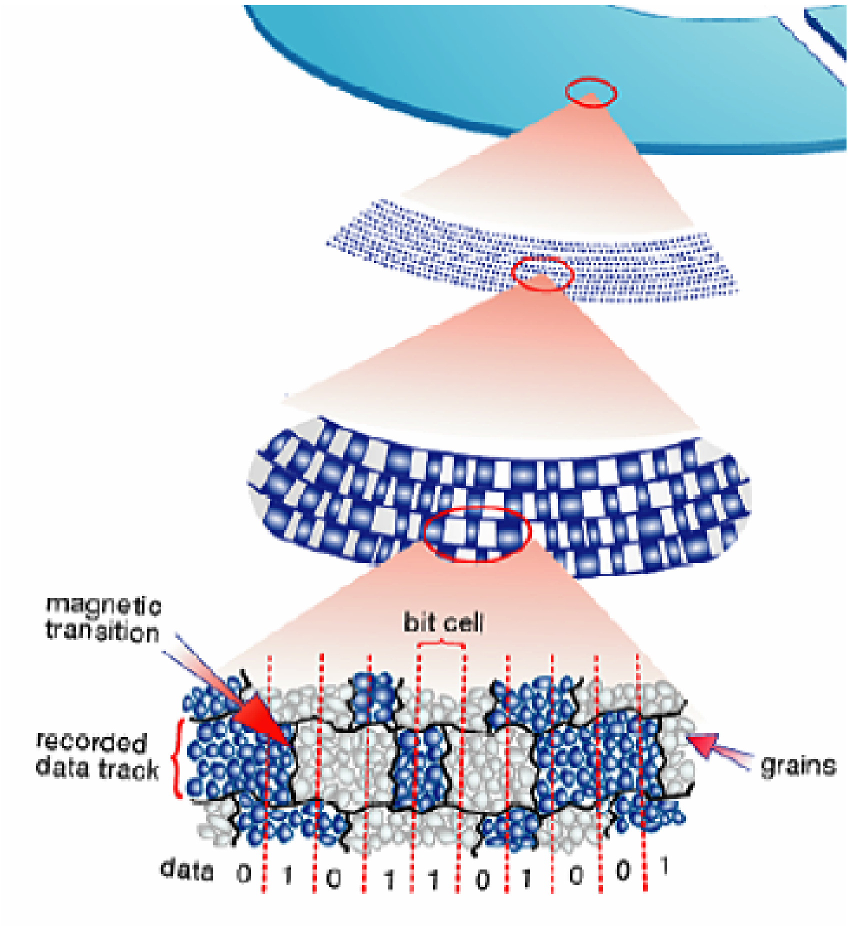
Inside a Hard Disk



Aluminum (sometimes glass) platters

Deep Inside a Hard Disk

- Bit-cell composed of about 50-100 magnetic grains
- 0 has uniform polarity, 1 has a boundary between magnetizations
- magnetized in direction of disk head (longitudinal) or perpendicular (more complex, but more density)
- in development: HAMR
    - heat-assisted (with lasers)
    - potentially 50 Tb/in²



Disk Operation

- Platters start moving from rest (*spinup time*)
  - lots of mass to start moving
- Heads find the right track (*seek time*)
  - arm powered by actuator motor, accelerates and coasts, slows down and settles on correct track (servo-guided)
- Disk rotates until correct sector found (*rotational latency*)
  - contingent on platter diameter and RPM (Savvio 15K rotates 300 times/second)
  - Have to stop the platters (*spindown time*)

Addressing Disks

- Old days: CHS (cylinder-head-sector)
  - supply physical characteristics of the disk to the operating system
  - it specifies exactly where on the physical disk to read and write data
- Nowadays: cylinders not uniform
  - can store more data on outer tracks than inner tracks (zoned bit recording)
    - why?
      - function of constant angular velocity (CAV) vs constant linear velocity (CLV) found in CD-ROM

Logical Block Addressing (LBA)

- OS sees drive as an array of blocks
  - first block LBA = 0, next block LBA = 1 etc.
- disk firmware takes care of managing the physical location of data
- Block: smallest unit of data accessible through the OS
  - can be the size of a sector (512 bytes) up to the size of a page (often 4 KB): defined by kernel

Disk Scheduling

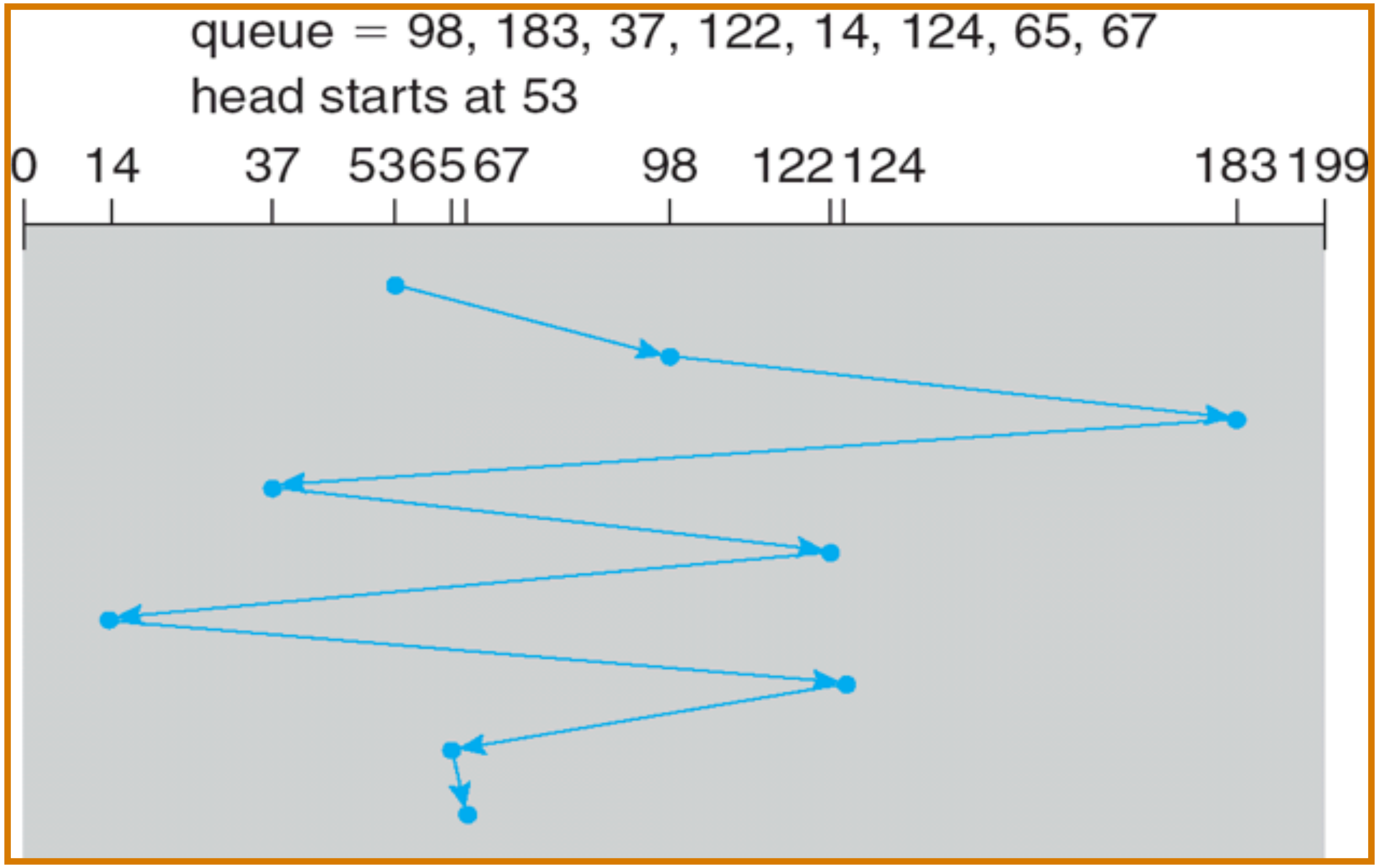
- Why does the OS need to schedule?
  - Improves access time (seek time & rotational latency)
  - even with LBA, assumption is that blocks are written in essentially contiguous order
  - maximizes bandwidth
    - transferred bytes / service + transfer time

Disk Scheduling Algorithms

- Consider the following request queue
  - min cylinder = 0, max cylinder = 199
    - requests at the following cylinders:
    - 98, 183, 37, 122, 14, 124, 65, 67
    - drive head is at cylinder 53

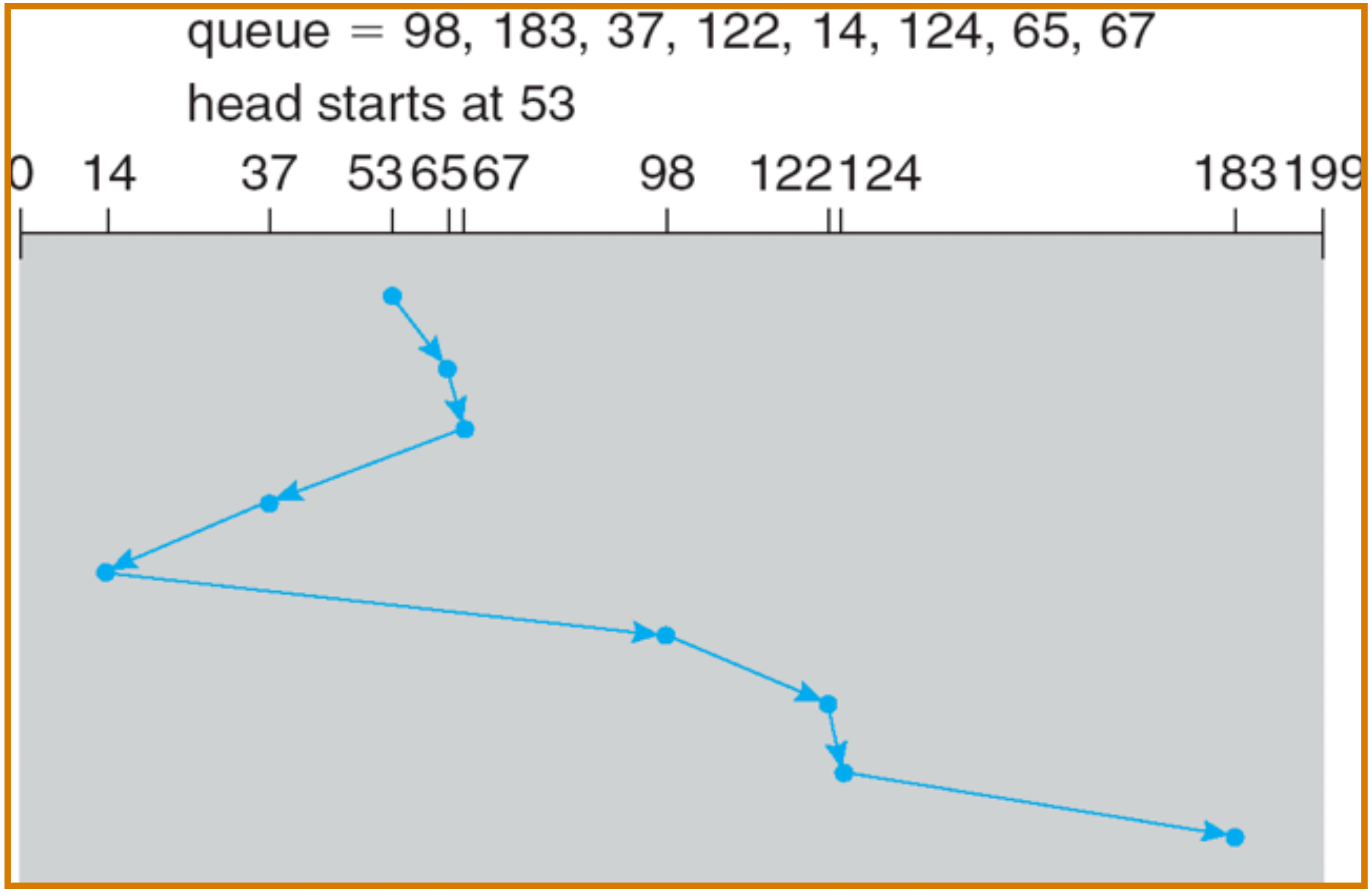
First-come First-served (FCFS)

- Service the requests in order of arrival
- Head movement of 640 cylinders



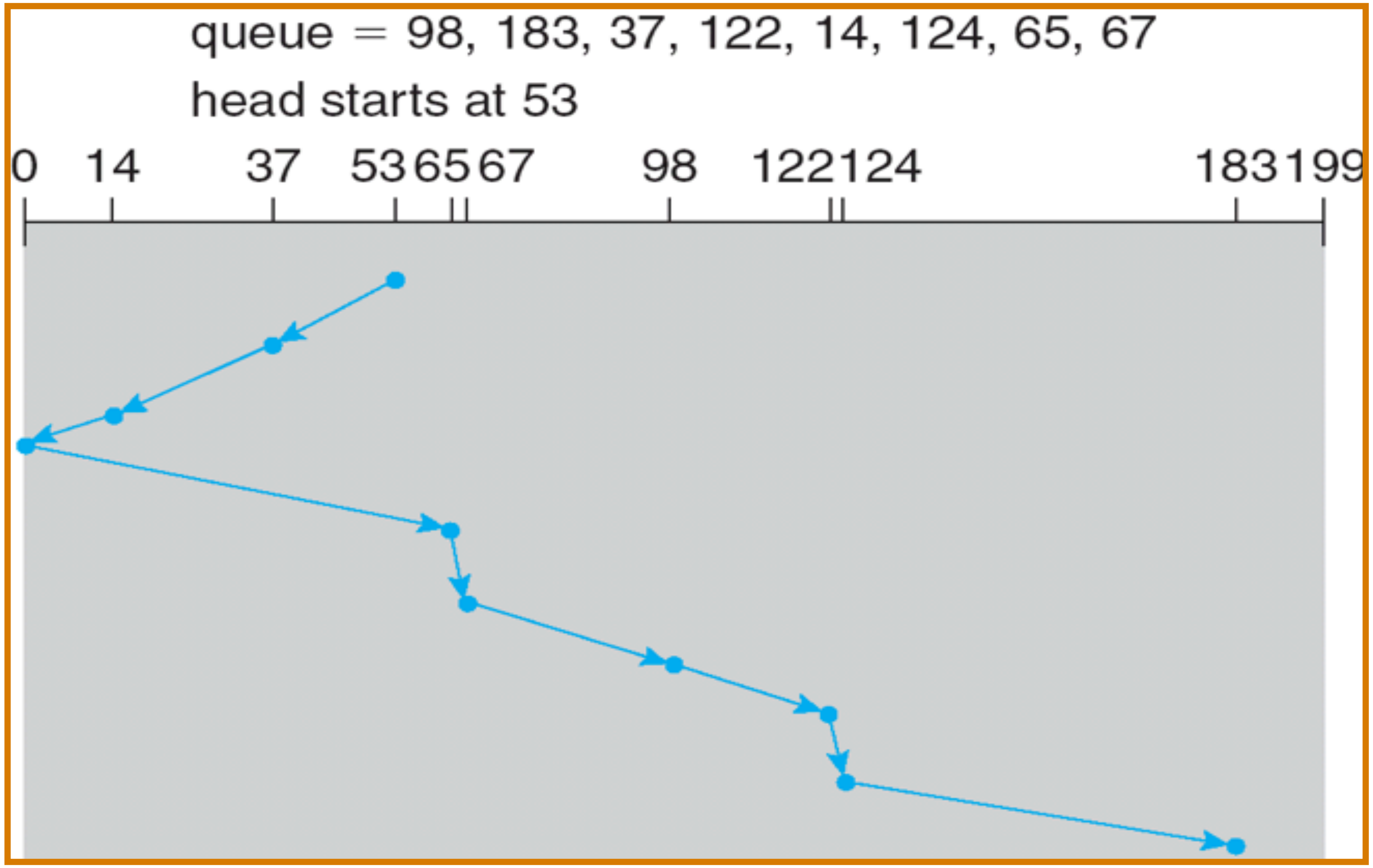
Shortest Seek Time First (SSTF)

- Min. seek time from head position (like SJF)
- Head movement of 236 cylinders



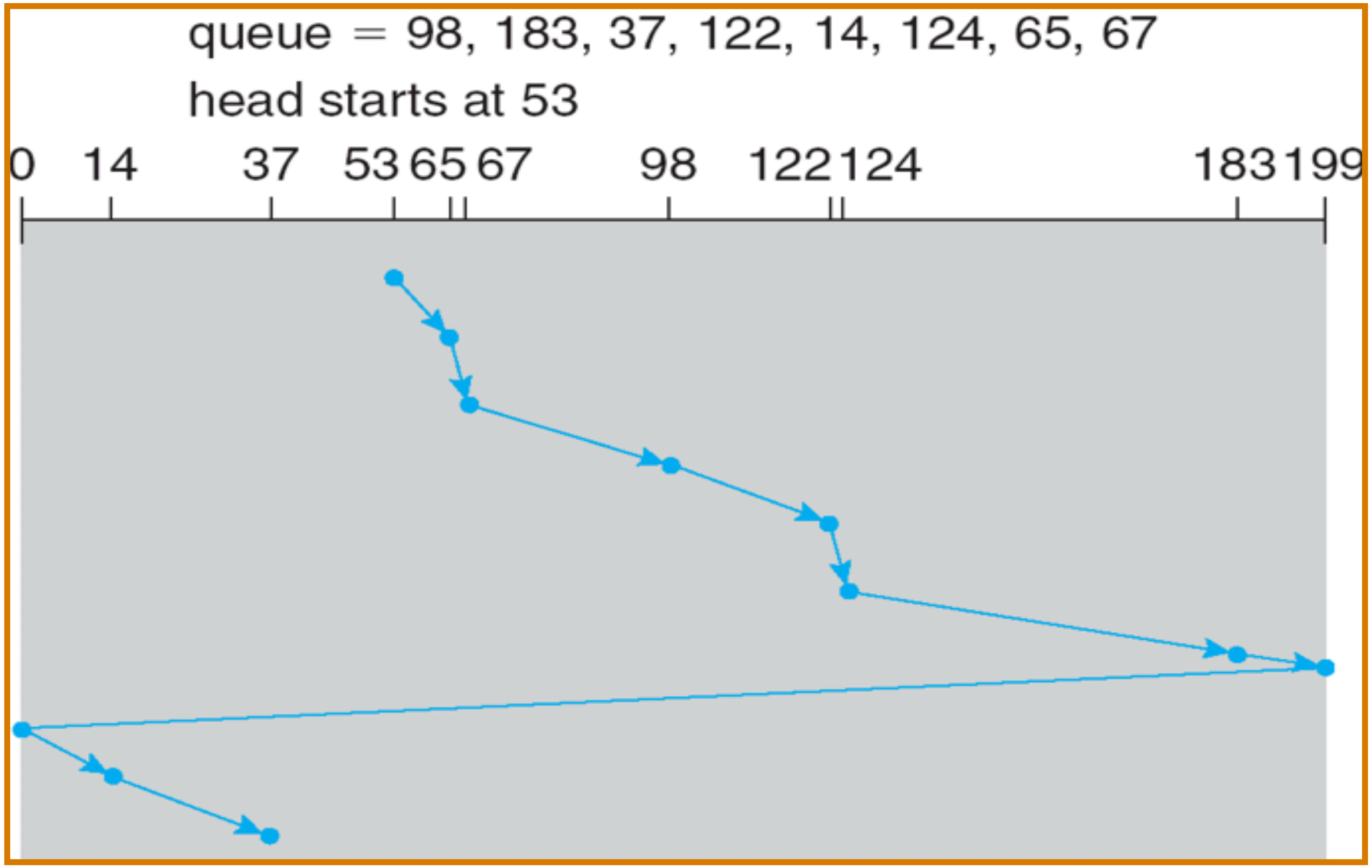
SCAN (Elevator) Algorithm

- Arm moves from one end of disk to the other then reverses (like an elevator)
- Head movement of 208 cylinders



C-SCAN Algorithm

- More uniform wait time than SCAN
- Head services requests in one direction then returns to beginning of disk (like circular list)

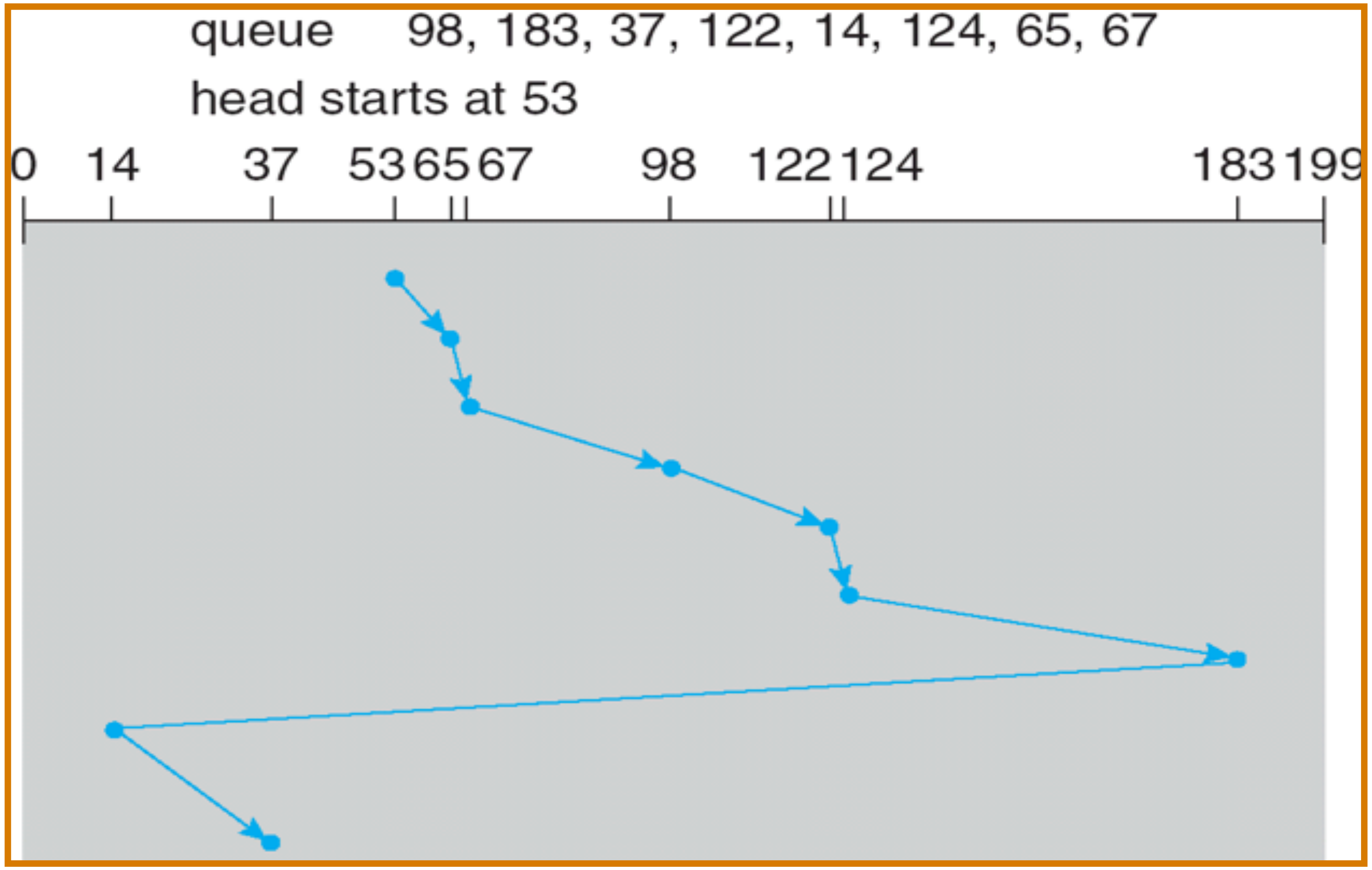


queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

C-LOOK Algorithm

- Like C-SCAN but only seeks to farthest request in queue
- Returns to lowest request (not start of disk)



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

head starts at 53

Choosing a Disk Scheduling Algo.

- SSTF: increased performance over FCFS
- SCAN, C-SCAN: good for heavy loads
  - less chance of starvation
- C-LOOK: good overall
- File allocation plays a role
  - contiguous allocation limits head movement
- Note: only considering seek time
  - rotational latency also important but hard for OS to know (doesn't have physical drive characteristics)
    - drive controllers implement some queueing and request coalescing

Drive Controller Scheduling?

- Why not have the drive controller in the disk perform all of the disk scheduling?
- Would be more efficient, but…
- OS knows about constraints that the disk doesn’t
  - demand paging > application I/O
  - write > read if cache is almost full
  - guaranteeing write ordering (e.g. journaling, data flushing)

Linux I/O Schedulers

- Linus Elevator (default in 2.4 kernel)
  - merges adjacent requests and sorts request queue
  - can lead to starvation in some cases though: big push to change for 2.6 kernel
- Deadline I/O Scheduler
  - merges & sorts request + expiration timer
  - multiple queues to minimize seeks while ensuring request don't starve
- Anticipatory I/O Scheduler
  - waits a few ms after a read request to see if another one is made (high probability); acts like deadline scheduler otherwise
  - loses time if wrong but big win if right

Linux Schedulers (ctd.)

- Complete Fair Queueing (CFQ) I/O Scheduler
  - different than the others: assigns queues based on originating process
  - queues are serviced round-robin, usually picking 4 requests from each queue at a time
  - good for multimedia (e.g., ensuring audio buffers are full)
- When to use which?
  - Linus Elevator: obsolete
  - Deadline: good for lots of seeks, critical workloads
  - Anticipatory: good for servers
  - CFQ: desktops

Disk Management

- Low-level formatting
- Logical formatting
- Booting
- Bad block recovery
- Swap space

Low-Level (Physical) Formatting

- divide disk into sectors for disk controller to read and write
  - sector numbers, error-correcting codes (ECC), other identifying information (e.g., servo control data) written to each sector
- usually only done at factory
  - can restore factory configuration (reinitialize)

High-Level (Logical) Formatting

- Before formatting, OS needs to partition the disk into 1 or more cylinder groups
  - why more than 1? root vs swap partitions, dual boot, etc.
- write a file system onto the disk
  - structures such as file allocation table (FAT - DOS) or inodes (UNIX)
- write the boot block (boot sector)

Boot Process

- Bootstrapping starts from a process in ROM
- Boot loader reads a bootstrap program from the bootblock
  - on PCs: Master boot record (MBR): first sector on disk (446 bytes, then 64 byte partition table)
- Second-stage boot loader: program whose location is pointed to from MBR
  - NTLDR on Windows, LILO/GRUB on Linux
    - choose the partition to boot from to start to OS

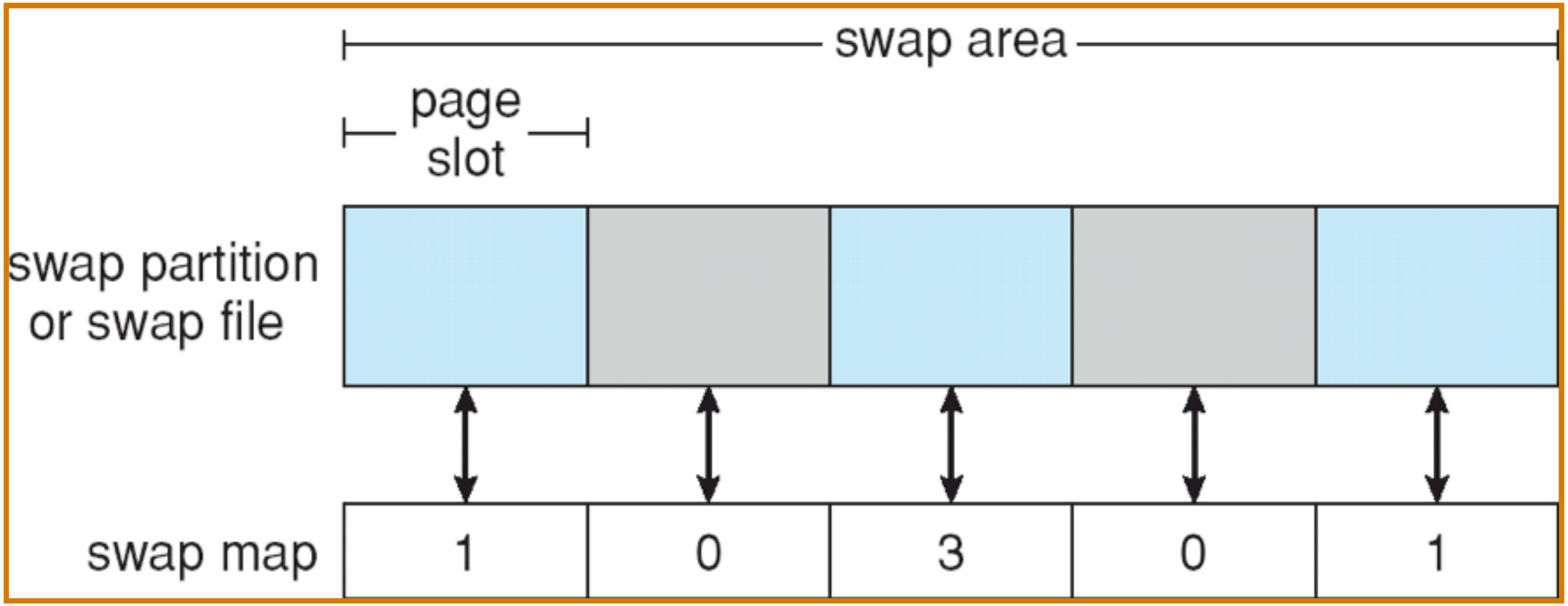
Bad Block Recovery

- Most disks have some bad blocks even from the factory
- ECC used (Reed-Solomon encoding on modern disks) to try and recover
- *Sector Sparing*: drive marks bad block and maps to a spare block the OS doesn't see
- *Sector Slipping:* drive remaps blocks in order on disk, skipping over bad one
  - Disk does lots of background tasks
    - Still, Avoid head crashes

Swap-Space Management

- Swap space: used for virtual memory (extension of main memory)
- Often given its own disk partition
  - Can hold process images or memory pages
- Linux and Solaris: page slots within swap files or partitions
  - only allocate swap page slot when page forced out of memory
  - swap map indicates how many processes using page

Linux Swap Structures



Attaching Disks to Networks

- NAS: network attached storage - RPCs between host and storage
  - e.g., NFS (what we use), iSCSI
- SAN: storage area network
  - multiple connected storage arrays, servers connect directly to SAN
- Becoming more like each other
  - e.g., Open Storage Networking proposal (from NetApp) combines elements of each

SCSI vs IDE/ATA

- Originally speed but with serial ATA (SATA) interface speeds have caught up
- SCSI supports more drives on a bus but SATA can be beneficial for small numbers
- Why pay more for SCSI? Disks manufactured differently
  - assumed to be server (enterprise) vs personal
    - often faster (e.g., 15K disks usually only SCSI)
    - SCSI drives better constructed (O-ring sealing, air flow, more rigidity); stronger actuator motors;more reliable
    - ATA cheap though: 1 TB SATA < 73 GB SCSI

Summary

- Storage is critical and getting more so
- physical characteristics: cylinders (tracks), heads, sectors
- seek, rotation time
- Scheduling algorithms affect system performance
- Storage management: boot process, swap space
- On your own: look over NAS and SAN figs
  - Recommended: RAID (0,1,5 most common)

- Next time: final review, wrapup