Chap. 12) Secondary-Storage Architecture

경희대학교 컴퓨터공학과

이 승 룡

Secondary Storage

- Secondary storage usually
  - ✓ is anything that is outside of "primary memory"
  - ✓ does not permit direct execution of instructions or data retrieval via machine load/store instructions

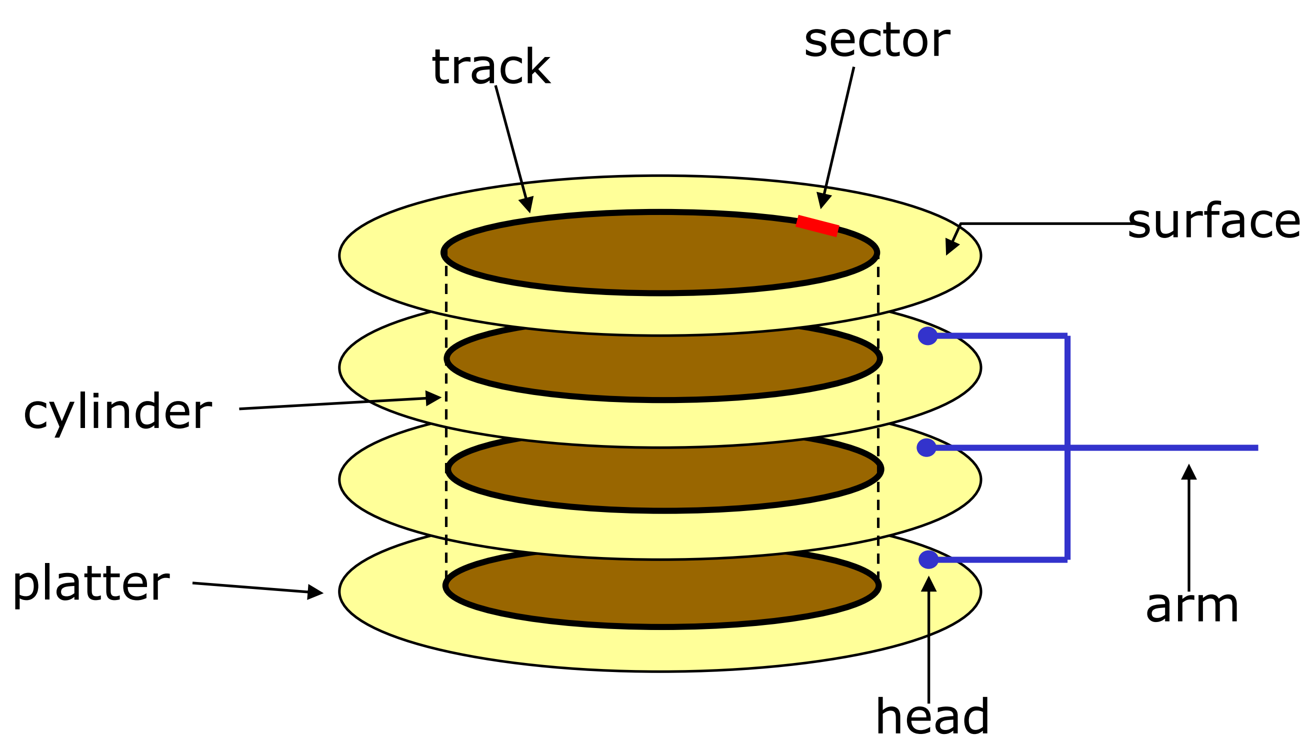
- Characteristics
  - ✓ It's large: 100GB and more
  - ✓ It's cheap: 120GB IDE disk costs ₩250,000
  - ✓ It's persistent: data survives power loss
  - ✓ It's slow: milliseconds to access

Disks

- Disks and the OS
  - ✓ Disks are messy physical devices:
    - Errors, bad blocks, missed seeks, etc.
  - ✓ The job of the OS is to hide this mess from higher-level software
    - Low-level device drivers (initiate a disk read, etc)
    - Higher-level abstractions (files, databases, etc.)
  - ✓ The OS may provide different levels of disk access to different clients
    - Physical disk block (surface, cylinder, sector)
    - Disk logical block (disk block #)
    - Logical file (filename, block or record or byte #)

Disks (Cont'd)

- Physical disk structure
  - ✓ platters
  - ✓ surfaces
  - ✓ tracks
  - ✓ sectors
  - ✓ cylinders
  - ✓ arm
  - ✓ heads



Disks (Cont'd)

- Interacting with disks
  - ✓ Specifying disk requests requires a lot of info:
    - Cylinder #, surface #, track #, sector #, transfer size, etc.
  - ✓ Older disks required the OS to specify all of this
    - The OS needs to know all disk parameters
  - ✓ Modern disks are more complicated
    - Not all sectors are the same size, sectors are remapped, etc.
  - ✓ Current disks provide a higher-level interface (e.g. SCSI)
    - The disks exports its data as a logical array of blocks [0..N]
    - Disk maps logical blocks to cylinder/surface/track/sector
    - Only need to specify the logical block # to read/write
    - As a result, physical parameters are hidden from OS

Disks (Cont'd)

- Disk performance
  - ✓ Performance depends on a number of steps
    - Seek: moving the disk arm to the correct cylinder

      → depends on how fast disk arm can move (increasing very slowly)
    - Rotation: waiting for the sector to rotate under head

      → depends on rotation rate of disk (increasing, but slowly)
    - Transfer: transferring data from surface into disk controller, sending it back to the host

      → depends on density of bytes on disk (increasing, and very quickly)
  - ✓ Disk scheduling:
    - Because seeks are so expensive, the OS tries to schedule disk requests that are queued waiting for the disk

Disk Attachment

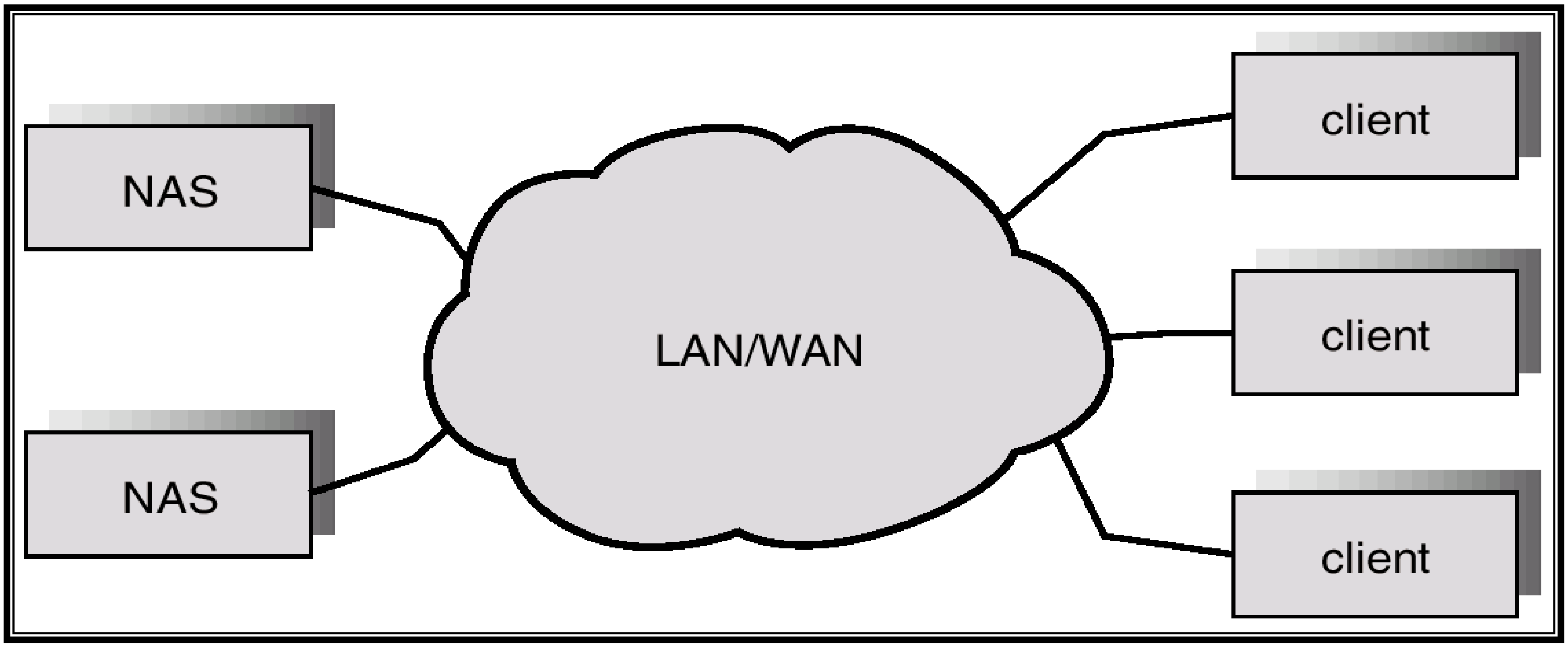
- Disks may be attached one of two ways:

  1. **Host attached** via an I/O port

  2. **Network attached** via a network connection

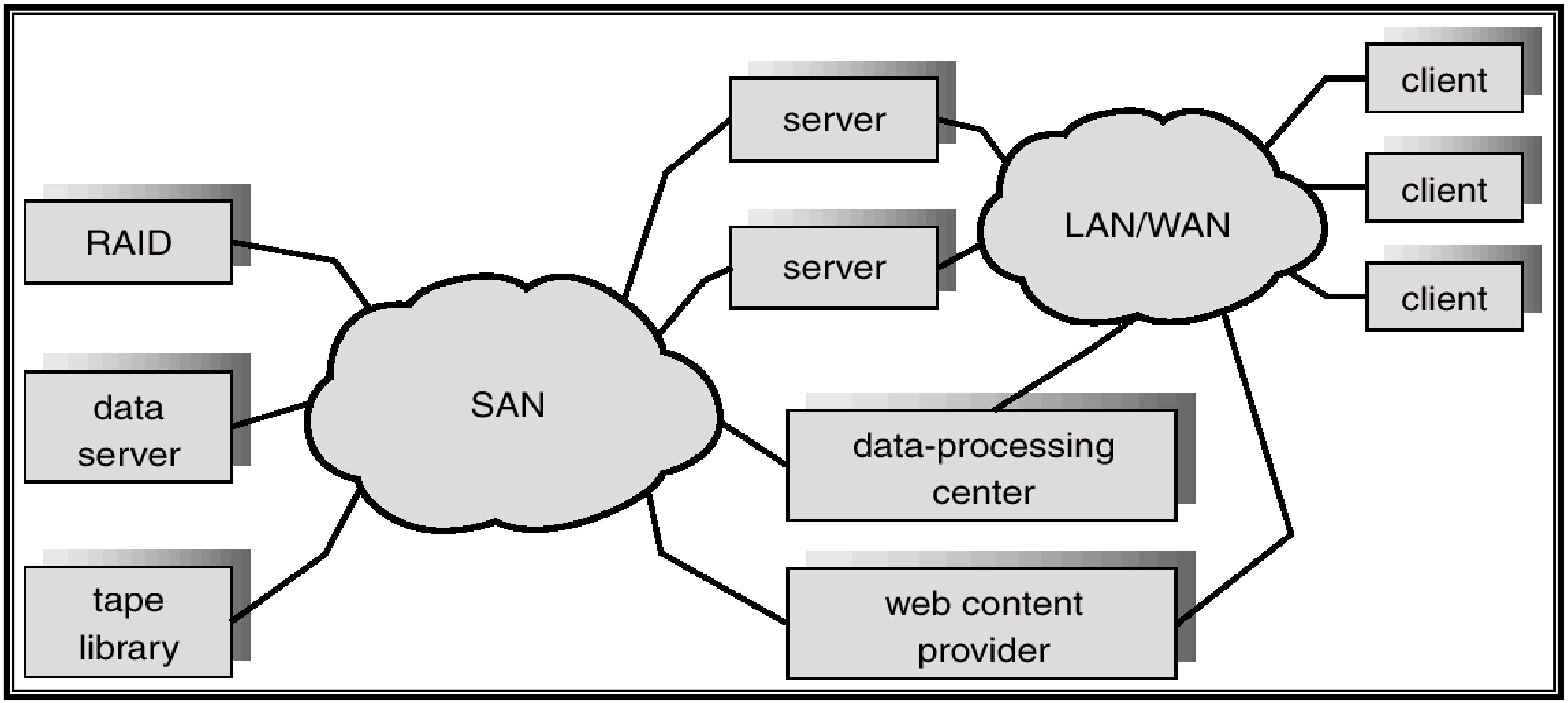
Network-Attached Storage

- Accessed via TCP/IP- or UDP/IP-based networks
- NFS, CIFS, etc.
- File-level access



Storage-Area Network

- Accessed via private network dedicated for storages
- Use storage protocols such as SCSI or Fibre channel
- Block-level access
- File systems for SAN is another story (e.g. GFS)



Storage Architecture

Protocols

Interconnection	Block	File
Non-IP Networks	**IDE / SCSI** (Direct) **SAN** (Storage Area Network: Fibre Channel)	**DAFS** (Direct Access File System: NFS over VIA)
IP-based Networks	**NBD** (Network Block Device) **iSCSI** (SCSI over TCP/IP)	**NAS** (Network Attached Storage: NFS, CIFS)

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 $\approx$ 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'd)

- Several algorithms exist to schedule the servicing of disk I/O requests

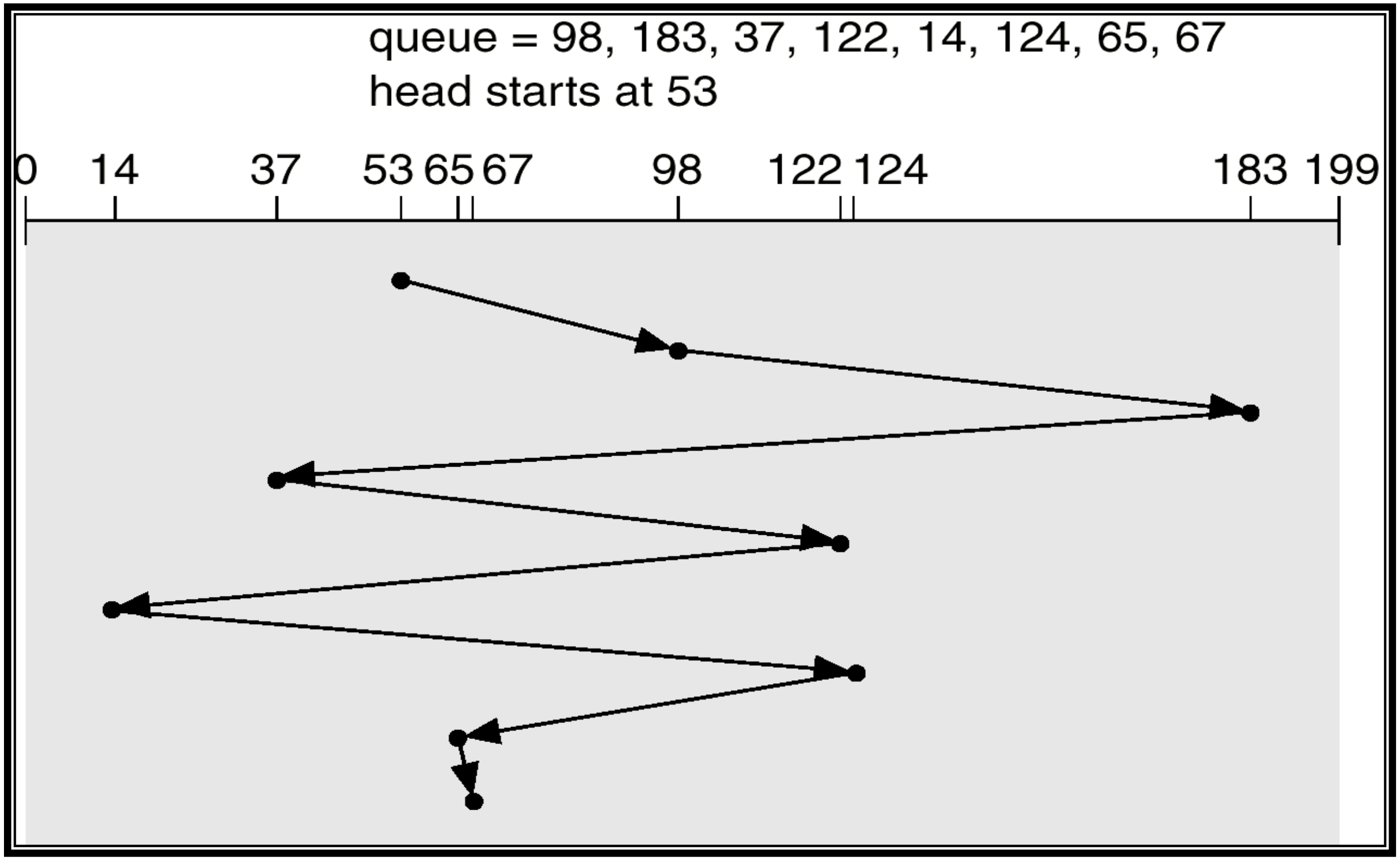
- We illustrate them with a request queue (0-199)

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

  Head pointer 53

FCFS

- Illustration shows total head movement of 640 cylinders



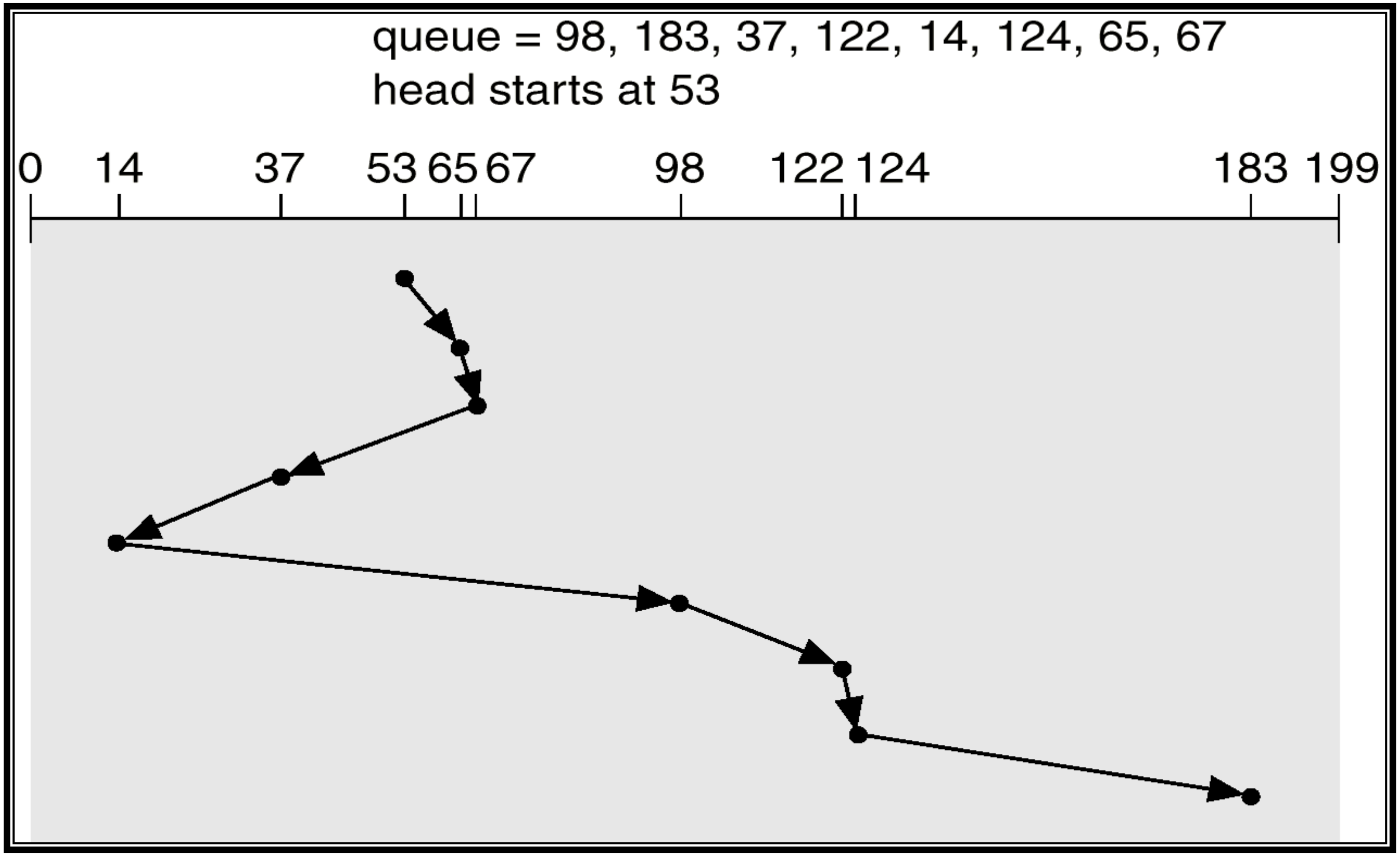
SSTF

- 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

SSTF (Cont'd)

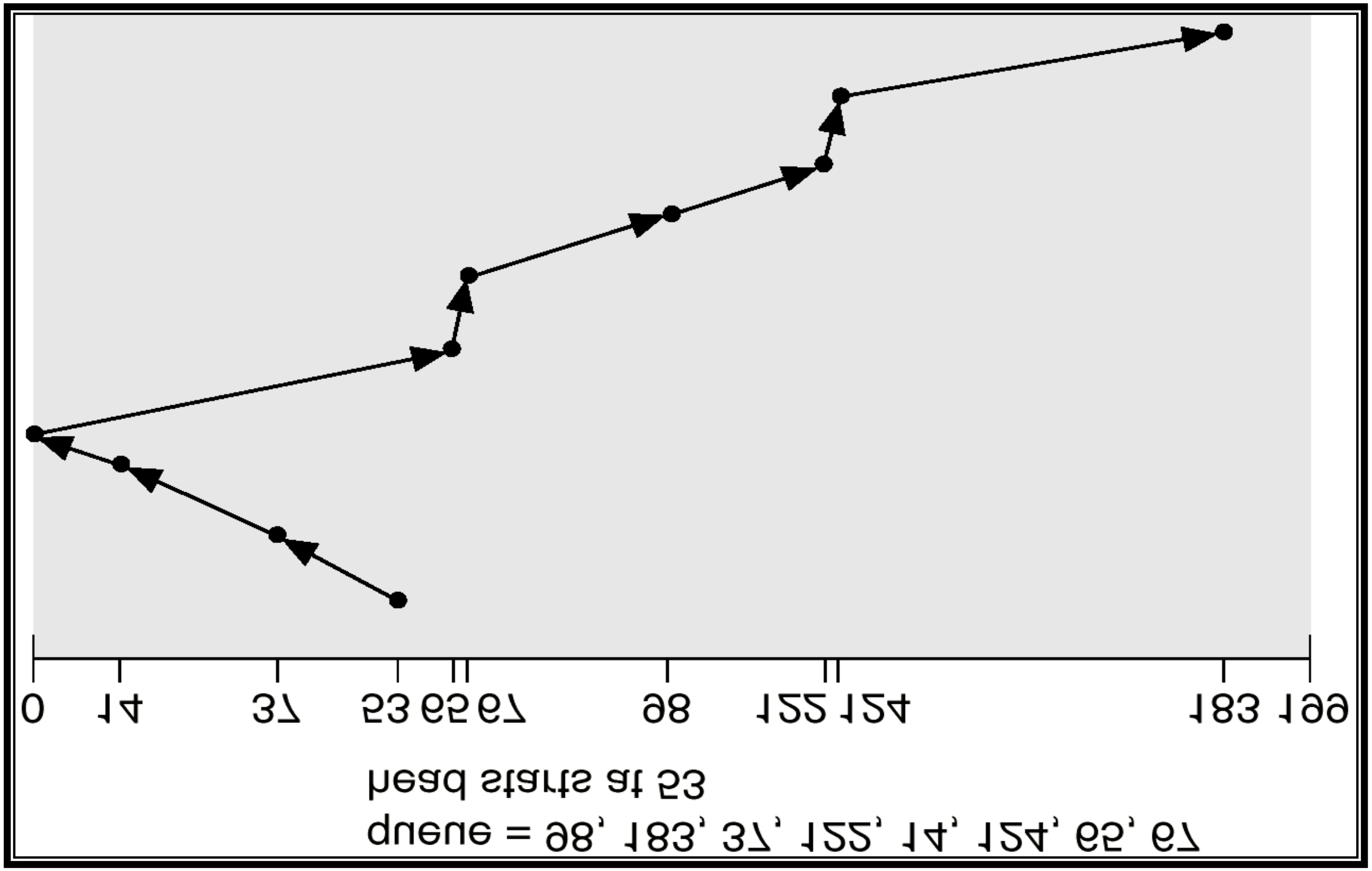


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0 14 37 53 65 67 98 122 124 183 199

SCAN

- 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
- Sometimes called the *elevator algorithm*
- Illustration shows total head movement of 208 cylinders

SCAN (Cont'd)

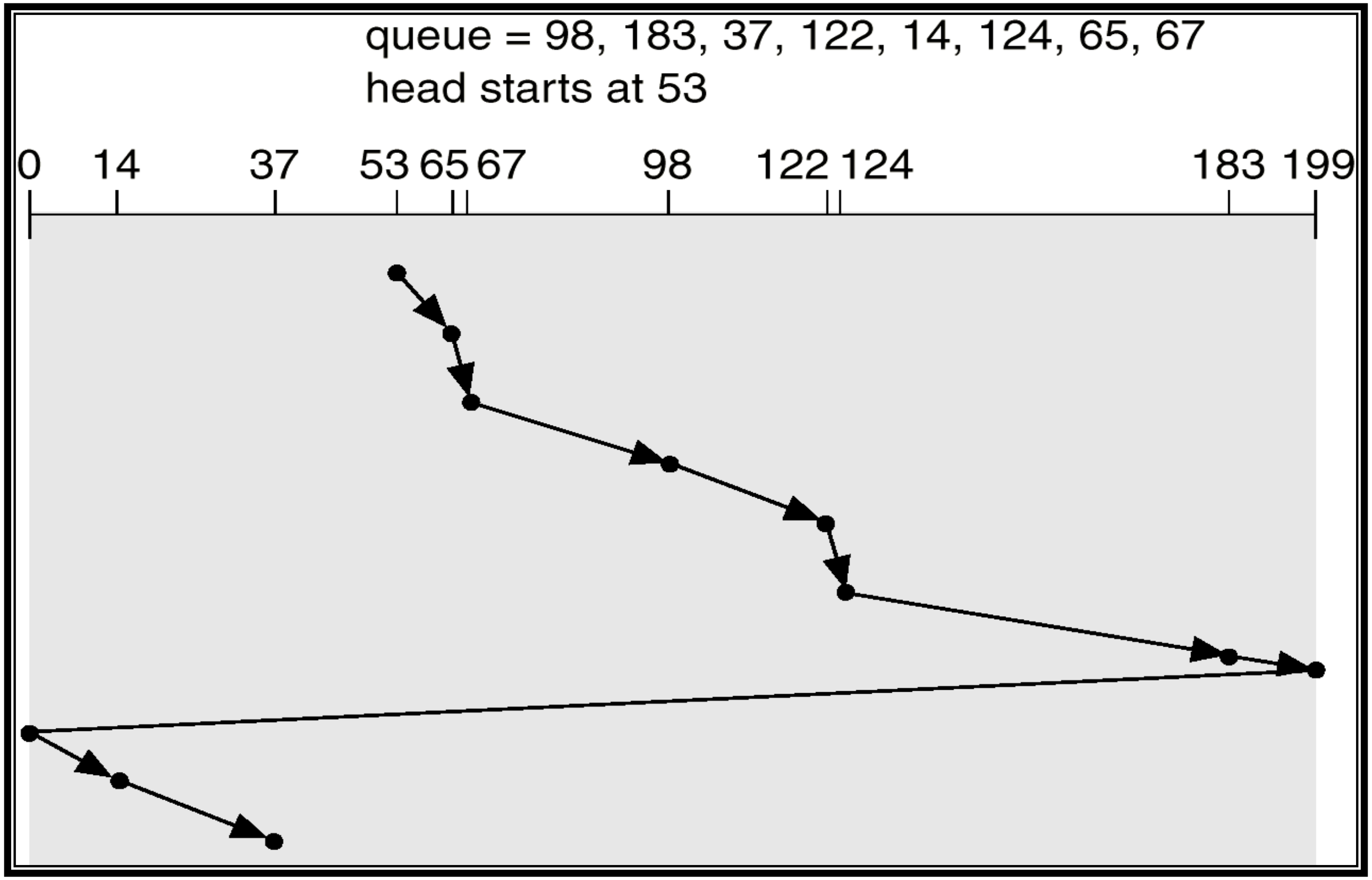


0 14 37 53 65 67 98 122 124 183 199
head starts at 53
queue = 98, 183, 37, 122, 14, 124, 65, 67

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

C-SCAN (Cont'd)

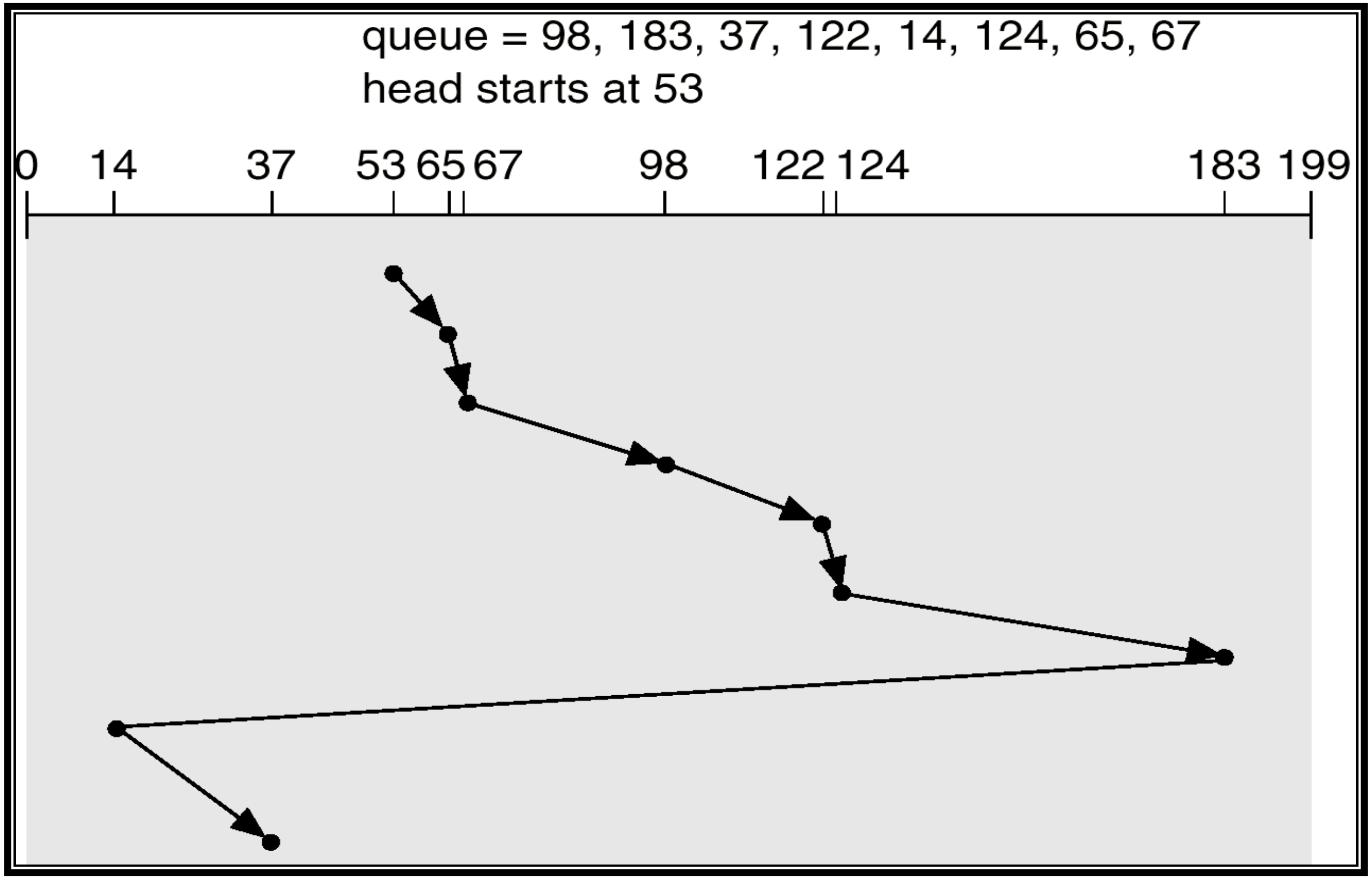


queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-LOOK

- 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
- Cf) Look

C-LOOK (Cont'd)



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

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
- In general, unless there are request queues, disk scheduling does not have much impact
  - ✓ Important for servers, less so for PCs
- Modern disks often do the disk scheduling themselves
  - ✓ Disks know their layout better than OS, can optimize better
  - ✓ Ignores, undoes any scheduling done by OS

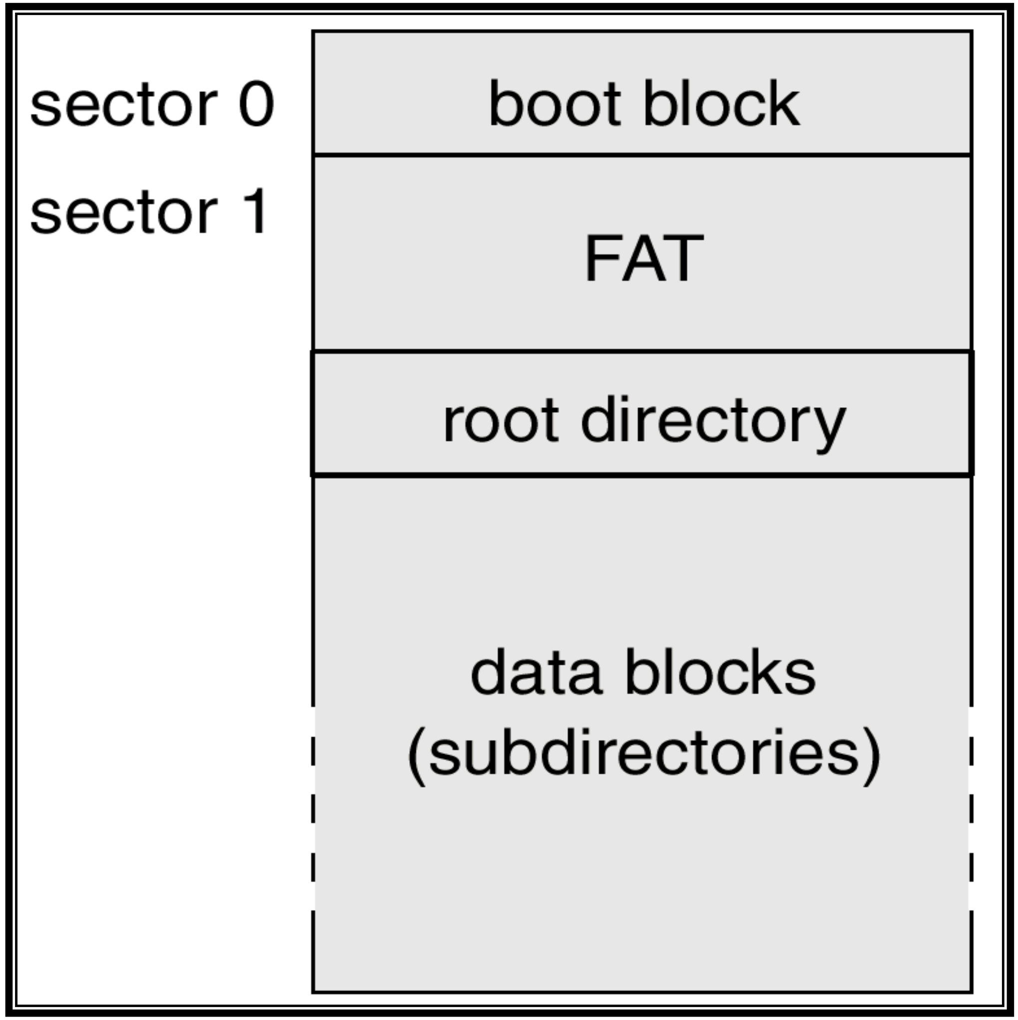
Disk Management

- *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”
- Boot block initializes system
  - ✓ The bootstrap is stored in ROM
  - ✓ *Bootstrap loader* program
- Methods such as *sector sparing* used to handle bad blocks

Disk Controllers

- Intelligent controllers
  - ✓ Nowadays, most disk controllers are built around a small CPU and have many kilobytes of memory
  - ✓ They run a program written by the controller manufacturer to process I/O requests from the CPU and satisfy them
  - ✓ Intelligent features:
    - Read-ahead: the current track
    - Caching: frequently-used blocks
    - Request reordering: for seek and/or rotational optimality
    - Request retry on hardware failure
    - Bad block identification
    - Bad block remapping: onto spare blocks and/or tracks

MS-DOS Disk Layout



Swap-Space Management

- Swap-space
  - ✓ Virtual memory uses disk space as an extension of main memory
- Swap-space can be carved out of the normal file system,or, more commonly, it can be in a separate disk partition
- Swap-space management
  - ✓ 4.3BSD allocates swap space when process starts
    - holds *text segment* (the program) and *data segment*
  - ✓ Kernel uses *swap maps* to track swap-space use
  - ✓ Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created

RAID

- Redundant Array of Inexpensive Disks
  - ✓ A storage system, not a file system
- Motivations
  - ✓ Use small cheap disks as a cost-effective alternative to large, expensive disks (I = Inexpensive)
  - ✓ Provide higher reliability and higher data-transfer

    (I = Independent)
- Improving reliability via redundancy
  - ✓ Mirroring (shadowing)
  - ✓ Parity or error-correcting codes
- Improving performance via parallelism
  - ✓ Data striping: bit-level vs. block-level

RAID Levels

- RAID 0
  - ✓ Non-redundant striping (no reliability)
  - ✓ Data is broken down into blocks and each block is striped across multiple disks
  - ✓ I/O performance is greatly improved by spreading the I/O load across many channels and drives
  - ✓ Typically used in data rate intensive applications (e.g. video editing)

0	1	2	3
A0	A1	A2	A3
A4	A5	A6	A7
A8	A9	A10	A11
...	...	...	...

RAID Levels (Cont'd)

- RAID 1
  - ✓ Mirrored disks
  - ✓ Twice the read transaction rate of single disks, same write transaction rate as single disks
  - ✓ Expensive, highest disk overhead
  - ✓ No rebuild is necessary in case of a disk failure

0	1	2	3	4	5	6	7
A0	B0	C0	D0	A0	B0	C0	D0
A1	B1	C1	D1	A1	B1	C1	D1
A2	B2	C2	D2	A2	B2	C2	D2
…	…	…	…	…	…	…	…

RAID Levels (Cont'd)

- RAID 2
  - ✓ Memory-style error-correcting codes (ECC)
  - ✓ Each data word has its Hamming Code ECC word recorded on the ECC disks
  - ✓ On read, the ECC code verifies correct data or corrects single bit errors
  - ✓ ECC is embedded in almost all modern disk drives (e.g. SCSI)

0	1	2	3	4	5	6
A0	A0	A0	A0	Px	Py	Pz
A1	A1	A1	A1	Px	Py	Pz
A2	A2	A2	A2	Px	Py	Pz
A3	A3	A3	A3	Px	Py	Pz
…	…	…	…	…	…	…

RAID Levels (Cont'd)

- RAID 3
  - ✓ Bit-interleaved parity
  - ✓ Stripe parity is generated on writes, recorded on the parity disk and checked on reads
  - ✓ Less storage overhead than RAID 2
  - ✓ Requires hardware support for efficient use

0	1	2	3	4
A0	A0	A0	A0	P
A1	A1	A1	A1	P
A2	A2	A2	A2	P
A3	A3	A3	A3	P
...	...	...	...	...

RAID Levels (Cont'd)

- RAID 4
  - ✓ Block-interleaved parity
  - ✓ Parity for same rank blocks is generated on writes, recorded on the parity disk and checked on reads
  - ✓ Very good read performance (same as RAID 0)
  - ✓ Writes, however, require parity data be updated each time
  - ✓ No advantages over RAID 5 and does not support multiple simultaneous write operations

0	1	2	3	4
A0	A1	A2	A3	P
A4	A5	A6	A7	P
A8	A9	A10	A11	P
...	...	...	...	...

RAID Levels (Cont'd)

- RAID 5
  - ✓ Block-interleaved distributed parity
  - ✓ Parity for blocks in the same rank is generated on writes, recorded in a distributed location
  - ✓ Can speed up small writes in multiprocessing systems, since the parity disk does not become a bottleneck

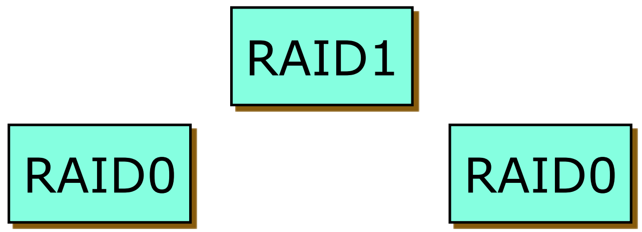
0	1	2	3	4
A0	A1	A2	A3	P
A4	A5	A6	P	A7
A8	A9	P	A10	A11
A12	P	A13	A14	A15
…	…	…	…	…

RAID Levels (Cont'd)

- RAID 6
  - ✓ P+Q redundancy scheme
  - ✓ RAID 5 + extra redundancy information to guard against multiple disk failure
  - ✓ Use error-correcting codes instead of parity

0	1	2	3	4	5
A0	A1	A2	A3	Px	Py
A4	A5	A6	Px	Py	A7
A8	A9	Px	Py	A10	A11
A12	Px	Py	A13	A14	A15
…	…	…	…	…	…

- RAID 0+1
  - ✓ A set of disks are striped, and then the stripe is mirrored to another, equivalent stripe
  - ✓ RAID 0 provides performance, while RAID 1 provides the reliability
  - ✓ A single drive failure will cause the whole array to become, in essence, a RAID 0 array



0	1	2	3	4	5	6	7
A0	A1	A2	A3	A0	A1	A2	A3
A4	A5	A6	A7	A4	A5	A6	A7
A8	A9	A10	A11	A8	A9	A10	A11
...	...	...	...	...	...	...	...

RAID Levels (Cont'd)

- RAID 10 (or RAID 1+0)
  - ✓ Disks are mirrored in pairs, and then the resulting mirror pairs are striped
  - ✓ Reliable better than RAID 0+1
    - RAID 0+1 is fault tolerant as long as the second through n-th disk is on the same stripe
    - RAID 1+0 is fault tolerant as long as no two disks are part of the same mirror

RAID0

R1 R1 R1 R1

0	1	2	3	4	5	6	7
A0	A0	A1	A1	A2	A2	A3	A3
A4	A4	A5	A5	A6	A6	A7	A7
A8	A8	A9	A9	A10	A10	A11	A11
…	…	…	…	…	…	…	…

Tertiary Storage Devices

- Low cost is the defining characteristic of tertiary storage
- Generally, tertiary storage is built using *removable media*
- Common examples of removable media are floppy disks and CD-ROMs
  - ✓ other types are available

Removable Disks

- Floppy disk
  - ✓ Thin flexible disk coated with magnetic material, enclosed in a protective plastic case
  - ✓ Most floppies hold about 1 MB
  - ✓ Similar technology is used for removable disks that hold more than 1 GB
  - ✓ Removable magnetic disks can be nearly as fast as hard disks, but they are at a greater risk of damage from exposure

Removable Disks (Cont'd)

- A magneto-optic disk records data on a rigid platter coated with magnetic material
  - ✓ Laser heat is used to amplify a large, weak magnetic field to record a bit
  - ✓ Laser light is also used to read data (Kerr effect)
  - ✓ The magneto-optic head flies much farther from the disk surface than a magnetic disk head, and the magnetic material is covered with a protective layer of plastic or glass
  - ✓ Resistant to head crashes

- Optical disks do not use magnetism
  - ✓ They employ special materials that are altered by laser light

WORM Disks

- The data on read-write disks can be modified over and over
- WORM (“Write Once, Read Many Times”) disks can be written only once
- Thin aluminum film sandwiched between two glass or plastic platters
- To write a bit, the drive uses a laser light to burn a small hole through the aluminum
  - ✓ Information can be destroyed by not altered
- Very durable and reliable
- *Read Only* disks, such as CD-ROM and DVD, com from the factory with the data pre-recorded

Tapes

- Compared to a disk, a tape is less expensive and holds more data, but random access is much slower
- Tape is an economical medium for purposes that do not require fast random access
  - ✓ e.g., backup copies of disk data, holding huge volumes of data
- Large tape installations typically use robotic tape changers that move tapes between tape drives and storage slots in a tape library
  - ✓ stacker – library that holds a few tapes
  - ✓ silo – library that holds thousands of tapes
- A disk-resident file can be *archived* to tape for low cost storage
  - ✓ The computer can *stage* it back into disk storage for active use

Hierarchical Storage Management (HSM)

- A hierarchical storage system extends the storage hierarchy beyond primary memory and secondary storage to incorporate tertiary storage
  - ✓ Usually implemented as a jukebox of tapes or removable disks
- Usually incorporate tertiary storage by extending the file system
  - ✓ Small and frequently used files remain on disk
  - ✓ Large, old, inactive files are archived to the jukebox
- HSM is usually found in supercomputing centers and other large installations that have enormous volumes of data

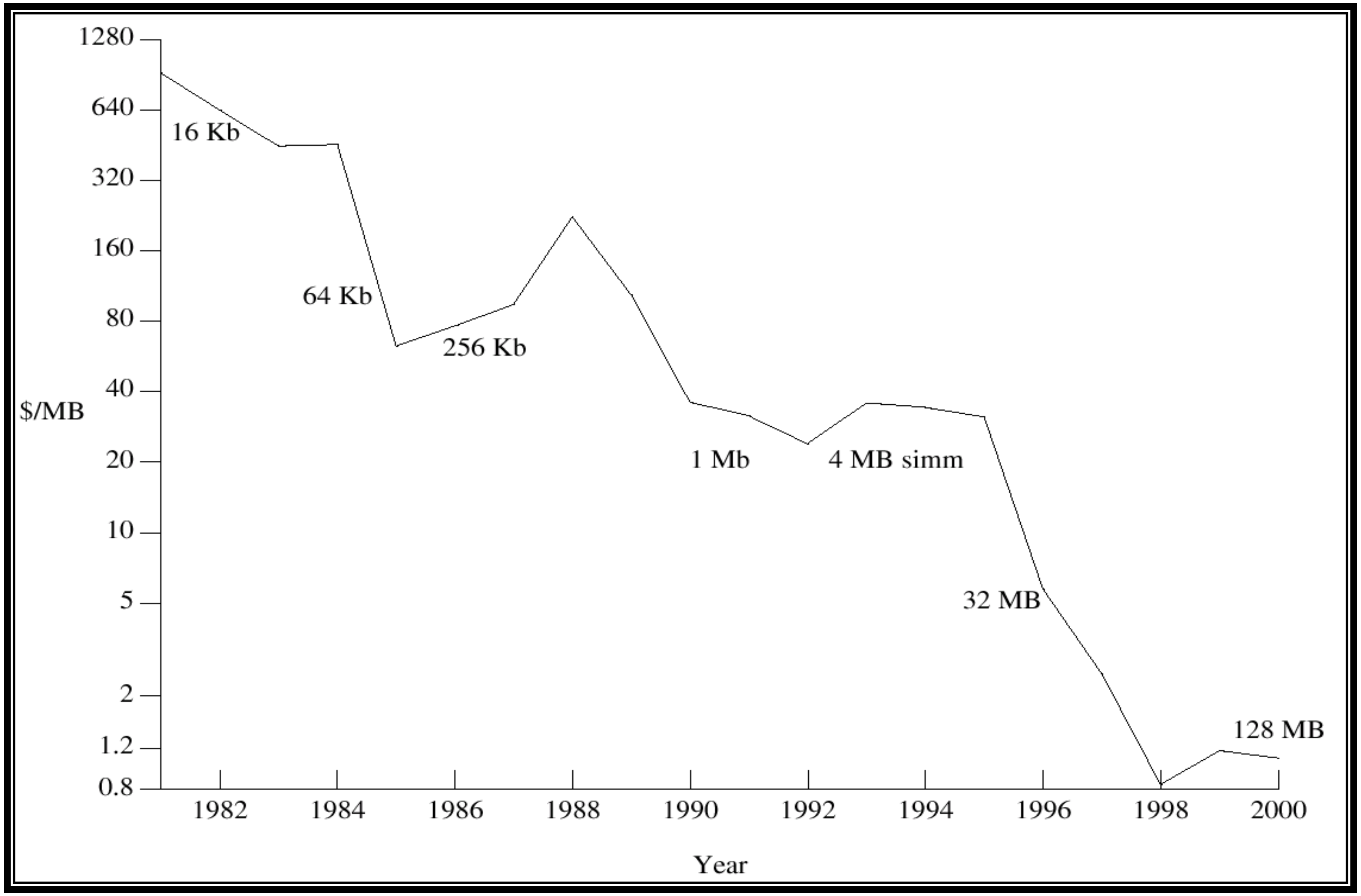
Reliability

- A fixed disk drive is likely to be more reliable than a removable disk or tape drive
- An optical cartridge is likely to be more reliable than a magnetic disk or tape
- A head crash in a fixed hard disk generally destroys the data, whereas the failure of a tape drive or optical disk drive often leaves the data cartridge unharmed

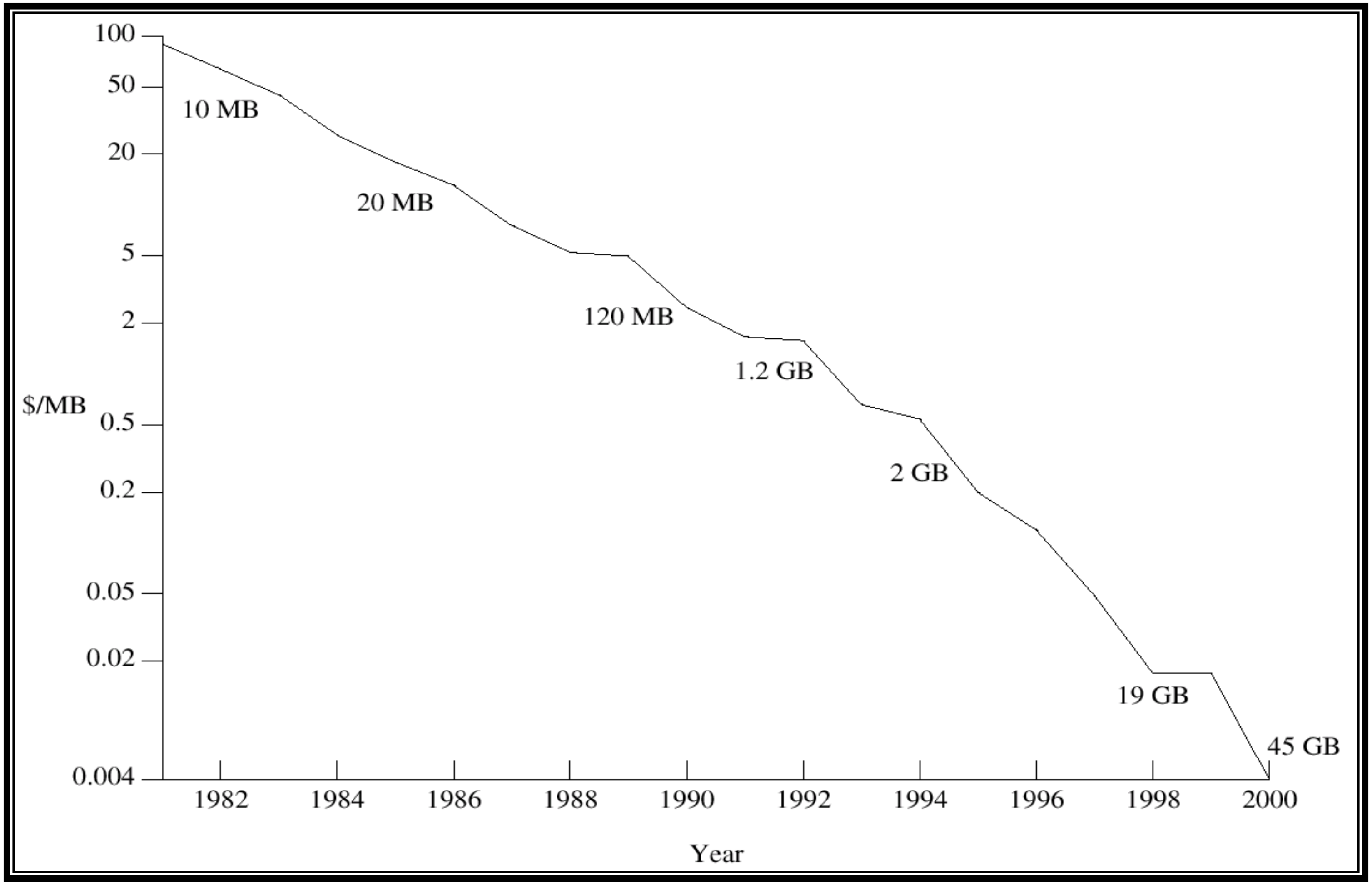
Cost

- Main memory is much more expensive than disk storage
- The cost per megabyte of hard disk storage is competitive with magnetic tape if only one tape is used per drive
- The cheapest tape drives and the cheapest disk drives have had about the same storage capacity over the years
- Tertiary storage gives a cost savings only when the number of cartridges is considerably larger than the number of drives

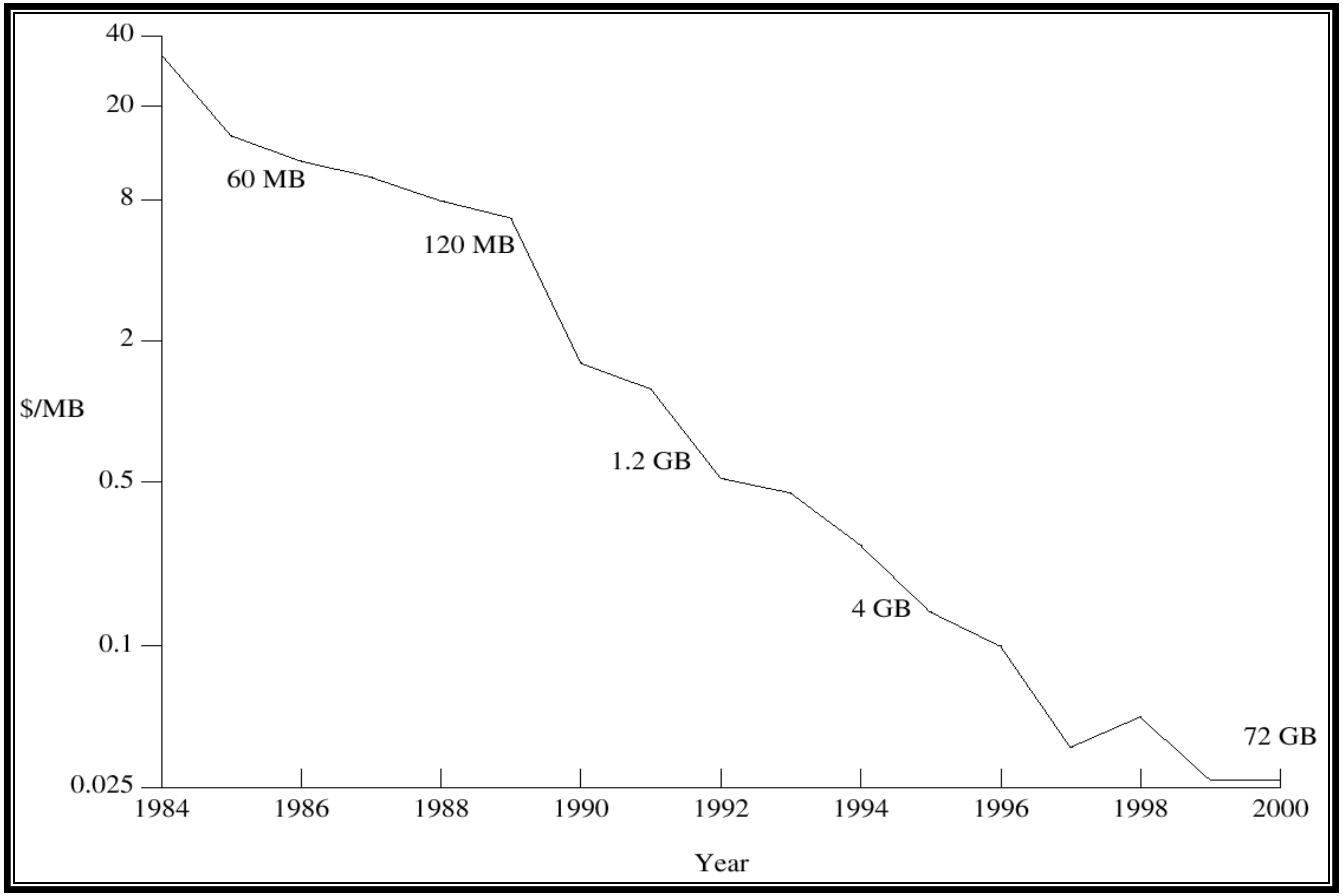
Price per Megabyte of DRAM, From 1981 to 2000



Price per Megabyte of Magnetic Hard Disk, From 1981 to 2000



Price per Megabyte of a Tape Drive, From 1984-2000



Summary

- Disk
  - ✓ Physical disk structure
    - Platter, surface, track, sector, cylinder, arm, and head
  - ✓ Current disk operation
    - Disk exports its data as a logical array of blocks [0, …, N]
    - Disk itself maps logical blocks to cylinder/surface/track/sector
    - OS needs to specify the logical block # to read/write instead of physical parameters
  - ✓ Disk performance
    - Seek: moving the disk arm to the correct cylinder
    - Rotation: waiting for the sector to rotate under head
    - Transfer: transferring data from surface into disk controller, sending it back to the host
    - Because seek time is much larger, OS should schedule disk requests to reduce seek time
  - ✓ Disk attachment
    - Host attached (IDE, SCSI) vs. Network attached (NAS, SAN)

Summary (Cont'd)

- Disk scheduling algorithms
  - ✓ FCFS
  - ✓ SSTF (Shortest Seek Time First)
    - Commonly implemented
  - ✓ SCAN (elevator algorithm)
    - Performs better on heavy load
  - ✓ C-SCAN (Circular-SCAN)
  - ✓ LOOK
  - ✓ C-LOOK

Summary (Cont'd)

- RAID (Redundant Array of Inexpensive Disks)
  - ✓ Motivation
    - Use small cheap disks as a cost-effective alternative to large, expensive disks
    - Provide higher reliability and higher data-transfer
  - ✓ RAID levels
    - RAID0
    - RAID1
    - RAID2
    - RAID3
    - RAID4
    - RAID5
    - RAID6 (P+Q)
    - RAID 0+1
    - RAID 1+0 (RAID10)