Chapter 12: Mass-Storage Systems

Chapter 12: Mass-Storage Systems

- Overview of Mass Storage Structure
- Disk Structure
- Disk Attachment
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure
- Disk Attachment
- Stable-Storage Implementation
- Tertiary Storage Devices
- Operating System Issues
- Performance Issues

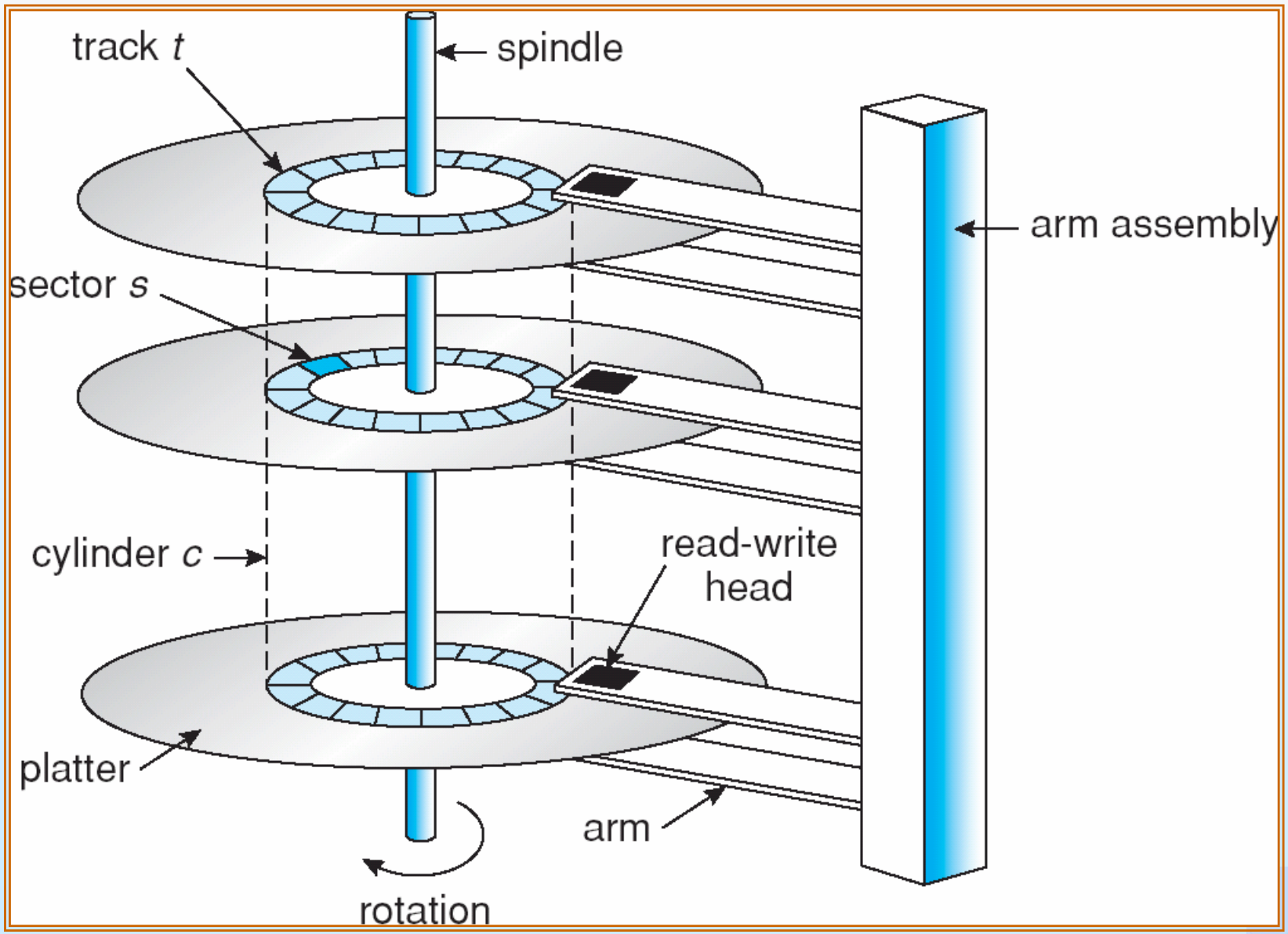
Objectives

- Describe the physical structure of **secondary** and **tertiary** storage devices and the resulting effects on the uses of the devices
- Explain the **performance** characteristics of mass-storage devices
- Discuss operating-system **services** provided for mass storage, including **RAID** and **HSM**

Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  - Drives rotate at 60 to 200 times per second
  - **Transfer rate** is rate at which data flow between drive and computer
  - **Positioning time** (**random-access time**): is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
  - **Head crash** results from disk head making contact with the disk surface
    - That's bad
- Disks can be removable

Moving-head Disk Machanism



Transfer Rates

- Stated Transfer Rates
  - The time at which bits are read from the magnetic surface by the read head.
- Effective Transfer Rates
  - The time that is needed for blocks to be delivered to the Operating System.

Disk Controller

- Drive attached to computer via **I/O bus**
  - Busses vary, including:
    - enhanced integrate drive electronics (**EIDE**)
    - advanced technology attachment (**ATA**)
    - serial ATA (**SATA**)
    - universal serial bus (**USB**)
    - fiber channel (**FC**)
    - Small Computer System Interface (**SCSI)**.
- The data transfers on a bus are carried out by special electronic processors called **controllers**.
- The **host controller** is the controller at the computer end of the bus. A **disk controller** is built into each disk drive.
- To perform a disk I/O operation, the computer places a command into the host controller,
- **Host controller** in computer uses bus to talk to **disk controller.**

Overview of Mass Storage Structure (Cont.)

- Magnetic tape
  - Was early secondary-storage medium
  - Relatively permanent and holds large quantities of data
  - Access time slow
  - Random access ~1000 times slower than disk
  - Mainly used for backup, storage of infrequently-used data, transfer medium between systems
  - Kept in spool and wound or rewound past read-write head
  - Once data under head, transfer rates comparable to disk
  - 20-200GB typical storage
  - Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT

Disk Structure

- Disk drives are addressed as large **1-dimensional arrays of *logical blocks***, where the logical block is the smallest unit of transfer.
- The size of a logical block is usually 512 bytes, although some disks can be **low-level formatted** to have a different logical block size, such as 1024 bytes.
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
  - Sector 0 is the first sector of the first track on the outermost cylinder.
  - **Mapping** proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

Disk Attachment

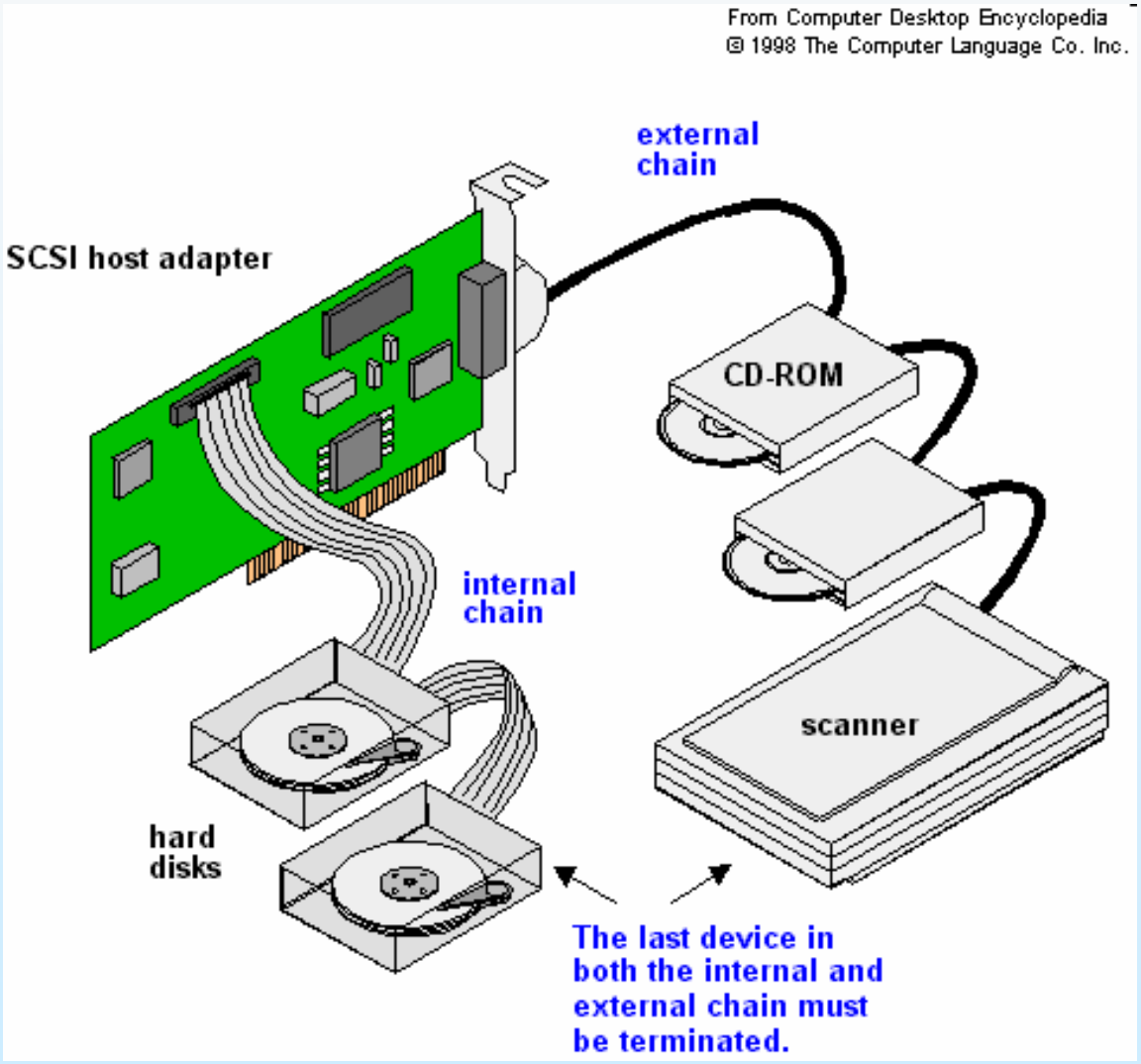
- **Host-attached storage:** accessed through **local** I/O ports talking to I/O busses

- The typical desktop PC uses an I/O bus architecture called IDE or ATA.
- This architecture supports a maximum of two drives per I/O bus.
- A newer, similar protocol that has simplified cabling is **SATA**.

SCSI

- SCSI itself is a bus, up to 16 devices on one cable, **SCSI initiator** requests operation and **SCSI targets** perform tasks
  - Each target can have up to 8 **logical units** (disks attached to device controller)
- FC is high-speed serial architecture
  - **switched fabric:** with 24-bit address space – the basis of **storage area networks** (**SAN**s) in which many hosts attach to many storage units
  - **arbitrated loop:** (**FC-AL**) of 126 devices

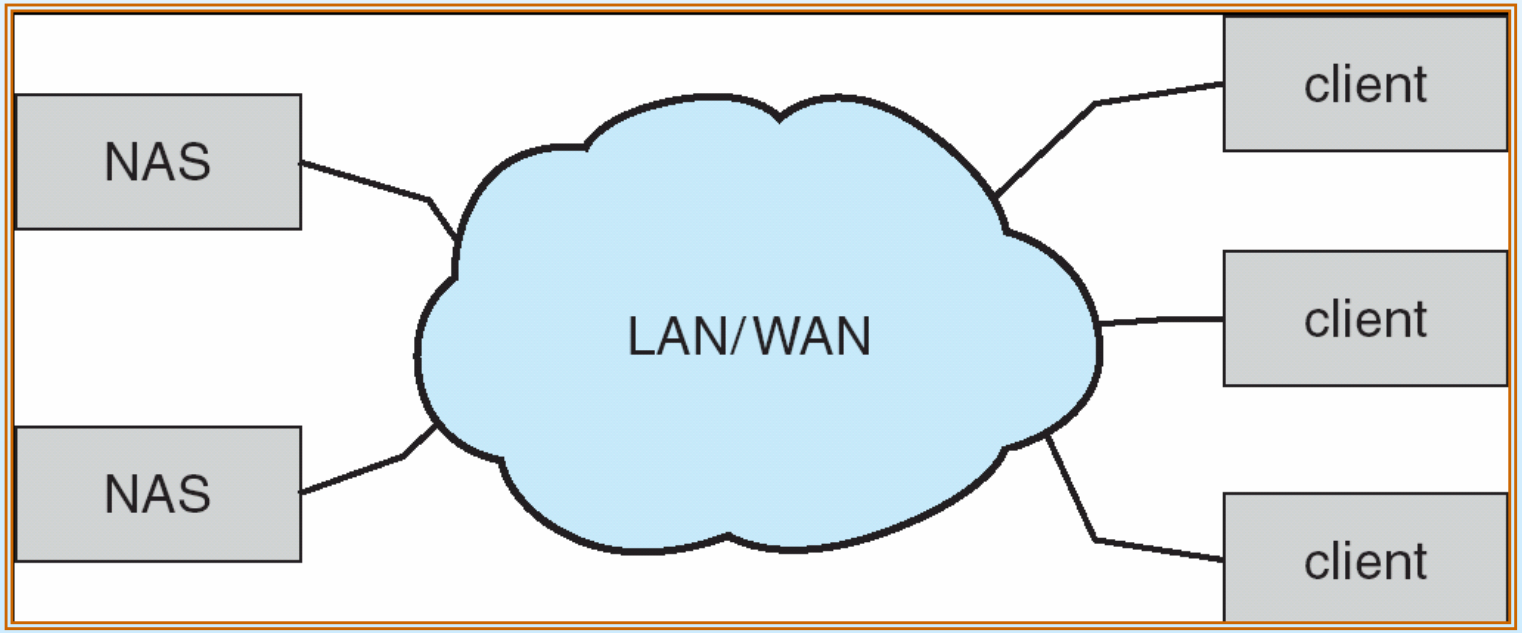
SCSI



- The advantage of SCSI is that several peripherals can be daisy chained to one host adapter, using only one slot in the bus.

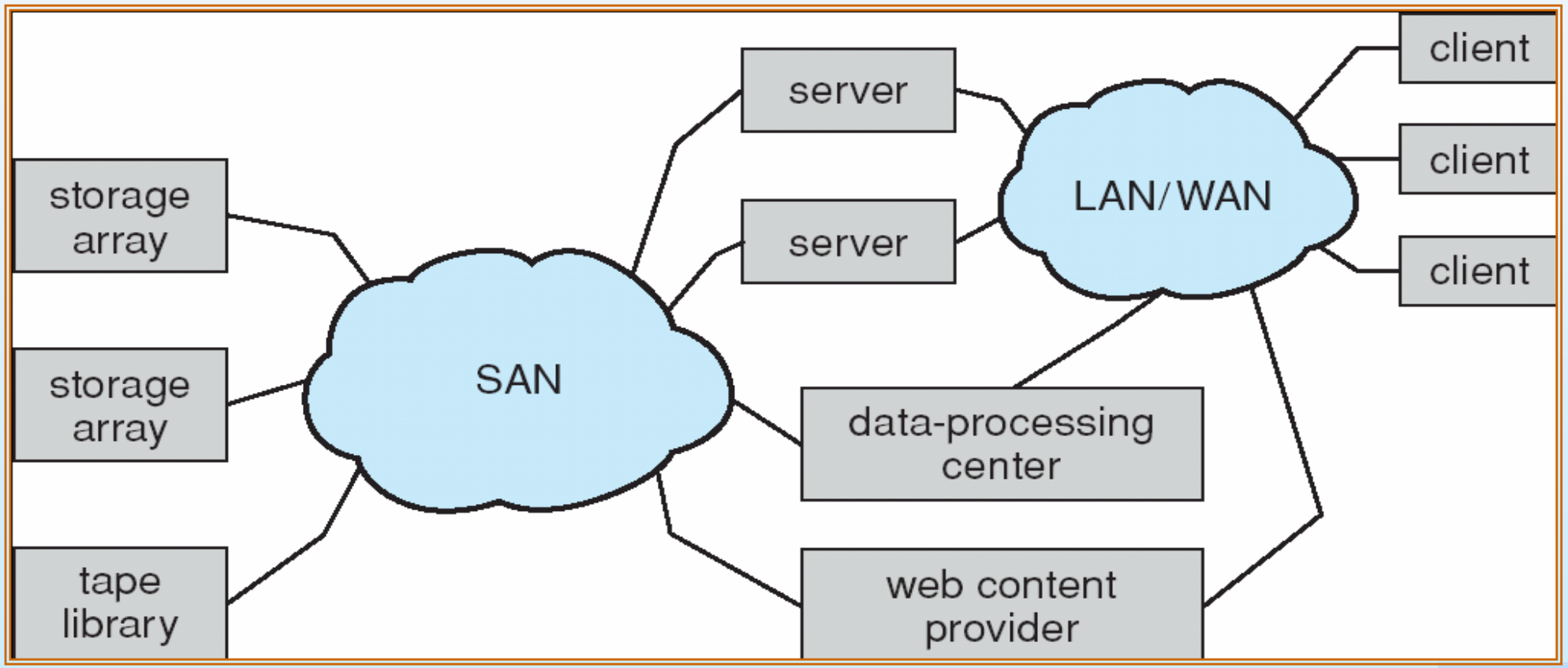
Network-Attached Storage

- Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)
- NFS and CIFS are common protocols
- Implemented via remote procedure calls (RPCs) between host and storage
- New iSCSI protocol uses IP network to carry the SCSI protocol



Storage Area Network

- Common in large storage environments (and becoming more common)
- Multiple hosts attached to multiple storage arrays - flexible



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

- Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. The request specifies several pieces of information:
  - Whether this operation is input or output
  - What the disk address for the transfer is
  - What the memory address for the transfer is
  - What the number of sectors to be transferred is
- If the desired disk drive and controller are available, the request can be serviced immediately.
- If the drive or controller is busy, any new requests for service will be placed in the queue of pending requests for that drive.
  - For a multiprogramming system with many processes, the **disk queue** may often have several pending requests.
- Thus, when one request is completed, the operating system chooses which pending request to service next.
- How does the operating system make this choice?
  - **Disk-scheduling**

Disk Scheduling (Cont.)

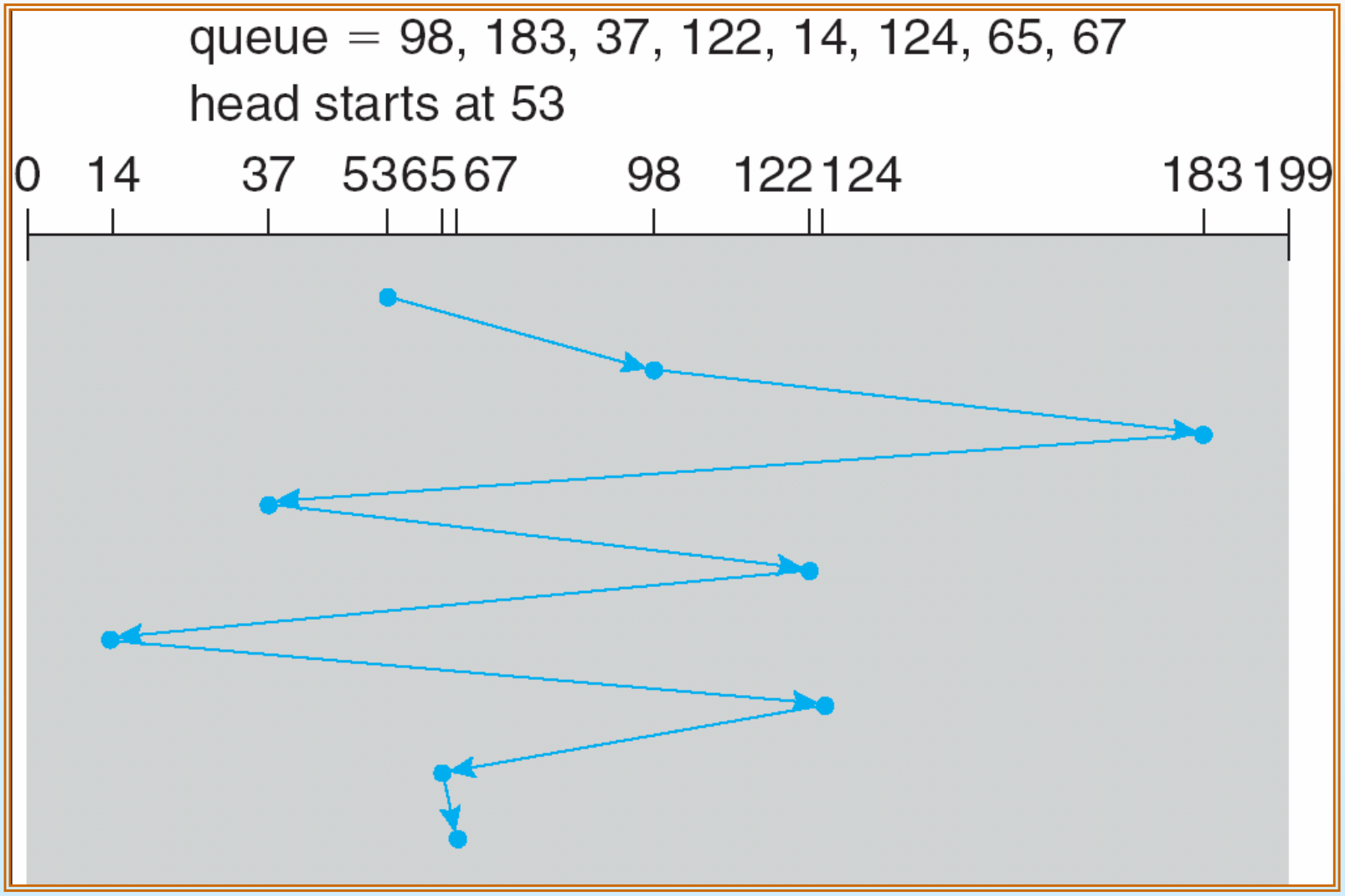
- Several algorithms exist to schedule the servicing of disk I/O requests.
- We illustrate them with a request queue (0-199). These are blocks on cylinders

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

Head pointer 53

FCFS

Illustration shows total head movement of 640 cylinders.



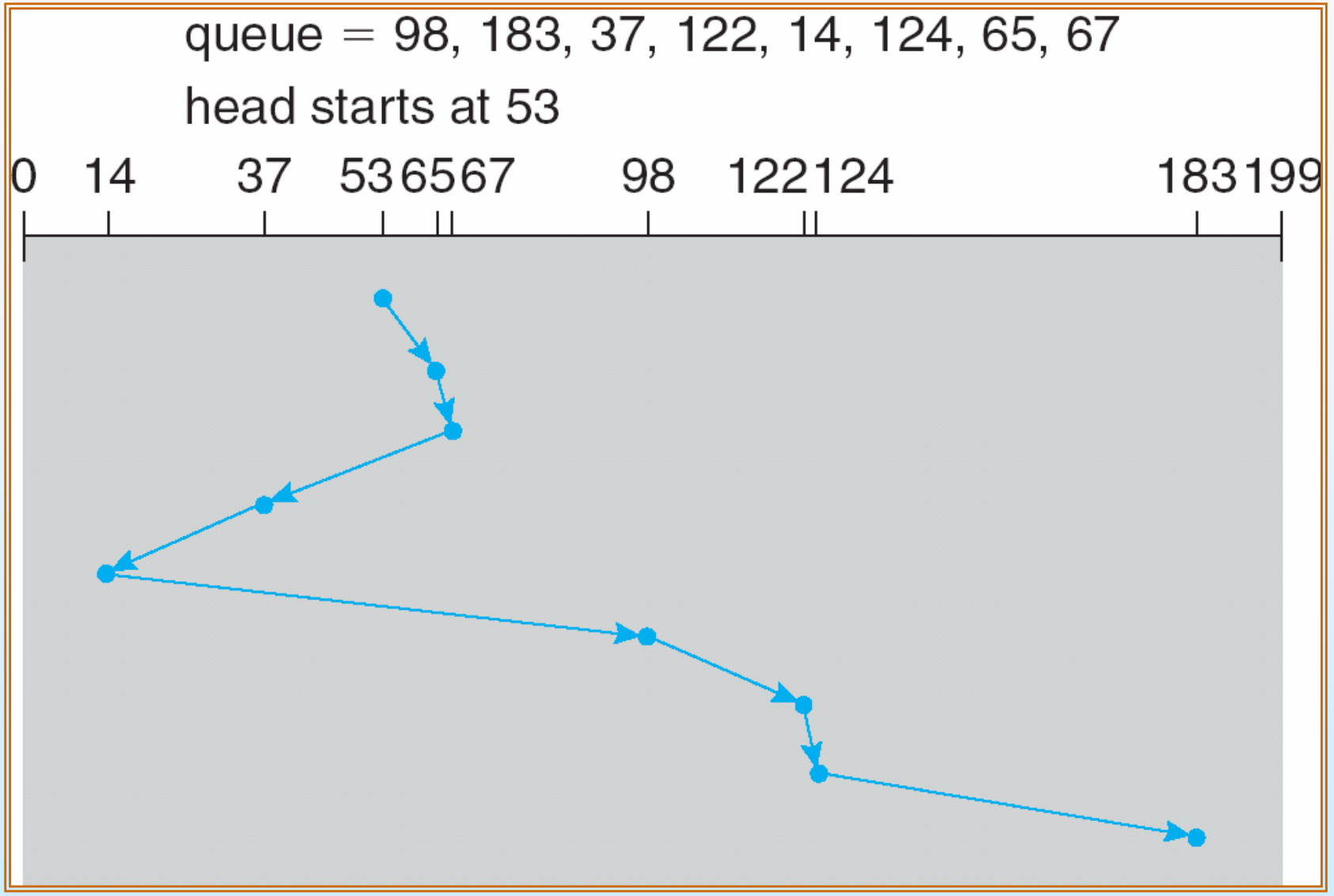
37 and 14 could be serviced together

There is a jump from 122 to 14!!!

SSTF

- SSTF: **shortest-seek-time-first (SSTF)** algorithm.
- 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.)

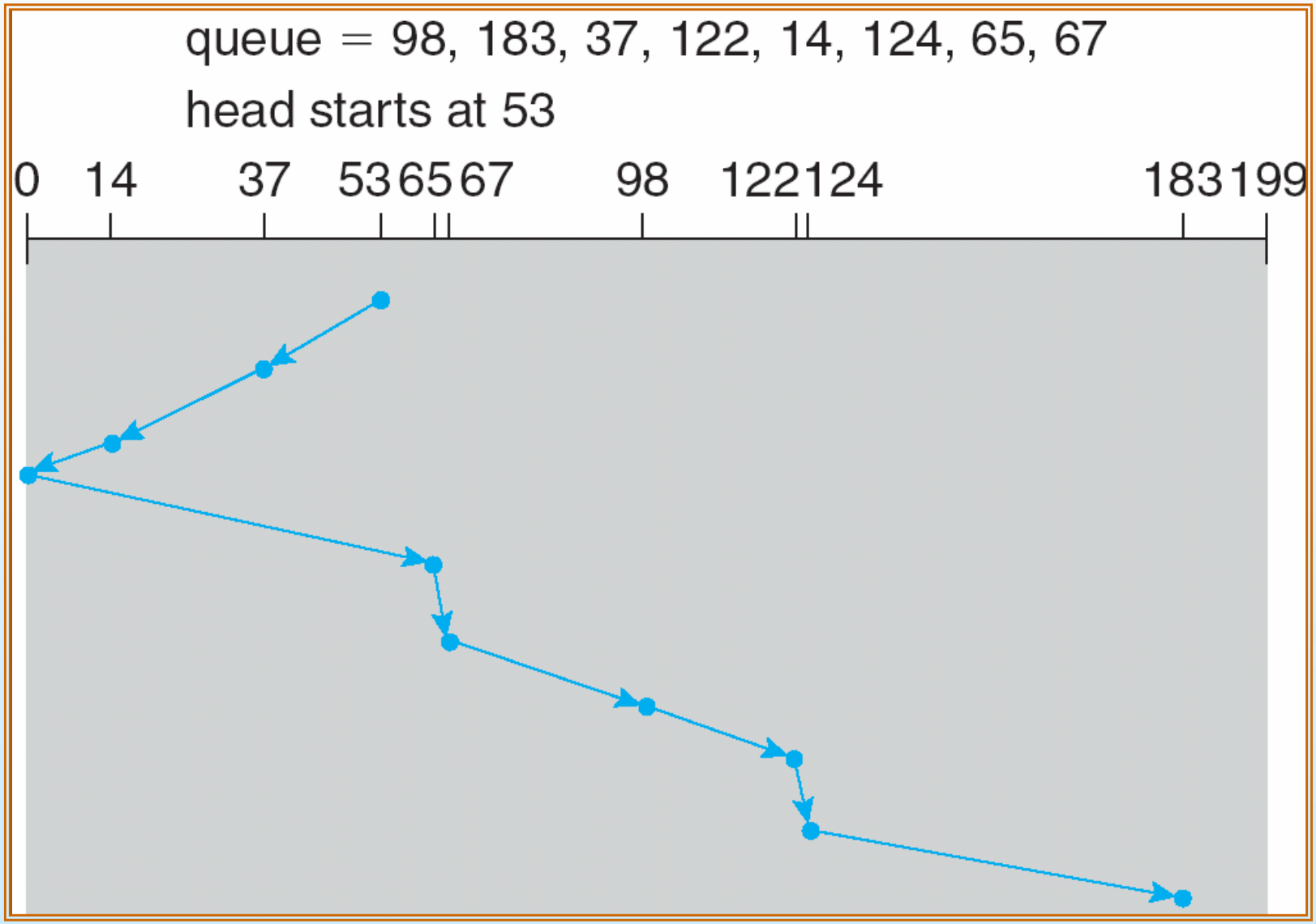


If new requests continue to come, 183 and 199 may starve!!!

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.)

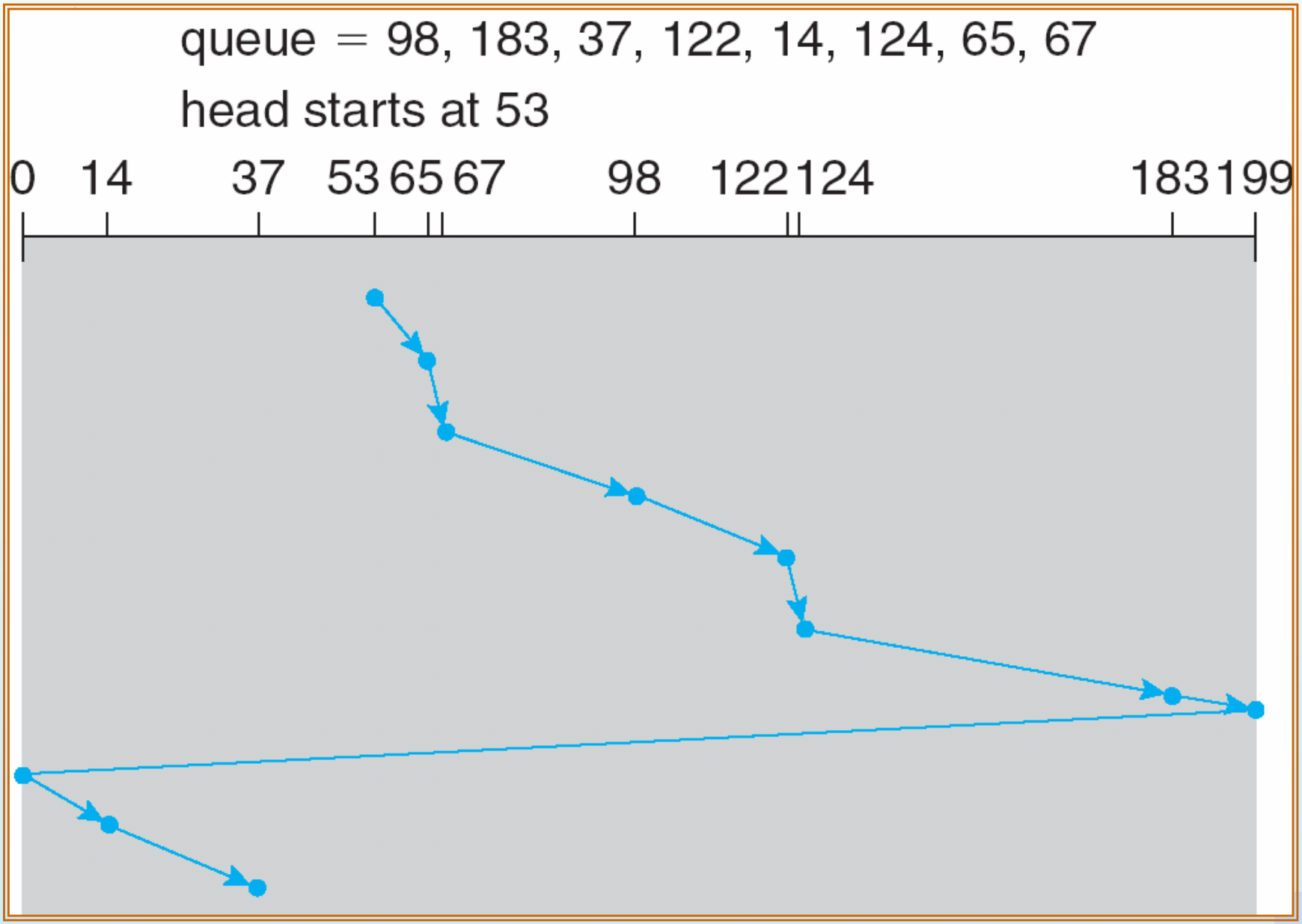

queue = 98, 183, 37, 122, 14, 124, 65, 67
head starts at 53
0
14
37
53
65
67
98
122
124
183
199

C-SCAN

- Consider the density of requests when the head reaches one end and reverses direction.
  - At this point relatively few requests are immediately in front of the head, since these cylinders have recently been serviced.
  - The heaviest density of requests is at the other end of the disk. These requests have also waited the longest, so why not go there first?

- 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.

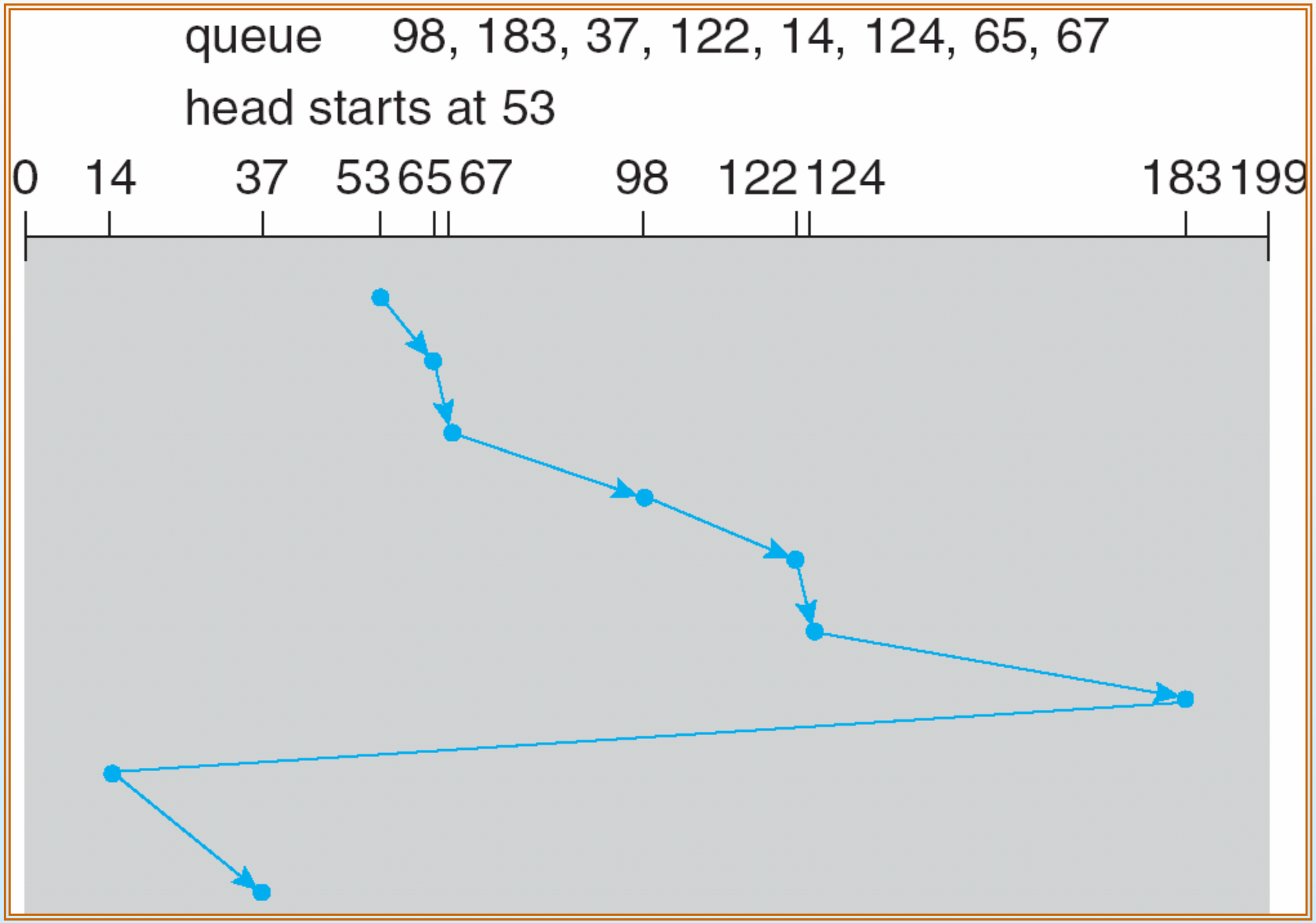
C-SCAN (Cont.)


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.

C-LOOK (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

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.
  - A program reading a **contiguously allocated file** will generate severed requests that are close together on the disk, resulting in limited head movement.
  - A **linked** or **indexed** file, in contrast, may include blocks that are widely scattered on the disk, resulting in greater head movement.
- 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

- The **location** of directories and index blocks is also important.
- Since every file must be opened to be used, and opening a file requires searching the directory structure, the directories will be accessed frequently.
- Suppose that a directory entry is on the **first cylinder** and a file's data are on the **final cylinder**.
  - In this case, the disk head has to move the entire width of the disk. If the directory entry were on the middle cylinder the head would have to move at most, one-half the width.
- **Caching the directories and index blocks** in main memory can also help to reduce the disk-arm movement particularly for read requests.

The role of OS in disk scheduling

- If I/O performance were the only consideration, the operating system would gladly turn over the responsibility of disk scheduling to the disk hardware.
- In practice, however, the operating system may have other constraints on the service order for requests.
  - For instance, **demand paging** may take priority over application I/0, and writes are more urgent than reads if the cache is running out of free pages.
  - Also, it may be desirable to **guarantee** the order of a set of disk writes to make the file system robust in the face of system **crashes**.
- To accommodate such requirements, an operating system may choose to do its own disk scheduling and to spoon-feed the requests to the disk controller, one by one, for some types of I/O.

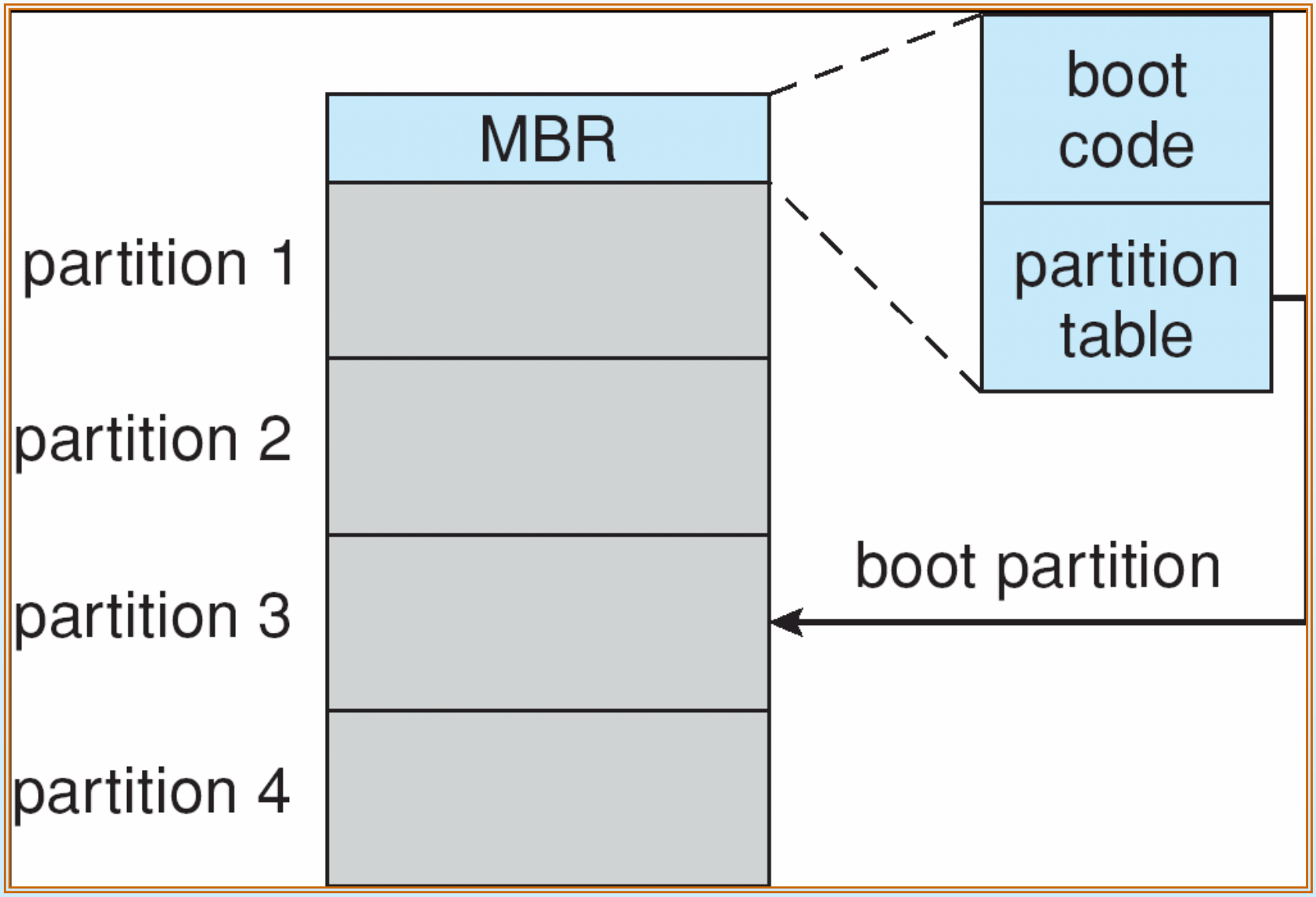
Disk Management

- ***Low-level formatting***, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
- Low-level formatting fills the disk with a special data structure for each sector. The data structure for a sector typically consists of a **header**, a **data area** (usually 512 bytes in size), and a **trailer**.
- The header and trailer contain information used by the disk controller, such as a sector number and an **error-correcting code (ECC).**
- When the controller writes a sector of data during normal I/O, the ECC is updated with a value calculated from all the bytes in the data area.
- When the sector is read, the ECC is recalculated and is compared with the stored value. If the stored and calculated numbers are different, this **mismatch** indicates that the data area of the sector has become corrupted and that the disk sector may be bad.

Disk Management

- 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.
  - The code in the boot ROM instructs the disk controller to read the boot blocks into memory (**no device drivers are loaded at this point**) and then starts executing that code.

Booting from a Disk in Windows 2000



Bad Blocks

- On simple disks, such as some disks with IDE controllers, bad blocks are handled manually.
- For instance, the MS-D0S **format** command performs logical formatting and, as a part of the process, scans the disk to find bad blocks.
  - If format finds a bad block it writes a special value into the corresponding FAT entry to tell the allocation routines not to use that block.
- If blocks go bad during normal operation, a special program (such as **chkdsk**) must be run manually to search for the bad blocks and to lock them away as before.
- Data that resided on the bad blocks usually are **lost**.

Bad Blocks

- More sophisticated disks, such as the SCSI disks used in high-end PCs and most workstations and servers, are smarter about bad-block recovery.
- The controller maintains a list of bad blocks on the disk. The list is initialized during the low-level formatting at the factory and is updated over the life of the disk.
- Low-level formatting also **sets aside spare sectors** not visible to the operating system.
- The controller can be told to **replace** each bad sector logically with one of the spare sectors.
- This scheme is known as **sector sparing** or **forwarding**.

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
  - 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.

Swap-space Linux

- Note that it may be safer to **overestimate** than to underestimate the amount of swap space required, because if a system runs out of swap space it may be forced to abort processes or may crash entirely.
- Overestimation wastes disk space that could otherwise be used for files, but it does no other harm. Some systems recommend the amount to be set aside for swap space:
  - Solaris, for example, suggests setting swap space **equal** to the amount by which virtual memory exceeds pageable physical memory.
  - Historically, Linux suggests setting swap space to **double** the amount of physical memory, although most
  - Linux systems now use considerably less swap space. In fact there is currently much debate in the Linux community about whether to set aside swap space at all!
- Some operating systems - including Linux -allow the use of **multiple swap spaces**. These swap spaces are usually put on separate disks so the load placed on the f/O system by paging and swapping can be spread over the system's f/O devices.

Swap-space in the File System

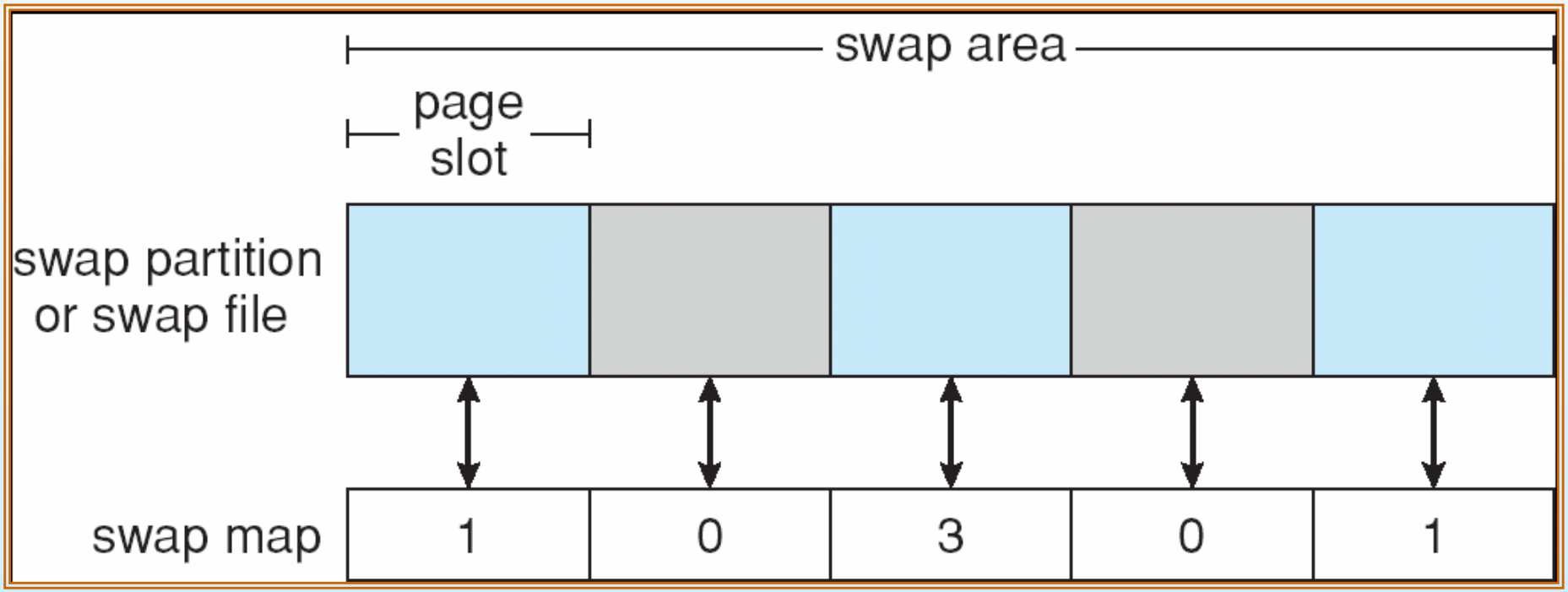
- A swap space can reside in one of two places:
  - It can be carved out of the normal file system, or
  - it can be in a separate disk partition.
- If the swap space is simply a **large file within the file system**, normal file-system routines can be used to create it, name it, and allocate its space.
  - **Inefficient**: navigating the directory-structure and the disk-allocation data structures **takes time**
  - **External fragmentation:** can greatly increase swapping times.

Swap-space in raw partition

- Alternatively, swap space can be created in a separate **raw partition**, as no file system or directory structure is placed in this space.
- Rather, a separate **swap-space storage manager** is used to allocate and deallocate the blocks from the raw partition.
- This manager uses algorithms **optimized for speed** rather than for storage efficiency because swap space is accessed much more frequently than file systems (when it is used).
- Internal fragmentation may increase, but this trade-off is acceptable because the life of data in the swap space generally is much shorter than that of files in the file system.
  - Swap space is **reinitialized** at boot time so any fragmentation is short-lived.

Data Structures for Swapping on Linux Systems

Some operating systems are flexible and can swap both in raw partitions and in file-system space. Linux is an example.



RAID Structure

- **RAID:** redundant arrays of inexpensive disks
- **RAID** – multiple disk drives provides **reliability** via **redundancy**.

- RAID is arranged into six different levels.

RAID (cont)

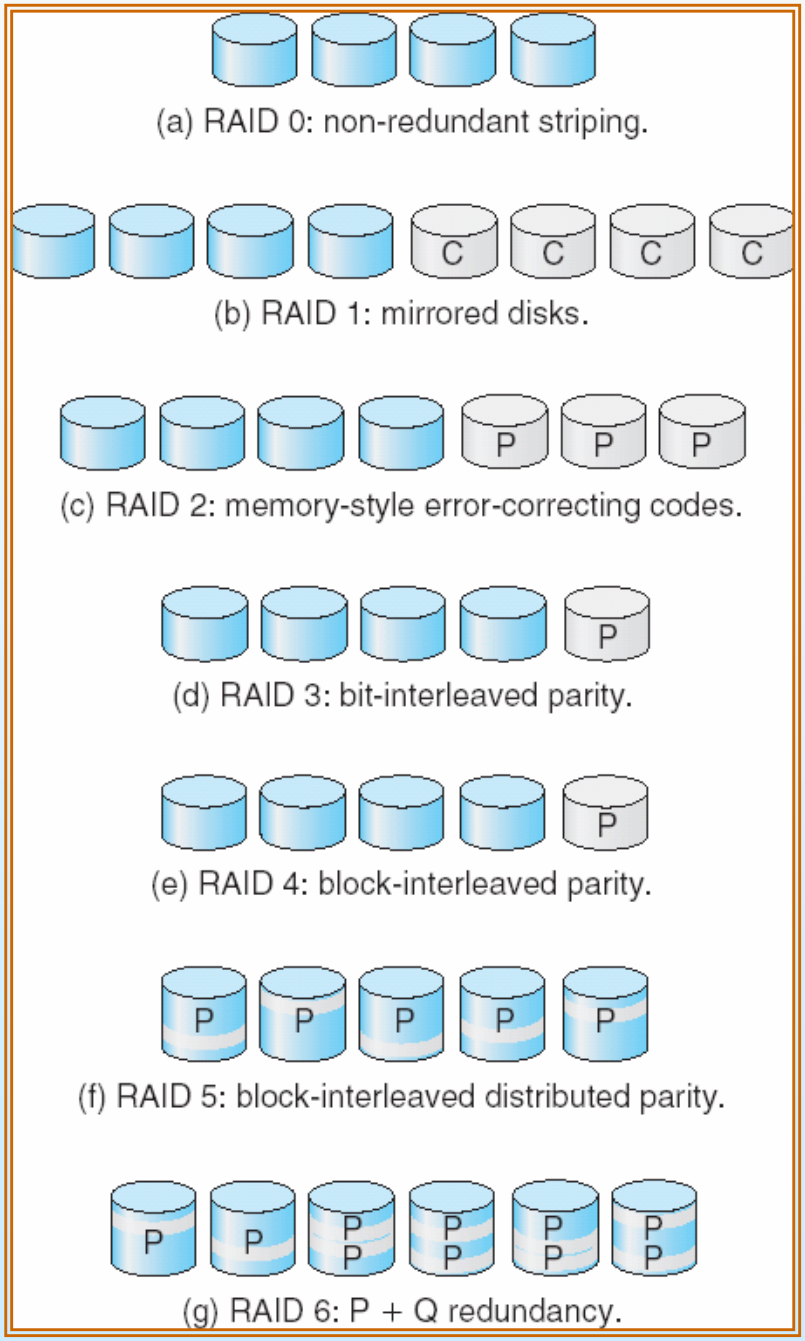
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively.
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data.
  - ***Mirroring*** or *shadowing* keeps duplicate of each disk.
  - *Block interleaved parity* uses much less redundancy.

Improvement through Parallelism

- With disk mirroring the rate at which read requests can be handled is doubled, since read requests can be sent to either disk.

- **Bit-level striping**
  - With multiple disks, we can improve the transfer rate as well by striping data across the disks. In its simplest form, data striping consists of **splitting** the bits of each byte across multiple disks; such striping is called bit-level striping.
- **Block-level striping**
  - Blocks of a file are striped across multiple disks

RAID Levels



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

(c) RAID 2: memory-style error-correcting codes.

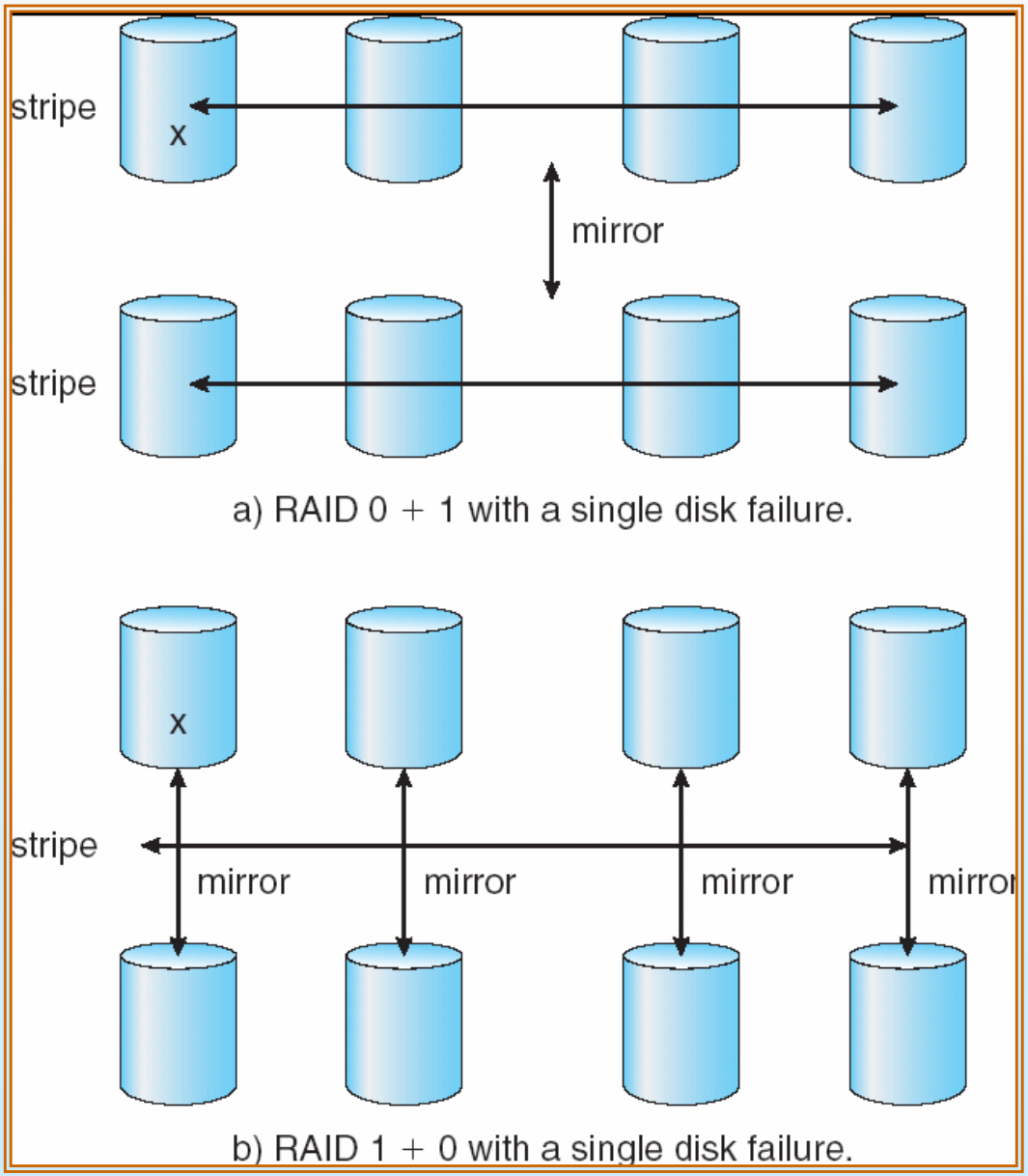
(d) RAID 3: bit-interleaved parity.

(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)



Replication

- Replication involves the automatic duplication of writes between separate sites for redundancy and disaster recovery.
- Replication can be **synchronous** or **asynchronous**.
- In synchronous replication, each block must be written locally and remotely before the write is considered complete, whereas in asynchronous replication, the writes are grouped together and written periodically.
- Asynchronous replication can result in data loss if the primary site fails but is faster and has no distance limitations.

Stable-Storage Implementation

- Write-ahead log scheme requires stable storage.
- To implement stable storage:
  - Replicate information on more than one nonvolatile storage media with independent failure modes.
  - Update information in a controlled manner to ensure that we can recover the stable data after any failure during data transfer or recovery.

Recovery

- During recovery from a failure, each pair of physical blocks is examined.
- If both are the same and no detectable error exists, then no further action is necessary.
- If one block contains a detectable error, then we replace its contents with the value of the other block.
- If neither block contains a detectable error, but the blocks differ in content, then we replace the content of the first block with that of the second.
- **This recovery procedure ensures that a write to stable storage either succeeds completely or results in no change.**

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.)

- 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

- Optical disks do not use magnetism at all.
- Instead, they use special materials that can be altered by laser light to have relatively dark or bright spots.
- One example of optical-disk technology is the **phase-change disk** which is coated with a material that can freeze into either a crystalline or an amorphous state.
- The most common examples of this technology are the re-recordable CD-RW and DVD-RW.

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 but not altered.
- Very durable and reliable.
- *Read Only* disks, such ad CD-ROM and DVD, come 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.

Operating System Issues

- Major OS jobs are:
  - to manage physical devices
  - to present a virtual machine abstraction to applications
- For hard disks, the OS provides two abstractions:
  - Raw device – an array of data blocks.
  - File system – the OS queues and schedules the interleaved requests from several applications.

Application Interface

- Most OSs handle removable disks almost exactly like fixed disks — a new cartridge is formatted and an empty file system is generated on the disk.
- Tapes are presented as a **raw storage medium**, i.e., and application does not not open a file on the tape, it opens the whole tape drive as a raw device.
  - Usually the tape drive is reserved for the exclusive use of that application.
  - Since the OS does not provide file system services, the application must decide how to use the array of blocks.
- Since every application makes up its own rules for how to organize a tape, a tape full of data can generally only be **used by the program that created it**.

Tape Drives

- The basic operations for a tape drive differ from those of a disk drive.
- **Locate:** positions the tape to a specific logical block, not an entire track (corresponds to **seek**).
- The **read_position** operation returns the **logical block number** where the tape head is.
- The **space** operation enables relative motion.
- Tape drives are “append-only” devices;
  - updating a block in the middle of the tape also effectively erases everything beyond that block.
- An EOT (end-of-tape) mark is placed after a block that is written.

File Naming

- The issue of naming files on removable media is especially difficult when we want to write data on a removable cartridge on one computer, and then use the cartridge in another computer.
- Contemporary OSs generally leave the name space problem unsolved for removable media, and depend on applications and users to figure out how to access and interpret the data.
- Some kinds of removable media (e.g., CDs) are so well standardized that all computers use them the same way.

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.

Speed

- Two aspects of speed in tertiary storage are bandwidth and latency.

- Bandwidth is measured in bytes per second.
  - **Sustained bandwidth** – average data rate during a large transfer; # of bytes/transfer time.
    Data rate when the data stream is actually flowing.
  - **Effective bandwidth** – average over the entire I/O time, including **seek** or **locate**, and cartridge switching.
    Drive's overall data rate.

Speed (Cont.)

- Access latency – amount of time needed to locate data.
  - **Access time for a disk** – move the arm to the selected cylinder and wait for the rotational latency; < 35 milliseconds.
  - **Access on tape** requires winding the tape reels until the selected block reaches the tape head; tens or hundreds of seconds.
  - Generally say that random access within a tape cartridge is about a **thousand times slower** than random access on disk.
- The low cost of tertiary storage is a result of having many cheap cartridges share a few expensive drives.
- A removable library is best devoted to the storage of **infrequently used data**, because the library can only satisfy a relatively small number of I/O requests per hour.

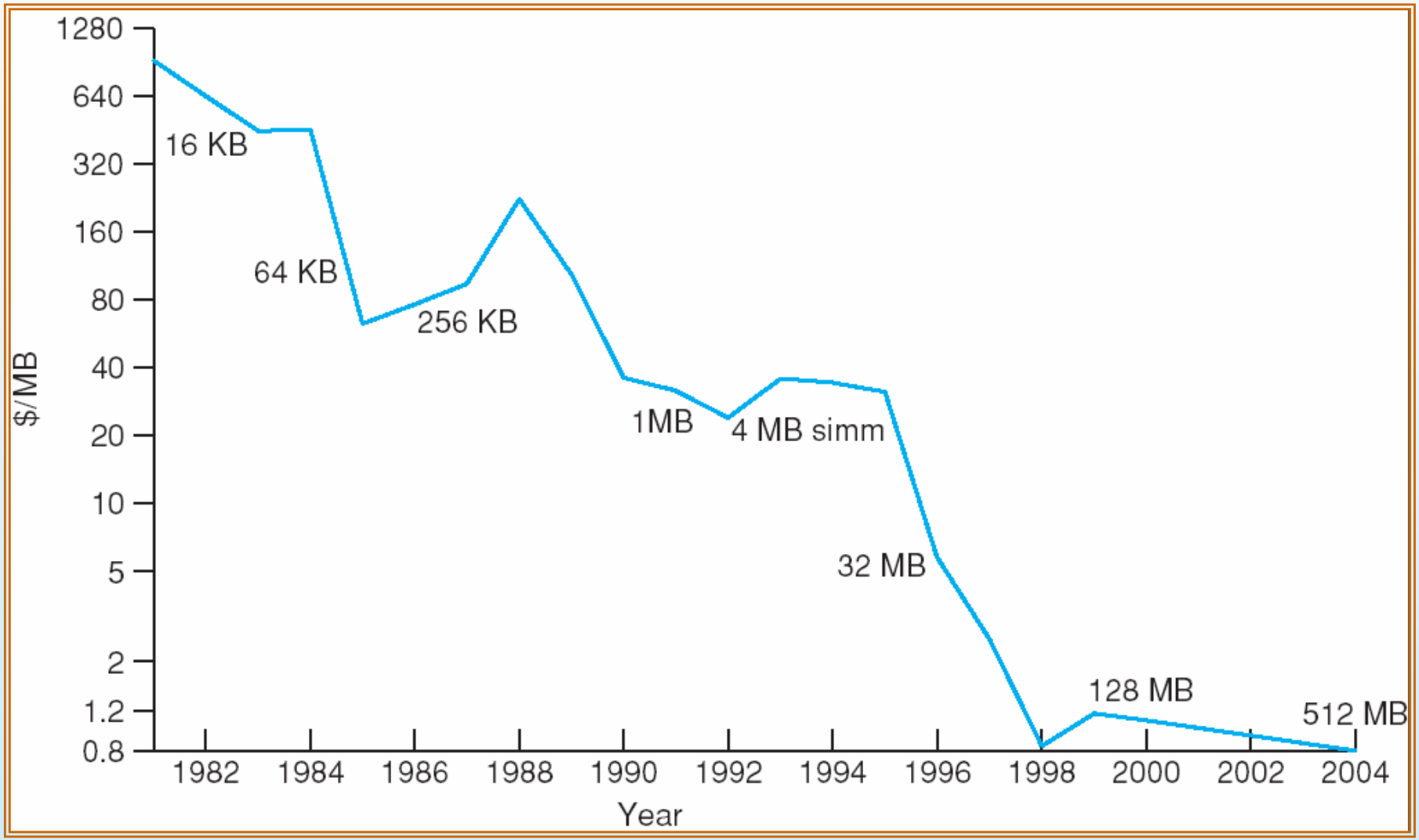
Reliability

- 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.

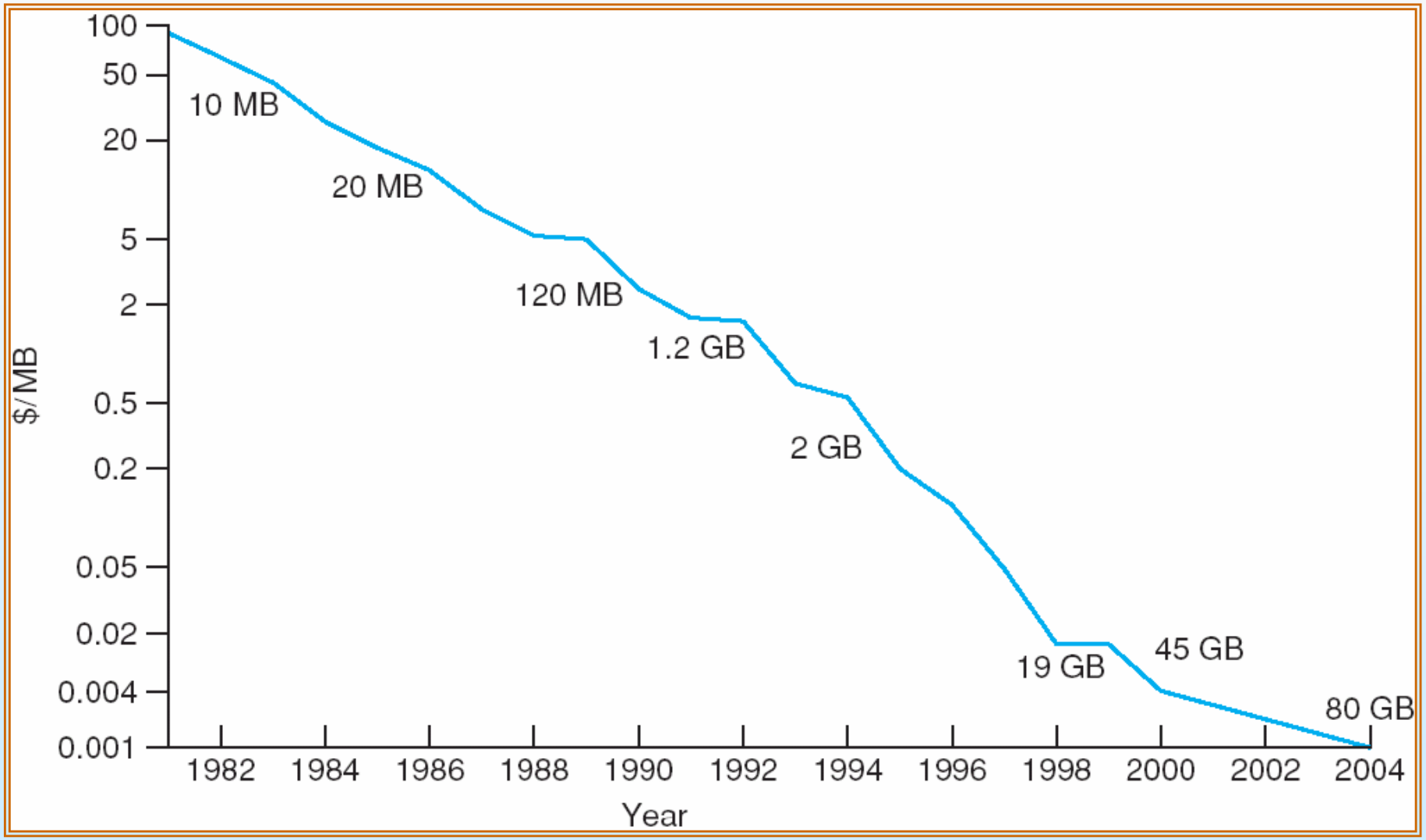
Cost

- 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.

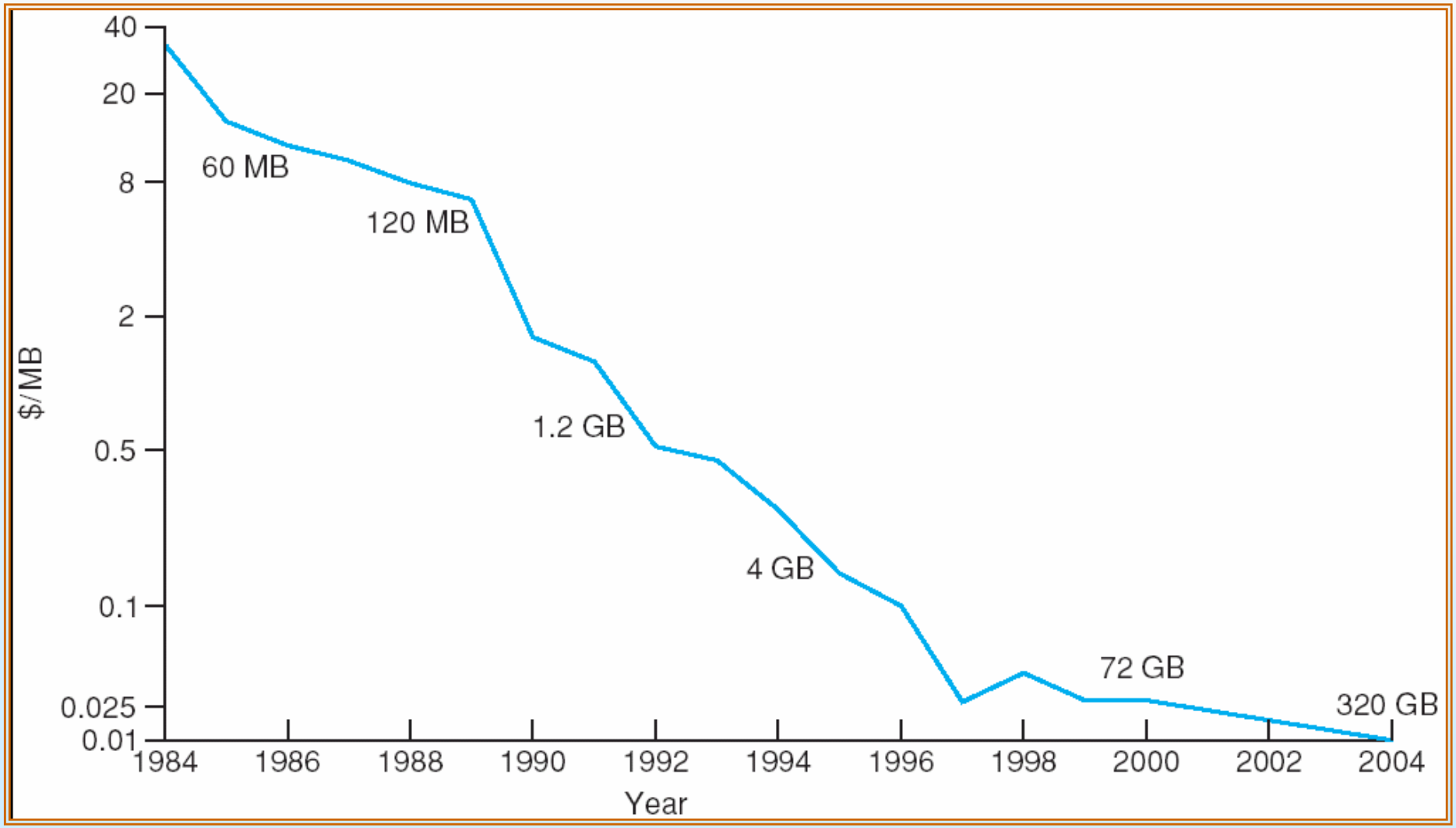
Price per Megabyte of DRAM, From 1981 to 2004



Price per Megabyte of Magnetic Hard Disk, From 1981 to 2004



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



End of Chapter 12