Mass-Storage Systems

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 Support
- Performance Issues

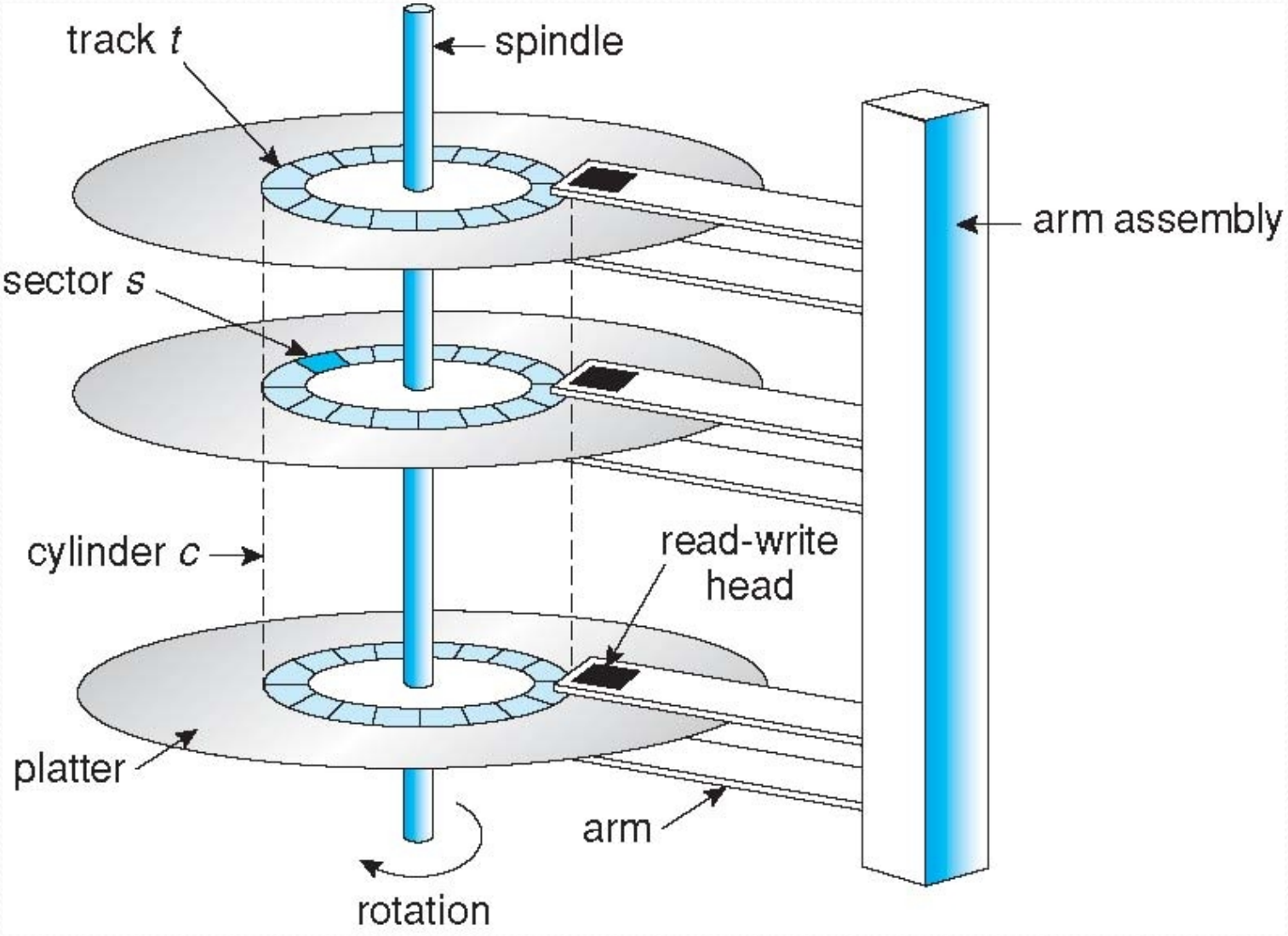
Objectives

- 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
- Drive attached to computer via **I/O bus**
  - Busses vary, including **EIDE**, **ATA**, **SATA**, **USB**, **Fibre Channel**, **SCSI**
  - **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

Moving-head Disk Mechanism



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

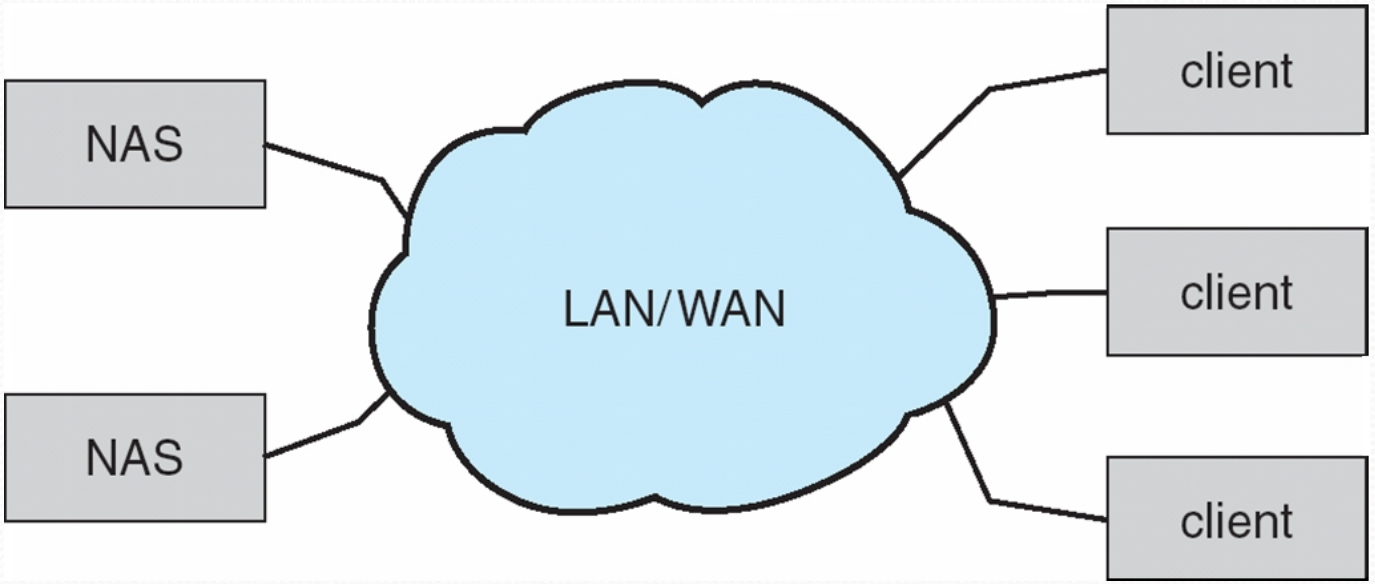
- Disk drives are addressed as large 1-dimensional arrays of **logical blocks**, where the logical block is the smallest unit of transfer
- The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially
  - Sector 0 is the first sector of the first track on the outermost 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 I/O ports talking to I/O busses
- 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
  - Can be switched fabric with 24-bit address space – the basis of **storage area networks (SAN**s**)** in which many hosts attach to many storage units
  - Can be **arbitrated loop (FC-AL)** of 126 devices

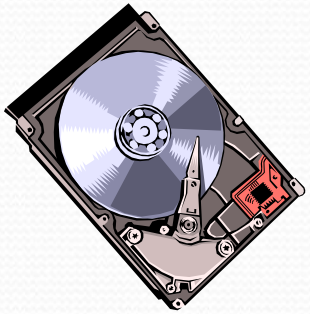
Network-Attached Storage

- 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



Categories of I/O Devices

External devices that engage in I/O with computer systems can be grouped into three categories:



Human readable

- suitable for communicating with the computer user
- printers, terminals, video display, keyboard, mouse

Machine readable

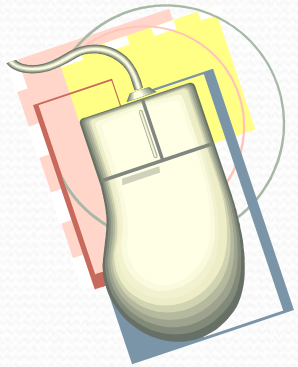
- suitable for communicating with electronic equipment
- disk drives, USB keys, sensors, controllers

Communication

- suitable for communicating with remote devices
- modems, digital line drivers

Differences in I/O Devices

- Devices differ in a number of areas:



Data Rate

- there may be differences of magnitude between the data transfer rates

Application

- the use to which a device is put has an influence on the software

Complexity of Control

- the effect on the operating system is filtered by the complexity of the I/O module that controls the device

Unit of Transfer

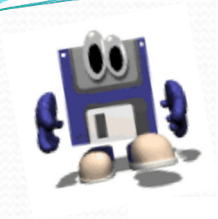
- data may be transferred as a stream of bytes or characters or in larger blocks

Data Representation

- different data encoding schemes are used by different devices

Error Conditions

- the nature of errors, the way in which they are reported, their consequences, and the available range of responses differs from one device to another



Data Rates

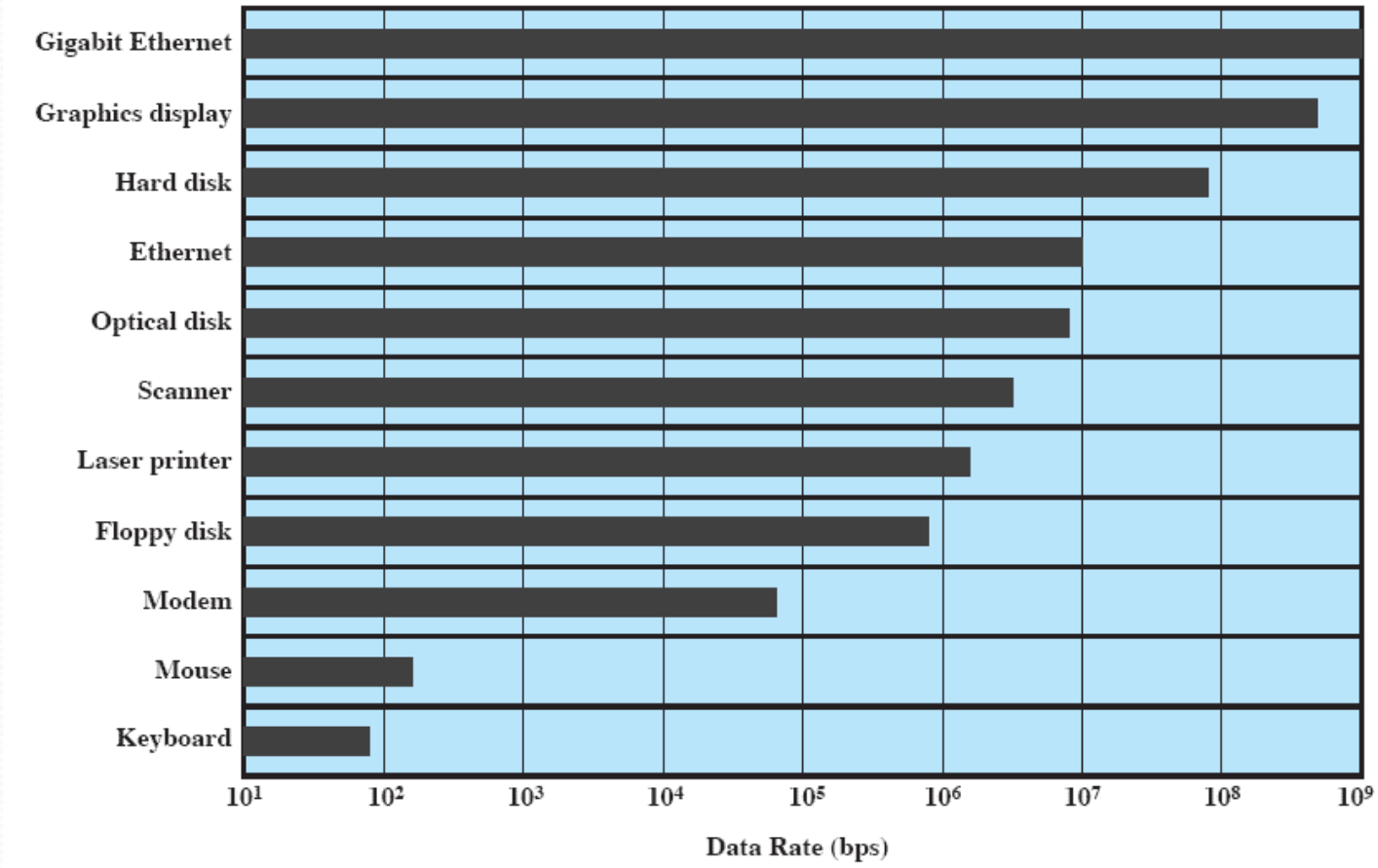


Figure 11.1 Typical I/O Device Data Rates

Organization of the I/O Function

- Three techniques for performing I/O are:
- **Programmed I/O**
  - the processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding
- **Interrupt-driven I/O**
  - the processor issues an I/O command on behalf of a process
    - if non-blocking – processor continues to execute instructions from the process that issued the I/O command
    - if blocking – the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process
- **Direct Memory Access (DMA)**
  - a DMA module controls the exchange of data between main memory and an I/O module

Techniques for Performing I/O

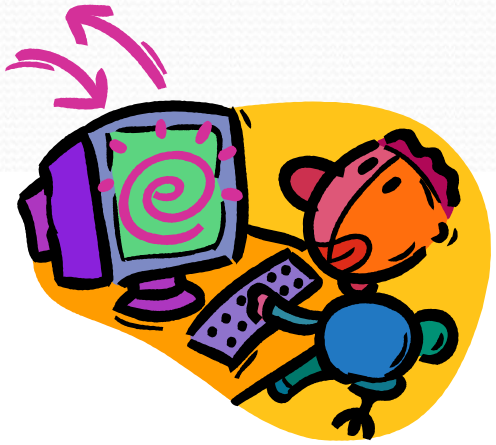


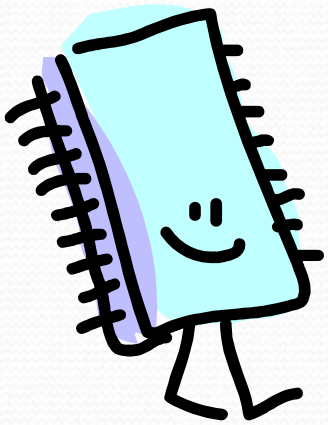
Table 11.1 I/O Techniques

	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

Evolution of the I/O Function

1. Processor directly controls a peripheral device
2. A controller or I/O module is added
3. Same configuration as step 2, but now interrupts are employed
4. The I/O module is given direct control of memory via DMA
5. The I/O module is enhanced to become a separate processor, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right

Direct Memory Access



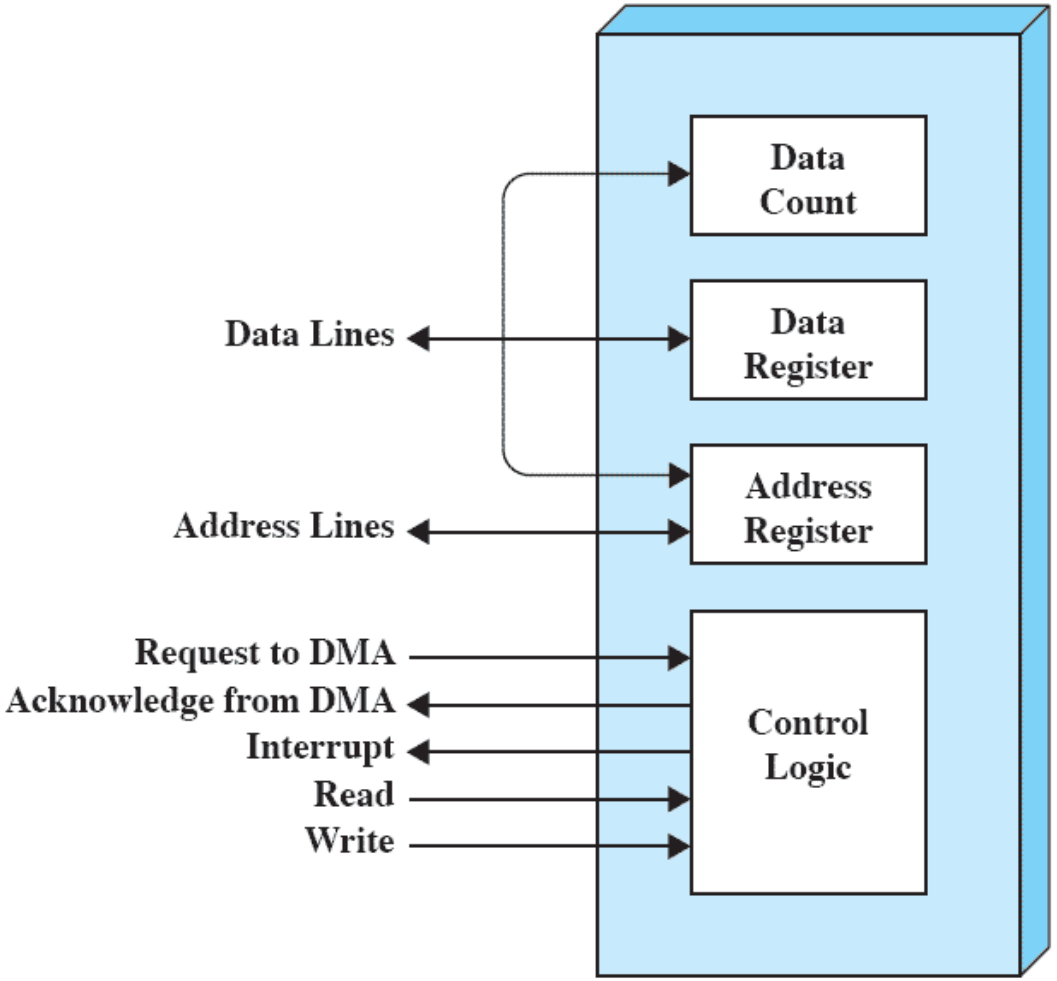
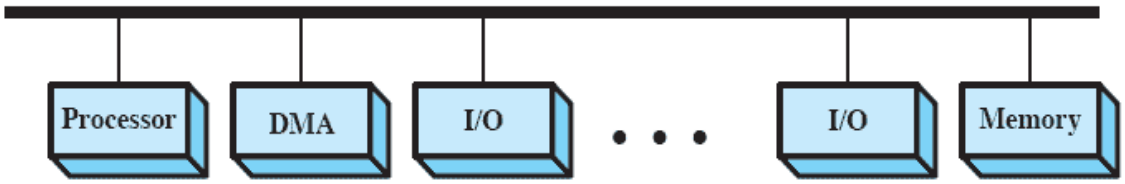


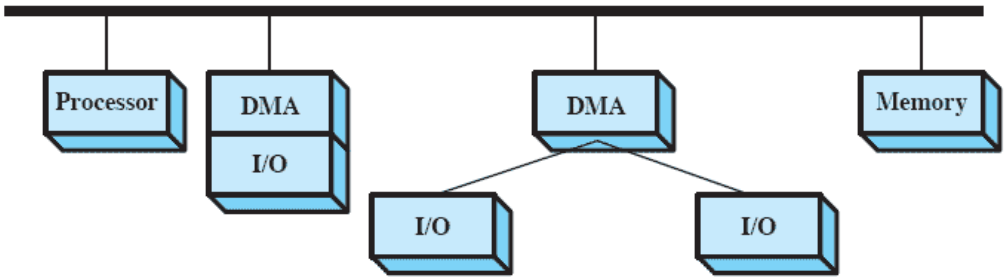
Figure 11.2 Typical DMA Block Diagram



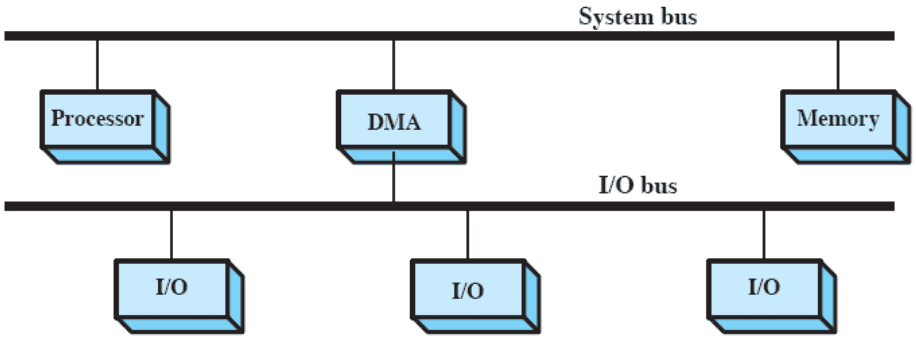
(a) Single-bus, detached DMA

DMA

Alternative



(b) Single-bus, Integrated DMA-I/O



(c) I/O bus

Configurations

Design Objectives

Efficiency

- Major effort in I/O design
- Important because I/O operations often form a bottleneck
- Most I/O devices are extremely slow compared with main memory and the processor
- The area that has received the most attention is disk I/O

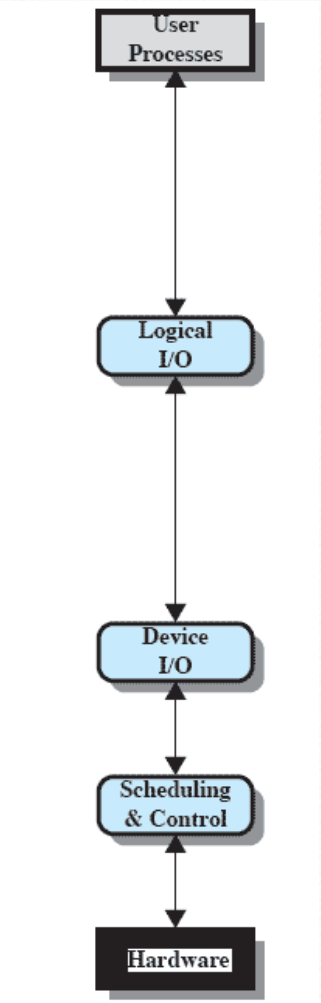
Generality

- Desirable to handle all devices in a uniform manner
- Applies to the way processes view I/O devices and the way the operating system manages I/O devices and operations
- Diversity of devices makes it difficult to achieve true generality
- Use a hierarchical, modular approach to the design of the I/O function

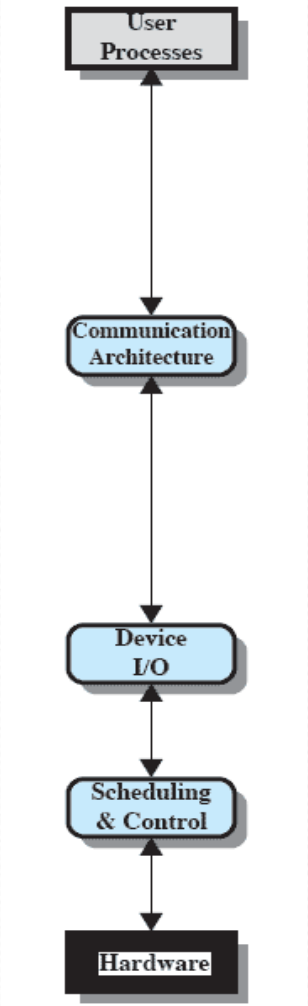
Hierarchical Design

- Functions of the operating system should be separated according to their complexity, their characteristic time scale, and their level of abstraction
- Leads to an organization of the operating system into a series of layers
- Each layer performs a related subset of the functions required of the operating system
- Layers should be defined so that changes in one layer do not require changes in other layers

A Model of I/O Organization



(a) Local peripheral device



(b) Communications port

(c) File system

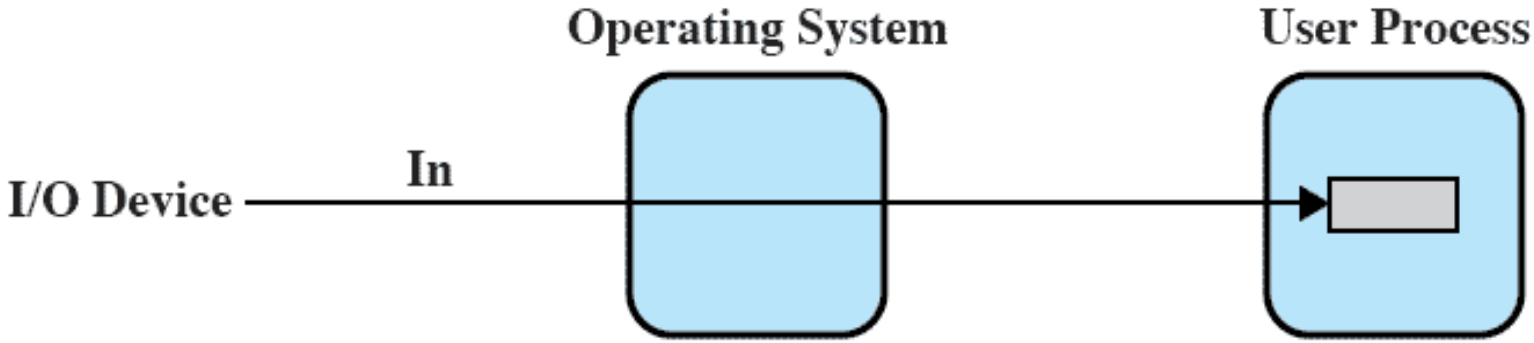
Buffering

- Perform input transfers in advance of requests being made and perform output transfers some time after the request is made

Block-oriented device	Stream-oriented device
• stores information in blocks that are usually of fixed size • transfers are made one block at a time • possible to reference data by its block number • disks and USB keys are examples	• transfers data in and out as a stream of bytes • no block structure • terminals, printers, communications ports, and most other devices that are not secondary storage are examples

No Buffer

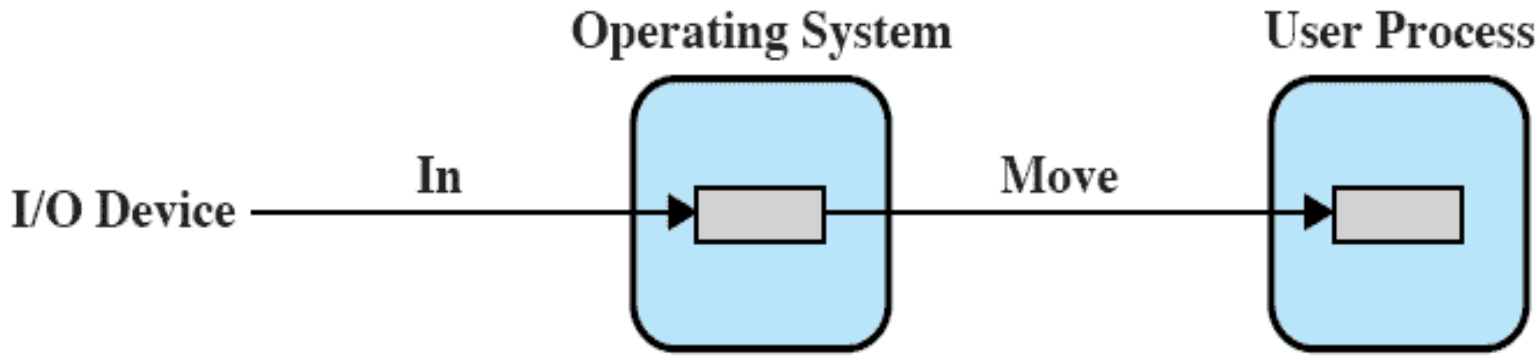
- Without a buffer, the OS directly accesses the device when it needs



(a) No buffering

Single Buffer

- Operating system assigns a buffer in main memory for an I/O request



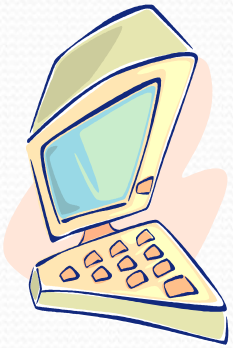
(b) Single buffering

Block-Oriented Single Buffer

- Input transfers are made to the system buffer
- Reading ahead/anticipated input
  - is done in the expectation that the block will eventually be needed
  - when the transfer is complete, the process moves the block into user space and immediately requests another block
  - Generally provides a speedup compared to the lack of system buffering
  - Disadvantages:
  - complicates the logic in the operating system
  - swapping logic is also affected

Stream-Oriented Single Buffer

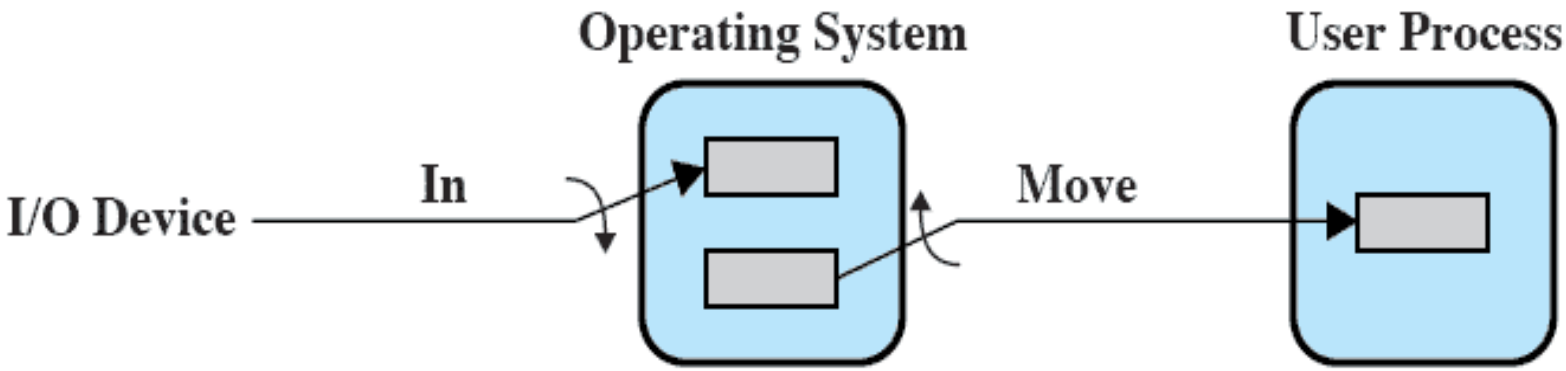
- Line-at-a-time operation
  - appropriate for scroll-mode terminals (dumb terminals)
  - user input is one line at a time with a carriage return signaling the end of a line
  - output to the terminal is similarly one line at a time



- Byte-at-a-time operation
- used on forms-mode terminals
- when each keystroke is significant
- other peripherals such as sensors and controllers

Double Buffer

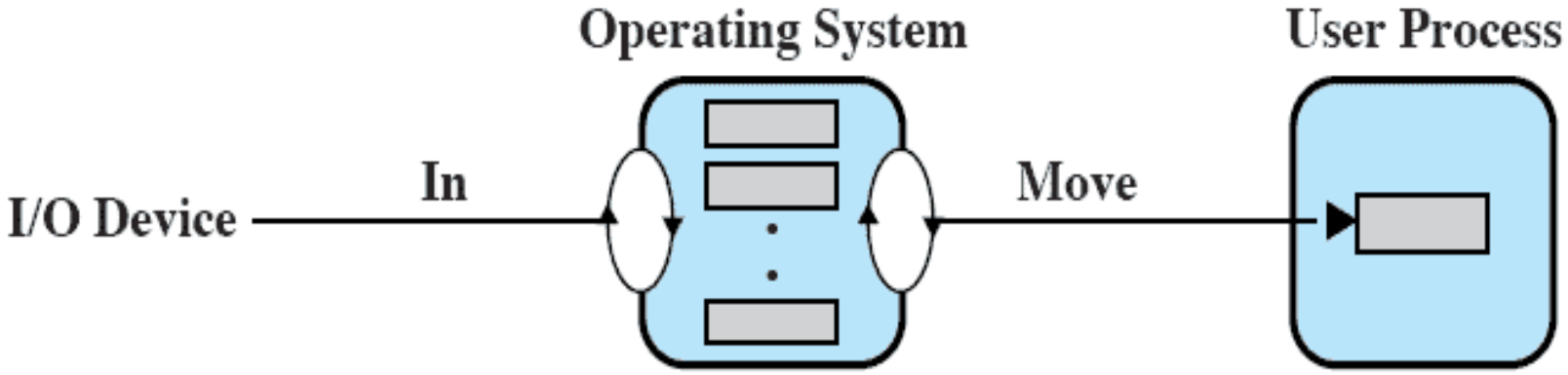
- Use two system buffers instead of one
- A process can transfer data to or from one buffer while the operating system empties or fills the other buffer
- Also known as buffer swapping



(c) Double buffering

Circular Buffer

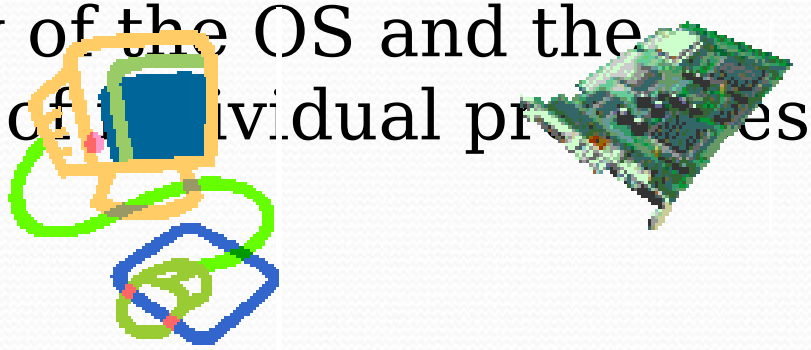
- Two or more buffers are used
- Each individual buffer is one unit in a circular buffer
- Used when I/O operation must keep up with process



(d) Circular buffering

The Utility of Buffering

- Technique that smoothes out peaks in I/O demand
  - with enough demand eventually all buffers become full and their advantage is lost
- When there is a variety of I/O and process activities to service, buffering can increase the efficiency of the OS and the performance of individual processes



Disk Performance Parameters

- The actual details of disk I/O operation depend on the:
  - computer system
  - operating system
  - nature of the I/O channel and disk controller hardware

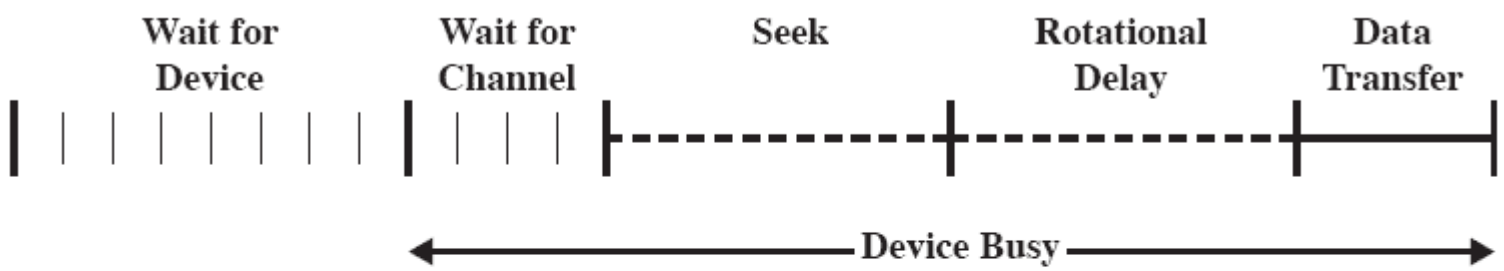


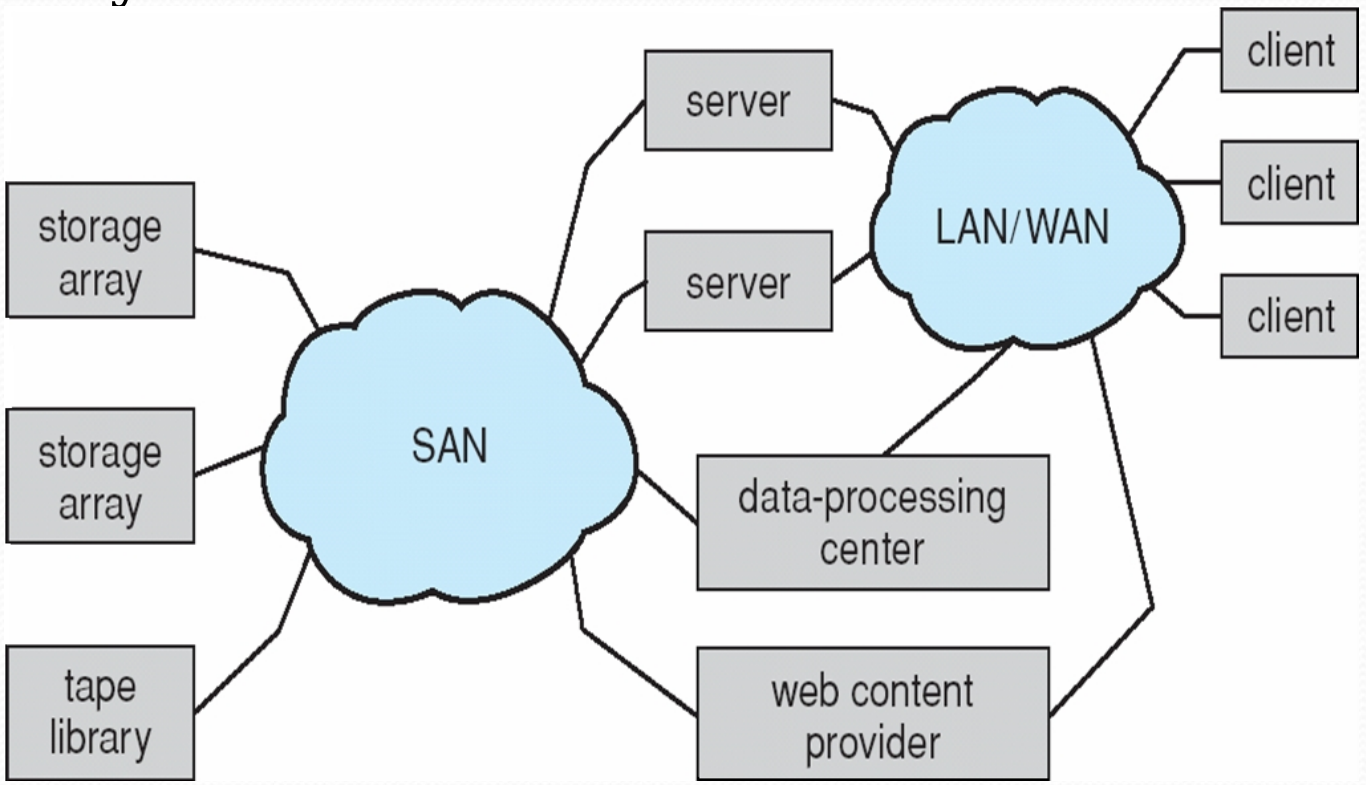
Figure 11.6 Timing of a Disk I/O Transfer

Positioning the Read/Write Heads

- When the disk drive is operating, the disk is rotating at constant speed
- To read or write the head must be positioned at the desired track and at the beginning of the desired sector on that track
- Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed-head system
- On a movable-head system the time it takes to position the head at the track is known as **seek time**
- The time it takes for the beginning of the sector to reach the head is known as **rotational delay**
- The sum of the seek time and the rotational delay equals the **access time**

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 (Cont.)

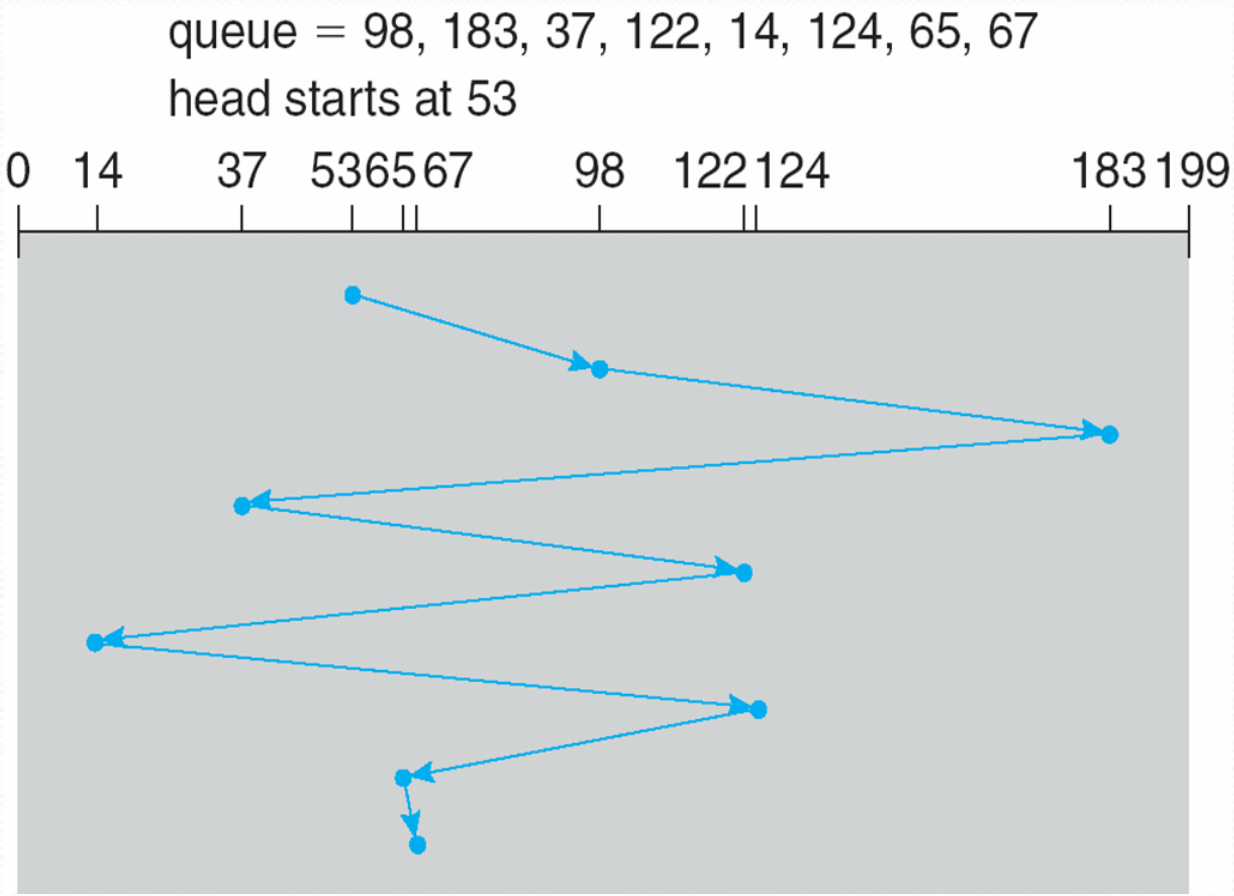
- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate them with a request queue (0-199)

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

Head pointer 53

FCFS

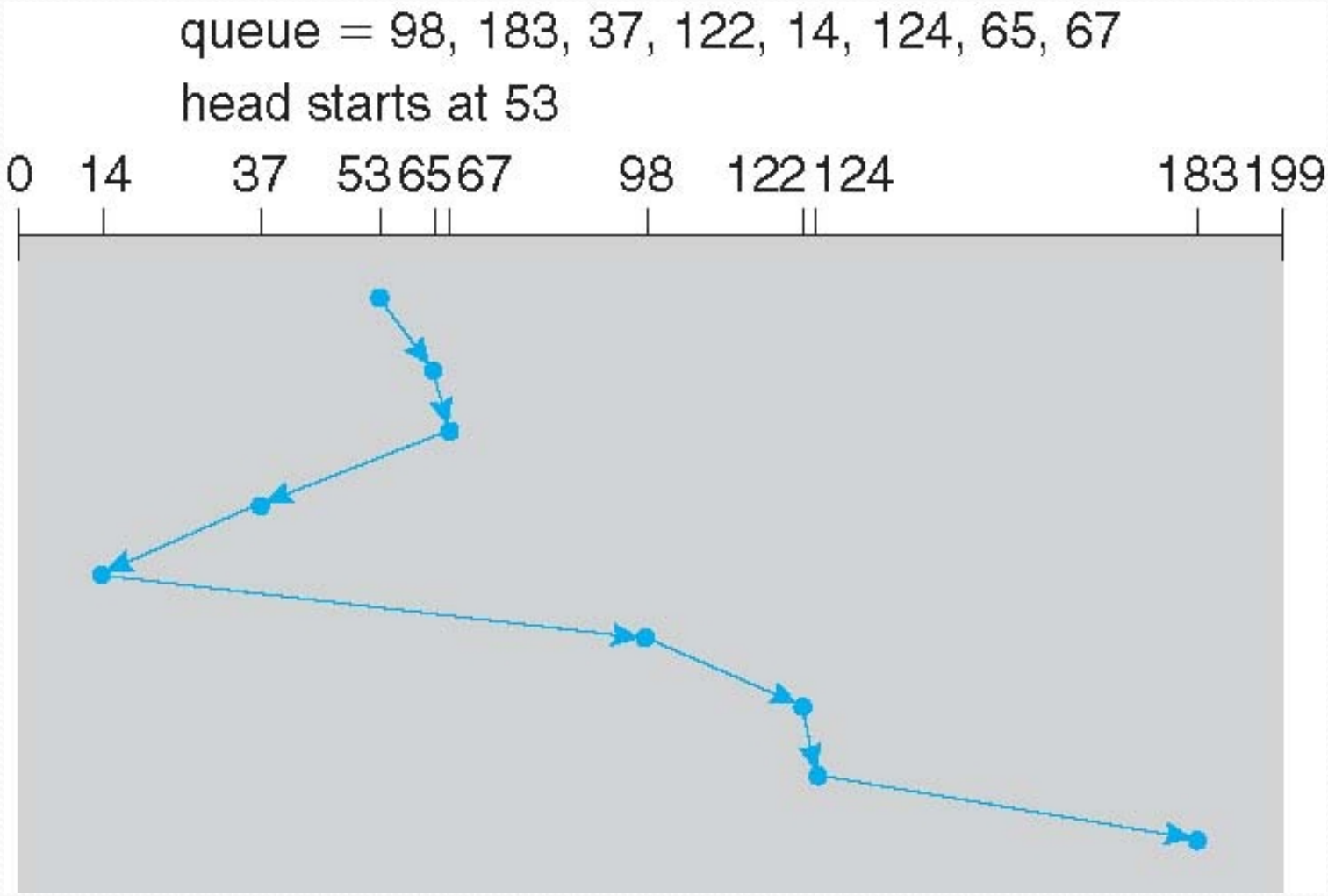
Illustration shows total head movement of 640 cylinders



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

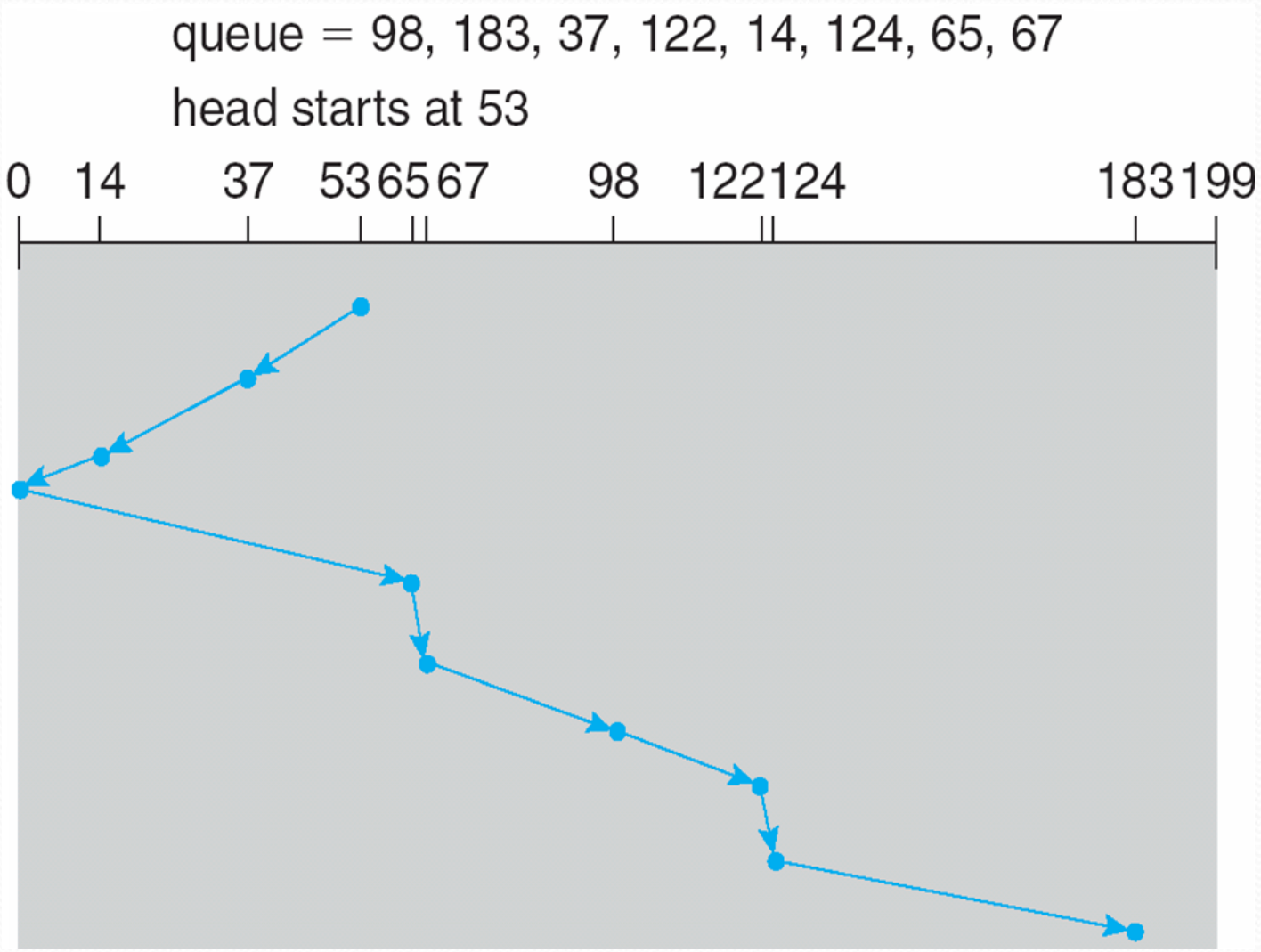
SSTF (Cont.)


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

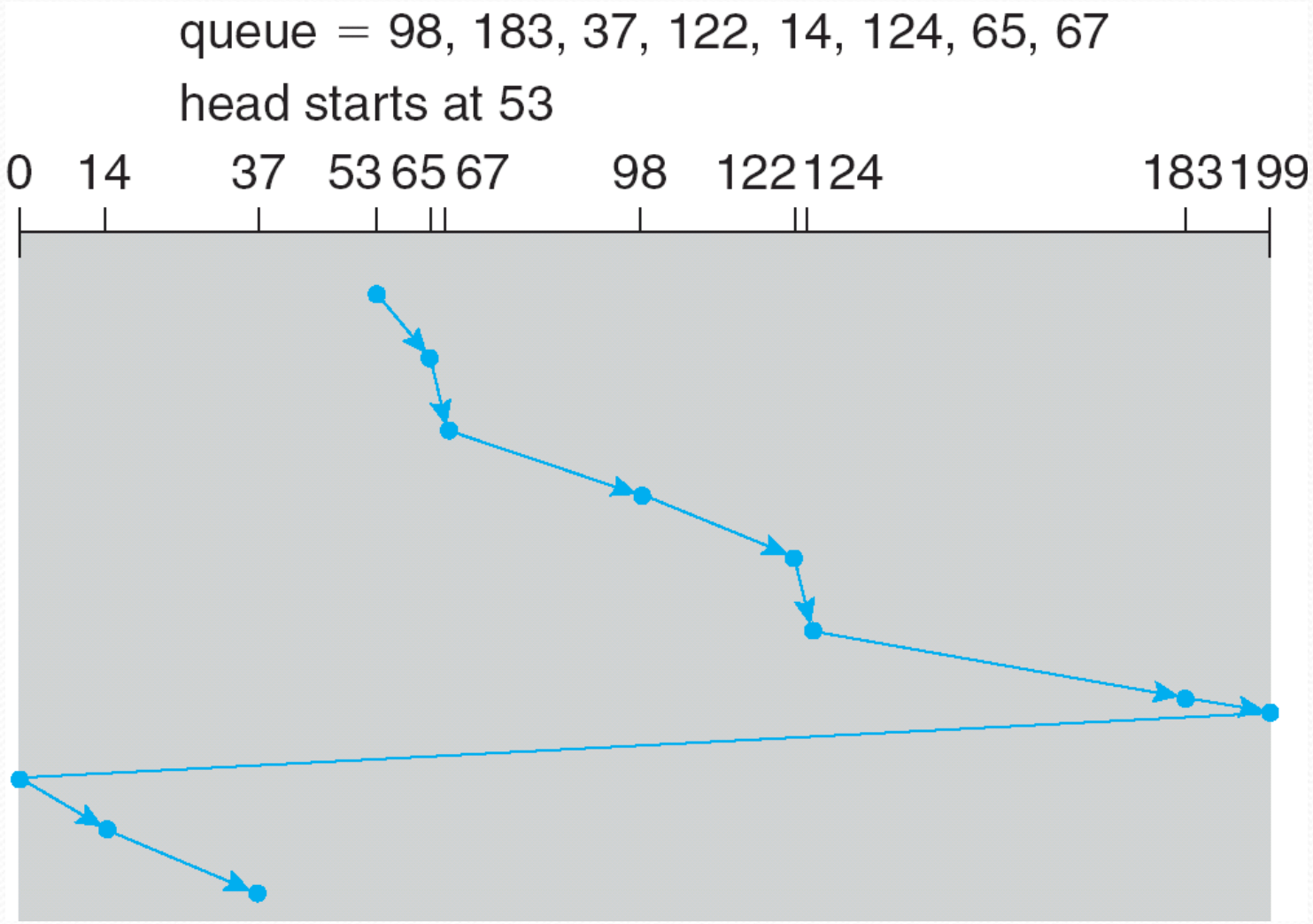
SCAN (Cont.)


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

- 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

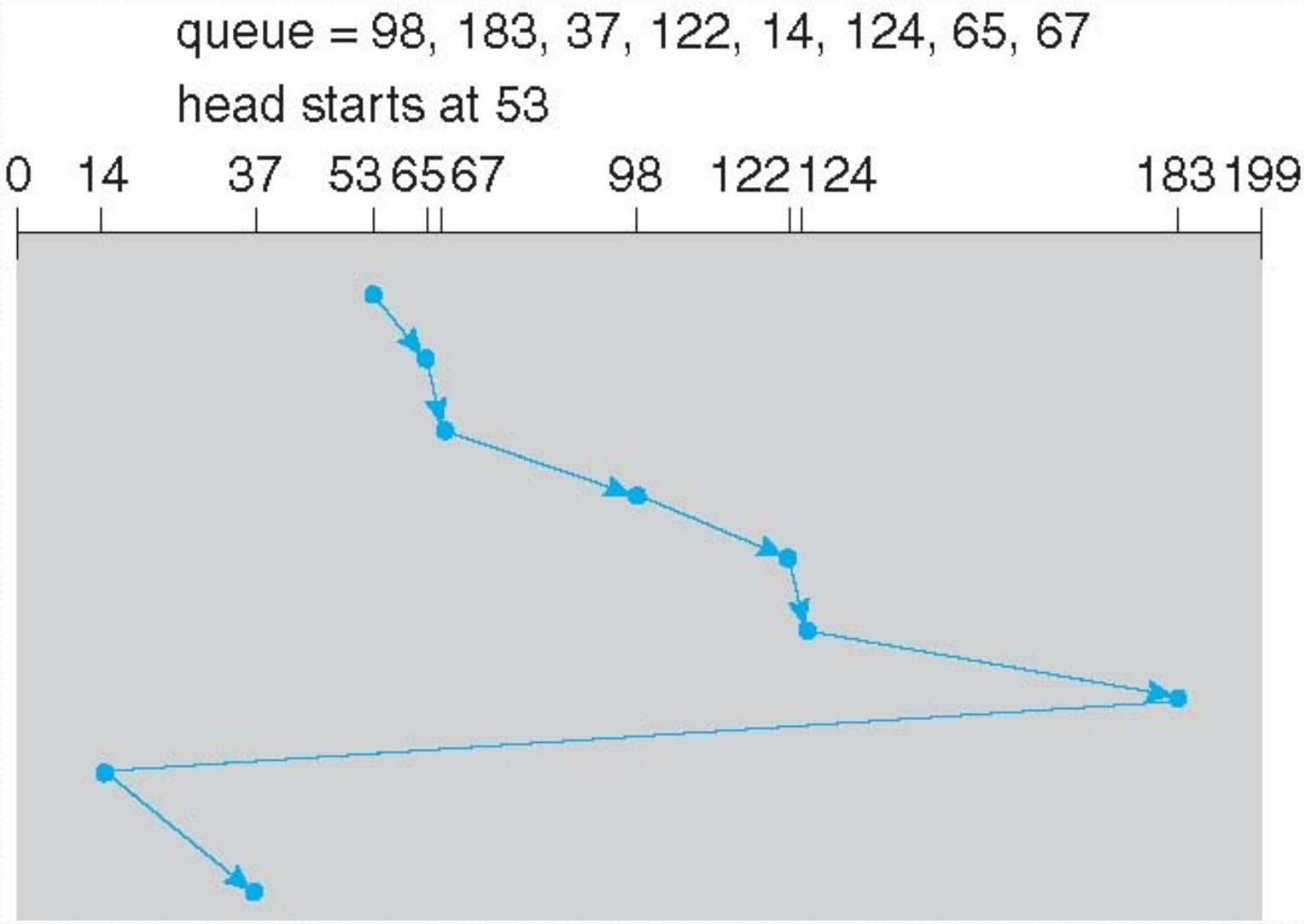
C-SCAN (Cont.)


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
- 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

Name	Description	Remarks
Selection according to requestor		
RSS	Random scheduling	For analysis and simulation
FIFO	First in first out	Fairest of them all
PRI	Priority by process	Control outside of disk queue management
LIFO	Last in first out	Maximize locality and resource utilization
Selection according to requested item		
SSTF	Shortest service time first	High utilization, small queues
SCAN	Back and forth over disk	Better service distribution
C-SCAN	One way with fast return	Lower service variability
N-step-SCAN	SCAN of *N* records at a time	Service guarantee
FSCAN	N-step-SCAN with *N* = queue size at beginning of SCAN cycle	Load sensitive

Table : Disk Scheduling Algorithms

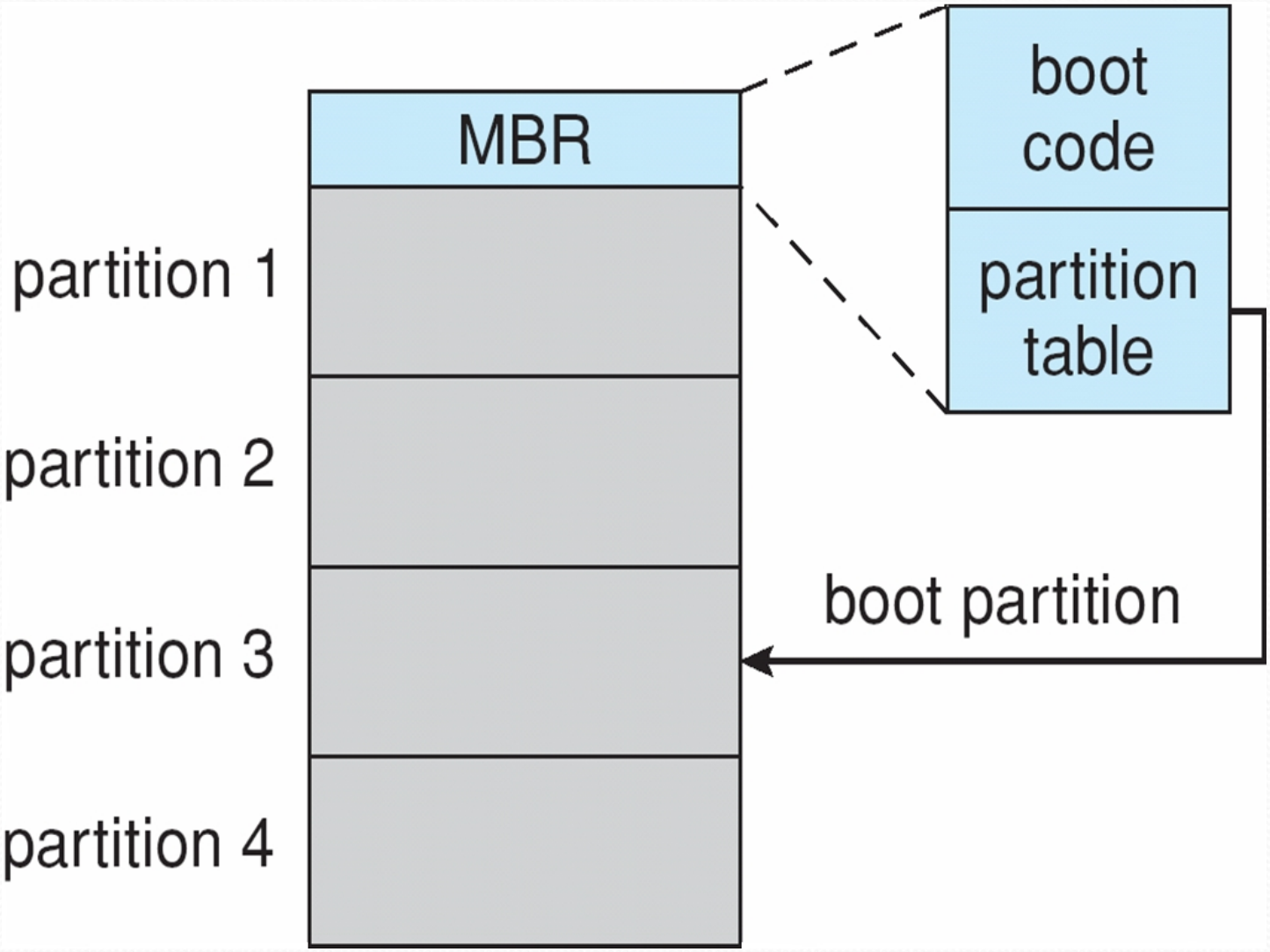
(a) FIFO (starting at track 100)		(b) SSTF (starting at track 100)		(c) SCAN (starting at track 100, in the direction of increasing track number)		(d) C-SCAN (starting at track 100, in the direction of increasing track number)	
Next track accessed	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**	**Next track accessed**	**Number of tracks traversed**
55	45	90	10	150	50	150	50
58	3	58	32	160	10	160	10
39	19	55	3	184	24	184	24
18	21	39	16	90	94	18	166
90	72	38	1	58	32	38	20
160	70	18	20	55	3	39	1
150	10	150	132	39	16	55	16
38	112	160	10	38	1	58	3
184	146	184	24	18	20	90	32
Average seek length	55.3	**Average seek length**	27.5	**Average seek length**	27.8	**Average seek length**	35.8

Table : Comparison of Disk Scheduling Algorithms

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”
  - To increase efficiency most file systems group blocks into **clusters**
    - Disk I/O done in blocks
    - File I/O done in clusters
- Boot block initializes system
  - The bootstrap is stored in ROM
  - **Bootstrap loader** program
- Methods such as **sector sparing** used to handle bad blocks

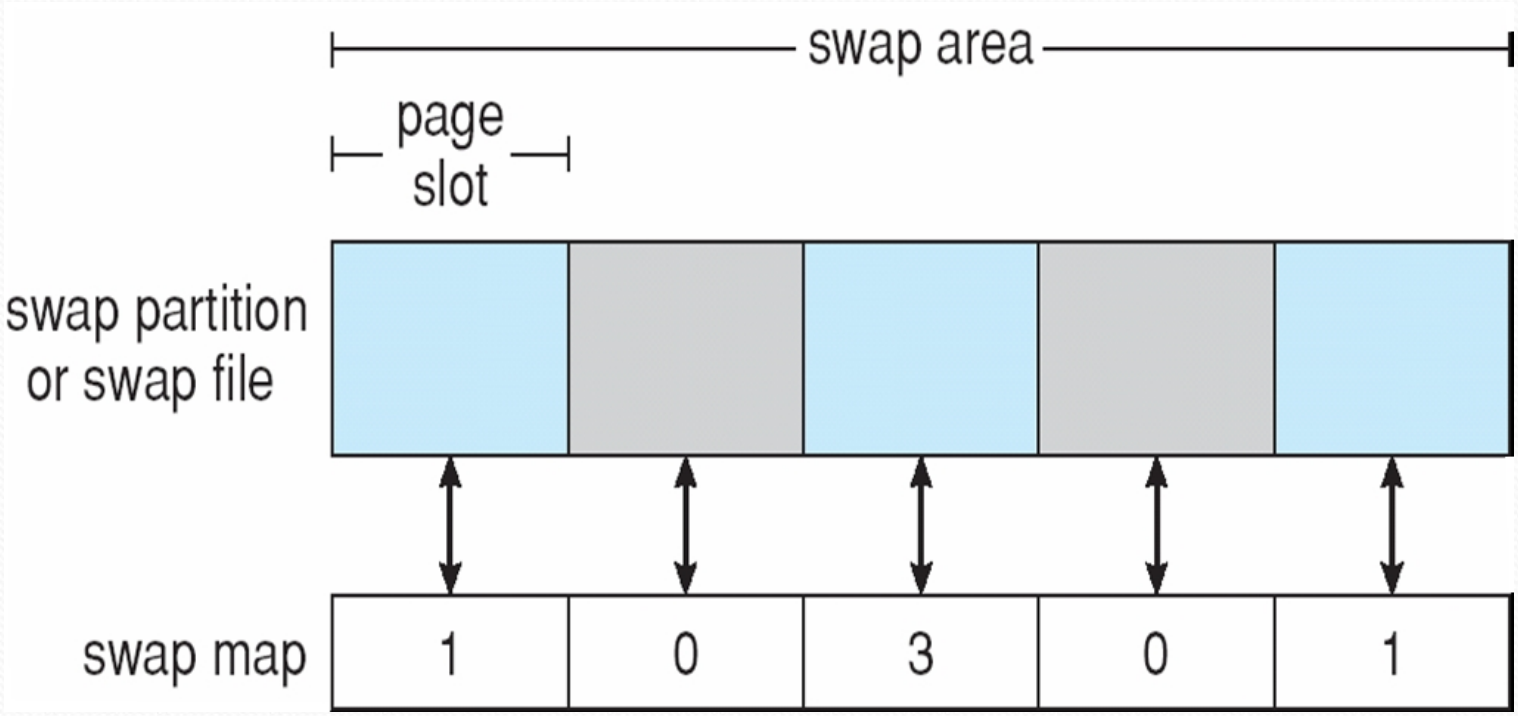
Booting from a Disk in Windows 2000



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

Data Structures for Swapping on Linux Systems



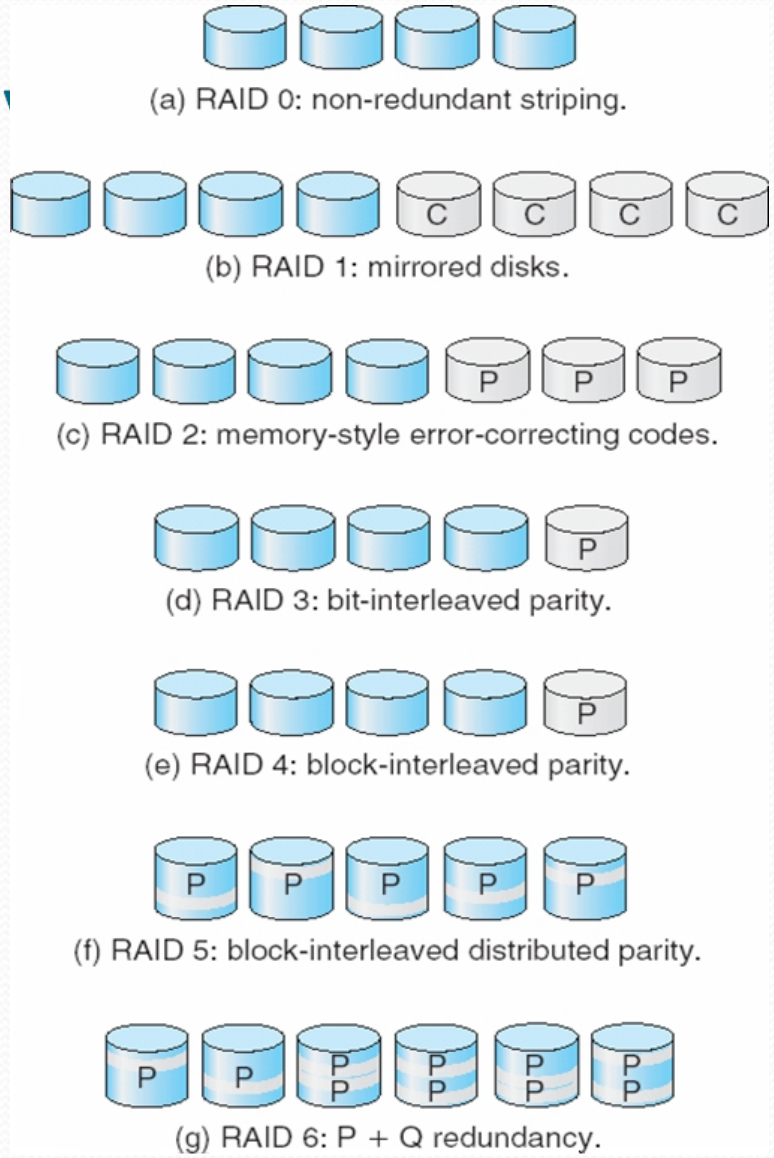
RAID Structure

- RAID – multiple disk drives provides reliability via **redundancy**
- Increases the **mean time to failure**
- Frequently combined with **NVRAM** to improve write performance
- RAID is arranged into six different levels

RAID (Cont.)

- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively
- Disk **striping** uses a group of disks as one storage unit
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  - **Mirroring** or **shadowing** (**RAID 1**) keeps duplicate of each disk
  - Striped mirrors (**RAID 1+0**) or mirrored stripes (**RAID 0+1**) provides high performance and high reliability
  - **Block interleaved parity** (**RAID 4, 5, 6**) uses much less redundancy
- RAID within a storage array can still fail if the array fails, so automatic **replication** of the data between arrays is common
- Frequently, a small number of **hot-spare** disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them

RAID Le



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

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

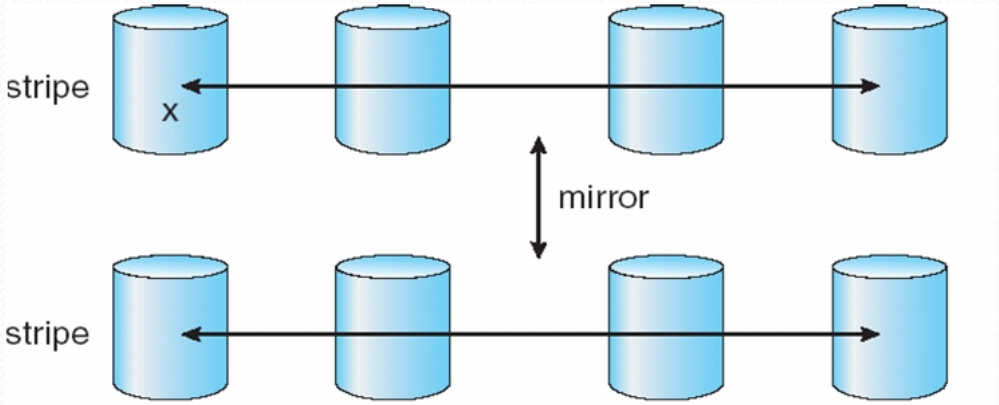
(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

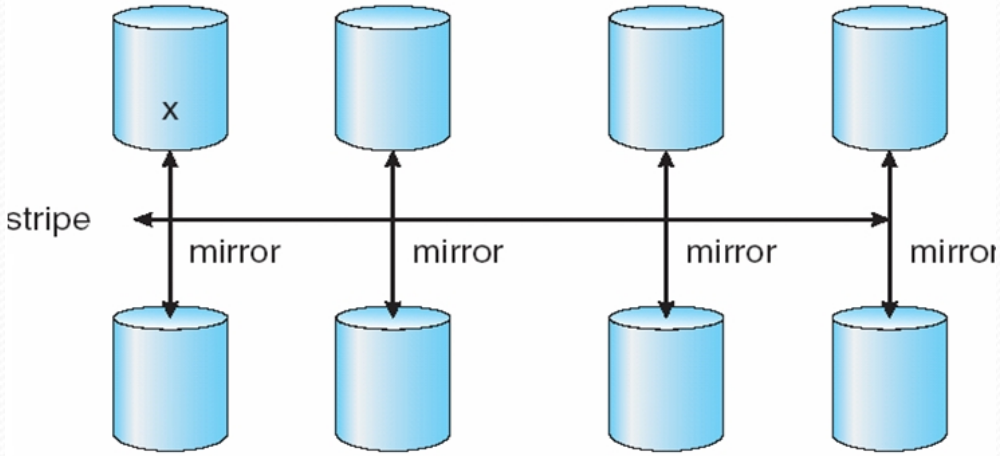
(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.

RAID (0 + 1) and (1 + 0)



a) RAID 0 + 1 with a single disk failure.

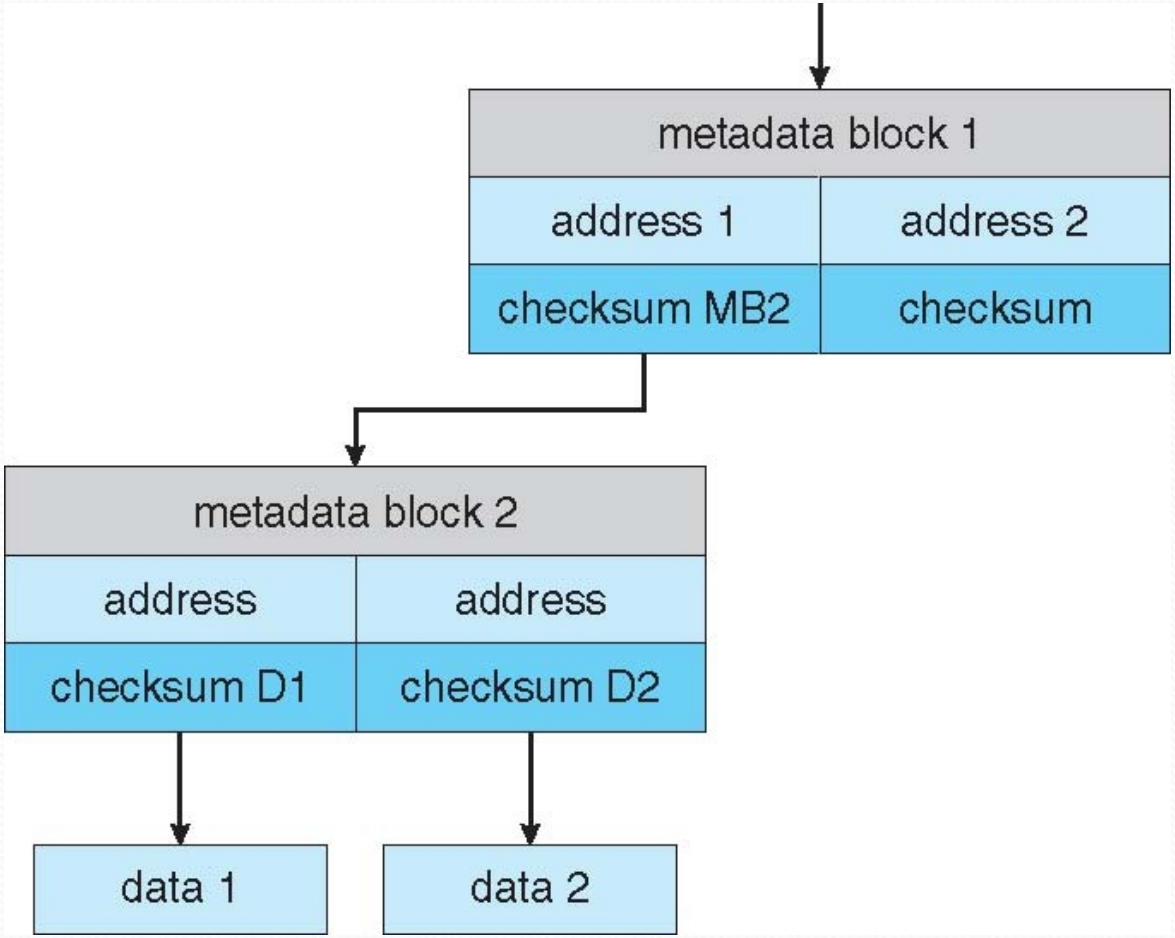


b) RAID 1 + 0 with a single disk failure.

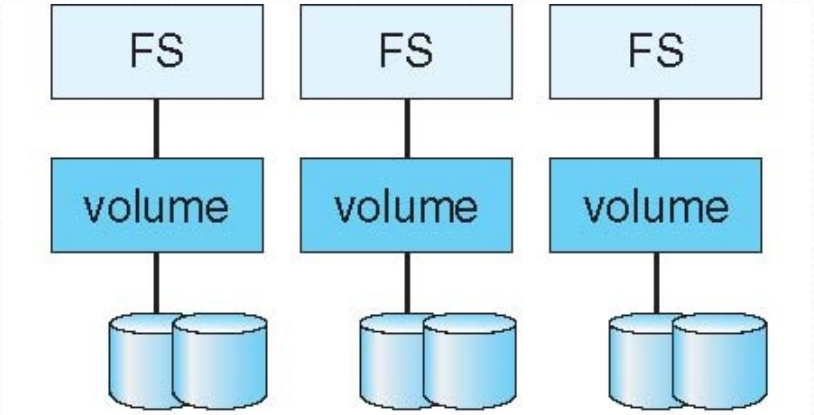
Extensions

- RAID alone does not prevent or detect data corruption or other errors, just disk failures
- Solaris ZFS adds **checksums** of all data and metadata
- Checksums kept with pointer to object, to detect if object is the right one and whether it changed
- Can detect and correct data and metadata corruption
- ZFS also removes volumes, partitions
  - Disks allocated in **pools**
  - Filesystems with a pool share that pool, use and release space like “malloc” and “free” memory allocate / release calls

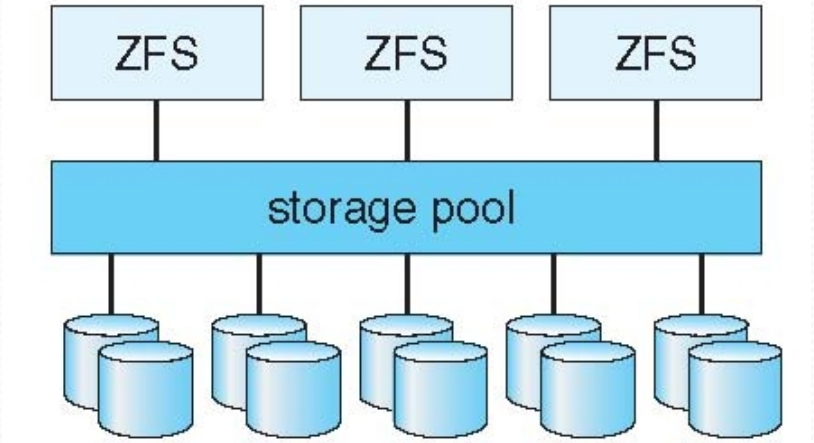
ZFS Checksums All Metadata and Data



Traditional and Pooled Storage



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

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

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 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 ad 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.

Operating System Support

- Major OS jobs are to manage physical devices and to present a virtual machine abstraction to applications
- For hard disks, the OS provides two abstraction:
  - 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 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

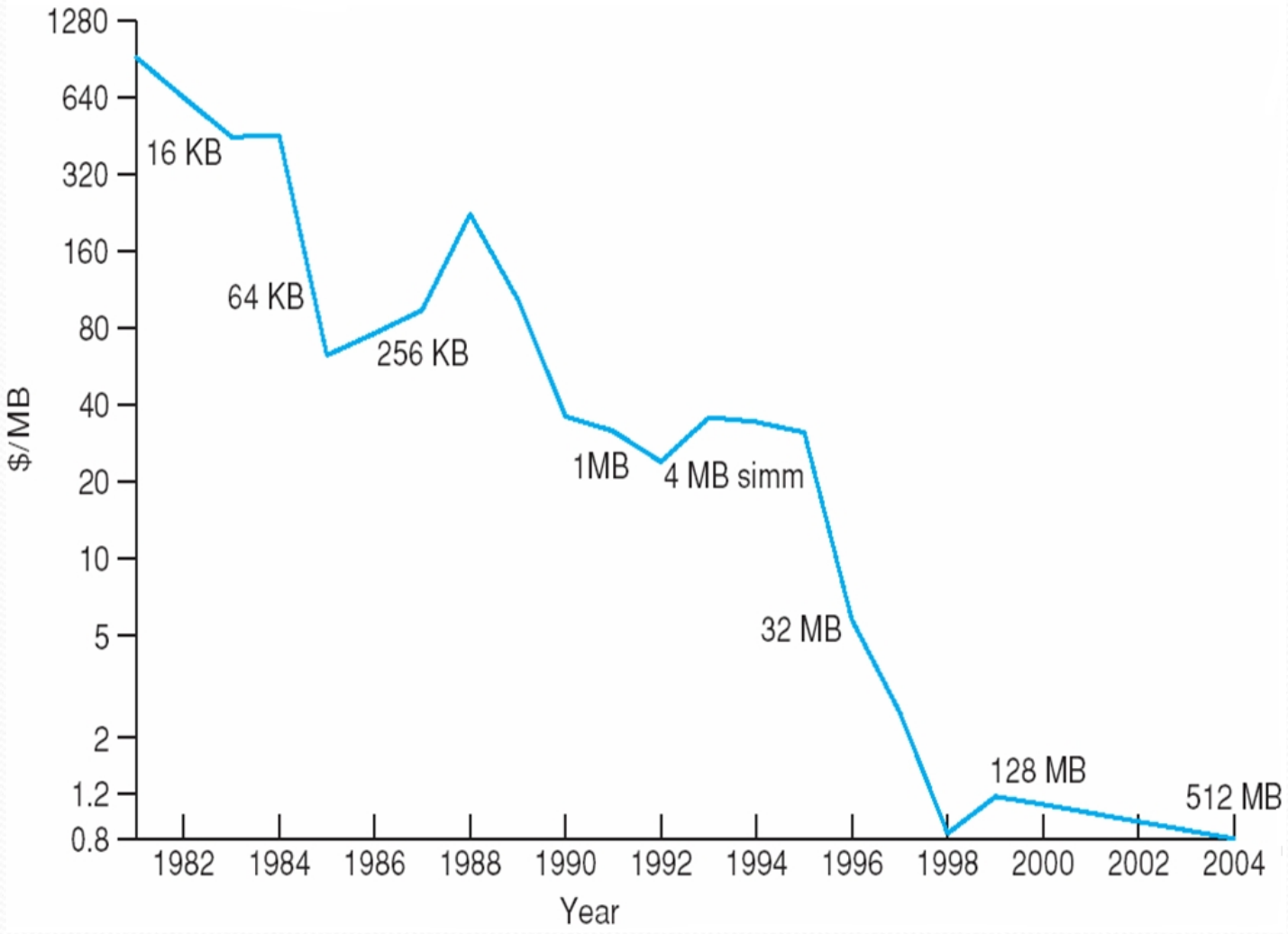
Reliability

- 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

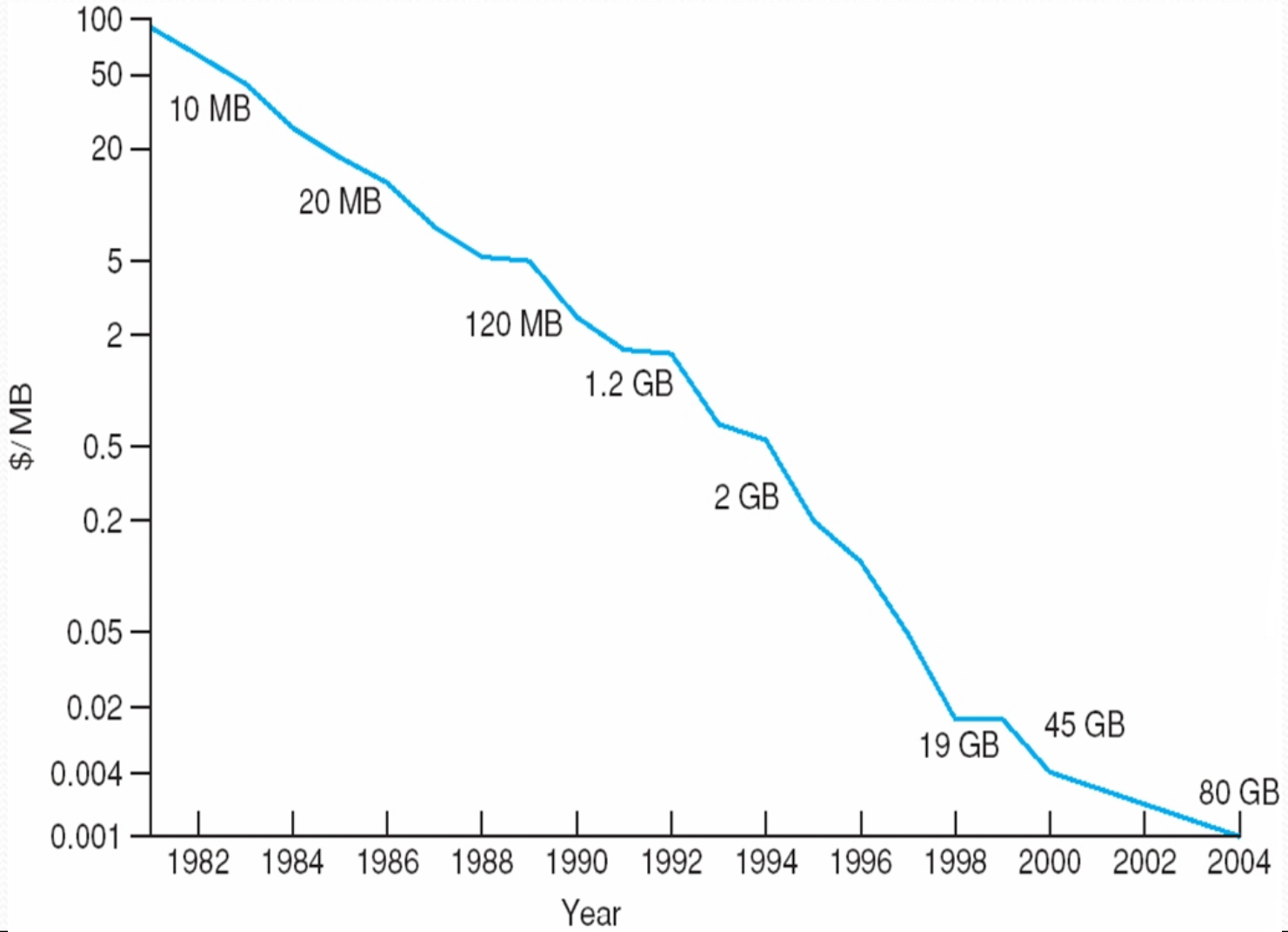
Cost

- 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

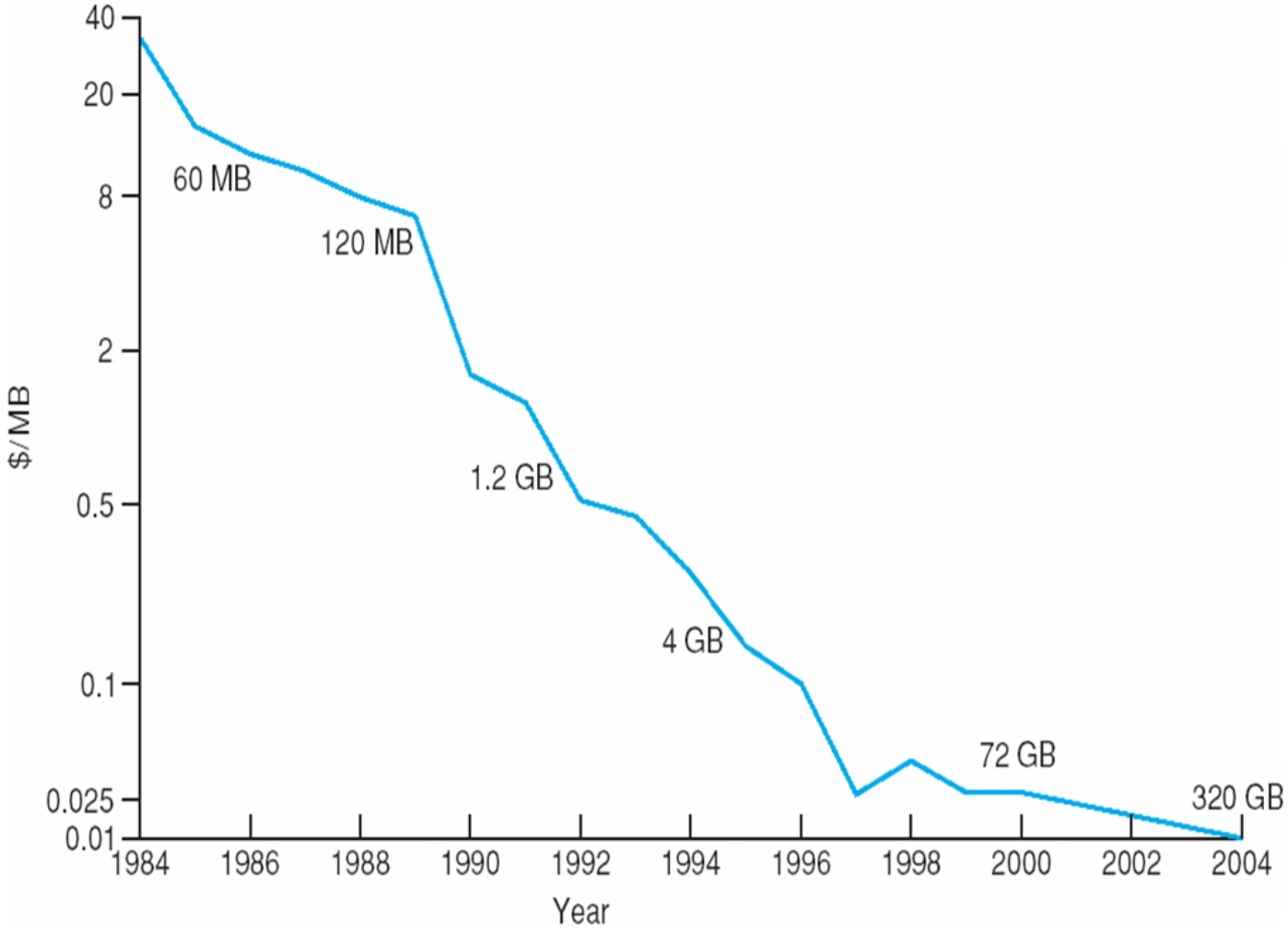
Price per Megabyte of DRAM From 1981 to 2004



Price per Megabyte of Magnetic Hard Disk From 1981 to 2004



Price per Megabyte of a Tape Drive From 1984-2000



End of Mass Storage System