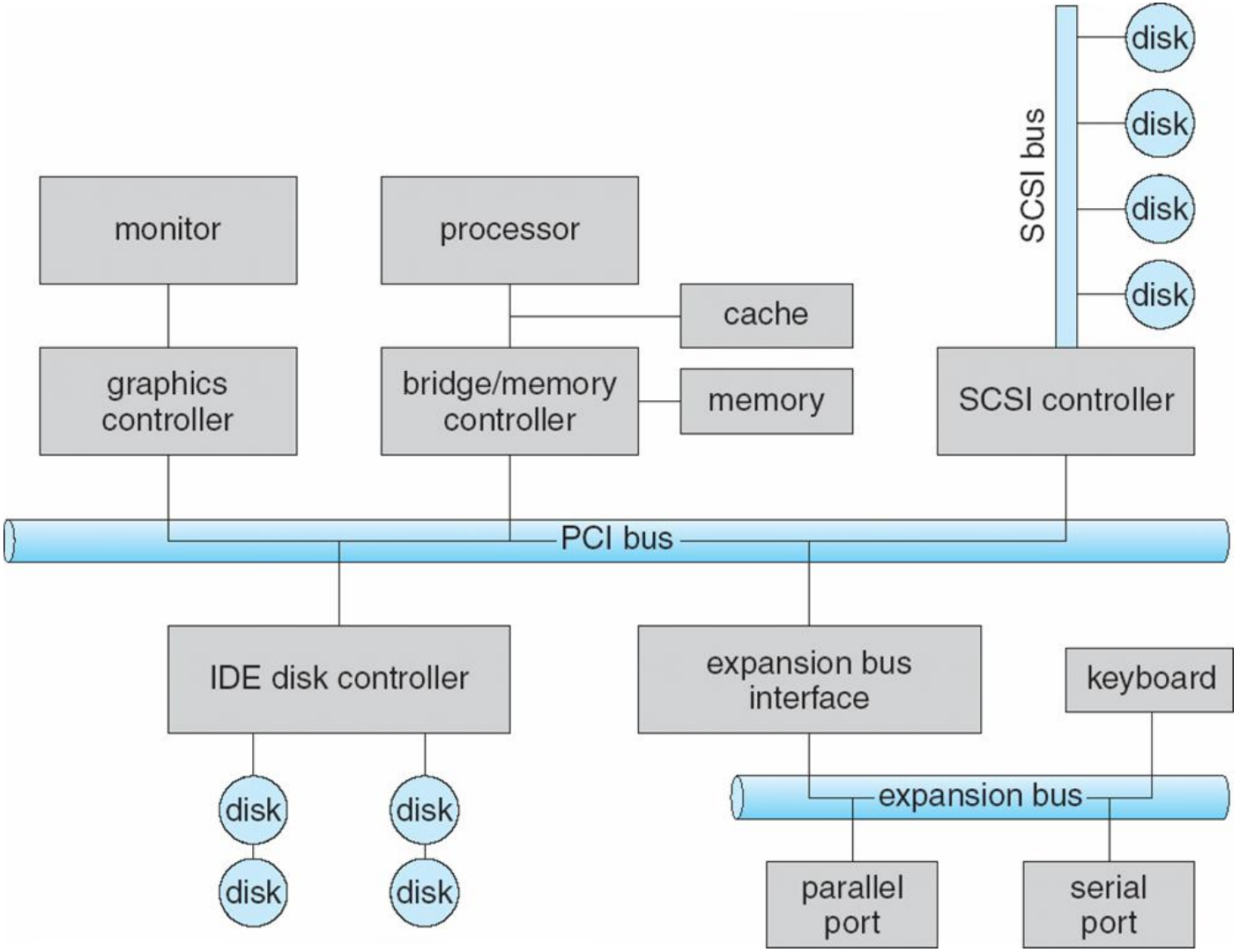
Part 9: I/O System & Disk

- I/O Hardware
- I/O System Design and Implementation
  - Design Objectives
  - Kernel I/O Structure
  - Kernel I/O Subsystem
  - Performance Consideration
- Disk
  - Disk Structure
  - Disk Scheduling
    - *FCFS, SSTF, SCAN, C-SCAN, C-LOOK*
  - Disk Management
  - Disk Reliability

I/O Hardware

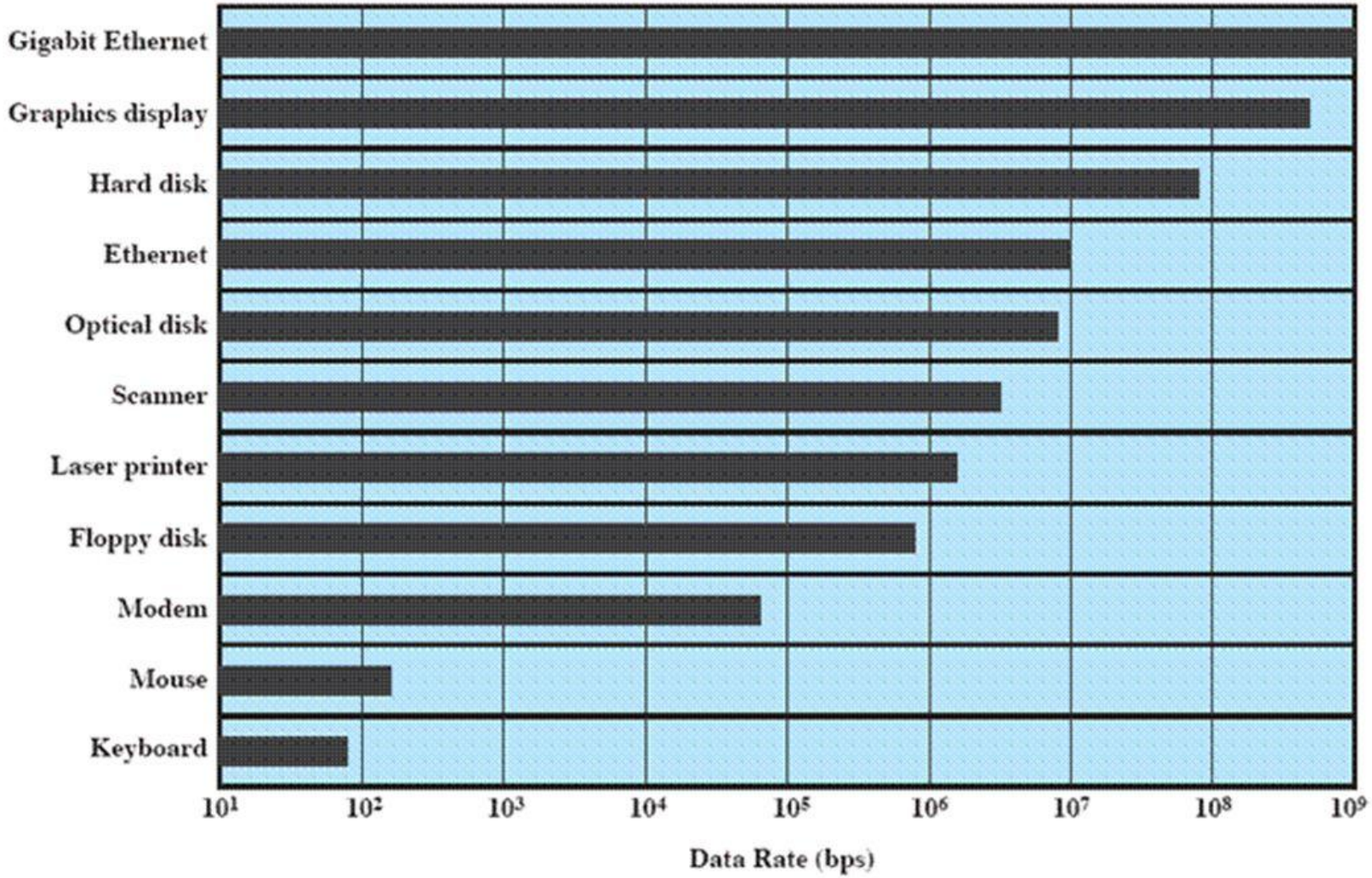

disk
disk
disk
disk
SCSI bus
monitor
processor
cache
graphics controller
bridge/memory controller
memory
SCSI controller
PCI bus
IDE disk controller
expansion bus interface
keyboard
disk
disk
disk
disk
expansion bus
parallel port
serial port

I/O Hardware

- Devices vary in many dimensions

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

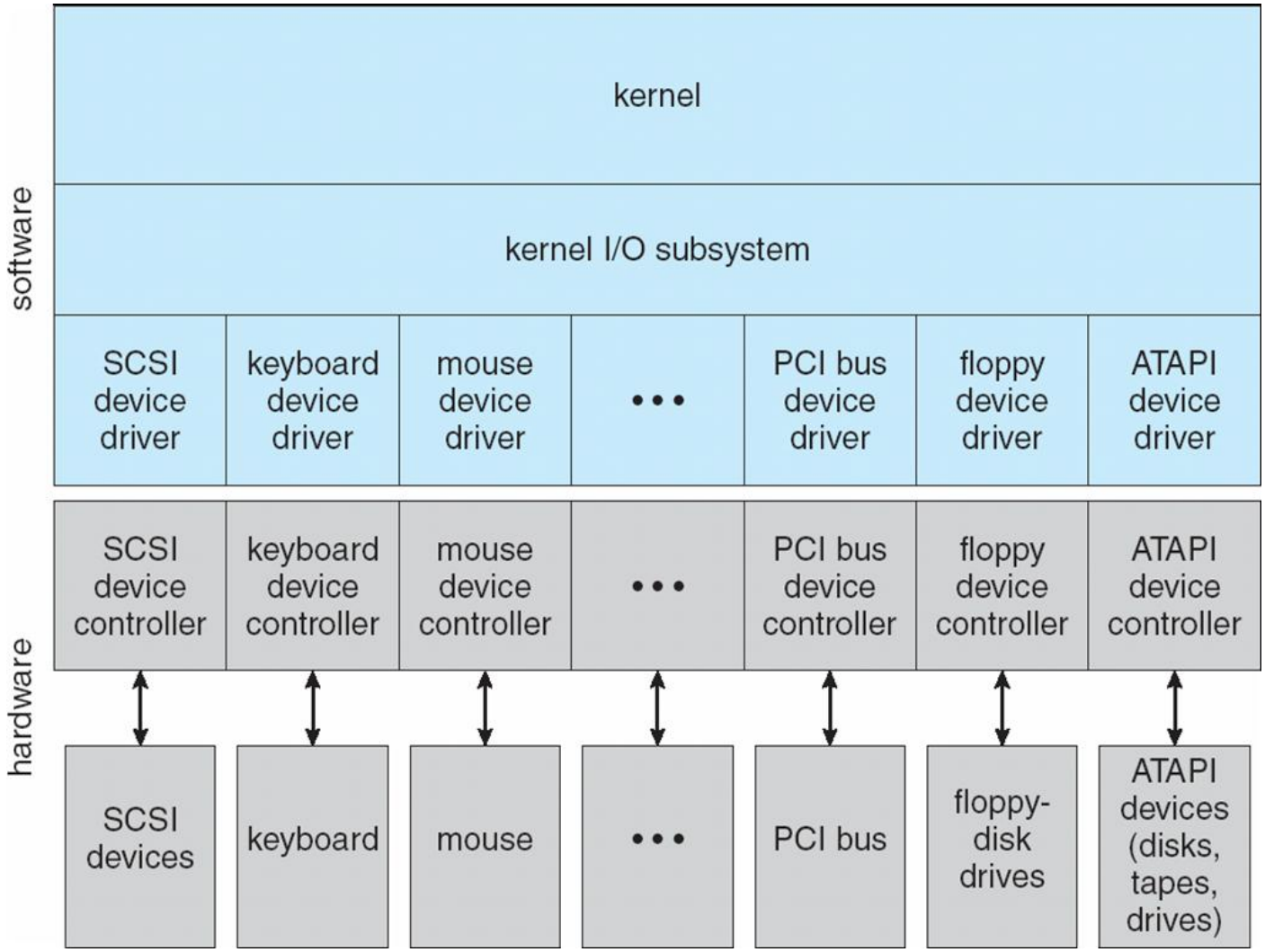
I/O Hardware


Gigabit Ethernet
Graphics display
Hard disk
Ethernet
Optical disk
Scanner
Laser printer
Floppy disk
Modem
Mouse
Keyboard
10¹
10²
10³
10⁴
10⁵
10⁶
10⁷
10⁸
10⁹
Data Rate (bps)

I/O System Design Objectives

- Efficiency
  - 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

Kernel I/O Structure



Kernel I/O Subsystem

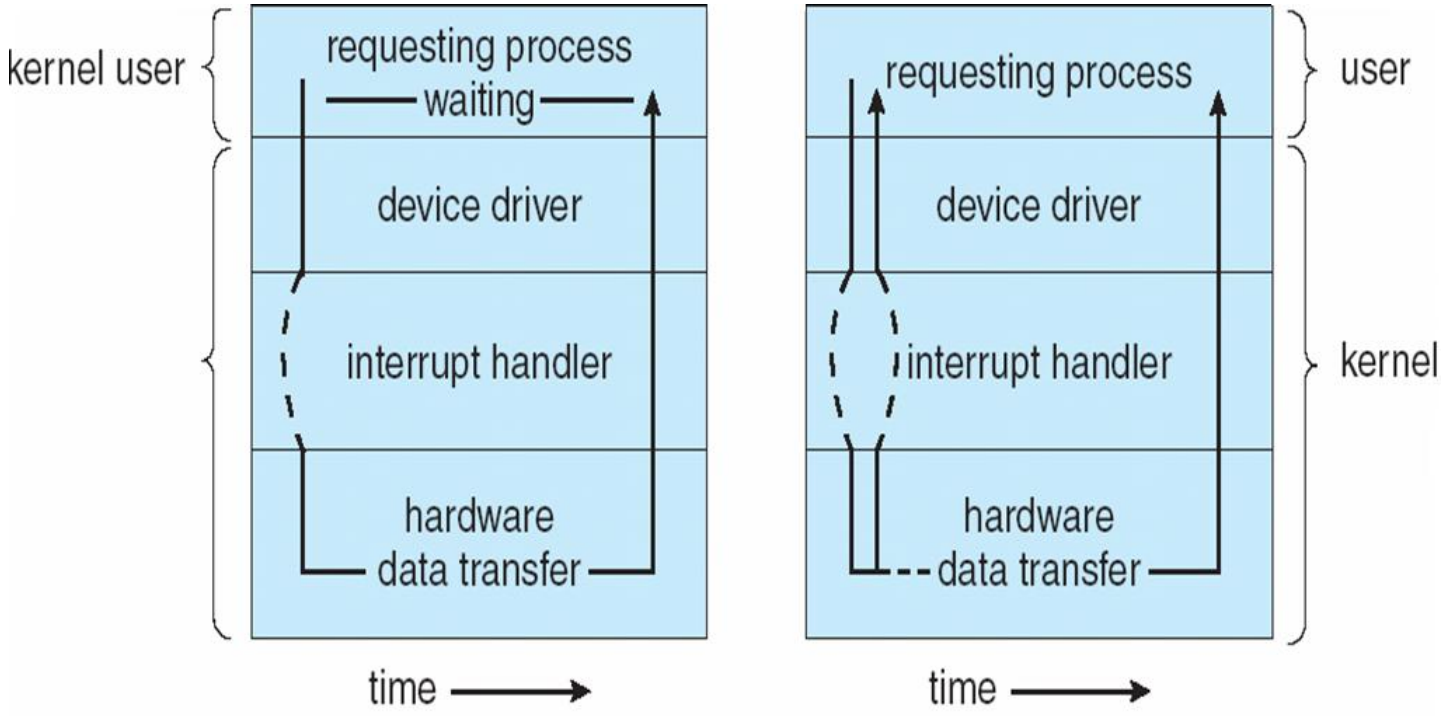
- I/O Scheduling
  - Schedule I/O request through rearranging the order of service, e.g. disk scheduling
  - Can improve the system efficiency
- Buffering
  - Store data in memory while transferring between devices
  - To cope with device speed mismatch
  - To cope with device transfer size mismatch, e.g. in networking

Kernel I/O Subsystem

- Caching
  - Fast memory holding copy of data
  - To improve I/O efficiency for files that are being written and reread rapidly.
- Spooling
  - Hold output for a device
  - Used in device that can serve only one request at a time, e.g. printing

Performance Consideration

- Two I/O Methods

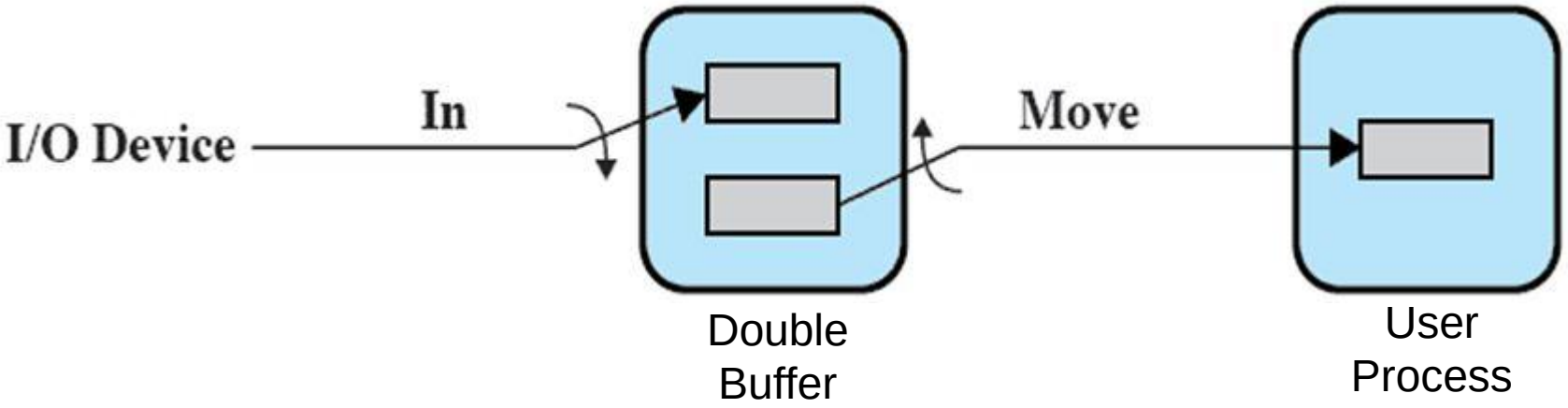


Synchronous / Blocking I/O

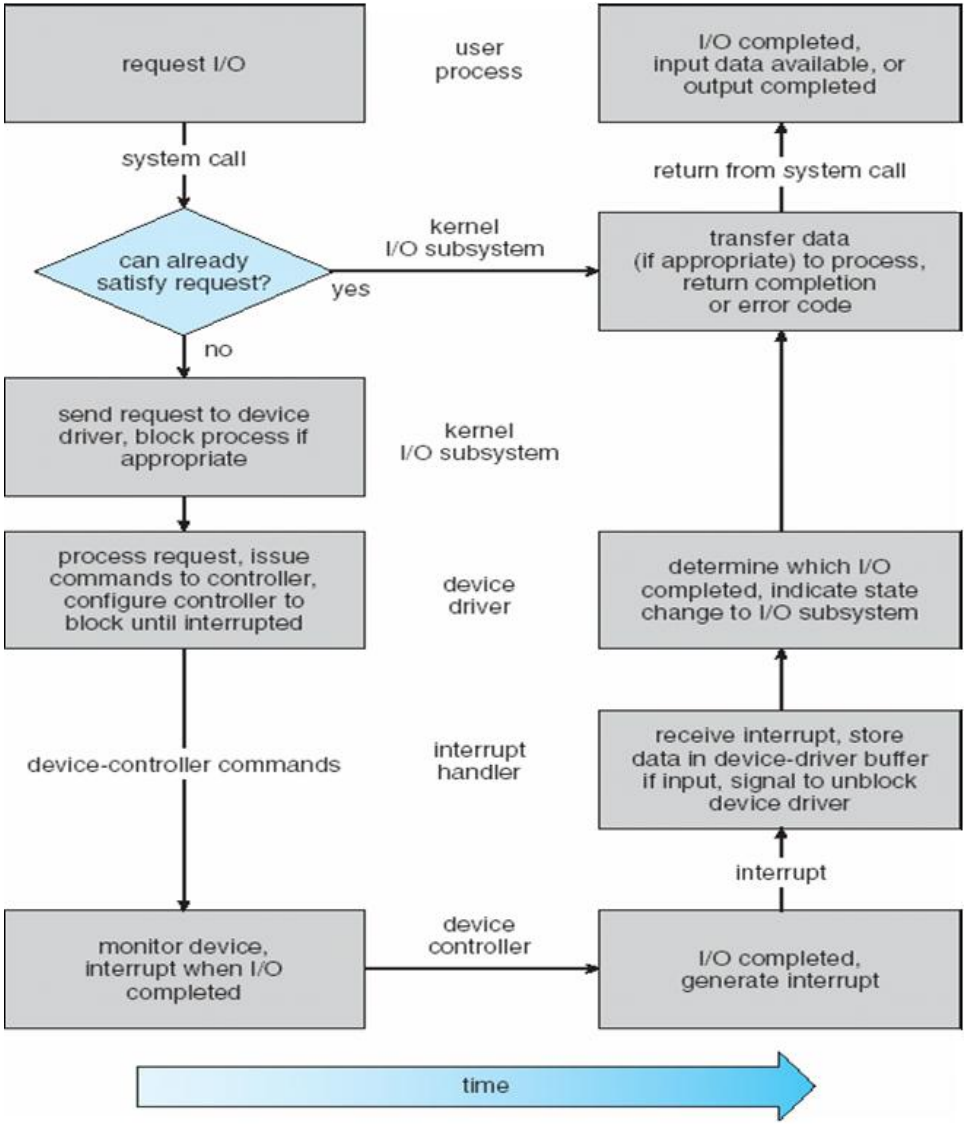
Asynchronous / Non-blocking I/O

Performance Consideration

- Using asynchronous I/O and double-buffer to improve performance



Performance Consideration



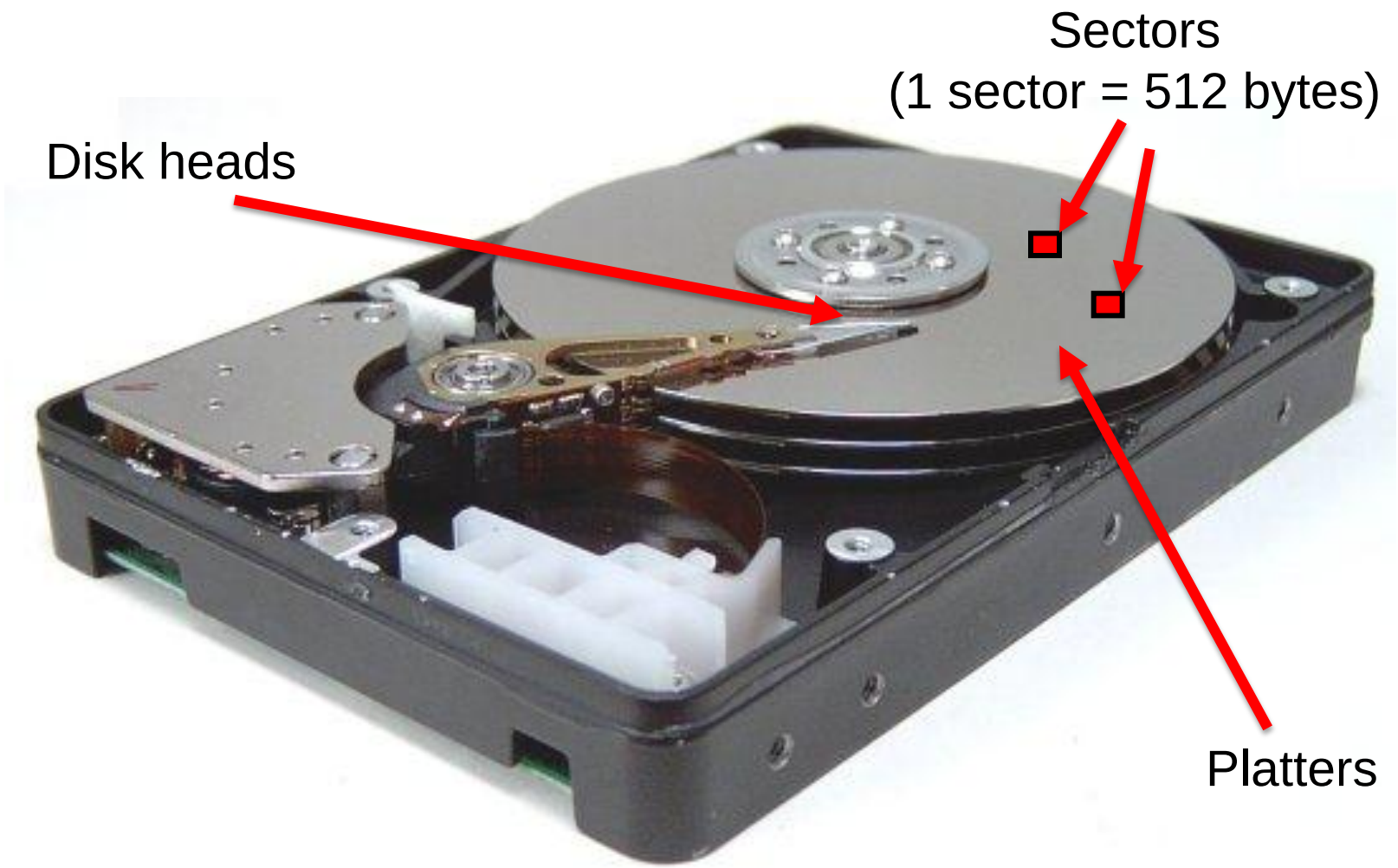
Performance Consideration

- I/O is a major factor in system performance
  - It demands CPU to execute device driver code
  - Context switches due to interrupts may stress CPU
  - Data copying between I/O controllers and physical memory also affect system performance.

Part 7: I/O System & Disk

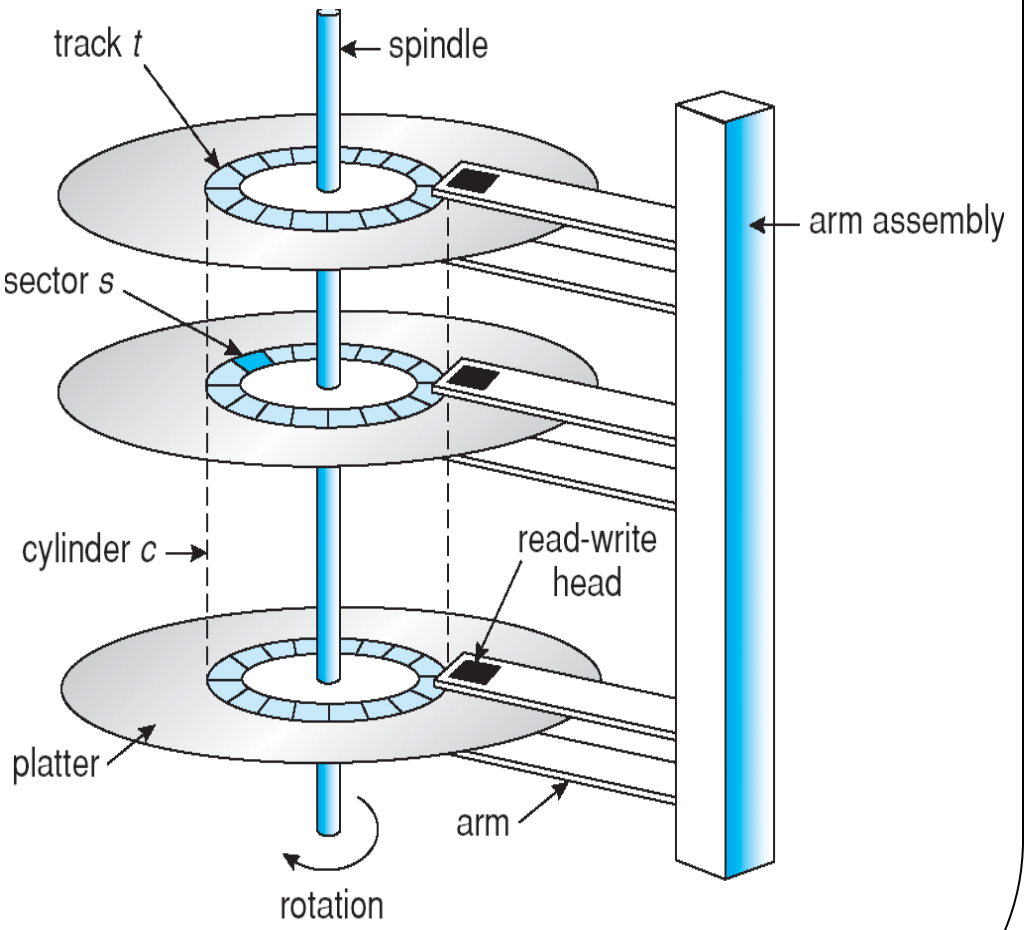
- I/O Hardware
- I/O System Design and Implementation
  - Design Objectives
  - Kernel I/O Structure
  - Kernel I/O Subsystem
  - Performance Consideration
- Disk
  - Disk Structure
  - Disk Scheduling
    - *FCFS, SSTF, SCAN, C-SCAN, C-LOOK*
  - Disk Management
  - Disk Reliability

Disk Structure



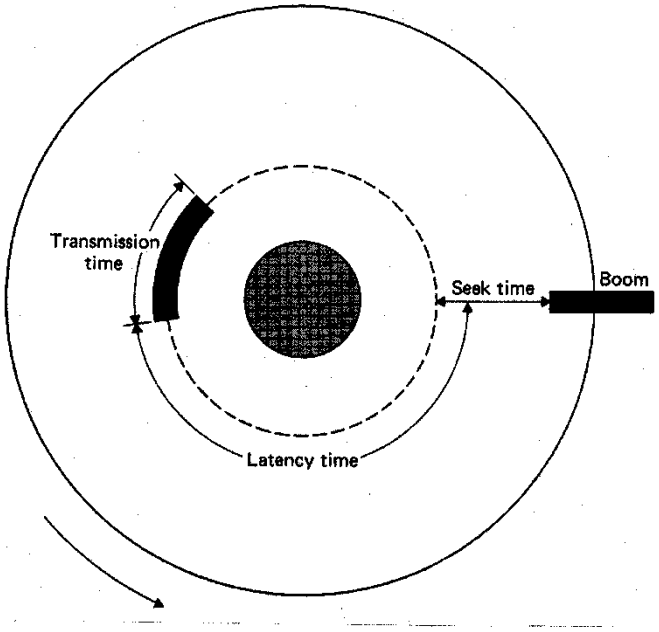
Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of *physical blocks*, where the physical block is the smallest unit of transfer (usually 512 bytes)
- The array of physical blocks is mapped into the sectors of the disk sequentially



Disk Scheduling

- The OS is responsible for using hardware efficiently — for disk drives this means having a fast access time and high disk bandwidth
- Access time has two major components
  - *Seek time* for the disk to move the heads to the track containing the desired sector (5 -25 milliseconds)
  - *Rotational latency* for the disk to rotate the desired sector to the disk head (8 milliseconds)



Disk Scheduling (Cont.)

- Minimize seek time
- Seek time $\approx$ seek distance
- Disk bandwidth is the total number of bytes transferred, divided by the total time between the 1st request and completion of the last transfer

Disk Scheduling (Cont.)

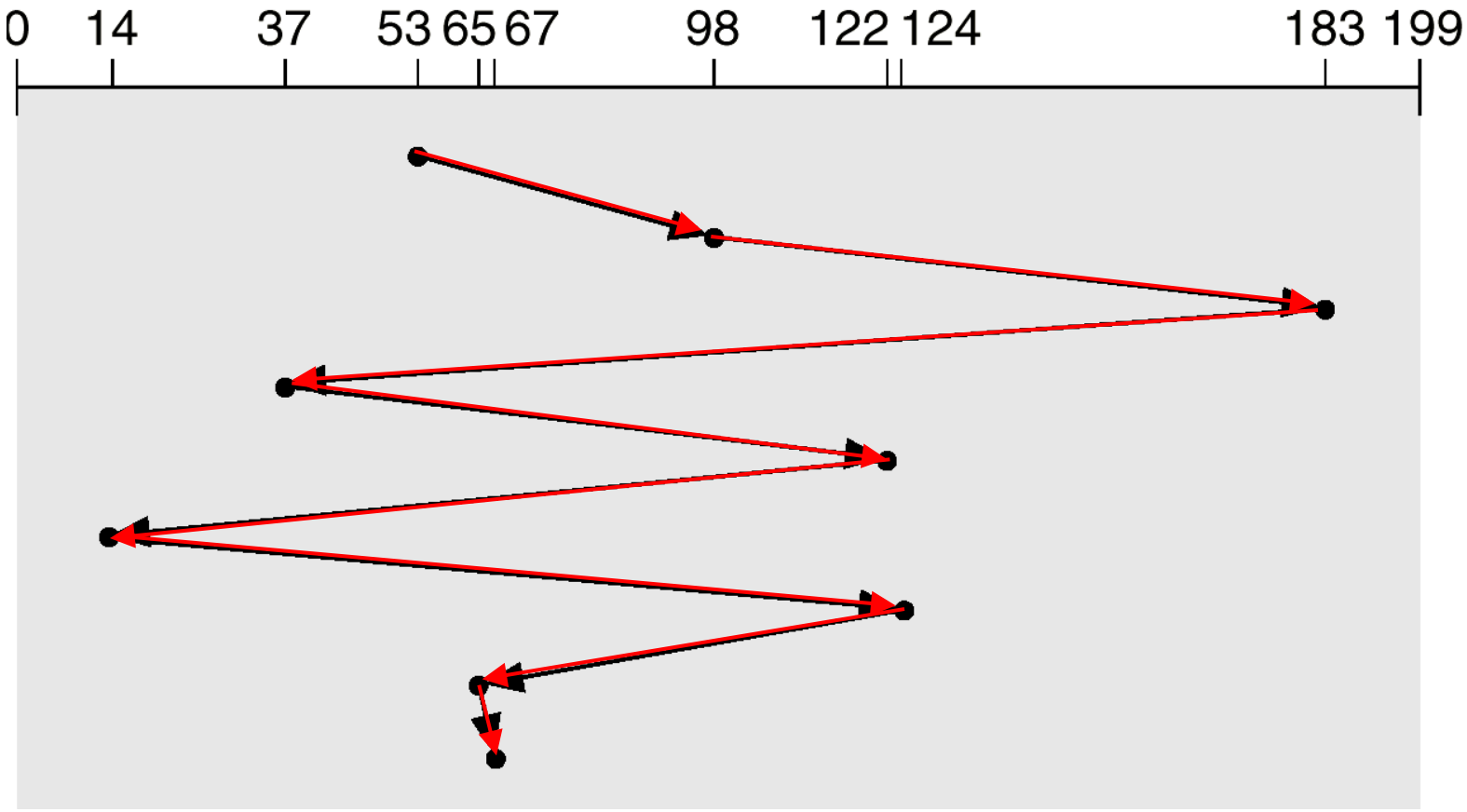
- Several algorithms exist to schedule the servicing of disk I/O requests
- We illustrate them with a request queue (0 -199)

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

- Assume that head is initially at cylinder 53

First-Come-First-Served (FCFS)

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



FCFS (Cont.)

Total head movement =

|98 - 53| + |183 - 98| + |37 - 183| + |122 - 37| +

|14 -122| + |124 -14| + |65 -124| + |67 - 65|

= 640

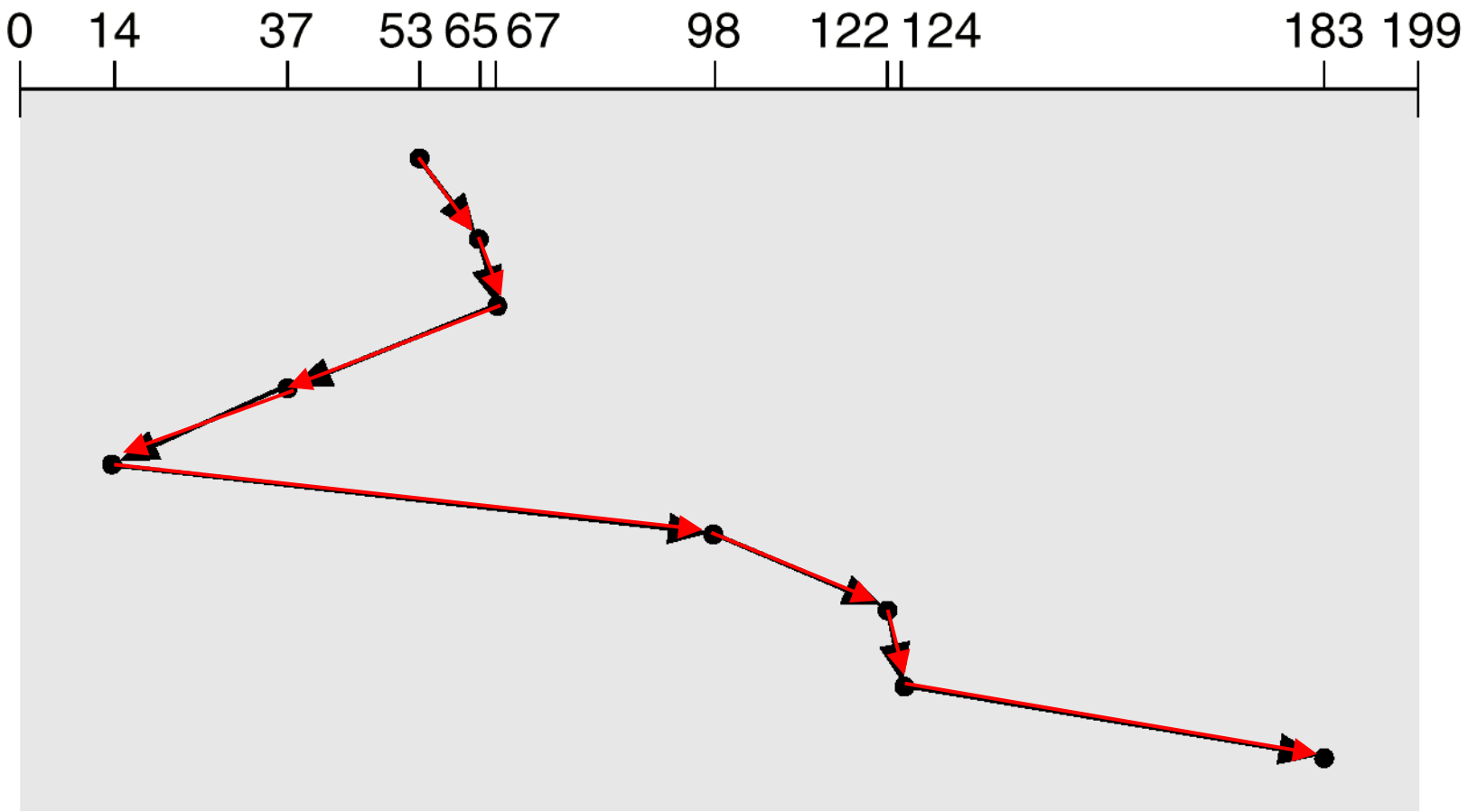
Problem: *FCFS may have wild swings back and forth!*

Shortest-Seek-Time-First (SSTF)

- Selects the request with the minimum seek time from the current head position
- SSTF scheduling is a form of SJF scheduling; may have starvation under heavy load, since distant requests may never be serviced
- Note: requests are from processes executing concurrently

SSTF (Cont.)

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



SSTF (Cont.)

Total head movement =

|65 - 53| + |67 - 65| + |37 - 67| + |14 - 37| +

|98 -14| + |122 - 98| + |124 - 122| + |183 - 124|

= 236

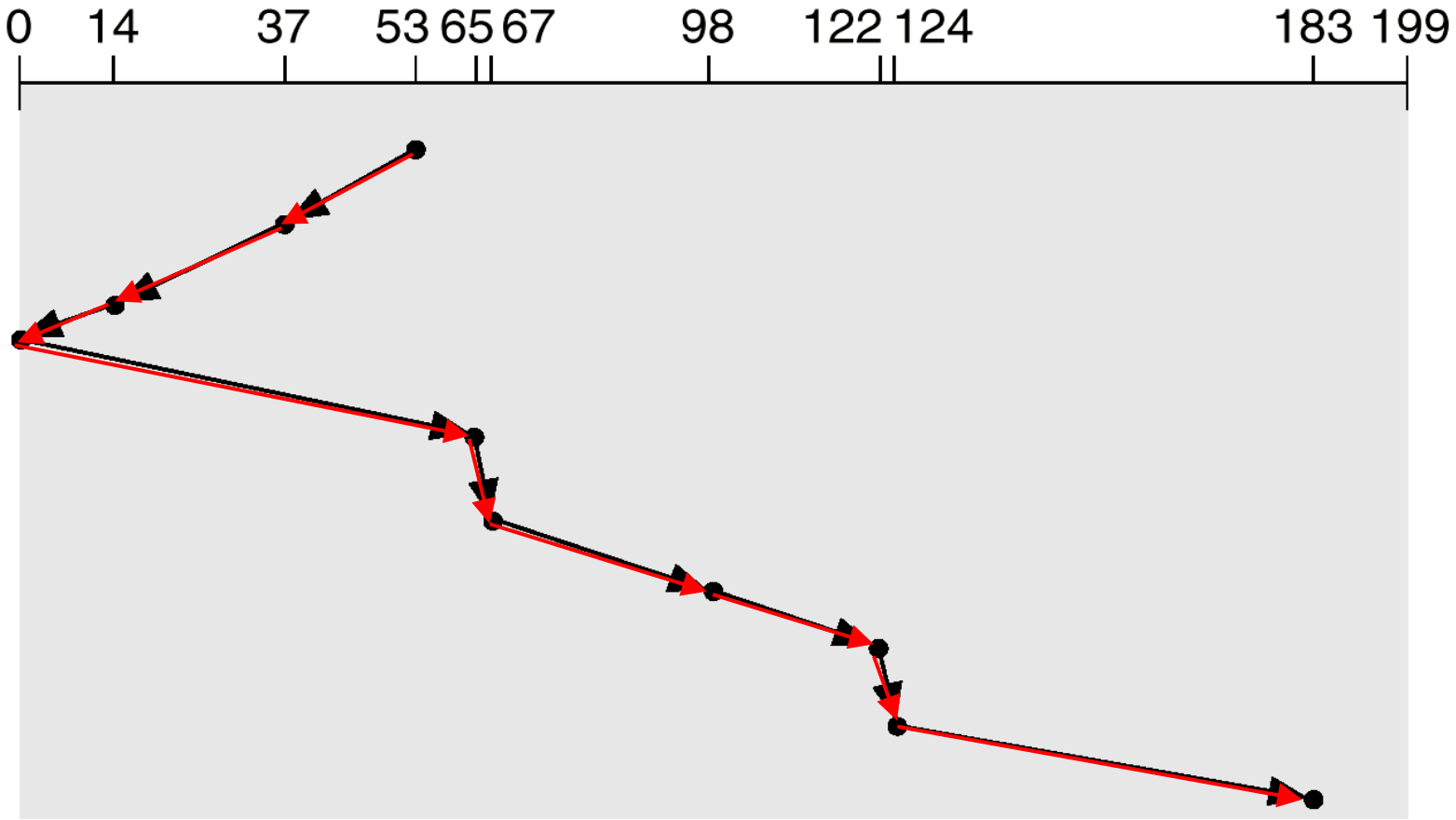
Problem: *SSTF may cause starvation!*

SCAN

- 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
- Also called the *elevator algorithm*
- Assuming here the head is moving toward 0
- Starvation unlikely but servicing requests may be delayed

SCAN (Cont.)

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



SCAN (Cont.)

Total head movement =

|37 - 53| + |14 - 37| + |0 - 14| + |65 - 0| + |67 - 65|

+ |98 - 67| + |122 - 98| + |124 - 122| + |183 - 124|

= 236

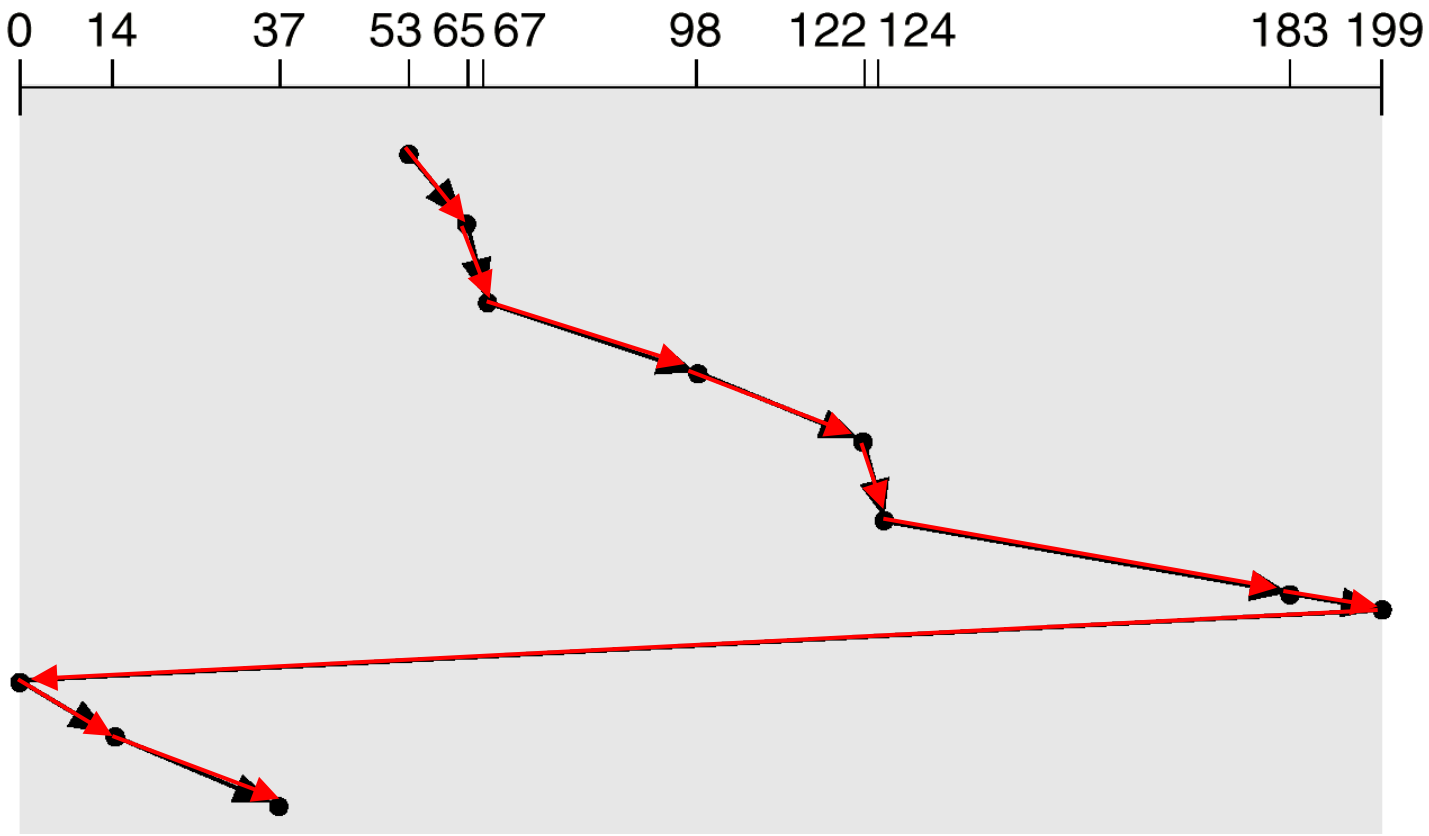
After the head reverses direction, does it needs to serve requests in the reversed direction immediately?

Circular-Scan (C-SCAN)

- Head moves from low end to high end of the disk, servicing requests along
- When reaching the high end, head immediately returns to the beginning of the disk, without servicing any requests

C-SCAN (Cont.)

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



C-SCAN (Cont.)

Total head movement =

|65 - 53| + |67 - 65| + |98 - 67| + |122 - 98| +

|124 - 122| + |183 -124| + |199 - 183| +

|0 -199| + |14 - 0| + |37 -14|

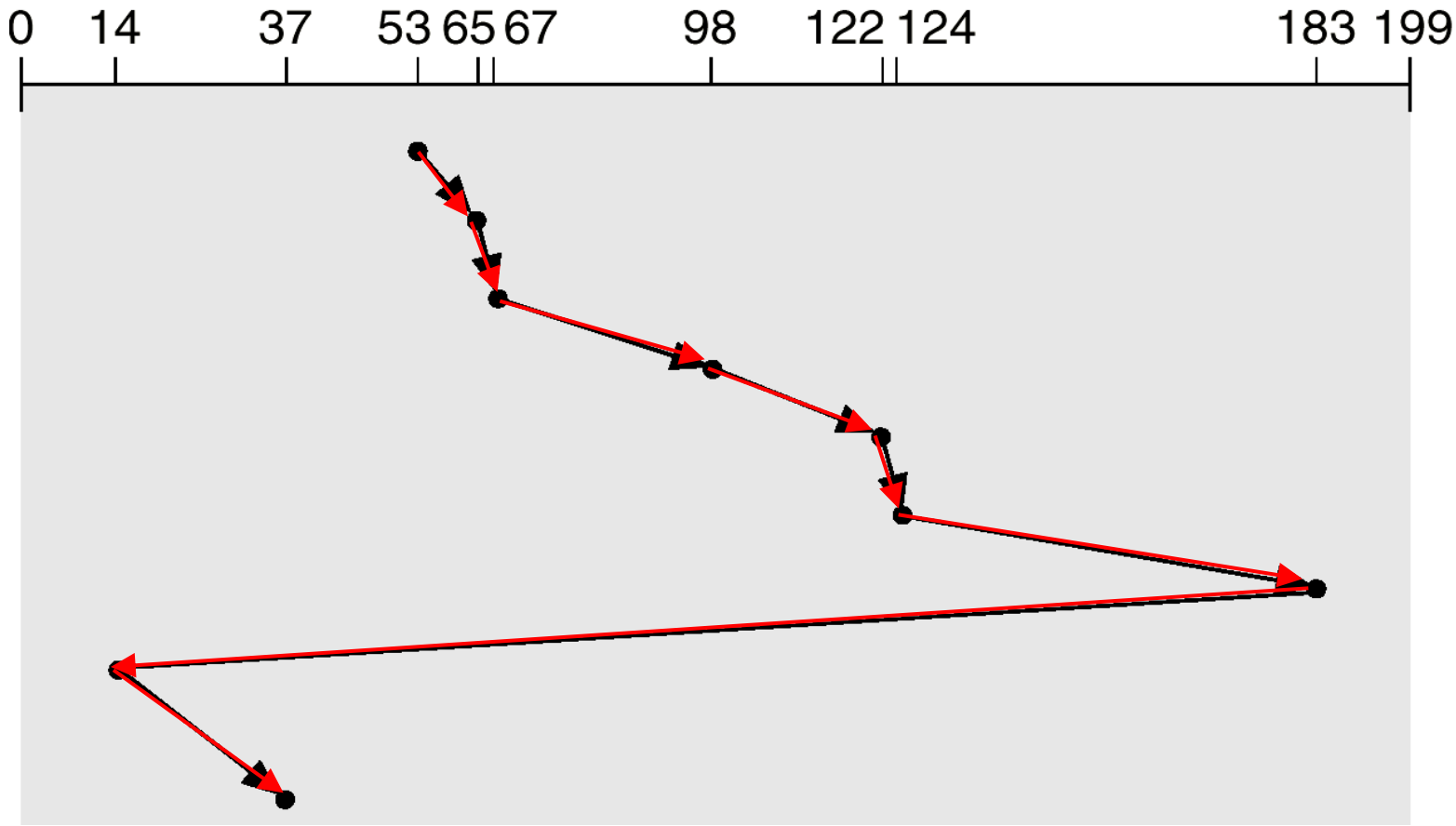
= 382

C-LOOK

- Version of C-SCAN
- Head moves from low end to high end of the disk, servicing requests along
- When reaching the last request, head returns immediately to the lowest cylinder # request
- From there it services the requests towards the high end of the disk

C-LOOK (Cont.)

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



C-LOOK (Cont.)

Total head movement =

|65 - 53| + |67 - 65| + |98 - 67| + |122 - 98| +

|124 -122| + |183 -124| + |14 -183| + |37 - 14|

= 322

Does SCAN, C-SCAN and C-LOOK really solve the starvation problem?

Selecting a Disk-Scheduling Algorithm

- SSTF is common and has a natural appeal
- SCAN and C-SCAN (or LOOK and C-LOOK) perform better for systems that place a heavy load on the disk (since starvation is unlikely)
- Performance depends on the number and types of requests

Selecting a Disk-Scheduling Algorithm (Cont.)

- Requests for disk service can be influenced by the file-allocation method:
  - A program reading a contiguously allocated file will generate requests that are close together
  - A linked or indexed file, may generate requests that are widely apart, resulting in greater head movement

Selecting a Disk-Scheduling Algorithm (Cont.)

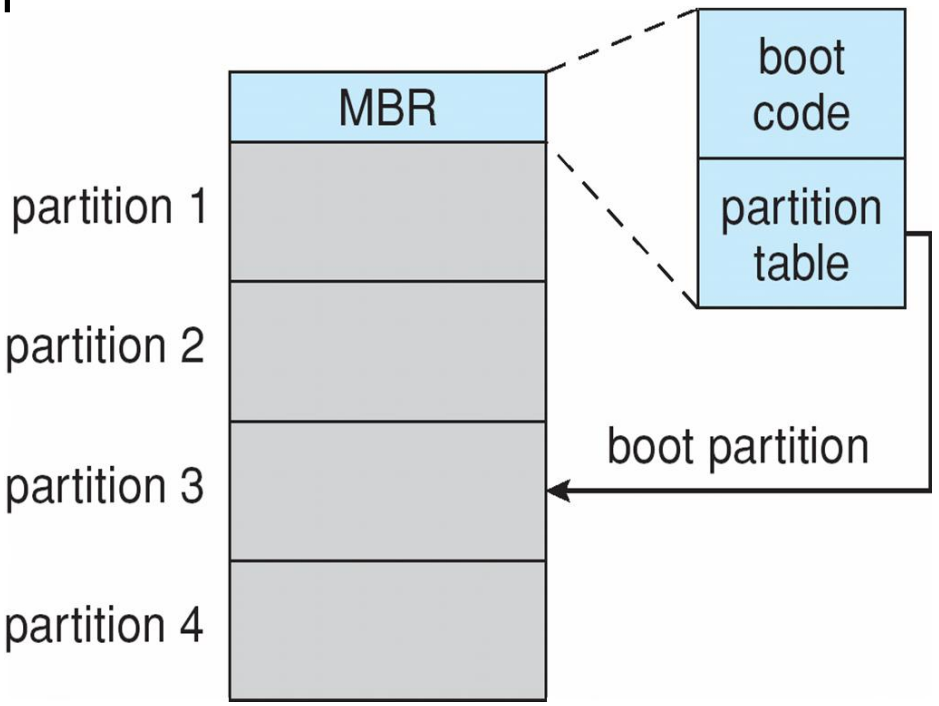
- The disk-scheduling algorithm should be written as a separate module of the OS, allowing it to be replaced with a different algorithm if necessary
- Either SSTF or C-LOOK is a reasonable choice for the default algorithm

Disk Management

- *Low-level formatting* (*physical formatting)* — Dividing a disk into sectors that the disk controller can read and write
- To use a disk to hold files, the OS needs to record its own data structures on the disk. Two steps:
  - *Partition* the disk into one or more groups of cylinders
  - *Logical formatting* or “making a file system”

Disk Management (Cont.)

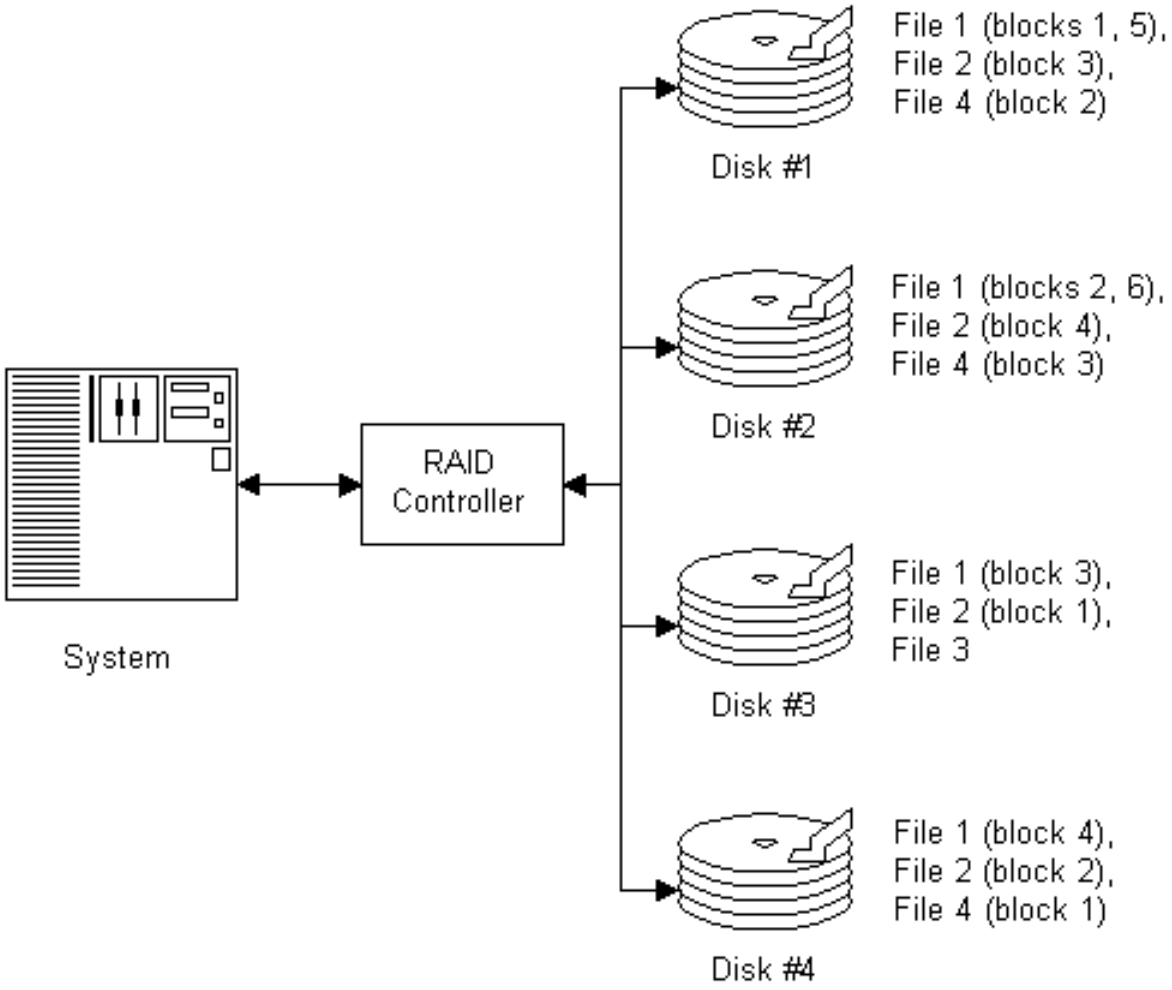
- Bootstrap program initializes system (i.e., starts computer)
  - A tiny bootstrap is stored in ROM, which loads the full bootstrap program (stored at disk boot blocks) into memory, and starts executing it
  - Finds the OS kernel on disk and loads it into memory
  - Then jumps to an initial address to start OS



Disk Reliability

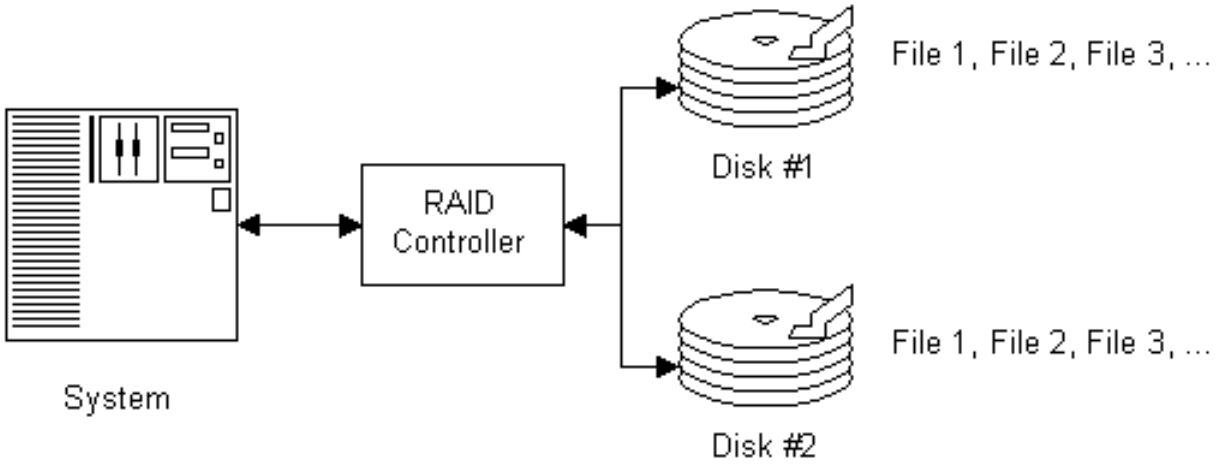
- Several improvements in disk-use techniques involve multiple disks working cooperatively
- *Disk striping* uses a group of disks as one storage unit
  - Each block is broken into several sub-blocks, with one sub-block stored on each disk
  - Time to transfer a block into memory is faster because all sub-blocks are transferred in parallel

Disk Reliability (cont.)



Disk Reliability (cont.)

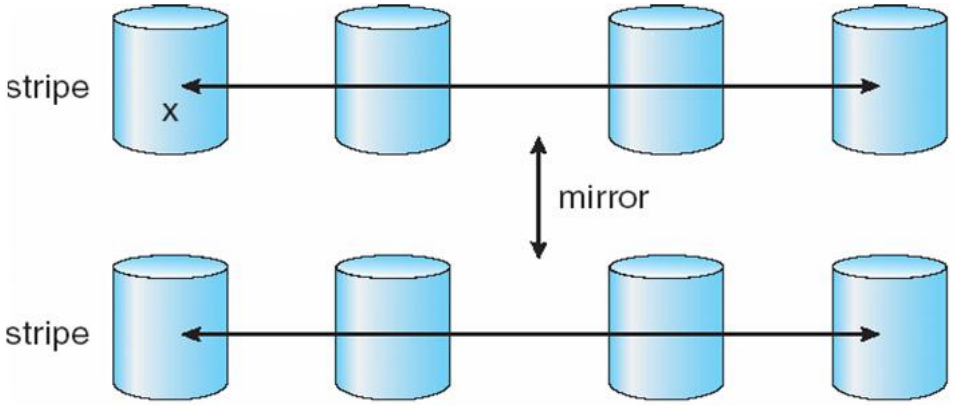
- *Mirroring* keeps duplicate of each disk
  - A logical disk consists of two physical disks
  - If one fails, the data can be read from the other



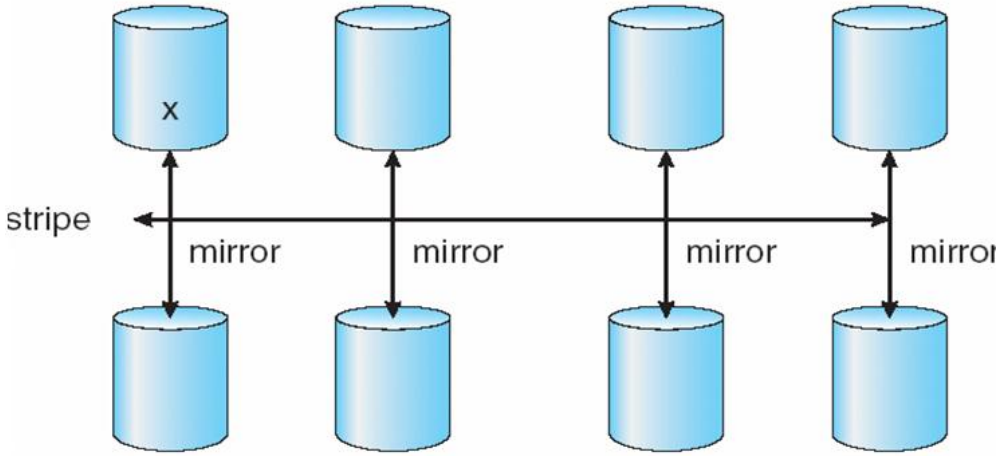
Disk Reliability (cont.)

- RAID (Redundant-Array-of-Independent-Disks) improves performance and reliability of the storage system by storing redundant data
  - RAID 0: non-redundant striping
  - Mirroring (RAID 1) keeps duplicate of each disk
  - Mirror of Strips (RAID 0+1) or Strips of Mirrors (RAID 1+0) provides high performance and high reliability

Disk Reliability (cont.)



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



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

Summary

- Device-driver layer hides the differences among device controllers from the I/O subsystem of the kernel.
- The I/O subsystem provides functions for: I/O scheduling, buffering, spooling, error handling, device reservation, and device name translation.
- I/O system calls are costly in terms of CPU consumption, because of the many layers of software between a physical device and the application.

Summary

- Disk scheduling algorithms optimize seek time:
  - FCFS, SSTF, SCAN, C-SCAN, C-LOOK
- Disk management requires disk partitioning and formatting
- Disk striping improve performance
- Mirroring improves reliability