Operating Systems 2023 Spring Term
Week 13

Dr. Emrah İnan (emrahinan@iyte.edu.tr)

Mass-Storage & I/O Systems

June 1, 2023

Week 12: Sample Glossary

- **volatile storage:** Storage whose content can be lost in a power outage or similar event. (on Page 1276)
- **nonvolatile storage (NVS):** Storage in which data will not be lost in a power outage or similar event. (on Page 1260)
- **bus:** A communication system; e.g., within a computer, a bus connects various components, such as the CPU and I/O devices, allowing them to transfer data and commands. (on Page 1241)

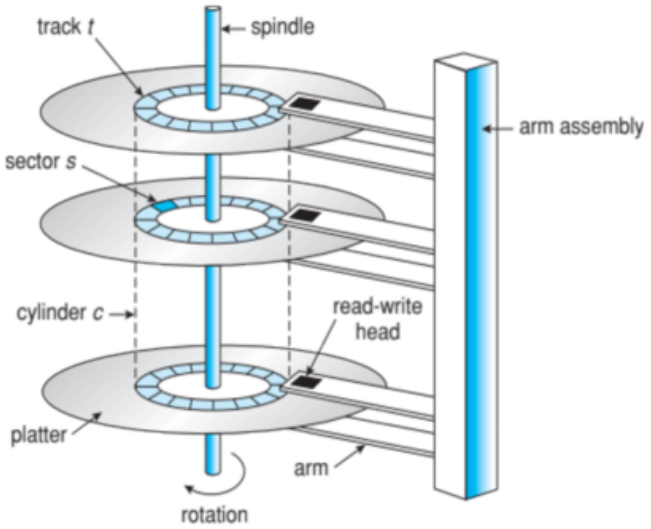
Introduction

- Disk Structure
- Disk Attachment
- Disk Scheduling
- Disk Management
- Swap-Space Management
- RAID Structure
- Stable-Storage Implementation
- Objectives
  - To describe the physical structure of secondary storage devices and its effects on the uses of the devices
  - To explain the performance characteristics of mass-storage devices
  - To evaluate disk scheduling algorithms
  - To discuss operating-system services provided for mass storage, including RAID

Overview of Mass Storage Structure

- Magnetic disks provide bulk of secondary storage of modern computers
  Drives rotate at 60 to 250 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, SAS, Firewire
  Host controller in computer uses bus to talk to disk controller built into drive or storage array

Moving-head Disk Mechanism



Each disk platter has a flat circular shape, like a CD
A read–write head “flies” just above each surface of every platter. The heads are attached to a disk arm that moves all the heads as a unit. The surface of a platter is logically divided into circular tracks, which are subdivided into sectors. The set of tracks at a given arm position make up a cylinder. A disk drive motor spins it at high speed. Most drives rotate 60 to 250 times per second, specified in terms of rotations per minute (RPM).

Hard Disks

- Platters range from .85" to 14" (historically)
  - Commonly 3.5", 2.5", and 1.8"
- Range from 30GB to 3TB per drive
- Performance
  - Transfer Rate – theoretical – 6 Gb/sec
  - Effective Transfer Rate – real – 1Gb/sec
  - Seek time from 3ms to 12ms – 9ms common for desktop drives
  - Average seek time measured or calculated based on 1/3 of tracks
  - Latency based on spindle speed
    - 1 / (RPM / 60) = 60 / RPM
  - Average latency = ½ latency

Spindle [rpm]	Average latency [ms]
4200	7.14
5400	5.56
7200	4.17
10000	3
15000	2

(From Wikipedia)

The transfer rate is the rate at which data flow between the drive and the computer. Another performance aspect, the positioning time, or random-access time, consists of two parts: the time necessary to move the disk arm to the desired cylinder, called the seek time, and the time necessary for the desired sector to rotate to the disk head, called the rotational latency.

Hard Disk Performance

Access Latency = **Average access time** = average seek time + average latency

For fastest disk 3ms + 2ms = 5ms

For slow disk 9ms + 5.56ms = 14.56ms

Average I/O time = average access time + (amount to transfer / transfer rate) + controller overhead

For example to transfer a 4KB block on a 7200 RPM disk with a 5ms average seek time, 1Gb/sec transfer rate with a .1ms controller overhead =

5ms + 4.17ms + 0.1ms + transfer time =

Transfer time = 4KB / 1Gb/s * 8Gb / GB * 1GB / 1024^2KB = 32 / (1024^2) = 0.031 ms

Average I/O time for 4KB block = 9.27ms + .031ms = 9.301ms

The First Commercial Disk Drive

1956
IBM RAMDAC computer included the IBM Model 350 disk storage system

5M (7 bit) characters
50 x 24” platters
Access time = < 1 second

Solid-State Disks

- Nonvolatile memory used like a hard drive (Many technology variations)
- Can be more reliable than HDDs
- More expensive per MB
- Maybe have shorter life span
- Less capacity But much faster
- Busses can be too slow -> connect directly to PCI for example
- No moving parts, so no seek time or rotational latency

Magnetic Tape

- Was early secondary-storage medium (Evolved from open spools to cartridges)
- 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
- 140MB/sec and greater
- 200GB to 1.5TB typical storage
- Common technologies are LTO-3,4,5 and T10000

Disk Structure

- Disk drives are addressed as large 1-dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer Low-level formatting creates logical blocks on physical media
- 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
  Logical to physical address should be easy
  Except for bad sectors
  Non-constant of sectors per track via constant angular velocity

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 (SANs) in which many hosts attach to many storage units
- I/O directed to bus ID, device ID, logical unit (LUN)

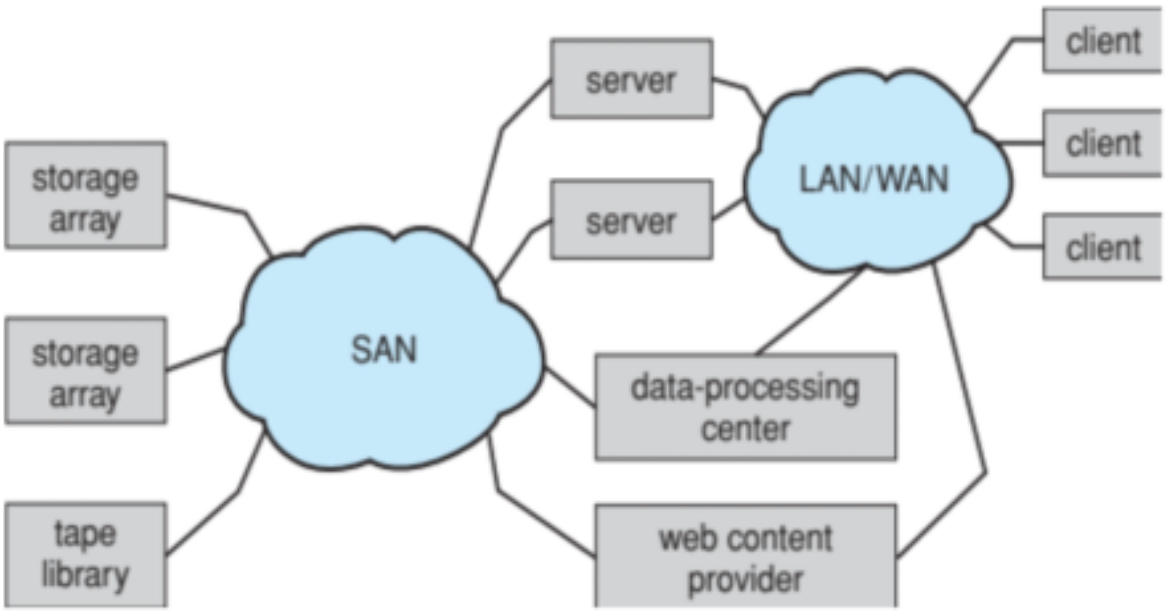
Storage Array

- Can just attach disks, or arrays of disks
- Storage Array has controller(s), provides features to attached host(s)
  Ports to connect hosts to array
  Memory, controlling software (sometimes NVRAM, etc)
  A few to thousands of disks
  RAID, hot spares, hot swap (discussed later)
  Shared storage -> more efficiency
  Features found in some file systems
  Snaphots, clones, thin provisioning, replication, deduplication, etc

Storage Area Network

Common in large storage environments

Multiple hosts attached to multiple storage arrays - flexible



Network-Attached Storage

Network-attached storage (**NAS**) is storage made available over a network rather than over a local connection (such as a bus)

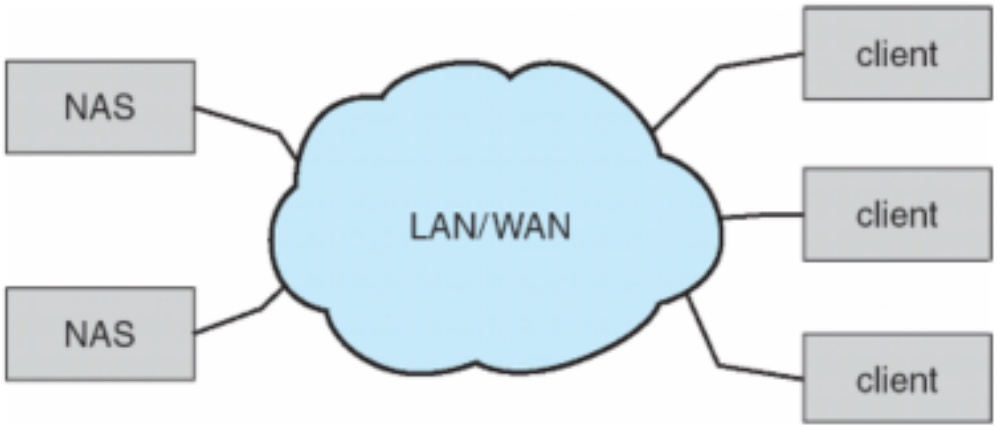
- Remotely attaching to file systems

NFS and CIFS are common protocols

Implemented via remote procedure calls (RPCs) between host and storage over typically TCP or UDP on IP network

iSCSI protocol uses IP network to carry the SCSI protocol

- Remotely attaching to devices (blocks)



Disk Scheduling I

- The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth
- 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
- There are many sources of disk I/O request -> OS, System processes, Users processes
- I/O request includes input or output mode, disk address, memory address, number of sectors to transfer OS maintains queue of requests, per disk or device Idle disk can immediately work on I/O request, busy disk means work must queue (Optimization algorithms only make sense when a queue exists)

Disk Scheduling II

Note that drive controllers have small buffers and can manage a queue of I/O requests (of varying “depth”)

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

The analysis is true for one or many platters

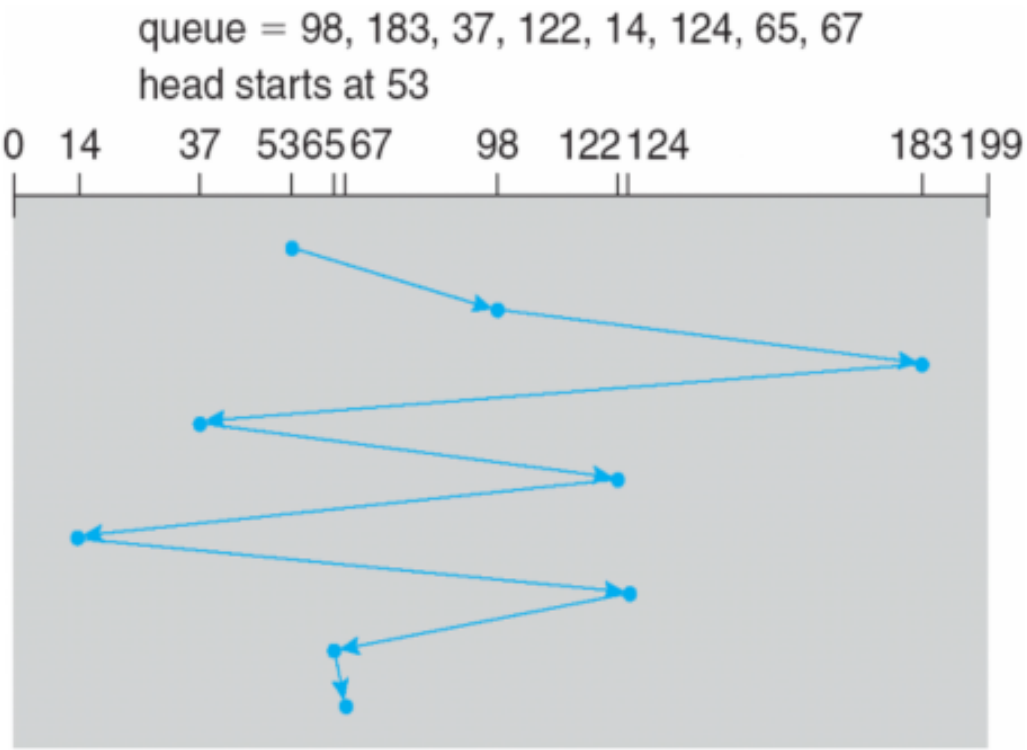
We illustrate scheduling algorithms with a request queue (0-199)

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

Head pointer 53

FCFS (the first-come, first-served algorithm or FIFO)

Illustration shows total head movement of 640 cylinders



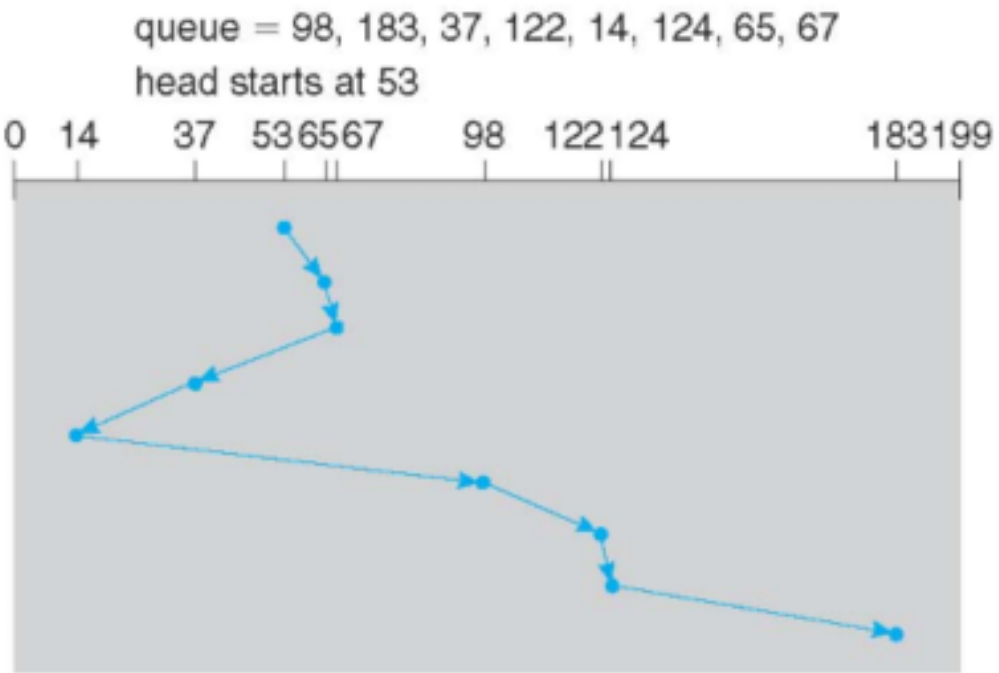
The wild swing from 122 to 14 and then back to 124 illustrates the problem with this schedule. If the requests for cylinders 37 and 14 could be serviced together, before or after the requests for 122 and 124, the total head movement could be decreased substantially, and performance could be thereby improved.

SSTF

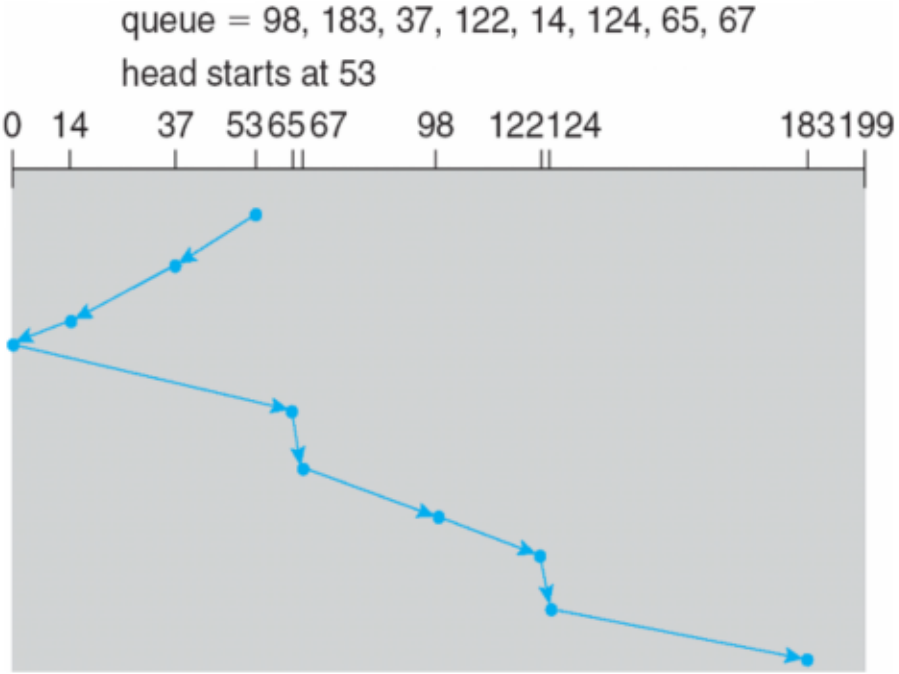
Shortest Seek Time First selects the request with the minimum seek time from the current head position

SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests

Illustration shows total head movement of 236 cylinders

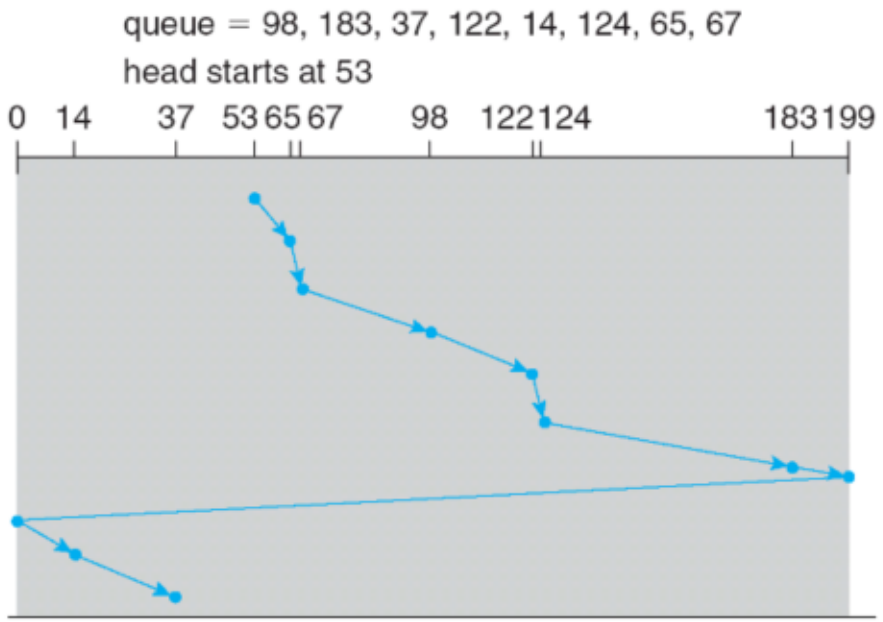


SCAN



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 236 cylinders. But note that if requests are uniformly dense, largest density at other end of disk and those wait the longest

C-SCAN



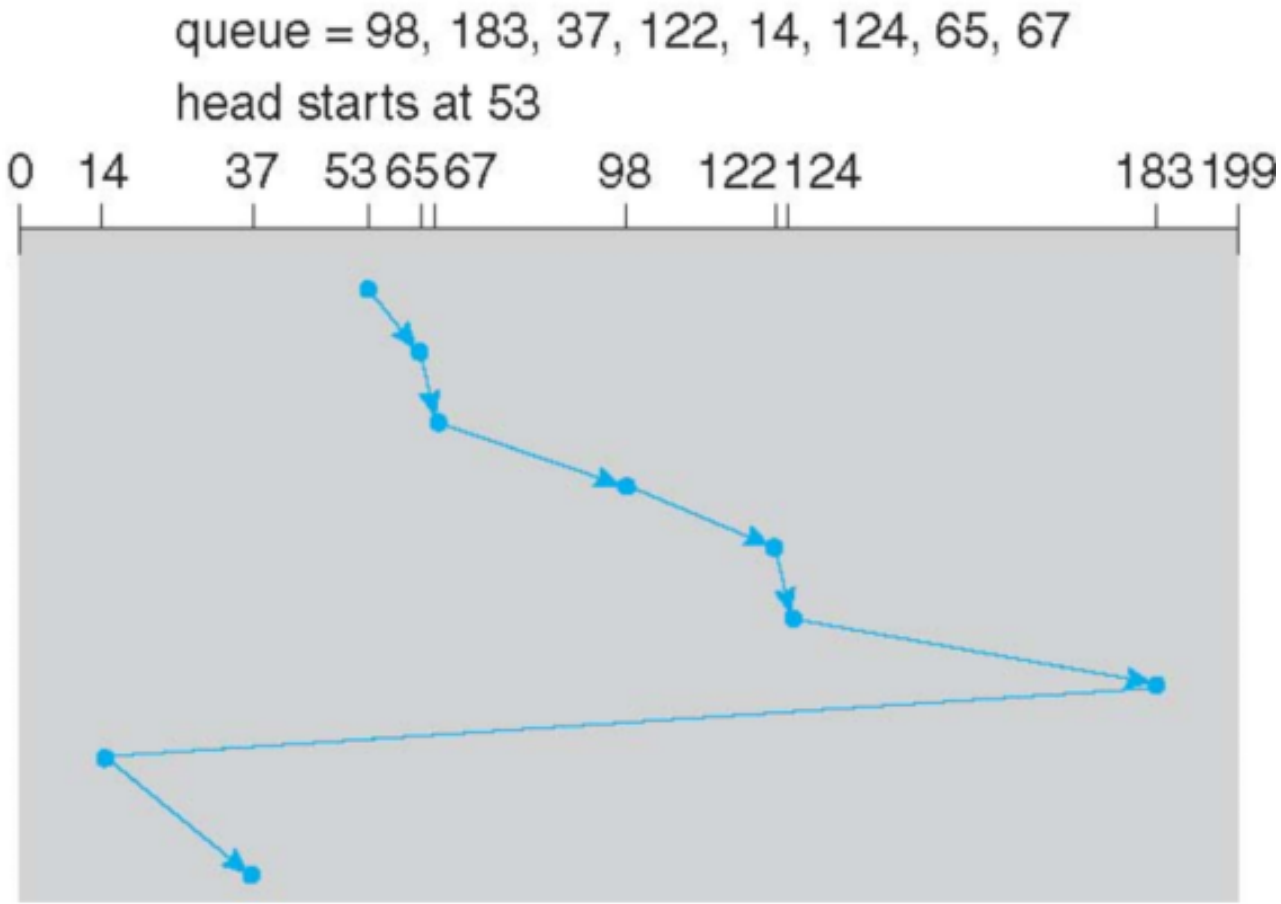
Provides a more uniform wait time than SCAN

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-LOOK



C-LOOK a 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

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 -> Less starvation
- Performance depends on the number and types of requests
- Requests for disk service can be influenced by the file-allocation method -> And metadata layout
- 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
- What about rotational latency? -> Difficult for OS to calculate

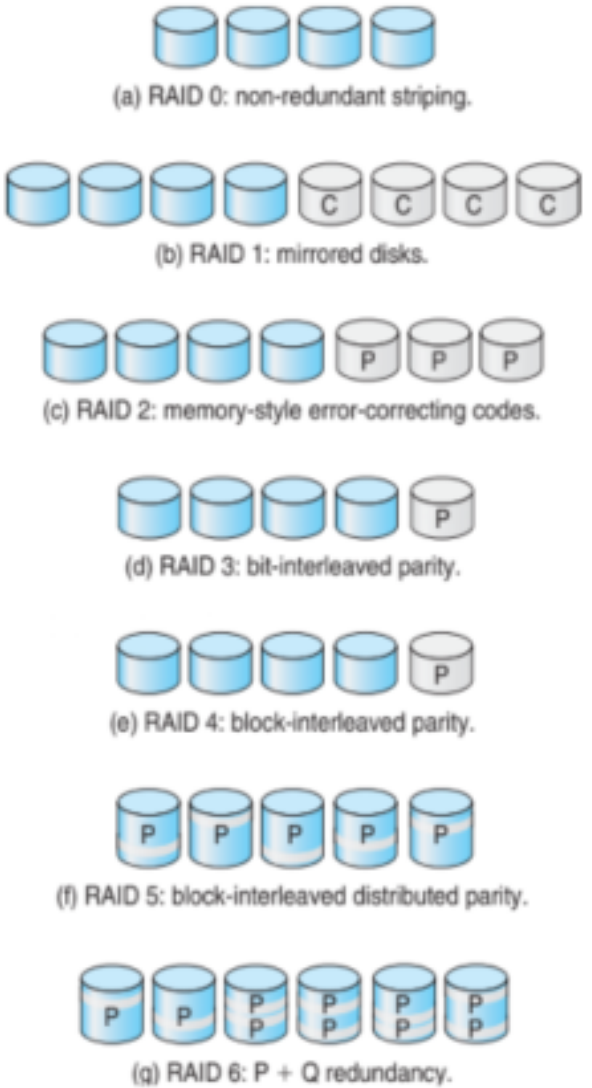
RAID Structure I

- RAID – redundant array of inexpensive disks multiple disk drives provides reliability via redundancy
- Increases the mean time to failure
- Mean time to repair – exposure time when another failure could cause data loss
- Mean time to data loss based on above factors
- If mirrored disks fail independently, consider disk with 1300,000 mean time to failure and 10 hour mean time to repair (Mean time to data loss is $100,000^2$ / $(2 * 10) = 500 * 10^6$ hours, or 57,000 years!)
- Frequently combined with NVRAM to improve write performance
- Several improvements in disk-use techniques involve the use of multiple disks working cooperatively

RAID Structure II

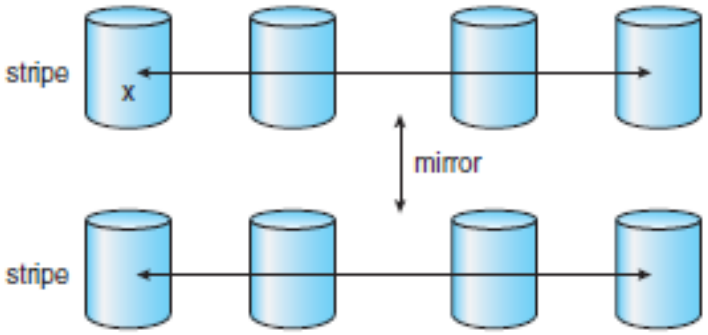
- Disk striping uses a group of disks as one storage unit
- RAID is arranged into six different levels
- RAID schemes improve performance and improve the reliability of the storage system by storing redundant data
  Mirroring or shadowing (RAID 1) keeps duplicate of each disk
  Striped mirrors (RAID 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 Levels

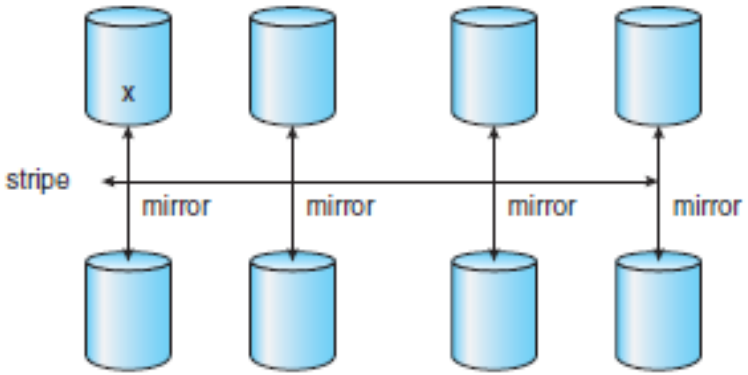


P indicates error-correcting bits and C indicates a second copy of the data

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



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

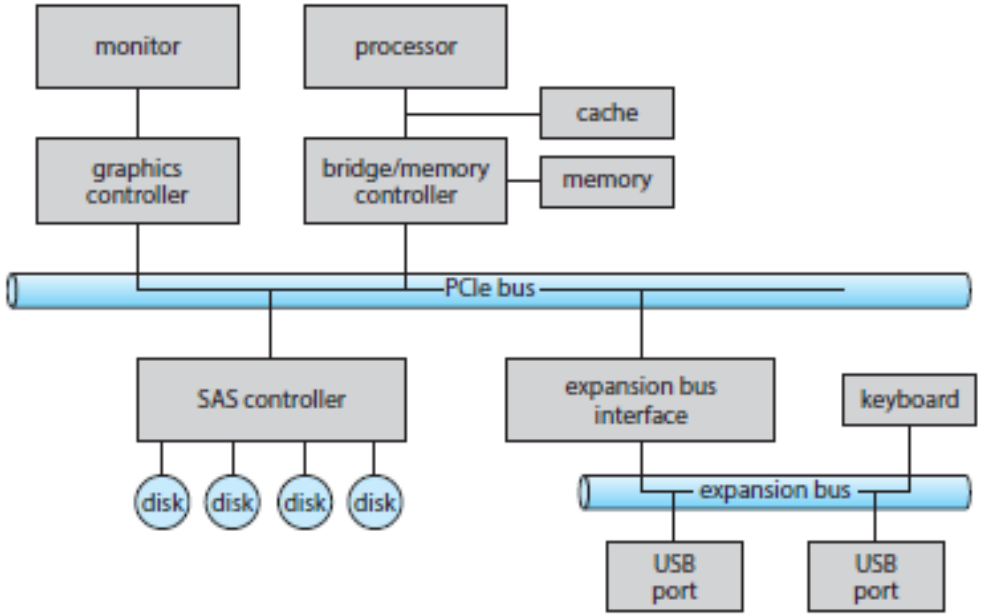


RAID 0 + 1 and 1 + 0 are used where both performance and reliability are important —for example, for small databases. Due to RAID 1's high space overhead, RAID 5 is often preferred for storing moderate volumes of data. RAID 6 and multidimensional RAID 6 are the most common formats in storage arrays.

I/O Systems: Overview

- I/O management is a major component of operating system design and operation (Important aspect of computer operation, I/O devices vary greatly, Various methods to control them, Performance management)
- Ports, busses, device controllers connect to various devices
- Device drivers encapsulate device details (Present uniform device-access interface to I/O subsystem)
- Common concepts – signals from I/O devices interface with computer
  Port – connection point for device
  Bus - daisy chain or shared direct access
  PCI bus common in PCs and servers, PCI Express (PCIe) expansion bus connects relatively slow devices
  Controller (host adapter) – electronics that operate port, bus, device (Sometimes integrated, Sometimes separate circuit board (host adapter))
  Contains processor, microcode, private memory, etc

A Typical PC Bus Structure



Devices have addresses, used by
Direct I/O instructions Memory-mapped I/O
Device data and command registers mapped to processor address space
Especially for large address spaces (graphics)

Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)

The data-in register is read by the host to get input. The data-out register is written by the host to send output.
The status register contains bits that can be read by the host. These bits indicate states, such as whether the current command has completed, whether a byte is available to be read from the data-in register, and whether a device error has occurred.
The control register can be written by the host to start a command or to change the mode of a device.

Polling

For each byte of I/O

1. Read busy bit from status register until 0
2. Host sets read or write bit and if write copies data into data-out register
3. Host sets command-ready bit
4. Controller sets busy bit, executes transfer
5. Controller clears busy bit, error bit, command-ready bit when transfer done

Step 1 is **busy-wait** cycle to wait for I/O from device

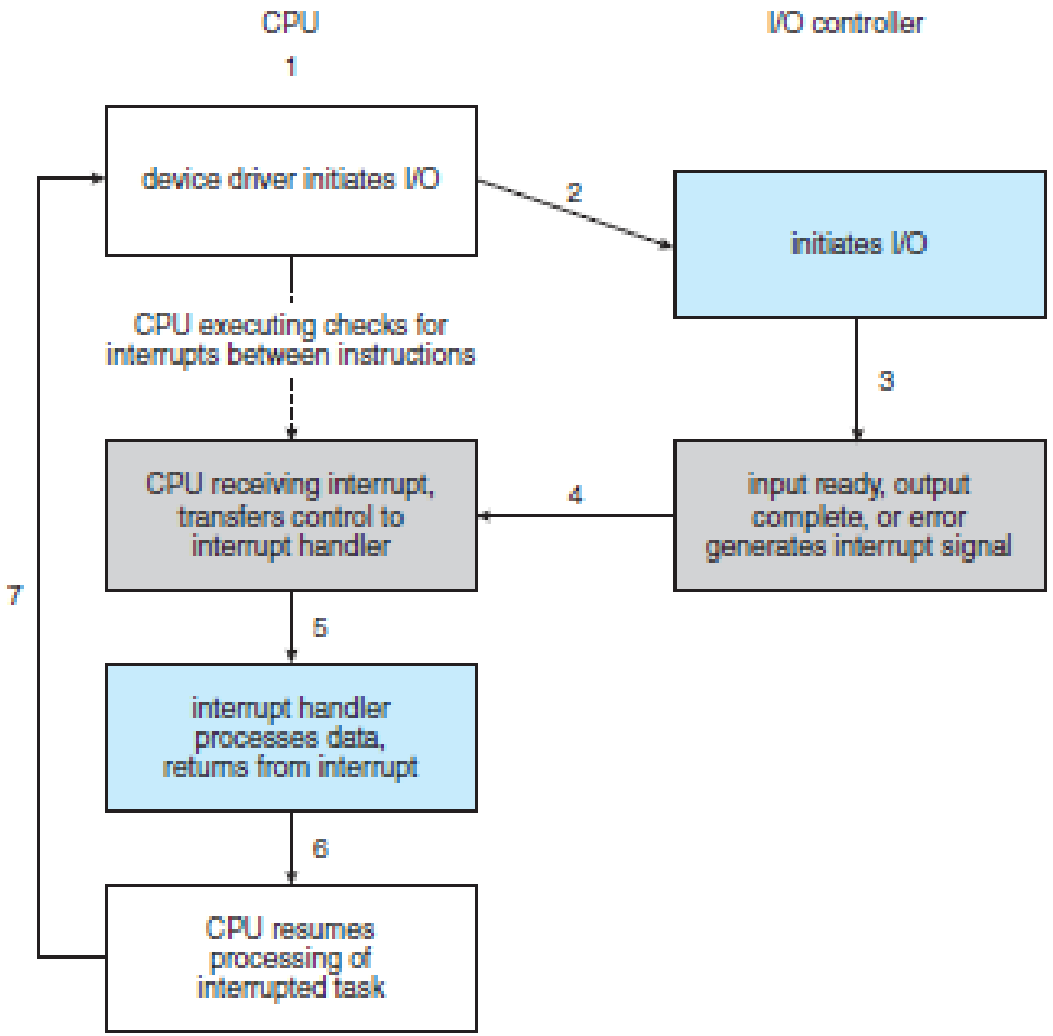
- Reasonable if device is fast
- But inefficient if device slow
- CPU switches to other tasks?
  - But if miss a cycle data overwritten / lost

In many computer architectures, three CPU-instruction cycles are sufficient to poll a device: read a device register, logical-and to extract a status bit, and branch if not zero. Clearly, the basic polling operation is efficient. But polling becomes inefficient when it is attempted repeatedly yet rarely finds a device ready for service,while other useful CPU processing remains undone.

Interrupts

- Polling can happen in 3 instruction cycles
  - Read status, logical-and to extract status bit, branch if not zero
  - How to be more efficient if non-zero infrequently?
- CPU **Interrupt-request line** triggered by I/O device
  - Checked by processor after each instruction
- **Interrupt handler** receives interrupts
  - **Maskable** to ignore or delay some interrupts
- **Interrupt vector** to dispatch interrupt to correct handler
  - Context switch at start and end
  - Based on priority
  - Some **nonmaskable**
  - Interrupt chaining if more than one device at same interrupt number

Interrupt-Driven I/O Cycle



Intel Pentium Processor Event-Vector Table

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Direct Memory Access

Used to avoid **programmed I/O** (one byte at a time) for large data movement

Requires **DMA** controller

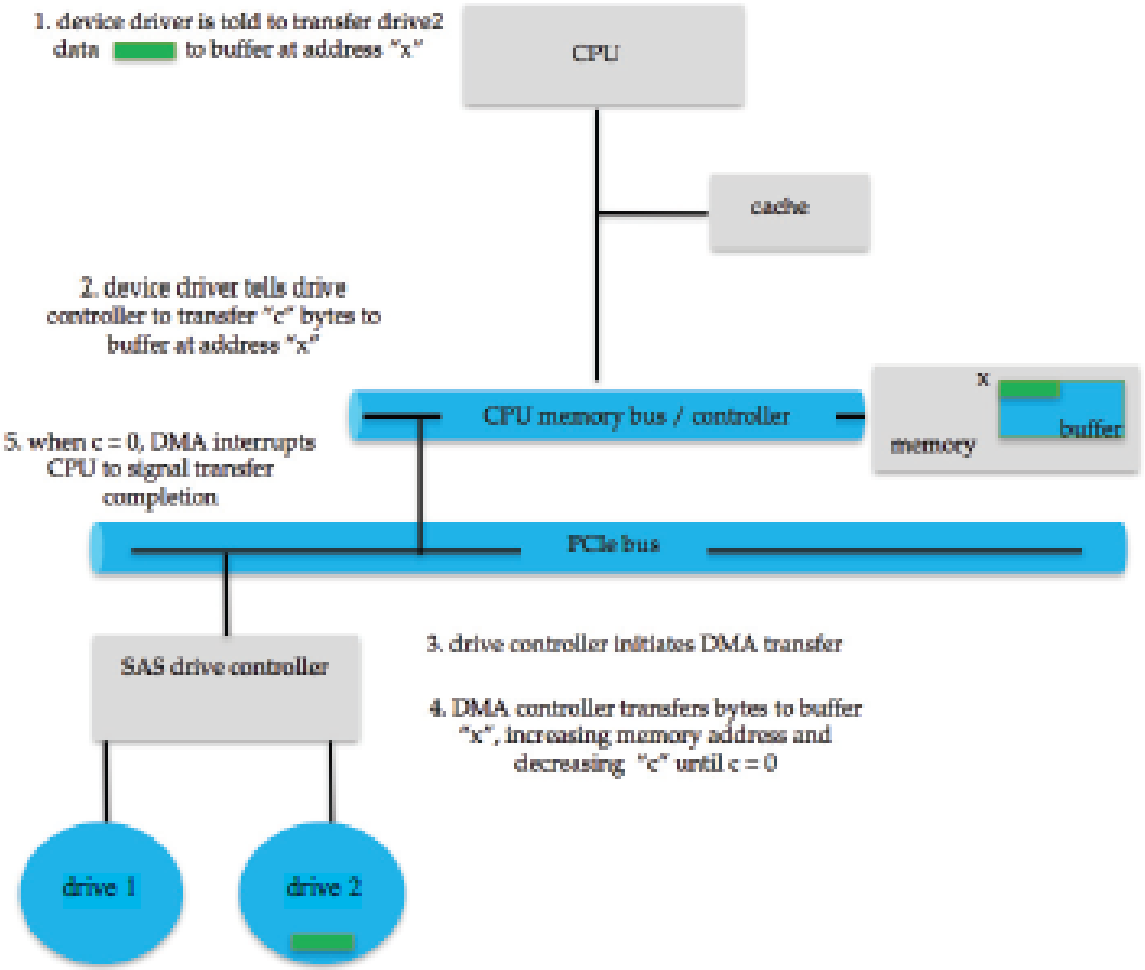
Bypasses CPU to transfer data directly between I/O device and memory

OS writes DMA command block into memory

- Source and destination addresses
- Read or write mode
- Count of bytes
- Writes location of command block to DMA controller
- Bus mastering of DMA controller – grabs bus from CPU
  - **Cycle stealing** from CPU but still much more efficient
- When done, interrupts to signal completion

Version that is aware of virtual addresses can be even more efficient - **DVMA**

Step Process to Perform DMA Transfer



Application I/O Interface

I/O system calls encapsulate device behaviors in generic classes

Device-driver layer hides differences among I/O controllers from kernel

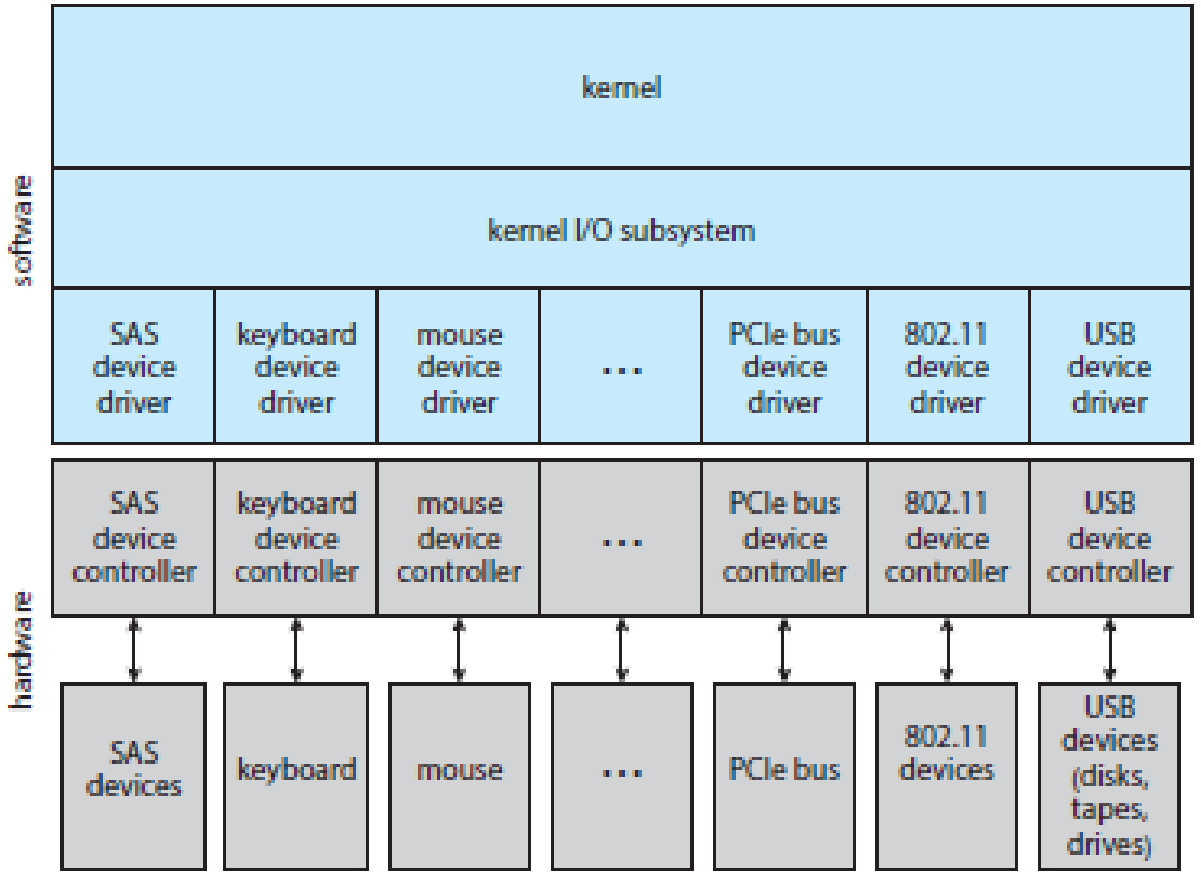
New devices talking already-implemented protocols need no extra work

Each OS has its own I/O subsystem structures and device driver frameworks

Devices vary in many dimensions

- **Character-stream** or **block**
- **Sequential** or **random-access**
- **Synchronous** or **asynchronous** (or both)
- **Sharable** or **dedicated**
- **Speed of operation**
- **read-write**, **read only**, or **write only**

A Kernel I/O Structure



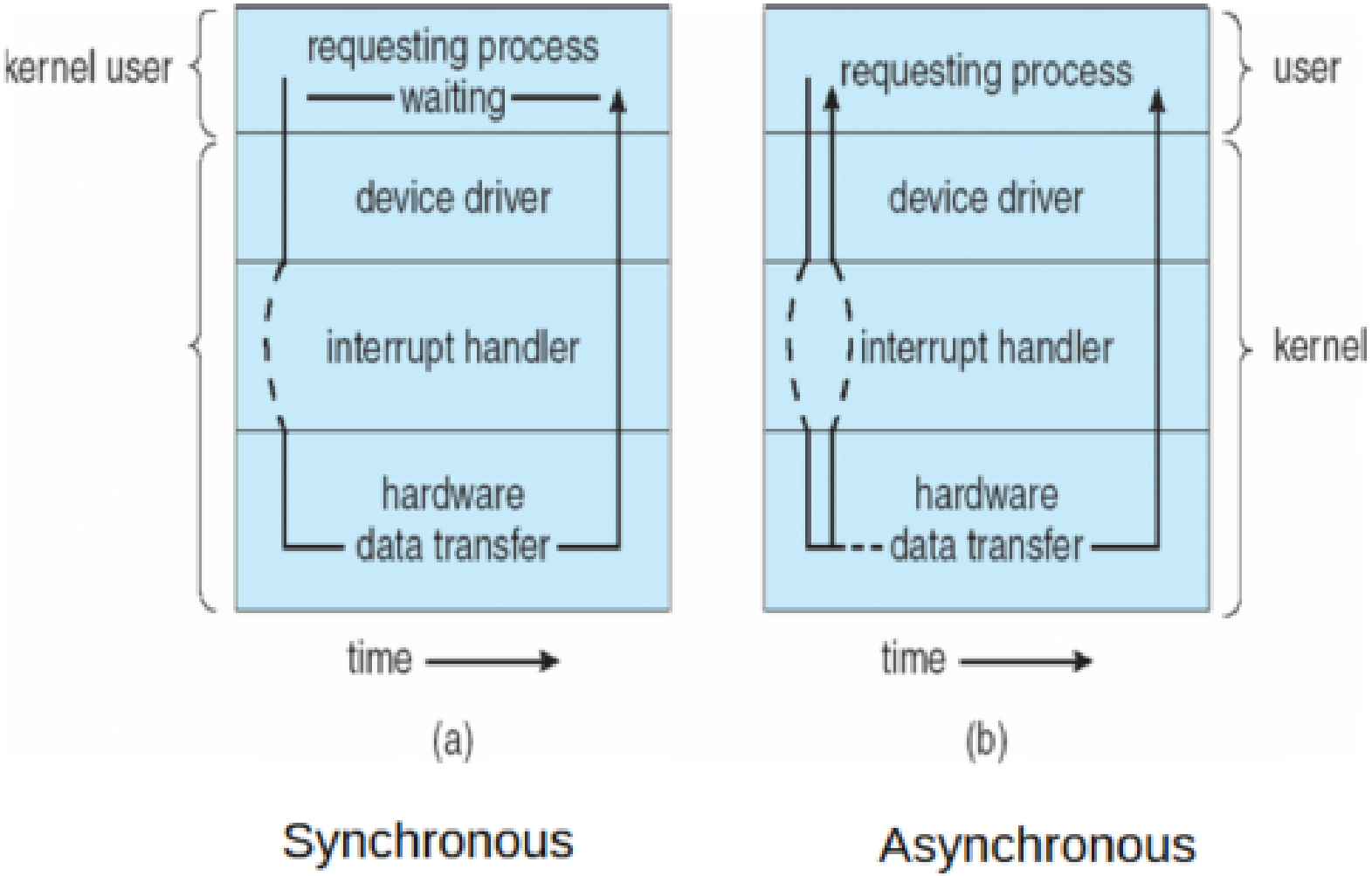
Characteristics of I/O Devices

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

Clocks and Timers

- Provide current time, elapsed time, timer
- Normal resolution about 1/60 second
- Some systems provide higher-resolution timers
- Programmable interval timer used for timings, periodic interrupts
- ioctl() (on UNIX) covers odd aspects of I/O such as clocks and timers
- Nonblocking and Asynchronous I/O
  - Blocking - process suspended until I/O completed
    Easy to use and understand
    Insufficient for some needs
  - Nonblocking - I/O call returns as much as available
    User interface, data copy (buffered I/O)
    Implemented via multi-threading
    Returns quickly with count of bytes read or written
    select() to find if data ready then read() or write() to transfer
  - Asynchronous - process runs while I/O executes
    Difficult to use, I/O subsystem signals process when I/O completed

Two I/O Methods



Kernel I/O Subsystem

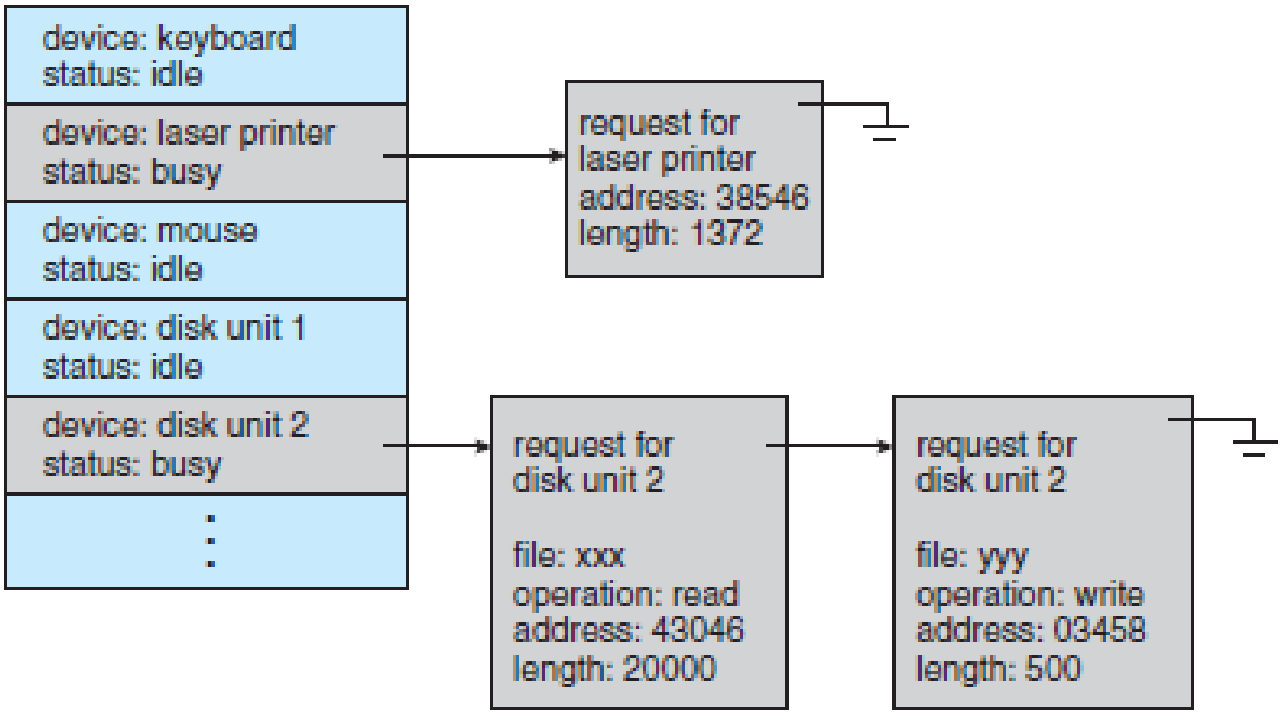
Scheduling

- Some I/O request ordering via per-device queue
- Some OSs try fairness
- Some implement Quality Of Service (i.e. IPQOS)

Buffering - store data in memory while transferring between devices

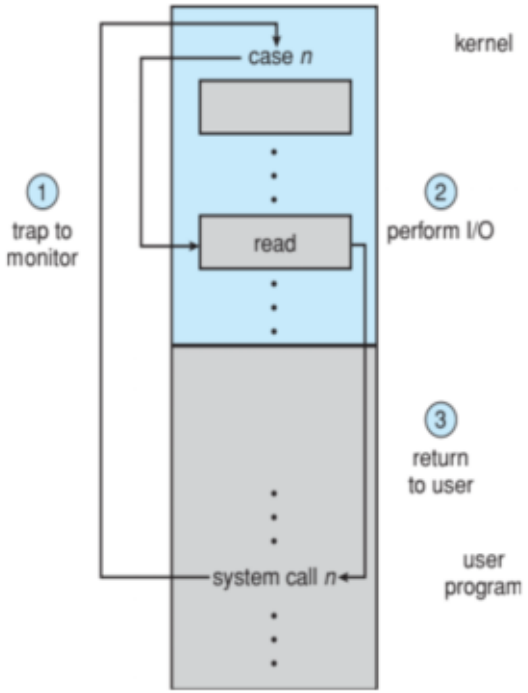
- To cope with device speed mismatch
- To cope with device transfer size mismatch
- To maintain “copy semantics”
- **Double buffering** – two copies of the data
  - Kernel and user
  - Varying sizes
  - Full / being processed and not-full / being used
  - Copy-on-write can be used for efficiency in some cases

Device-status Table



Kernel I/O Subsystem

- Caching - faster device holding copy of data (Always just a copy, Key to performance, Sometimes combined with buffering)
- Spooling - hold output for a device (If device can serve only one request at a time, i.e., Printing)
- Device reservation - provides exclusive access to a device
  System calls for allocation and de-allocation
  Watch out for deadlock



UNIX I/O Kernel Structure