Introduction

Adapted from Operating System Concepts – 9th Edition

Introduction

- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations

Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

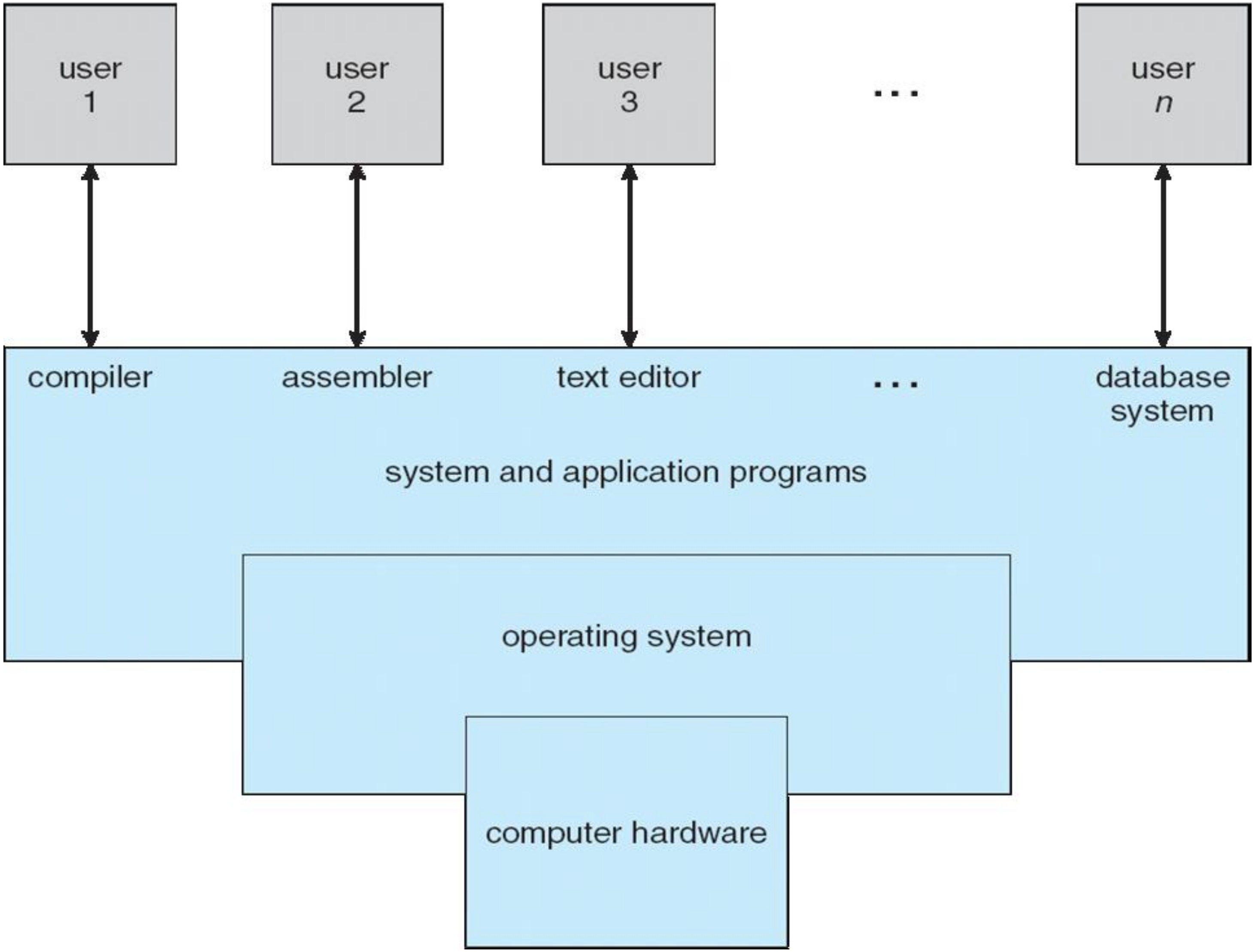
Goals of an Operating System

- Simplify the execution of user programs and make solving user problems easier.
- Use computer hardware efficiently.
- Allow sharing of hardware and software resources.
- Make application software portable and versatile.
- Provide isolation, security and protection among user programs.
- Improve overall system reliability error confinement, fault tolerance, reconfiguration.

Computer System Structure

- Computer system can be divided into four components:
  - **Hardware** – provides basic computing resources
    - CPU, memory, I/O devices
  - **Operating system**
    - Controls and coordinates use of hardware among various applications and users
  - **Application programs** – define the ways in which the system resources are used to solve the computing problems of the users
    - Word processors, compilers, web browsers, database systems, video games
  - **Users**
    - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer

Definition:

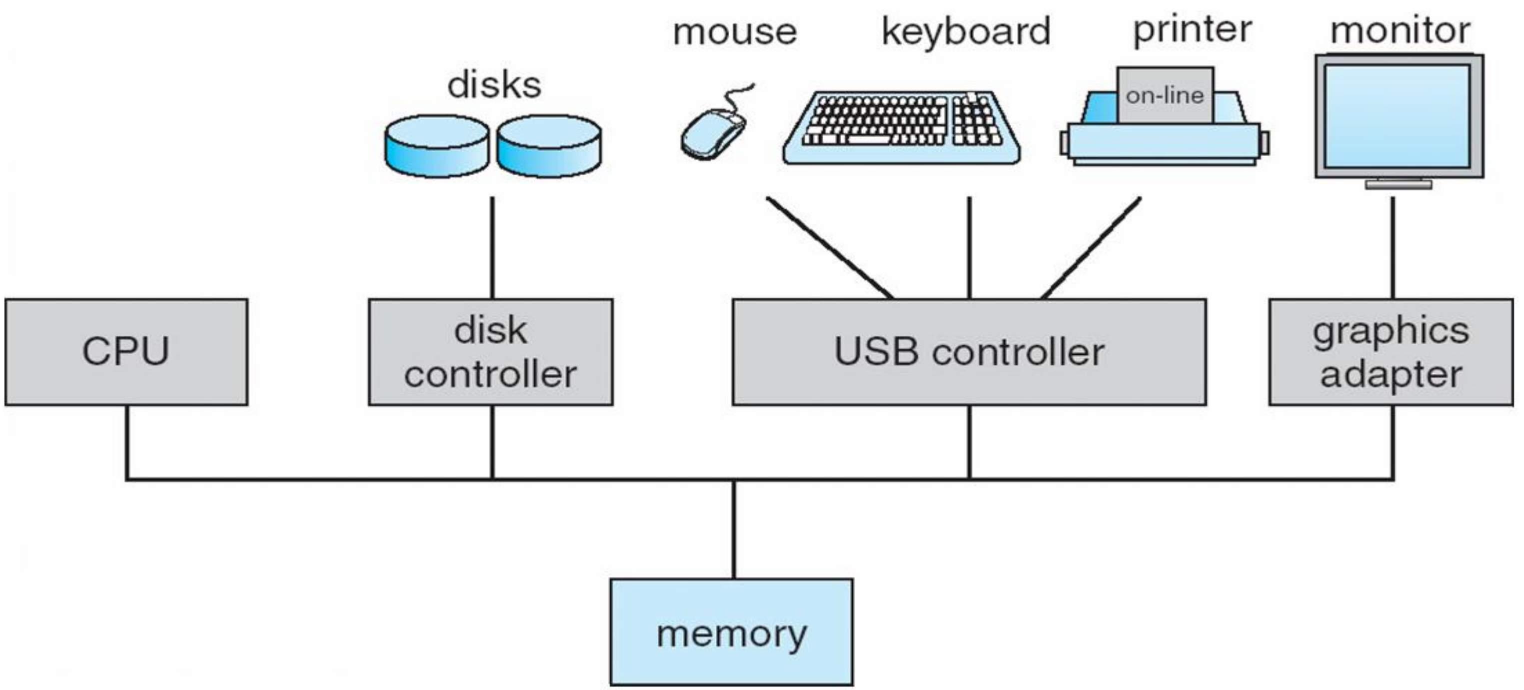
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
  - Typically stored in ROM or EPROM, generally known as **firmware**
  - Initializes all aspects of system
  - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
  - One or more CPUs, device controllers connect through common bus providing access to shared memory
  - Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation

- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs CPU that it has finished its operation by causing an **interrupt**

Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the **interrupt vector**, which contains the addresses of all the service routines
- Interrupt architecture must save the address of the interrupted instruction
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request
- An operating system is **interrupt driven**

I/O Structure

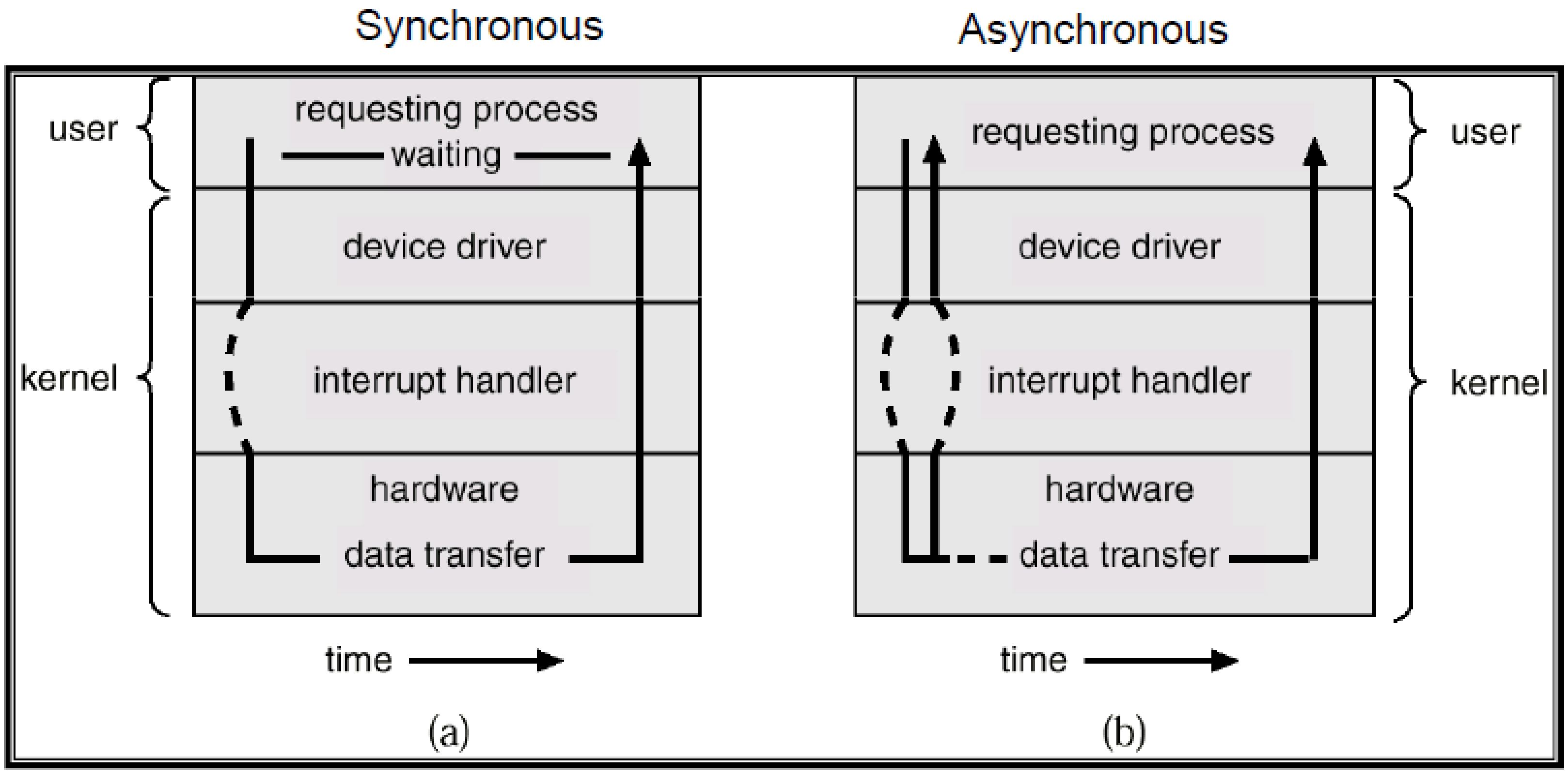
Synchronous I/O

- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing

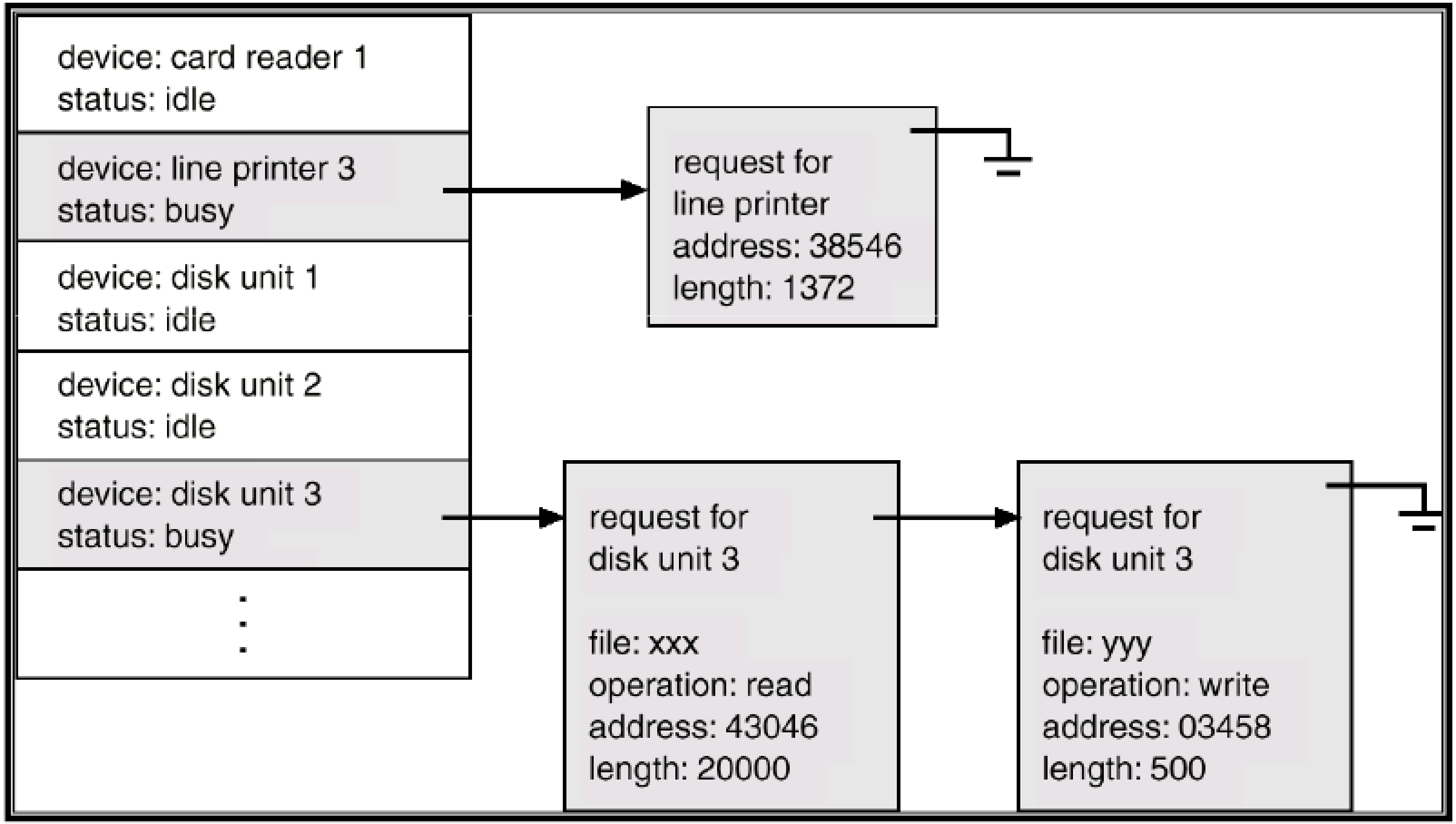
Asynchronous I/O

- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS to allow user to wait for I/O completion
  - **Device-status table** contains entry for each I/O device indicating its type, address, and state
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Two I/O Methods



Device-Status Table



Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from
- buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than one interrupt per byte

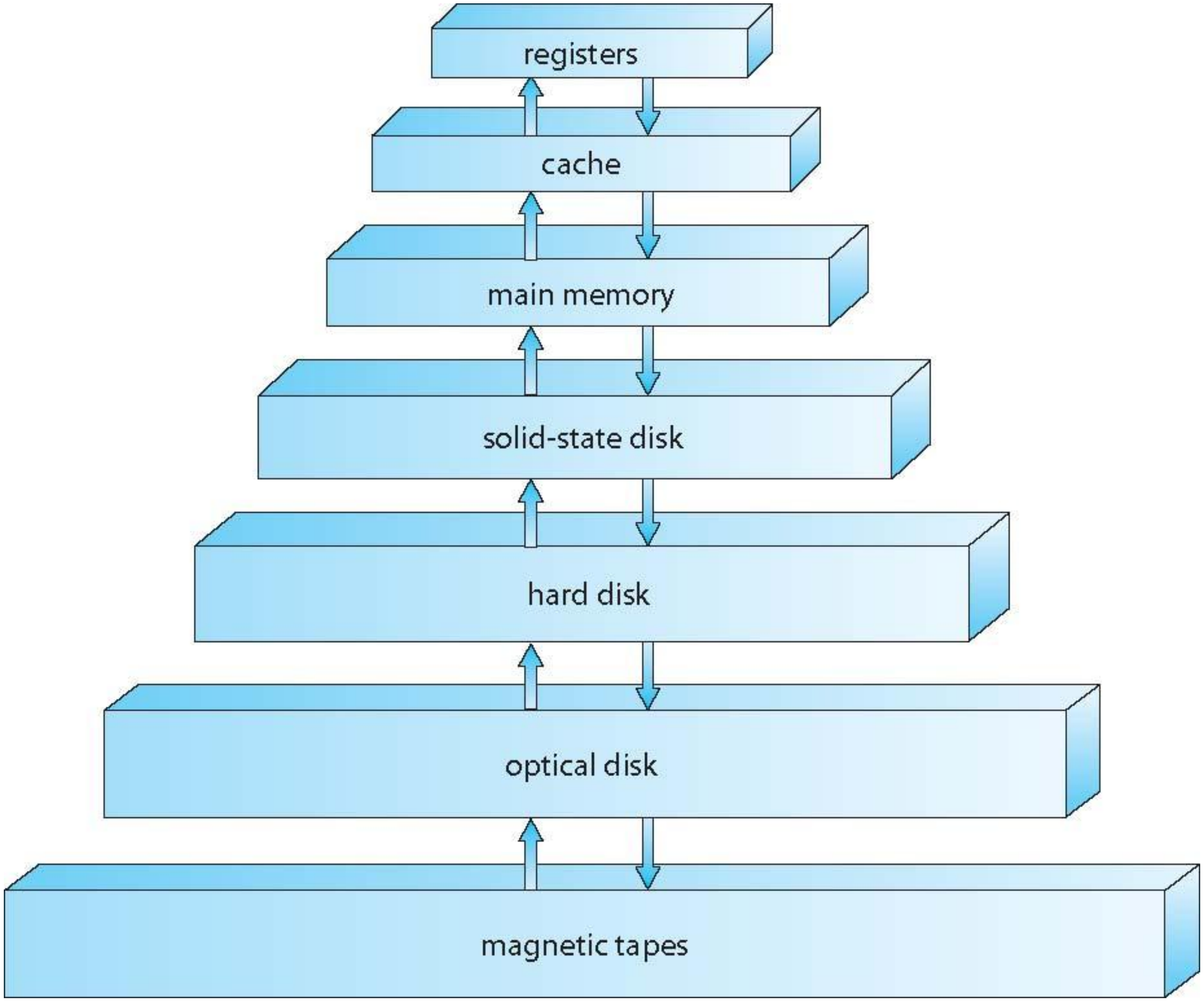
Storage Structure

- **Main memory** – only large storage media that the CPU can access directly
  - Typically **volatile**
- **Secondary storage** – extension of main memory that provides large **nonvolatile** storage capacity
- **Hard disks** – rigid metal or glass platters covered with magnetic recording material
  - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - The **disk controller** determines the logical interaction between the device and the computer
- **Solid-state disks** – faster than hard disks, nonvolatile
  - Various technologies
  - Becoming more popular

Storage Hierarchy

- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage

Storage-Device Hierarchy



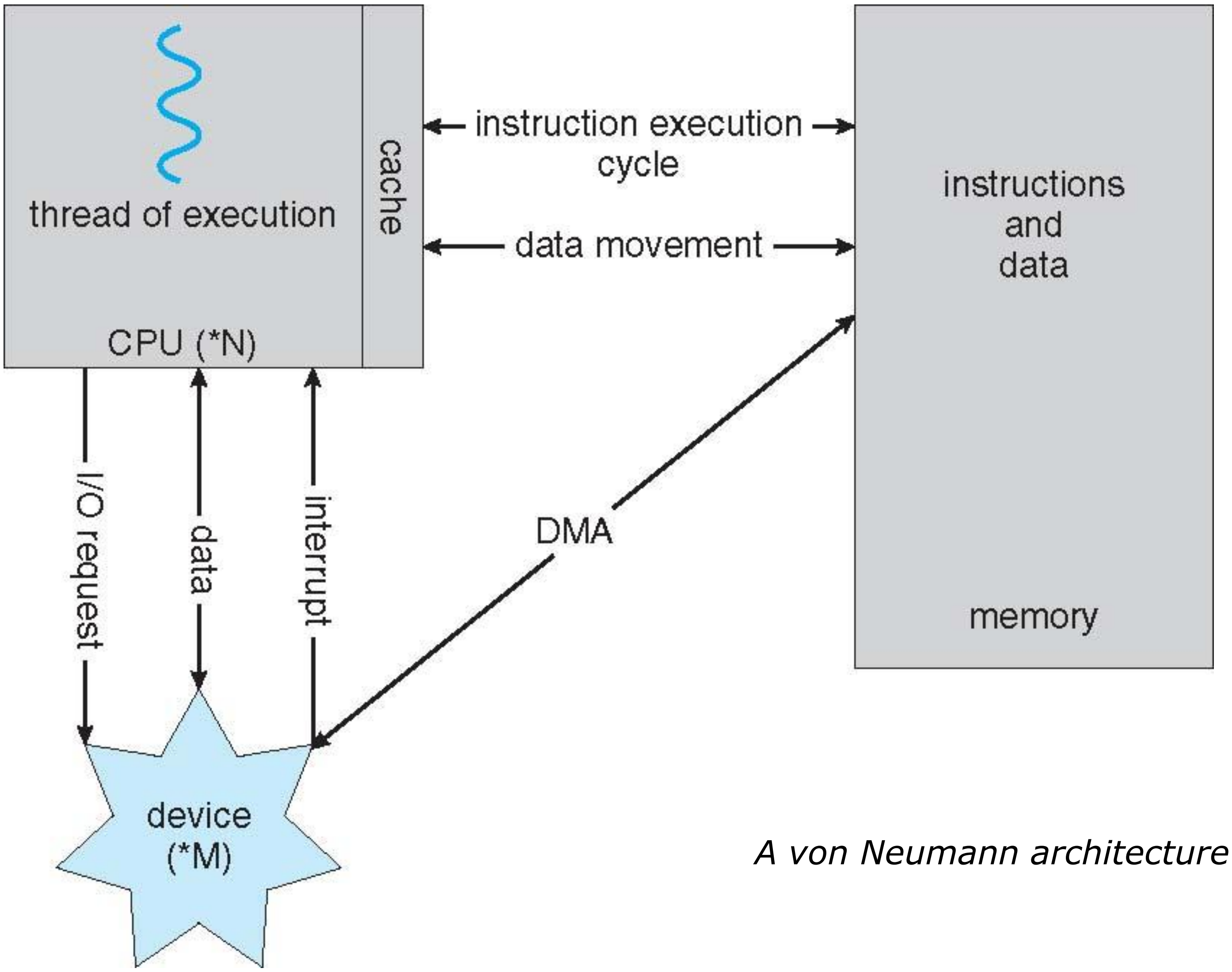
Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (fast)
  - If not, data copied to cache and used there
- Cache smaller than storage being cached
  - Cache management important design problem
  - Cache size and replacement policy

Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention
- Only one interrupt is generated per block, rather than the one interrupt per byte

How a Modern Computer Works



A von Neumann architecture

Evolution of OS

- Mainframe Systems
  - Batch Systems
  - Multiprogrammed Systems
  - Time sharing Systems
- Desktop Systems
- Multiprocessor Systems
- Distributed Systems
  - Client Server systems
  - Peer-to-Peer systems
- Clustered Systems
- Real-Time Systems
- Hand Held Systems

Evolution of OS

Mainframe Systems – Simple Batch Systems:

- User prepare a job and submit it to a computer operator, get output some time later
- No interaction between the user and the computer system
- Operator batches together jobs with similar needs to speedup processing

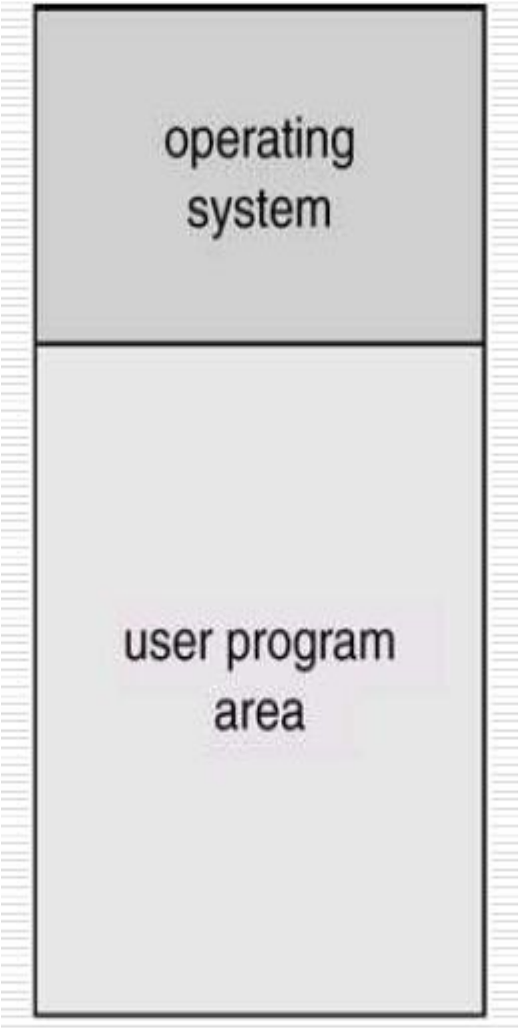
Task of OS: automatically transfers control from one job to another.

- OS always resident in memory

Disadvantages of one job at a time:

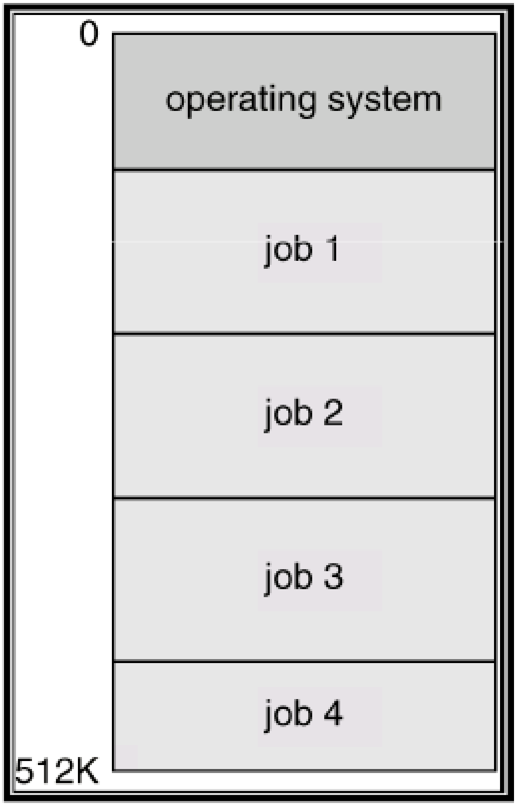
- CPU idle during I/O
- I/O devices idle when CPU busy
- OS is a resident monitor
  - initial control in monitor
  - control transfers to job
  - when job completes control transfers back to monitor

Memory Layout for a Simple Batch System



Multiprogrammed Batch Systems

- Several jobs are kept in main memory at the same time, and the
- CPU is multiplexed among them



OS Features in a Multiprogrammed System

- OS made decisions for users.
- Job Scheduling
  - Choose the jobs from the job pool to be loaded into Memory

- CPU Scheduling
  - Choosing the job to be run from a list of jobs ready to run at the same time.

Time-Sharing Systems–Interactive Computing

- Logical extension of Multiprogramming
- CPU executes multiple jobs by switching but the switching occurs so fast, that the user can interact with the program.
- Supports multiple users – little CPU time for every
- user- illusion that the system is dedicated to a single user.
- **Process: program in execution**

Time-Sharing Systems – Contd.

- Interactive (action/response)
  - when OS finishes execution of one command, it seeks the next control statement from user.
  - Eg: Switches jobs, when the current job needs input from the user who is slow.
- File systems
  - Resides on a collection of disks – disk mgmt is necessary.
- Virtual memory
  - Job is swapped in and out of memory to disk.

Desktop Systems

- *Personal computers – computer system dedicated to a* single user.
- I/O devices – keyboards, mouse, display screens, small printers.
- Single user systems may not need advanced CPU and peripheral utilization.
- So concentrates on user convenience and responsiveness.
- Due to the growth of intranets and internets, file protection feature was adopted.
- May run several different types of operating systems (Windows, MacOS, UNIX, Linux)

Multiprocessor Systems

- Also known as parallel systems or tightly coupled systems
- More than one processor in close communication, sharing computer bus, clock, memory, and usually peripheral devices
- Communication usually takes place through the shared memory.
- **Advantages**
- Increased throughput
- Economy of scale: cheaper than multiple single-processor systems
- Increased reliability: graceful degradation, fault tolerant

Multiprocessor Systems

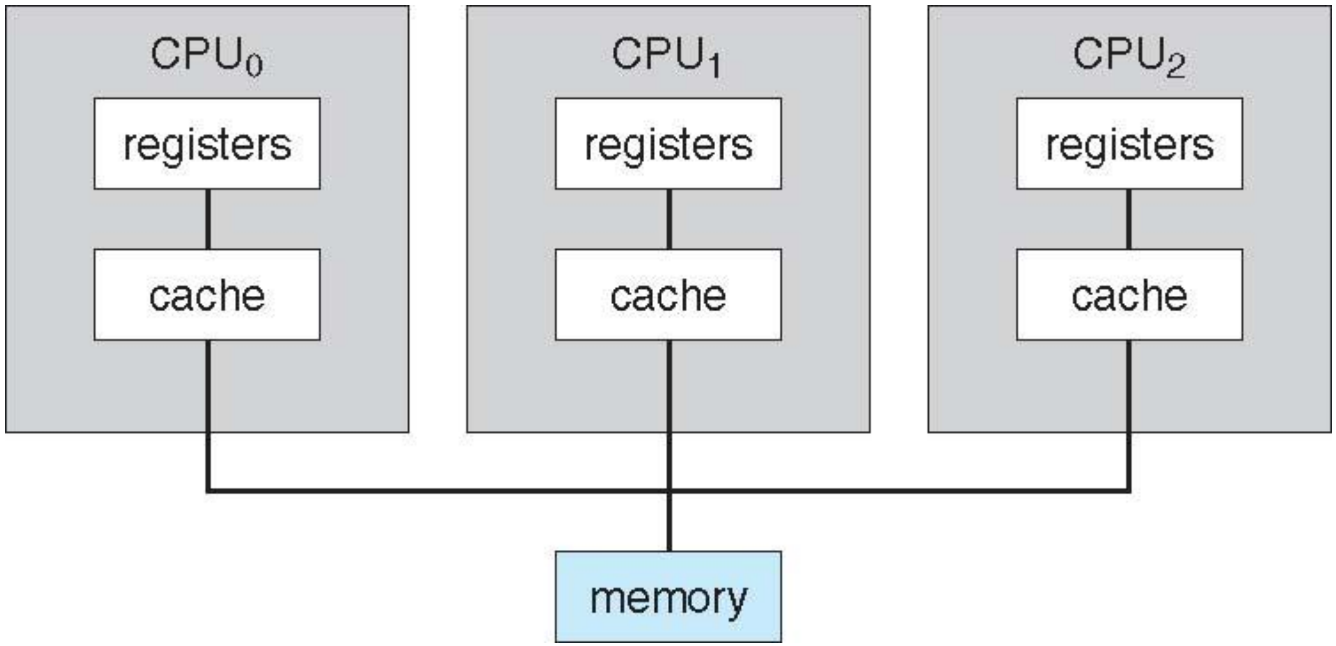
Symmetric multiprocessing (SMP)

- Each processor runs an identical copy of the operating system.
- All processors are peers: any processor can work on any task
- OS can distribute load evenly over the processors.
- Most modern operating systems support SMP

Asymmetric multiprocessing

- Master-slave relationship: a master processor controls the system, assigns works to other processors
- Each processor is assigned a specific task. Don't have the flexibility to assign processes to the least loaded CPU
- More common in extremely large systems

Symmetric Multiprocessing Architecture



Distributed Systems

- Based on the concept of networking
- Distribute the computation among several physical processors.
- ***Loosely coupled system*** *– each processor has its own* local memory; processors communicate with one another through various communications lines, such as high-speed buses or telephone lines.

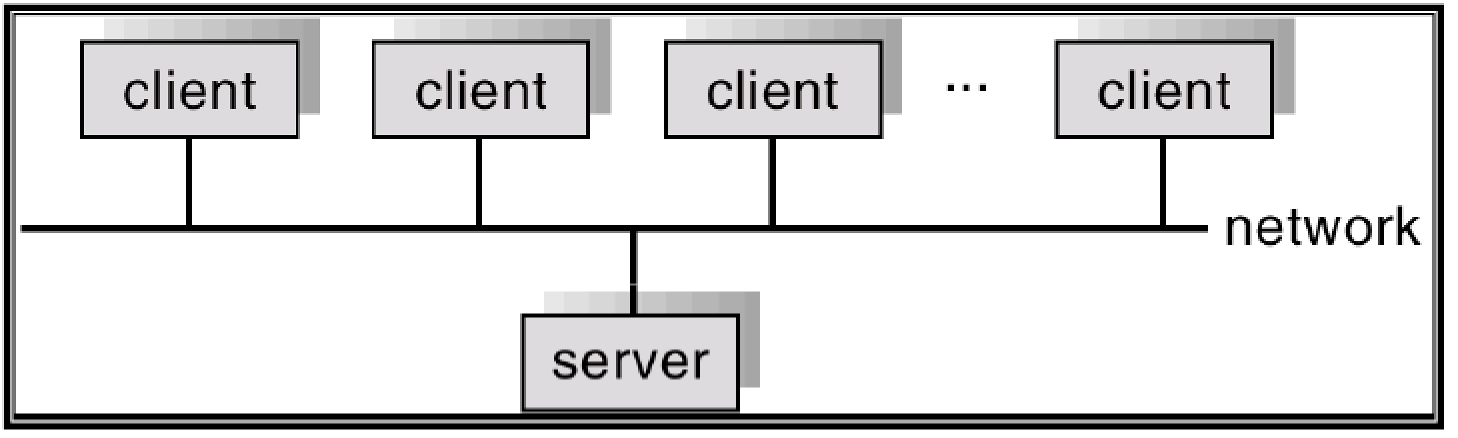
Advantages of distributed systems.

- Resource Sharing
- Computation speed up – load sharing
- Reliability

Distributed Systems – contd.

- Requires networking infrastructure.
- Local area networks (LAN) or Wide area networks (WAN)
- Two types:
  - client-server
    - Compute – severs
    - File-servers
  - peer-to-peer systems.

General Structure of Client-Server



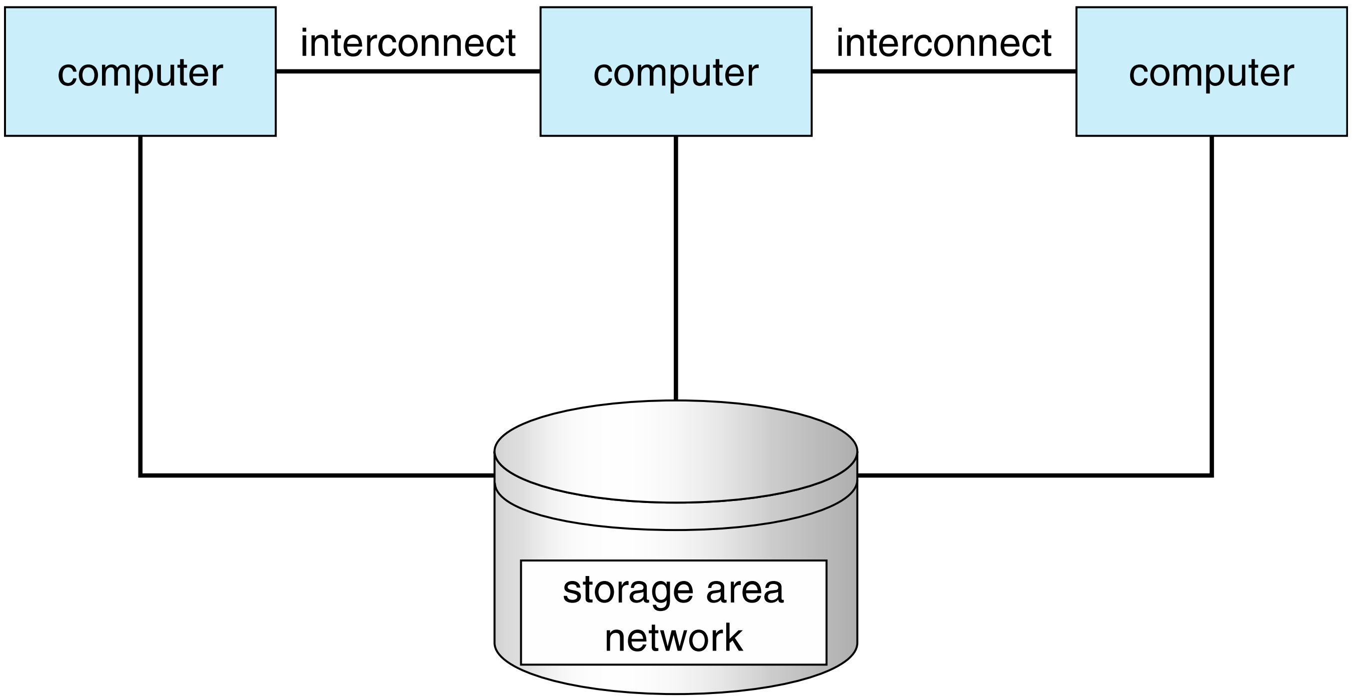
Clustered Systems

- Multiple CPUs to accomplish work but two or more systems are coupled together.
- Provides high reliability.
- *Asymmetric clustering:*
  - one server runs the application while the other server is in hot standby mode monitoring.
- *Symmetric clustering:*
  - all N hosts are running the application and monitoring each other.

Clustered Systems

- Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a **storage-area network (SAN)**
  - Provides a **high-availability** service which survives failures
    - **Asymmetric clustering** has one machine in hot-standby mode
    - **Symmetric clustering** has multiple nodes running applications, monitoring each other
  - Some clusters are for **high-performance computing (HPC)**
    - Applications must be written to use **parallelization**
  - Some have **distributed lock manager** (**DLM**) to avoid conflicting operations

Clustered Systems



Operating System Structure

- **Multiprogramming** (**Batch system**) needed for efficiency
  - Single user cannot keep CPU and I/O devices busy at all times
  - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
  - A subset of total jobs in system is kept in memory
  - One job selected and run via **job scheduling**
  - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing** (**multitasking**) is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
  - **Response time** should be < 1 second
  - Each user has at least one program executing in memory ⇨**process**
  - If several jobs ready to run at the same time ⇨ **CPU scheduling**
  - If processes don't fit in memory, **swapping** moves them in and out to run
  - **Virtual memory** allows execution of processes not completely in memory

Operating-System Operations

- **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, processes modifying each other or the operating system

Operating-System Operations (cont.)

- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time