# CECS 326 Operating Systems

## Introduction

Hailu Xu

California State University Long Beach

# Introduction

- Computer System Structure
- What Operating Systems Do
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management
- Kernel Data Structures
- Computing Environments
- Open-Source Operating Systems

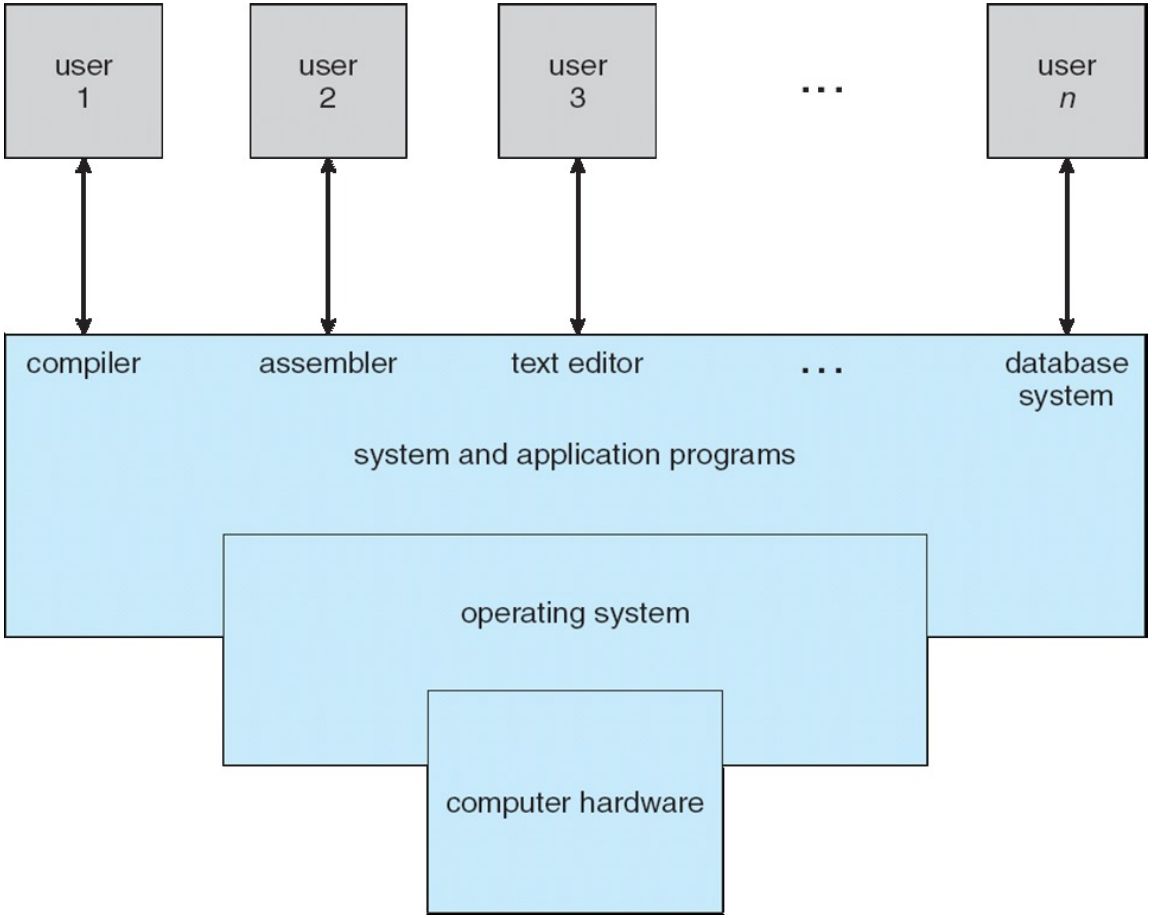
# Objectives

- To describe the basic organization of computer systems
- To provide a grand tour of the major components of operating systems
- To give an overview of the many types of computing environments
- To explore several open-source operating systems

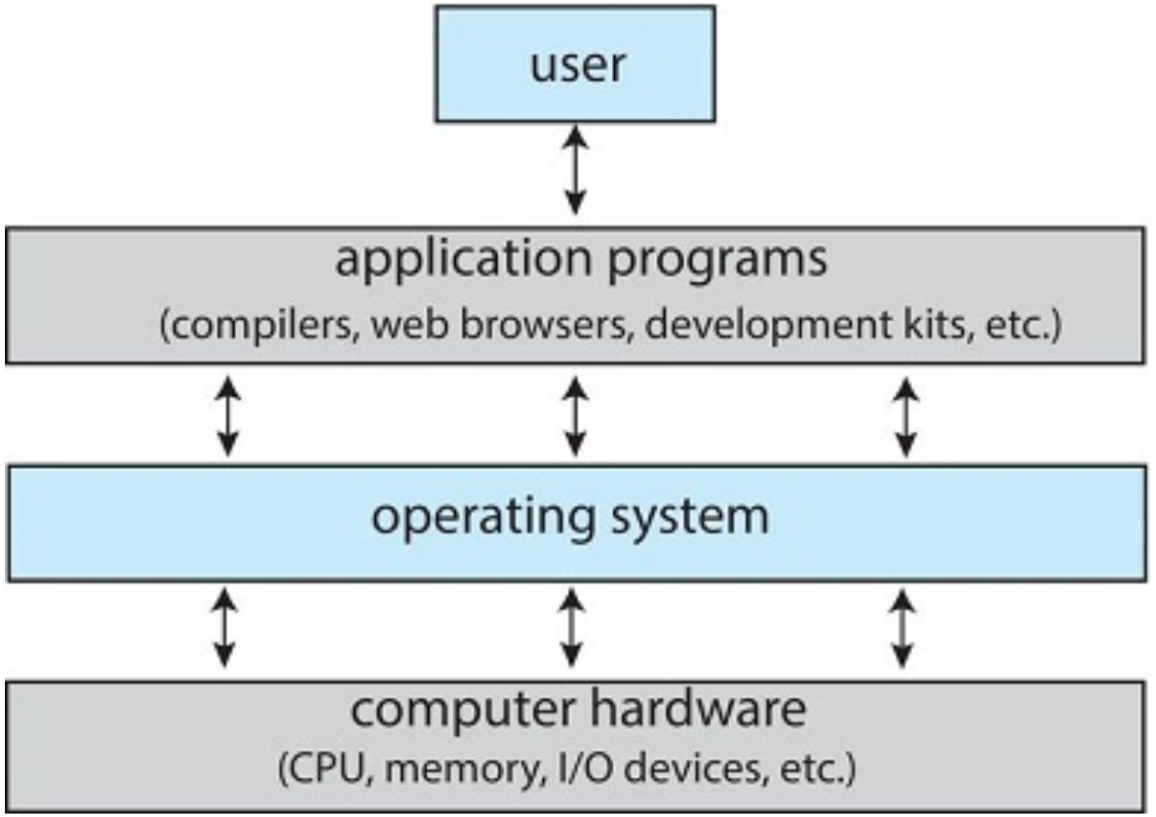
# Computer System Structure

- 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

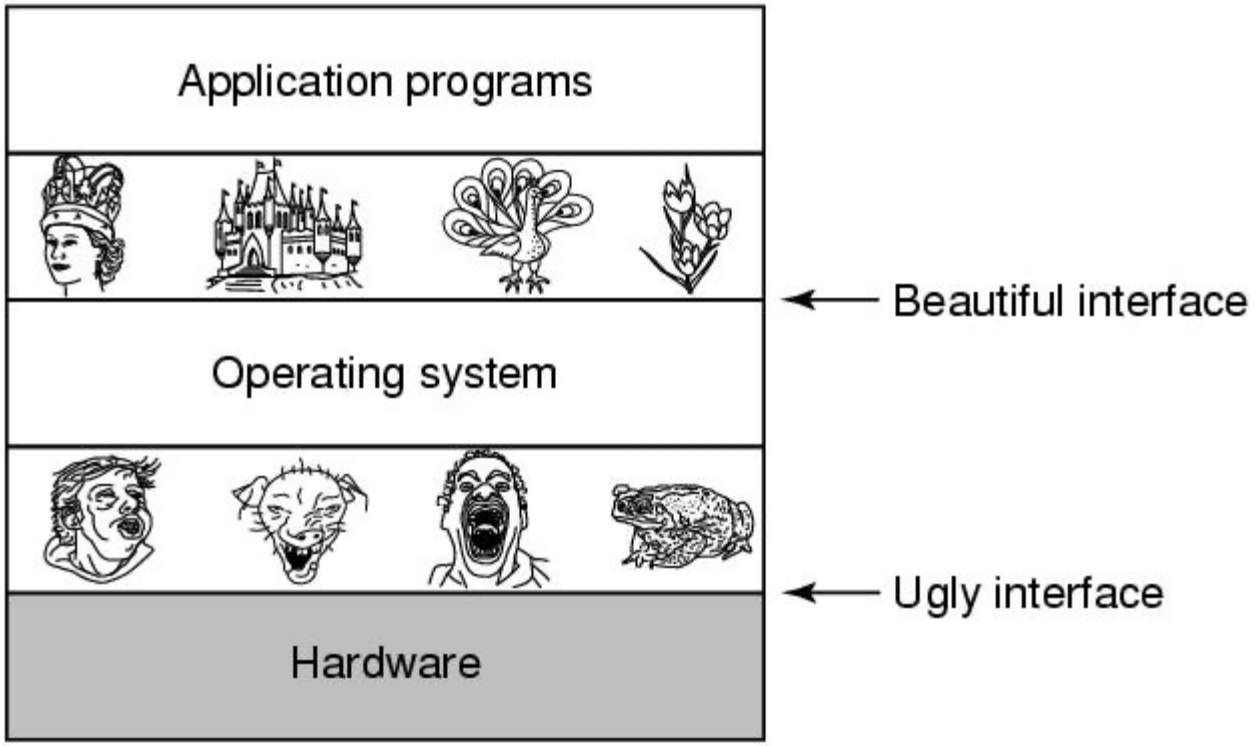
# Four Components of a Computer System



# Computer System Structure



# The Operating System as an Extended Machine



# What is an Operating System?

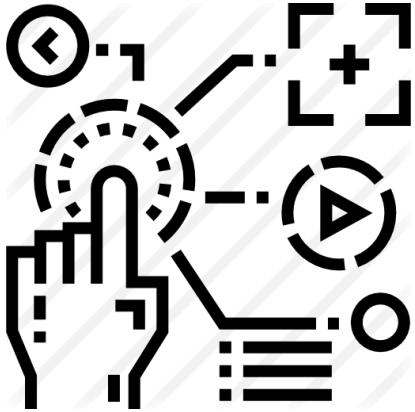
- A **program** that acts as an intermediary between a user of a computer and the computer hardware
- Operating system goals:
  - Execute user programs and make solving user problems easier
  - Make the computer system convenient to use
  - Use the computer hardware in an efficient manner

# What Operating Systems Do

- Depends on the point of view
- Users want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicate systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

# Operating System Definition

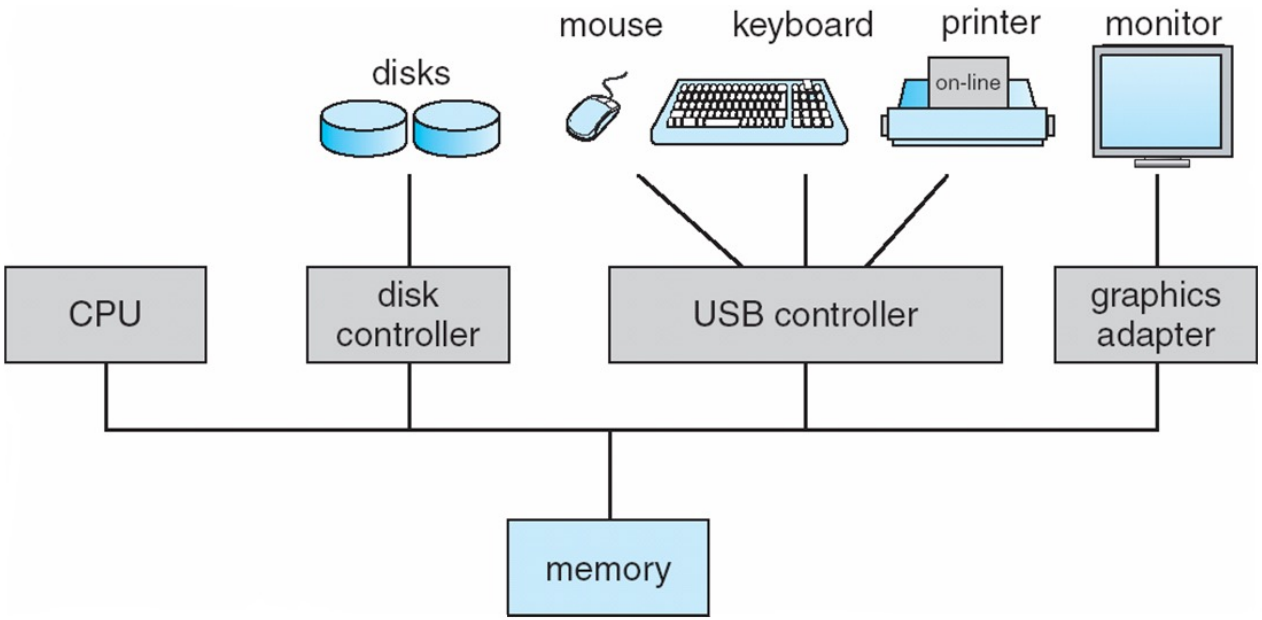
- OS is a **resource allocator**
  - Manages all resources
  - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
  - Controls execution of programs to prevent errors and improper use of the computer




What do we learn?
Operating system
What is OS?
Resource allocator
And ?
Control program

# 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

# 1.1.1 Why Interrupts?

- An operating system is **interrupt driven**
  - OS preserves state of CPU by storing registers and program counter
  - Determines which type of interrupt has occurred through polling of a vectored interrupt system
  - Different code performed action defined for different type of interrupt
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request

# 1.1.1 CPU – the brain

- **Central processing unit (CPU)**, also called a **central processor**, **main processor** or just **processor**,
- executes instructions that make up a computer program
- fetch, decode and execute steps in their operation, which are collectively known as the instruction cycle

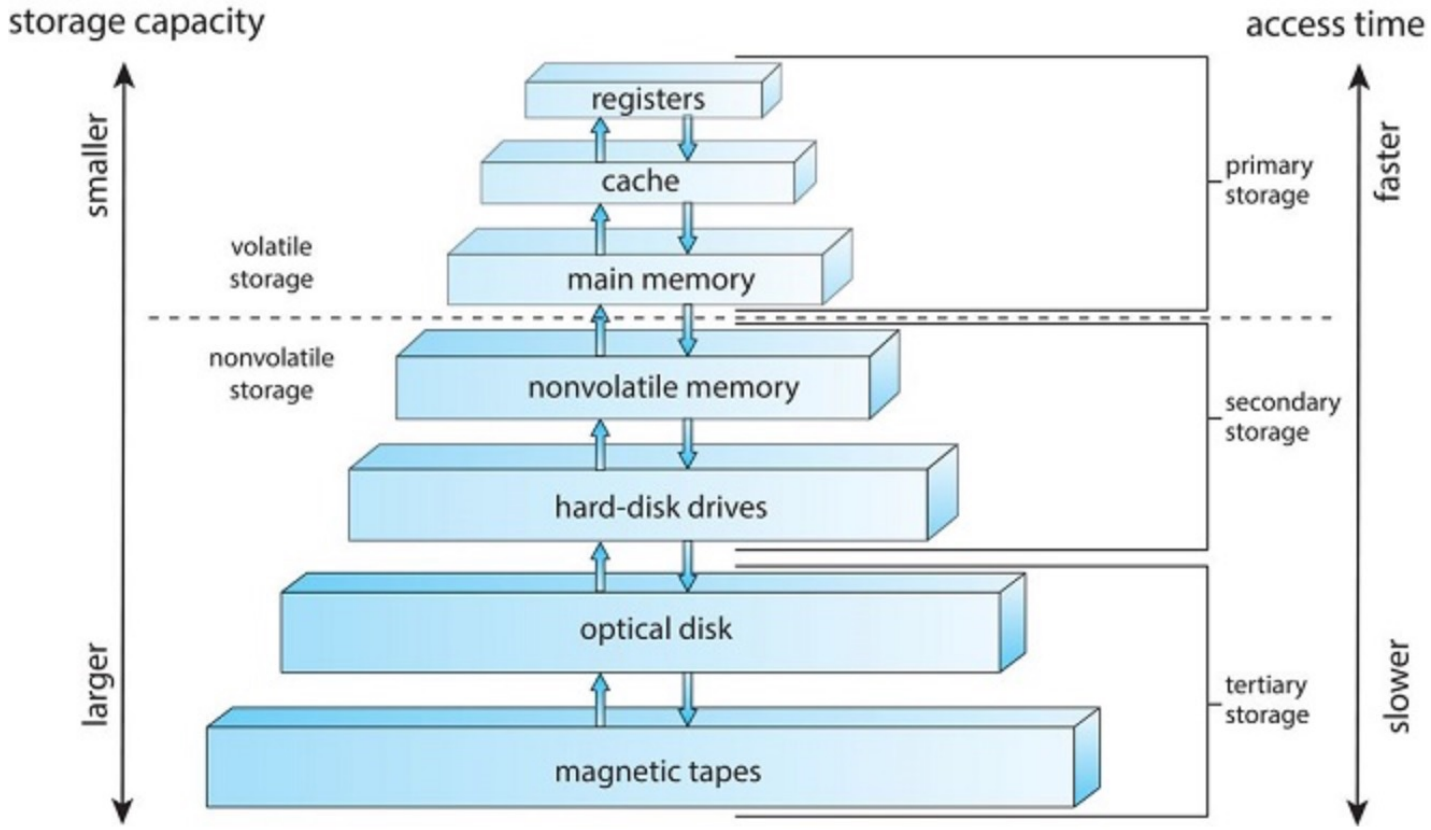
# 1.1.2 Storage Structure

- Main memory – only large storage media that the CPU can access directly
  - **Random access memory (RAM)**
  - Typically **volatile**
- Secondary storage – extension of main memory that provides large **nonvolatile** storage capacity
  - Hard disk drive (HD) – rigid metal or glass platters covered with magnetic recording material
    - Disk surface is logically divided into **tracks**, which are subdivided into **sectors**
  - **Solid-state disks** – faster than hard disks, nonvolatile (**We love it !!**)
    - Various technologies
    - Becoming more popular

# Storage Hierarchy

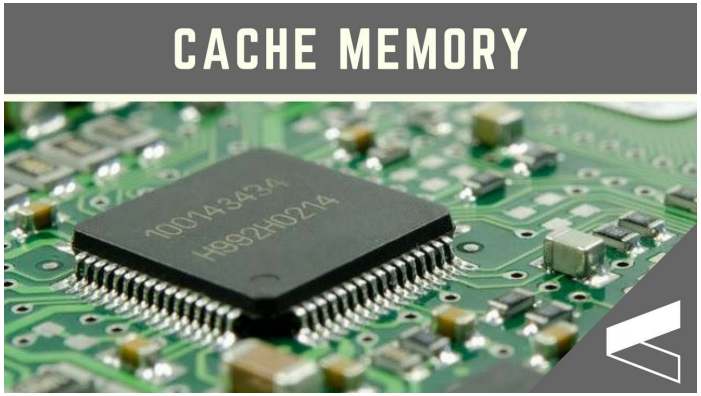
- Storage systems organized in hierarchy
  - Speed
  - Cost
  - Volatility
- **Caching** – copying information into faster storage system; main memory can be viewed as a cache for secondary storage
- **Device Driver** for each device controller to manage I/O

# Storage-Device Hierarchy



# Caching

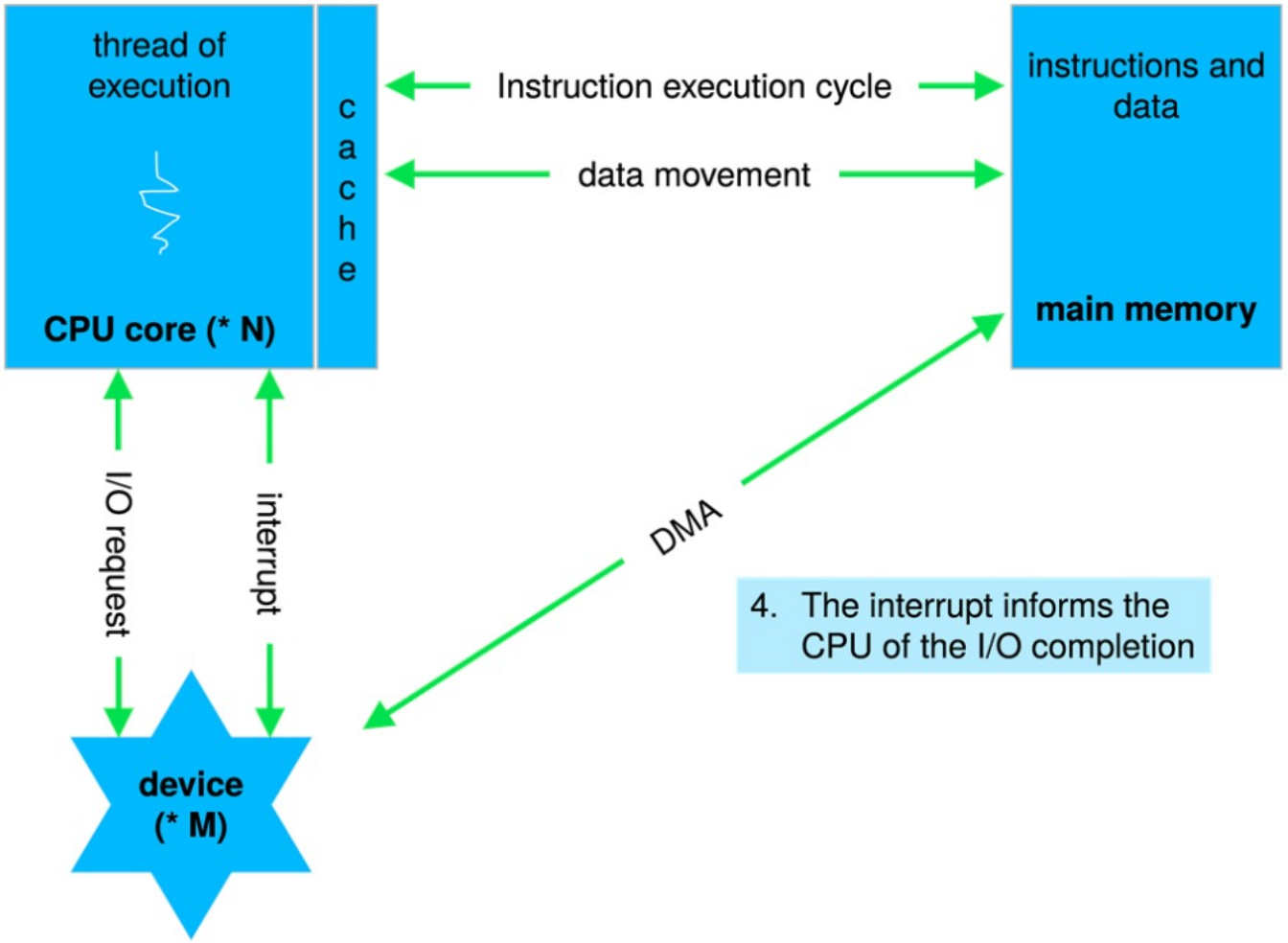
- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
  - If it is, information used directly from the cache (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

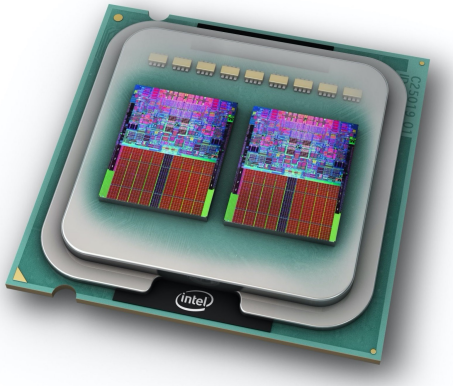
# How a Modern Computer Works



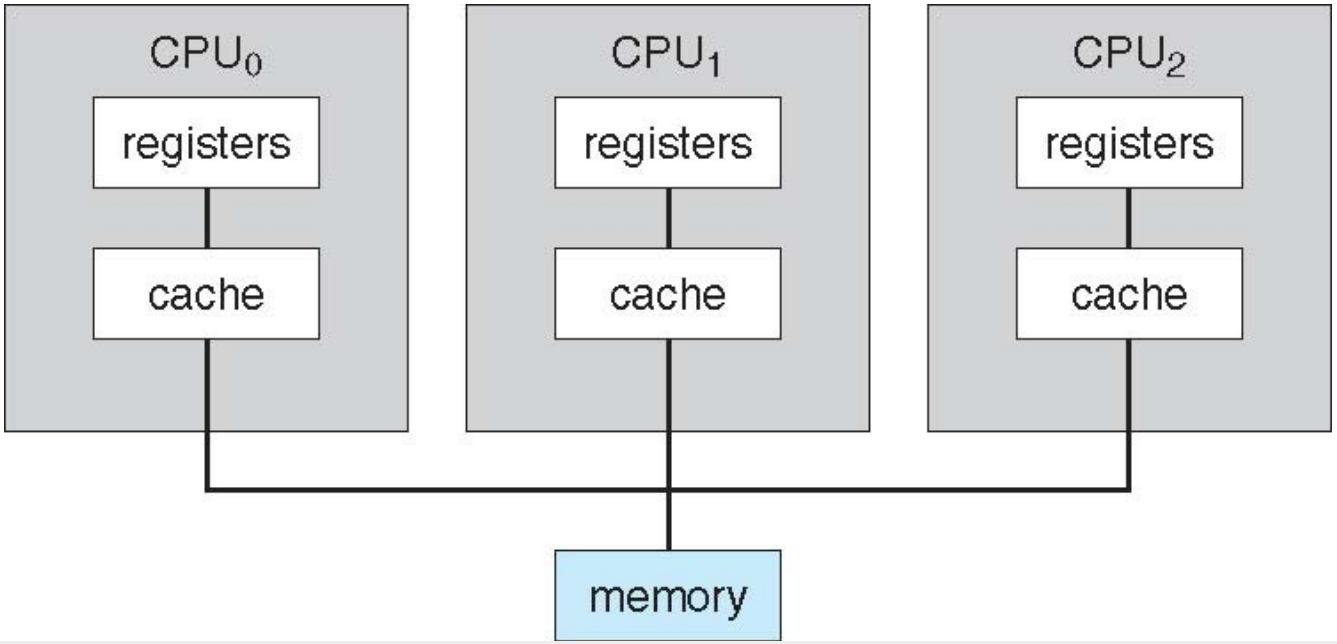

*A von Neumann architecture*

# 1.2 Computer-System Architecture

- **Single-processor system**
- **Multiprocessors system**
  - Growing in use and importance
  - Advantages include:
    1. Increased throughput
    2. Economy of scale
    3. Increased reliability – graceful degradation or fault tolerance



# Computer-System Architecture



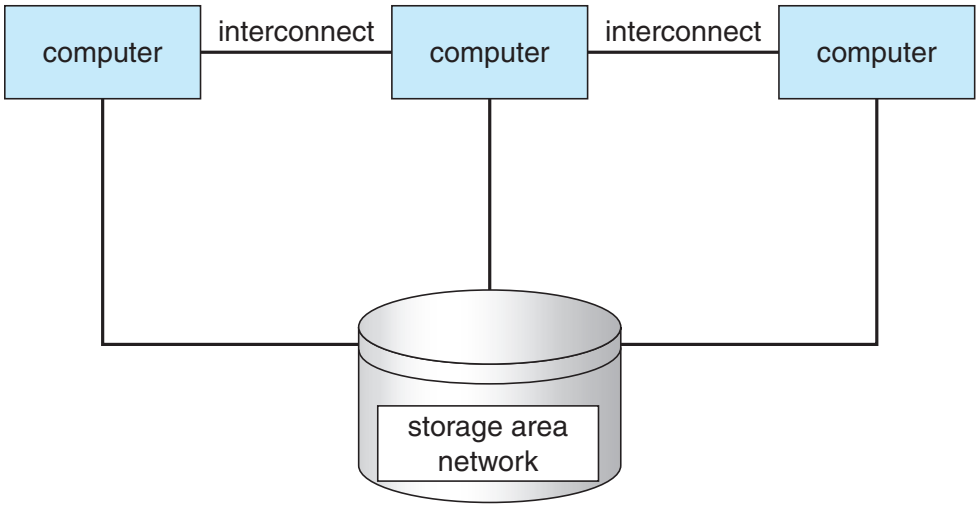

**DEFINITIONS OF COMPUTER SYSTEM COMPONENTS**

- **CPU**—The hardware that executes instructions.
- **Processor**—A physical chip that contains one or more CPUs.
- **Core**—The basic computation unit of the CPU.
- **Multicore**—Including multiple computing cores on the same CPU.
- **Multiprocessor**—Including multiple processors.

Although virtually all systems are now multicore, we use the general term ***CPU*** when referring to a single computational unit of a computer system and ***core*** as well as ***multicore*** when specifically referring to one or more cores on a CPU.

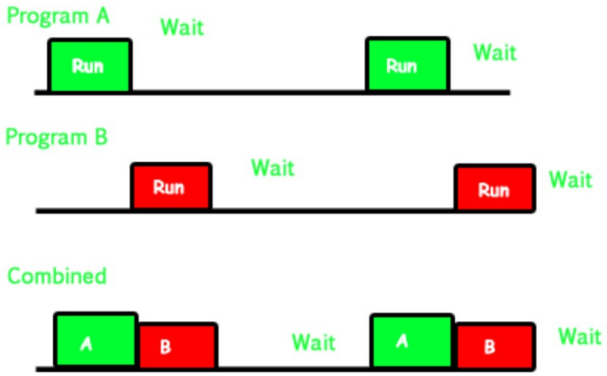
# Computer-System Architecture

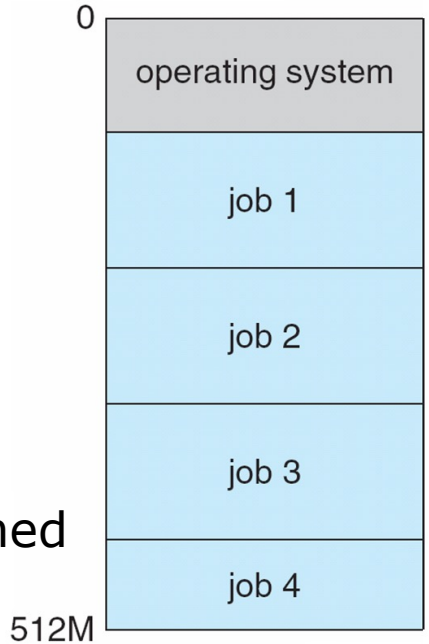
- Clustered systems: Like multiprocessor systems, but multiple systems working together
  - Usually sharing storage via a storage-area network (SAN)
  - Provides a high-availability service which survives failures
  - Some clusters are for high-performance computing (HPC)
    - Applications must be written to use parallelization




# 1.3 Operating System Structure

- **Multiprogramming** (**Batch system**)
  - 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

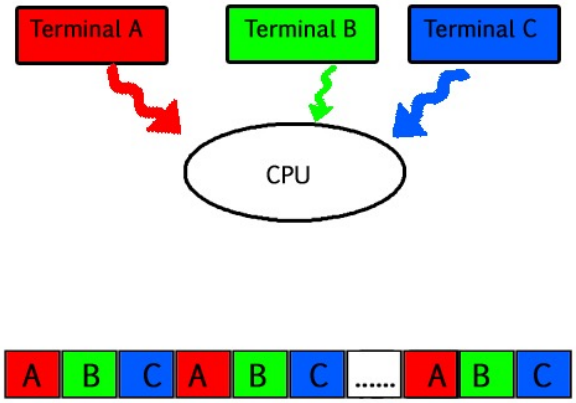



Memory Layout for Multiprogrammed System

# 1.3 Operating System Structure

- **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



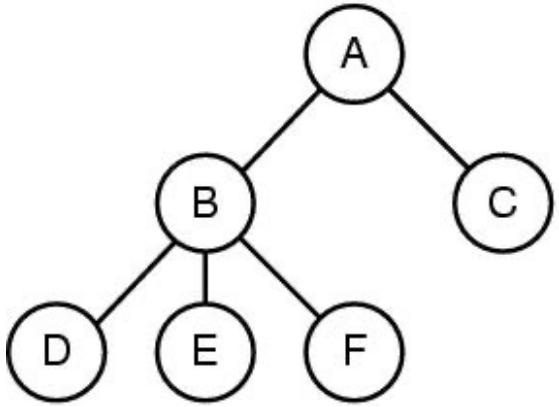
# 1.4 Resource Management

- Process Management
- Memory Management
- Storage Management
- I/O Management

# 1.4 Process Management

## What is a process?

- Program in execution
- Lives in address space
- Process table
  - Keeps info about process
  - Used to re-start process
- Can communicate with one another




Process creates child processes

Tree structure

# 1.4 Process Management

- Process needs resources to accomplish its task
  - CPU, memory, I/O, files
  - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
  - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread

# Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and when
  - Optimizing CPU utilization and computer response to users
- Memory management activities
  - Keeping track of which parts of memory are currently being used and by whom
  - Deciding which processes (or parts thereof) and data to move into and out of memory
  - Allocating and deallocating memory space as needed

# Storage Management

- OS provides uniform, logical view of information storage
  - Abstracts physical properties to logical storage unit - **file**
  - Each medium is controlled by device (i.e., disk drive, tape drive)
    - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
  - Files usually organized into directories
  - Access control on most systems to determine who can access what
  - OS activities include
    - Creating and deleting files and directories
    - Primitives to manipulate files and directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

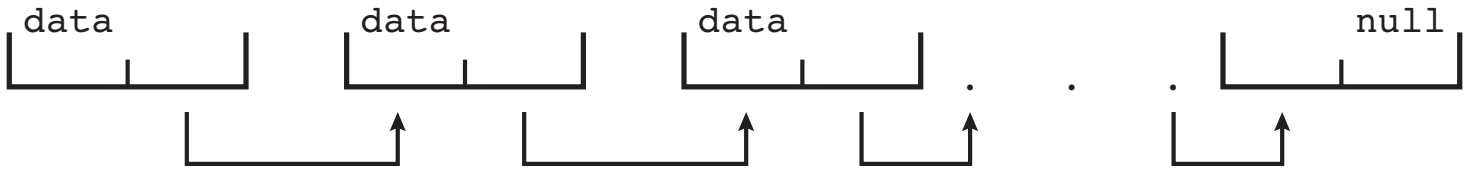
# Performance of Various Levels of Storage

| Level | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Name | registers | cache | main memory | solid state disk | magnetic disk |
| Typical size | < 1 KB | < 16MB | < 64GB | < 1 TB | < 10 TB |
| Implementation technology | custom memory with multiple ports CMOS | on-chip or off-chip CMOS SRAM | CMOS SRAM | flash memory | magnetic disk |
| Access time (ns) | 0.25 - 0.5 | 0.5 - 25 | 80 - 250 | 25,000 - 50,000 | 5,000,000 |
| Bandwidth (MB/sec) | 20,000 - 100,000 | 5,000 - 10,000 | 1,000 - 5,000 | 500 | 20 - 150 |
| Managed by | compiler | hardware | operating system | operating system | operating system |
| Backed by | cache | main memory | disk | disk | disk or tape |

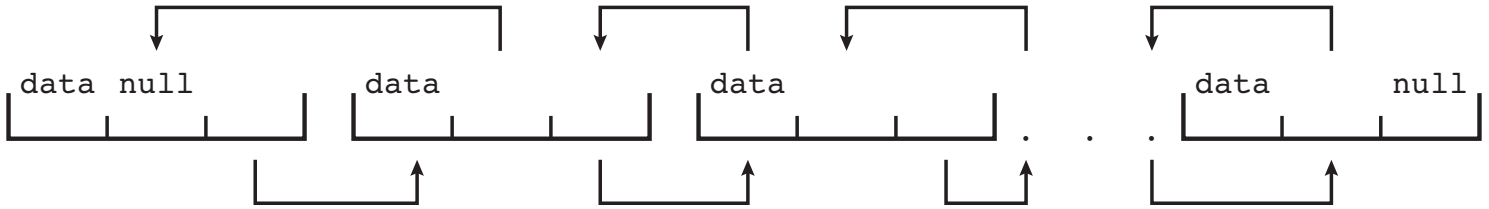
Movement between levels of storage hierarchy can be explicit or implicit

# 1.6 Kernel Data Structures

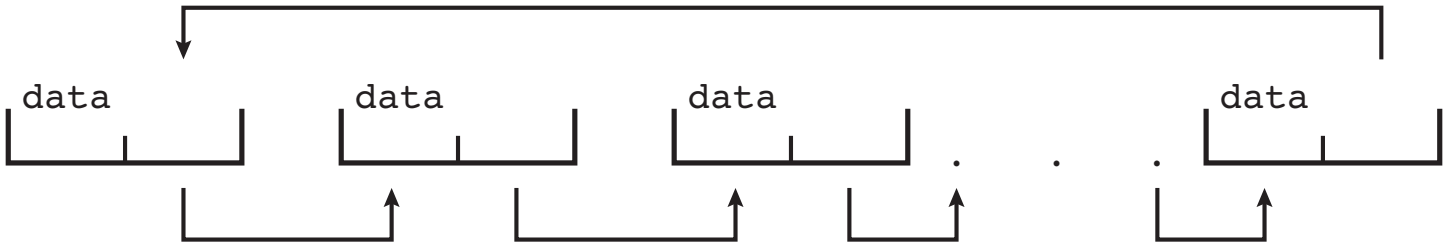
- Many similar to standard programming data structures
- ***Singly linked list***




- ***Doubly linked list***



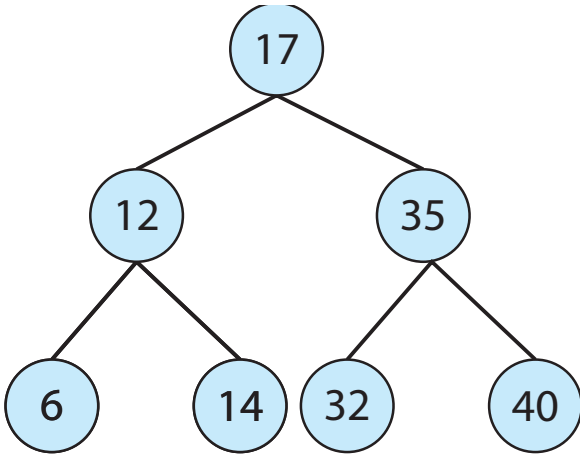

- ***Circular linked list***




- ***Stack and Queue***

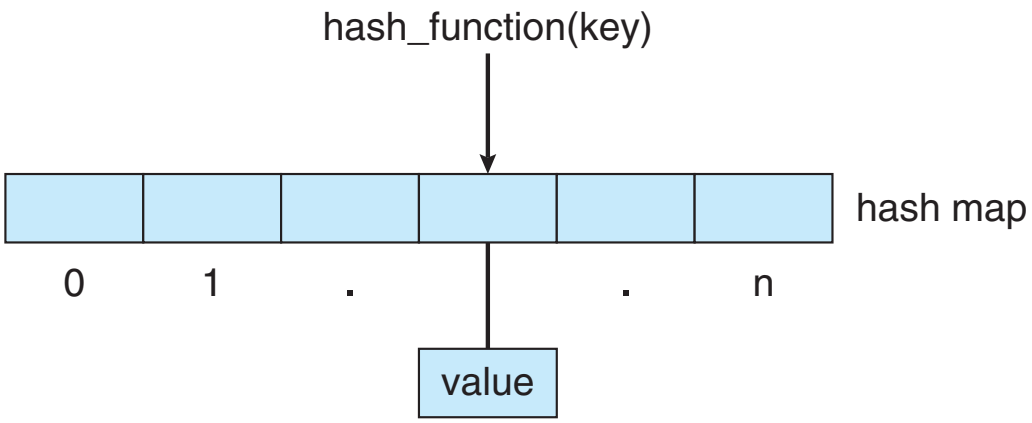
# 1.6 Kernel Data Structures

- **Binary search tree**
  left <= right
  - Search performance is *O(n)*
  - **Balanced binary search tree** is *O(lg n)*



# 1.6 Kernel Data Structures

- **Hash function** can create a **hash map**




- **Bitmap** – string of *n* binary digits representing the status of *n* items
- Linux data structures defined in

  ***include*** files `<linux/list.h>, <linux/kfifo.h>, <linux/rbtree.h>`

# 1.7 Computing Environments - Traditional

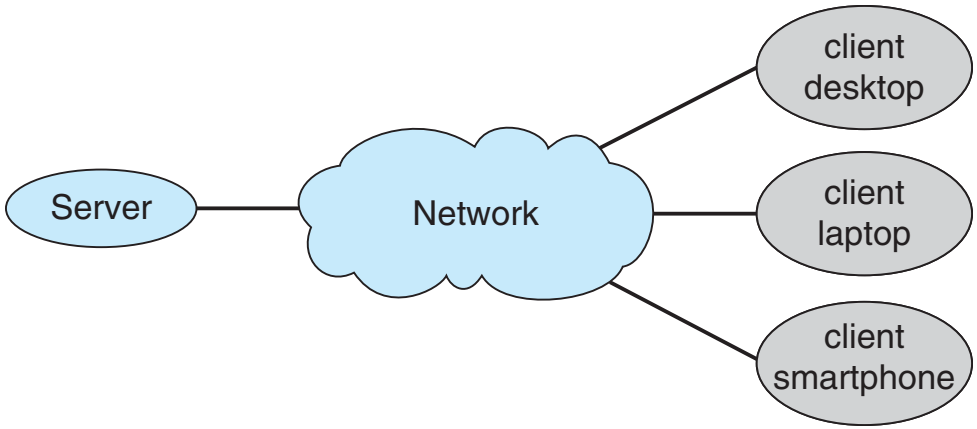
- Stand-alone general purpose machines
- But blurred as most systems interconnect with others (i.e., the Internet)
- Portals provide web access to internal systems
- **Network computers** are like Web terminals
- Mobile computers interconnect via **wireless networks**
- Networking becoming ubiquitous – even home systems use **firewalls** to protect home computers from Internet attacks

# Computing Environments - Mobile

- Handheld smartphones, tablets, etc
- What is the functional difference between them and a “traditional” laptop?
- Extra feature – more OS features (GPS, gyroscope)
- Allows new types of apps like ***augmented reality***
- Use IEEE 802.11 wireless, or cellular data networks for connectivity
- Leaders are **Apple iOS** and **Google Android**

# Computing Environments – Client-Server

- Client-Server Computing
  - Dumb terminals supplanted by smart PCs
  - Many systems now **servers**, responding to requests generated by **clients**
    - **Compute-server system** provides an interface to client to request services (i.e., database)
    - **File-server system** provides interface for clients to store and retrieve files

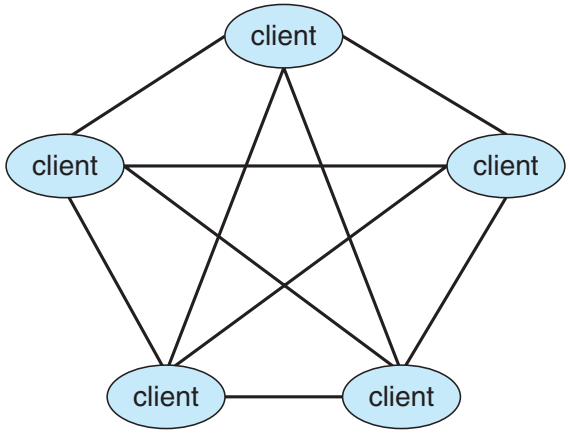


# Computing Environments – Distributed

- Distributed computing
  - Collection of separate, possibly heterogeneous, systems networked together
    - **Network** is a communications path, **TCP/IP** most common
      - Local Area Network (LAN)
      - Wide Area Network (WAN)
      - Metropolitan Area Network (MAN)
      - Personal Area Network (PAN)
  - **Network Operating System** provides features between systems across network
    - Communication scheme allows systems to exchange messages
    - Illusion of a single system

# Computing Environments - Peer-to-Peer

- Another model of distributed system
- P2P does not distinguish clients and servers
  - Instead all nodes are considered peers
  - May each act as client, server or both
  - Node must join P2P network
    - Registers its service with central lookup service on network, or
    - Broadcast request for service and respond to requests for service via ***discovery protocol***
  - Examples include Bitcoin, Ethereum

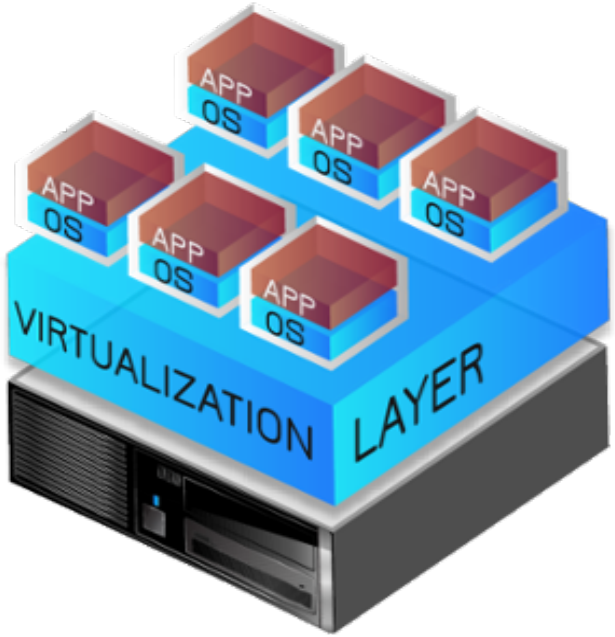



# Computing Environments - Virtualization

- Allows operating systems to run applications within other OSes
  - Vast and growing industry
- **Virtualization** – OS natively compiled for CPU, running **guest** OSes also natively compiled
  - Consider VMware running WinXP guests, each running applications, all on native WinXP **host** OS
  - **VMM** (virtual machine Manager) provides virtualization services


# Computing Environments - Virtualization


**Traditional Server Architecture**



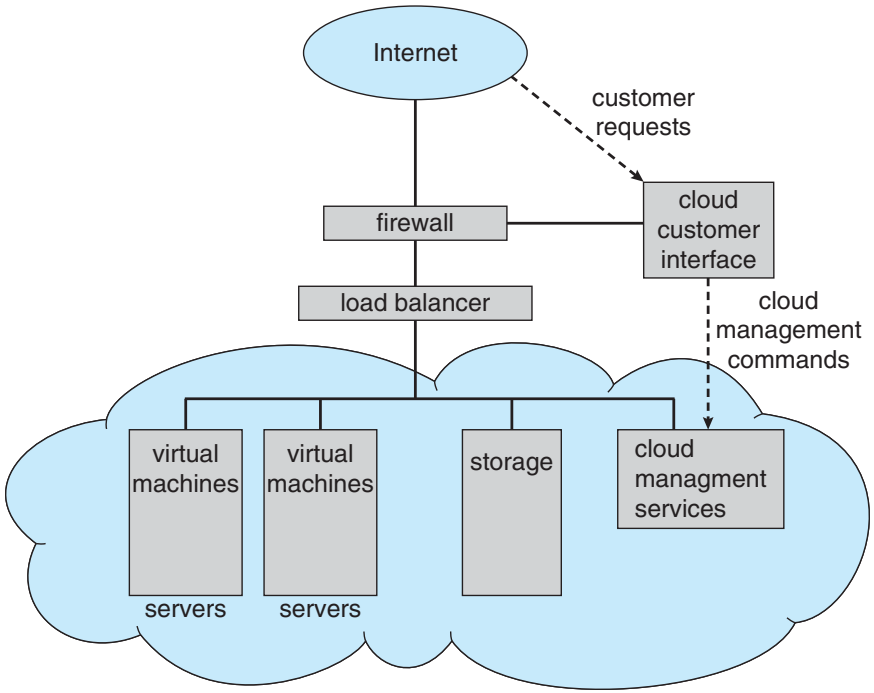

**Virtualized Server Architecture**

# Computing Environments – Cloud Computing

- Delivers computing, storage, even apps as a service across a network
- Logical extension of virtualization because it uses virtualization as the base for it functionality.
  - Amazon **EC2** has thousands of servers, millions of virtual machines, petabytes of storage available across the Internet, pay based on usage
- Many types
  - **Public cloud** – available via Internet to anyone willing to pay
  - **Private cloud** – run by a company for the company's own use
  - **Hybrid cloud** – includes both public and private cloud components
  - Software as a Service (**SaaS**) – one or more applications available via the Internet (i.e., word processor)
  - Platform as a Service (**PaaS**) – software stack ready for application use via the Internet (i.e., a database server)
  - Infrastructure as a Service (**IaaS**) – servers or storage available over Internet (i.e., storage available for backup use)

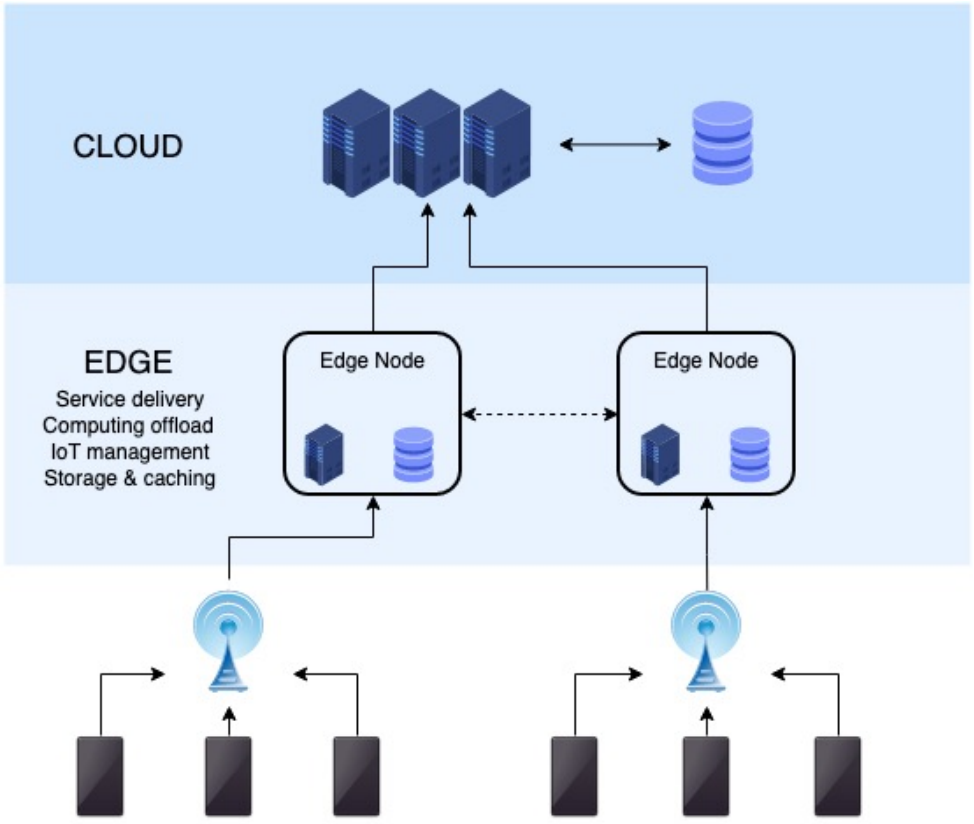
# Computing Environments – Cloud Computing

- Cloud computing environments composed of traditional OSes, plus VMMs, plus cloud management tools
  - Internet connectivity requires security like firewalls
  - Load balancers spread traffic across multiple applications



# Computing Environments – Edge Computing / IoT

- a distributed computing paradigm that brings computation and data storage closer to the location where it is needed, to improve response times and save bandwidth
  - Through virtualization technology
  - run a wider range of applications on the edge servers



# Open-Source Operating Systems

- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

# E.g., Apache Spark

- Spark and its RDDs were developed in 2012
  - Spark Core
  - RDDs
  - Spark SQL
  - Spark Mllib for machine learning
  - Spark Streaming


## Matei Zaharia

Stanford DAWN Lab and Databricks
Verified email at cs.stanford.edu - Homepage

Distributed Systems  Machine Learning  Databases  Security

| | | |
|---|---|---|
| **Spark: Cluster computing with working sets.**<br>M Zaharia, M Chowdhury, MJ Franklin, S Shenker, I Stoica<br>HotCloud 10 (10-10), 95 | 5290 | 2010 |
| **Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing**<br>M Zaharia, M Chowdhury, T Das, A Dave, J Ma, M McCauly, MJ Franklin, ...<br>Presented as part of the 9th {USENIX} Symposium on Networked Systems Design … | 4543 | 2012 |

Cited by VIEW ALL

| | All | Since 2015 |
|---|---|---|
| Citations | 49440 | 34269 |
| h-index | 50 | 44 |
| i10-index | 95 | 87 |