Organization & Architectures

Operating Systems

School of Data & Computer Science
Sun Yat-sen University

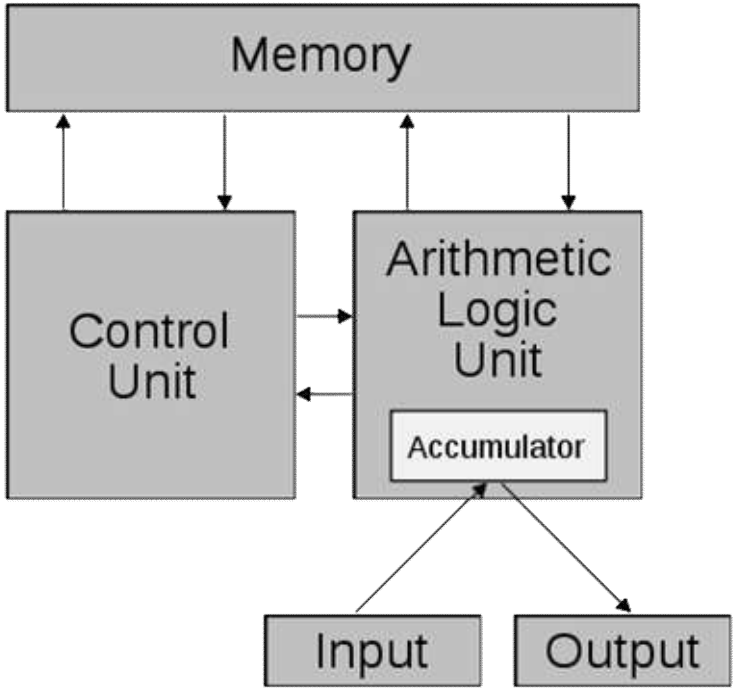
Lecture Notes: os_sysu@163.com
Instructor: Guoyang Cai
email: isscgy@mail.sysu.edu.cn

Content

- Computer Hardware Architectures
  - *von Neumann* Architecture
  - Harvard Architecture
- Computer System Organization
  - Interrupts
  - Storage Structures
  - Main Memory Management
  - I/O Structures
- Computer System Architectures & Environments
  - Multiprogrammed Batch Systems
  - Time-Sharing Systems
  - Real-Time Systems
  - Personal/Desktop Computers
  - Multiprocessor Systems
  - Clustered Systems
  - Networked Systems & Distributed Systems
  - Web-based Systems
  - Handheld Systems & Mobile Systems

Computer Hardware Architectures

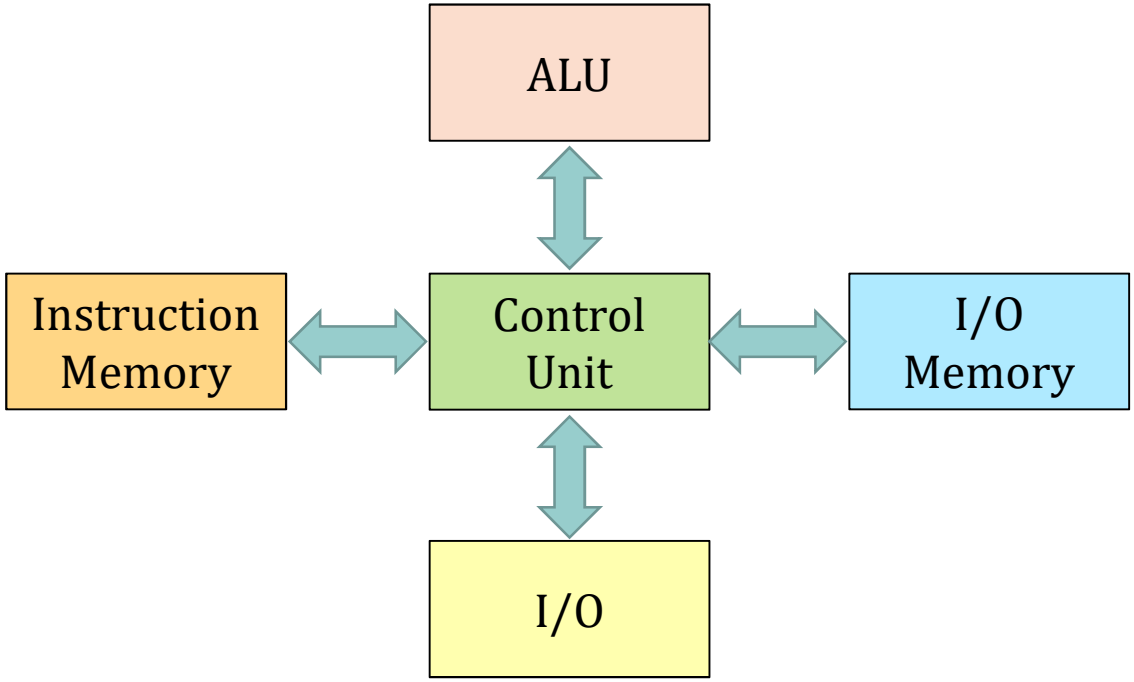
- *von Neumann* Architecture
  - Aka Princeton Architecture.
  - E.g., X86, …



von Neumann Architecture

Computer Hardware Architectures

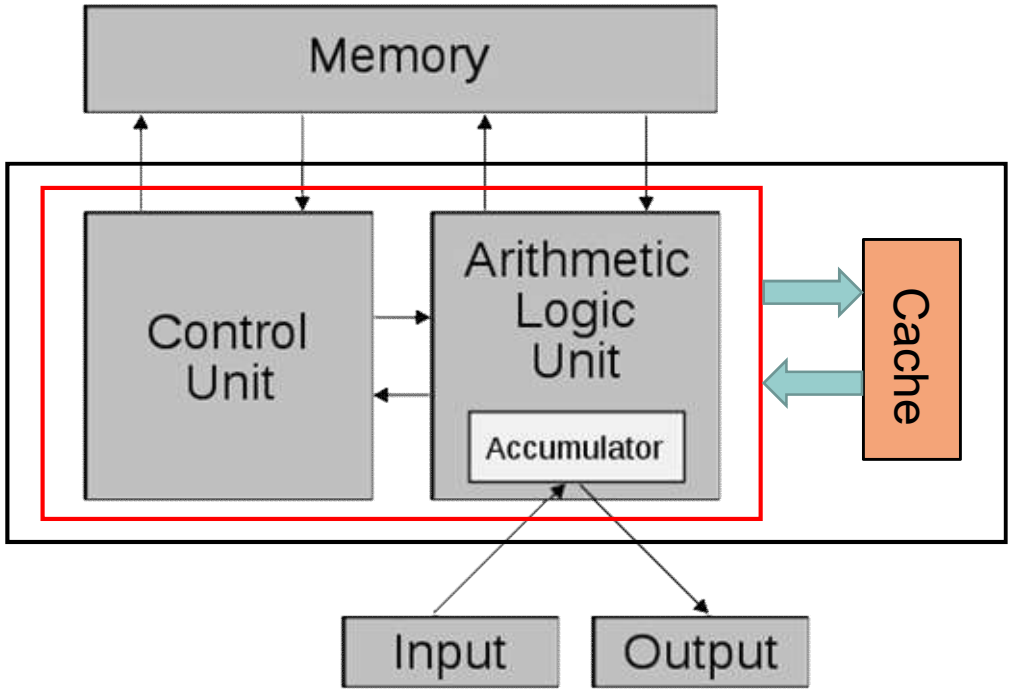
- Harvard Architecture
  - E.g., ARM9, MIC, most of DSP, …
  - Modified Harvard architecture …



Harvard Architecture

Computer Hardware Architectures

- Hybrid Architecture
  - Introducing caching within CPU

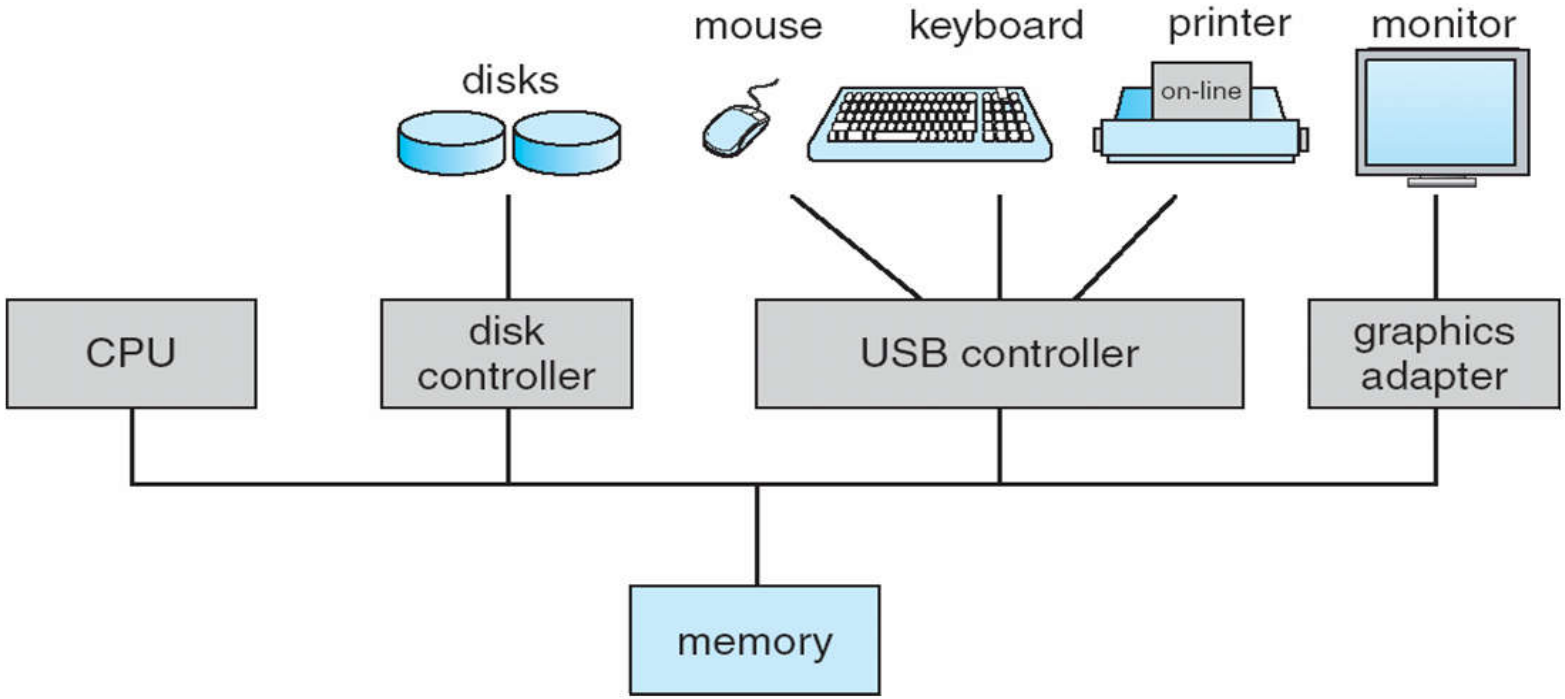


Computer Hardware Architectures

- Basic Attributes
  - Data representation
  - Addressing mode
  - Registers
  - Instruction system
  - Memory system
  - Interrupt controller
  - Input/output controller
  - Information protecting mechanism

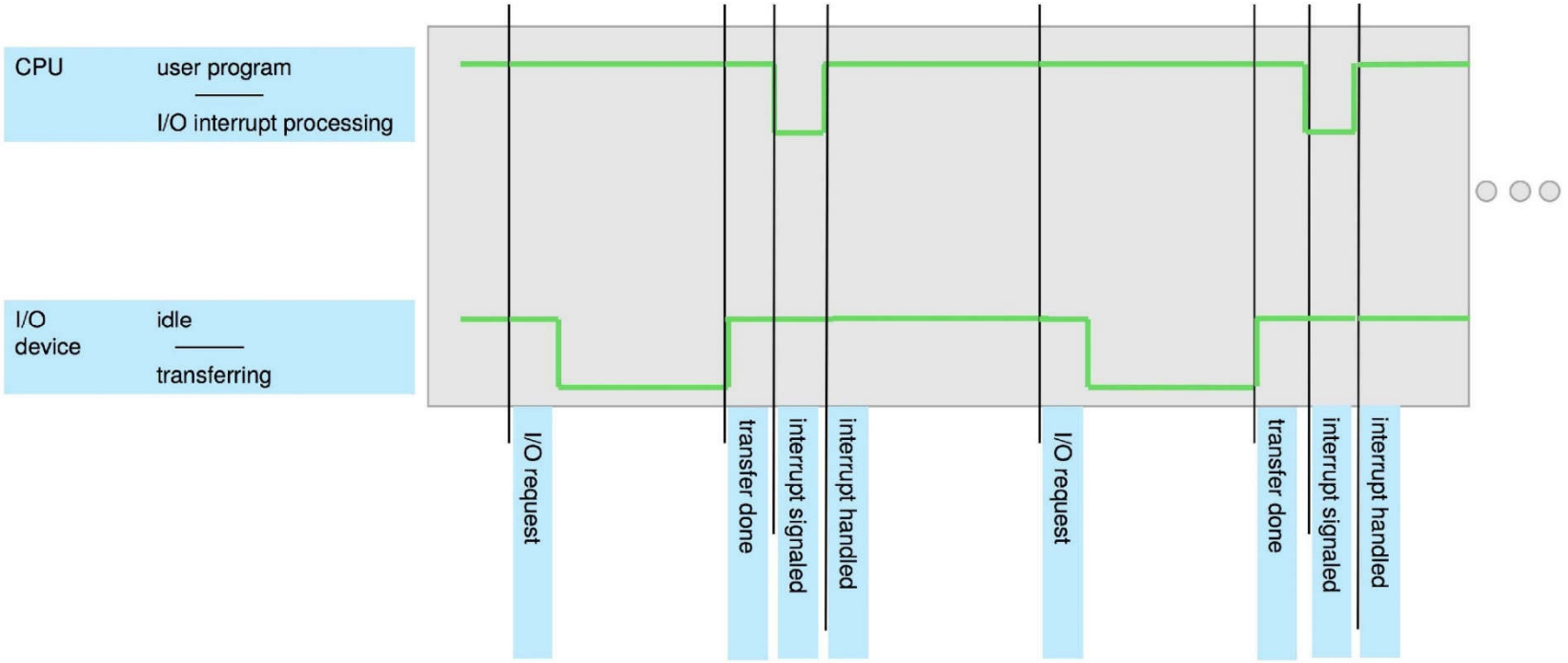
Computer System Organization

- A modern general-purpose computer system consists of one or more *CPUs* and a number of *device controllers* connected through a *common bus* that provides access between components and shared *memory*.
- Operating systems have a *device driver* for each device controller.
- The CPU and the device controllers can execute in parallel, competing for memory cycles.



Interrupts

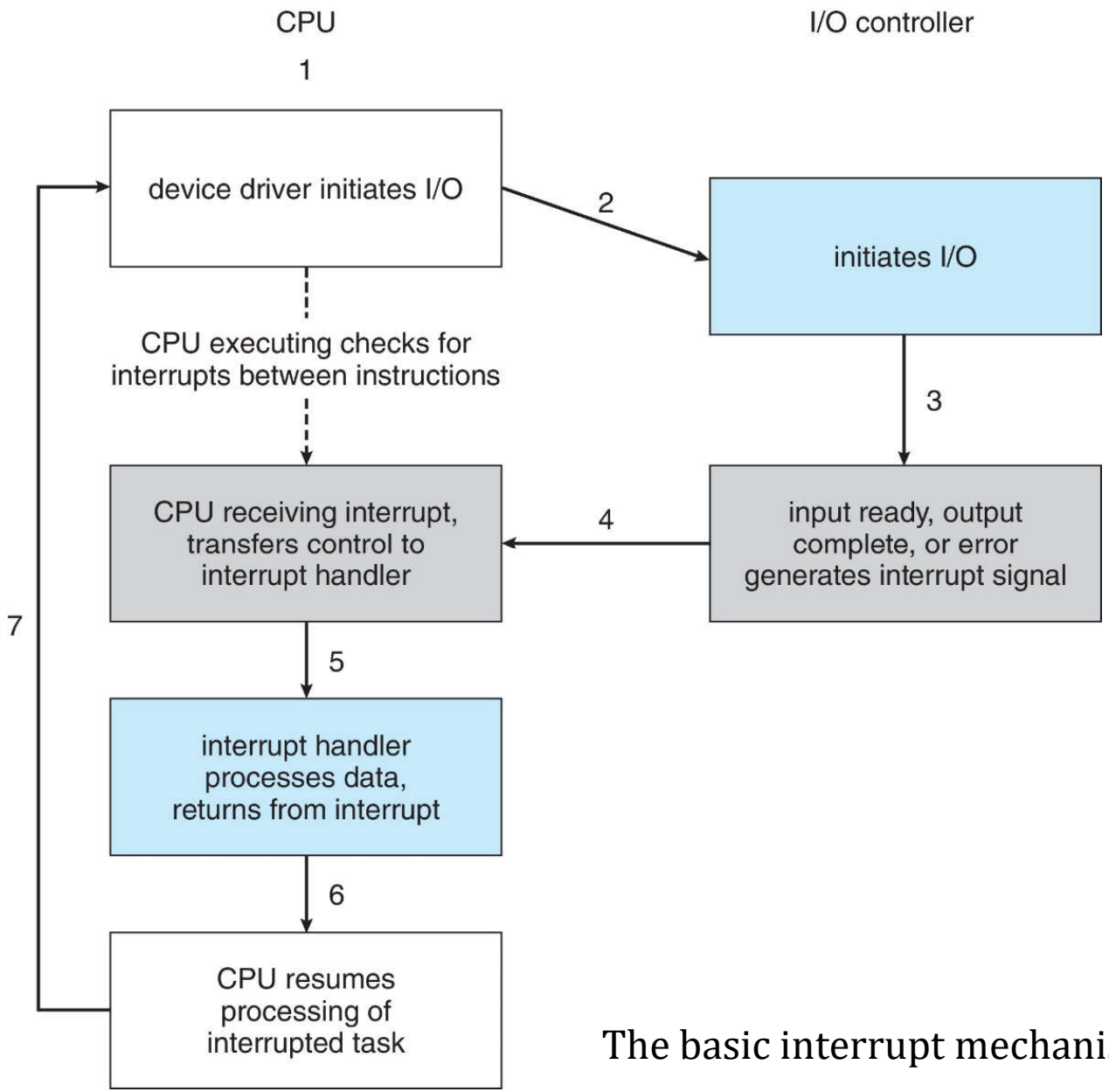
- Interrupts are a key part of how operating systems and hardware *interact.*
  - Hardware may trigger an interrupt at any time by sending a signal to the CPU, usually by way of the system bus.
  - The CPU stops what it is doing and immediately transfers execution to an interrupt service routine (ISR).
  - On completion, the CPU resumes the interrupted computation.



Interrupts

- Interrupts are an important part of a computer architecture. Each computer design has its own interrupt mechanism.
- The basic interrupt mechanism works as follows. It enables the CPU to respond to an *asynchronous* event, as when a device controller becomes ready for service.
  - CPU senses *interrupt-request line* after executing *every instruction*. A controller may assert a signal on the line. CPU reads the interrupt number and jumps to the *interrupt-handler routine*.
    - We say that the device controller ***raises*** an interrupt by asserting a signal on the interrupt request line, the CPU ***catches*** the interrupt and ***dispatches*** it to the interrupt handler, and the handler ***clears*** the interrupt by servicing the device.
  - A table of pointers to ISR addresses (this table aka the *interrupt vector*) is used to provide the necessary speed.
    - The table is stored in low memory. SRAM
    - It is indexed by a unique interrupt number given with the interrupt request.
    - Windows and UNIX dispatch interrupts in this manner.

Interrupts



The basic interrupt mechanism.

Interrupts

- Interrupt Controller
  - In a modern operating system, more sophisticated interrupt handling features are provided by the CPU and the *interrupt-controller* hardware.
    (1) the ability to defer interrupt handling during *critical processing*.
    (2) an *efficient* way to dispatch to the proper interrupt handler for a device.
    (3) *multilevel* interrupts, so that the operating system can distinguish between high- and low-priority interrupts and can respond with the appropriate degree of urgency.
- Maskable and Non-maskable Interrupts
  - Most CPUs have two interrupt request lines.
    - non-maskable interrupt line
      - reserved for events such as unrecoverable memory errors.
    - maskable interrupt line
      - used by device controllers to request service.
      - can be turned off by CPU before the execution of critical instruction sequences that must not be interrupted.

Interrupts

- INTEL processor event-vector table.

	vector number	description
non-maskable (0-31)	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)
maskable (32-255)	32–255	maskable interrupts

Interrupts

- Interrupt Chaining
  - In the case that computers have more devices (interrupt handlers) than they have address elements in the interrupt vector, *interrupt chaining* is used, in which each element in the interrupt vector points to the head of a list of interrupt handlers.
  - When an interrupt is raised, the handlers on the corresponding list are called one by one, until one is found that can service the request.
  - Interrupt chaining structure is a *compromise* between the overhead of a huge interrupt table and the inefficiency of dispatching to a single interrupt handler.
- Interrupt Priority
  - The interrupt mechanism also implements a system of *interrupt priority levels*. These levels enable the CPU to defer the handling of low-priority interrupts without masking all interrupts and makes it possible for a high-priority interrupt to *preempt* the execution of a low-priority interrupt.

Interrupts

- Interrupt Types and Attributes
  - An operating system is interrupt driven. Three interrupt types are:
    - Traps (Exceptions) 陷阱 (异常)
    - External Interrupts 外部中断
    - System Calls 系统调用
  - Various interrupt attributes
    - Asynchronous vs. Synchronous
    - External/Hardware vs. Internal/Software
    - Implicit vs. Explicit

	Interrupt Types		
Asynchronous	External Interrupts		Implicit
Synchronous		Traps	
		System Calls	Explicit
	External/ Hardware	Internal/ Software	

Storage Structures

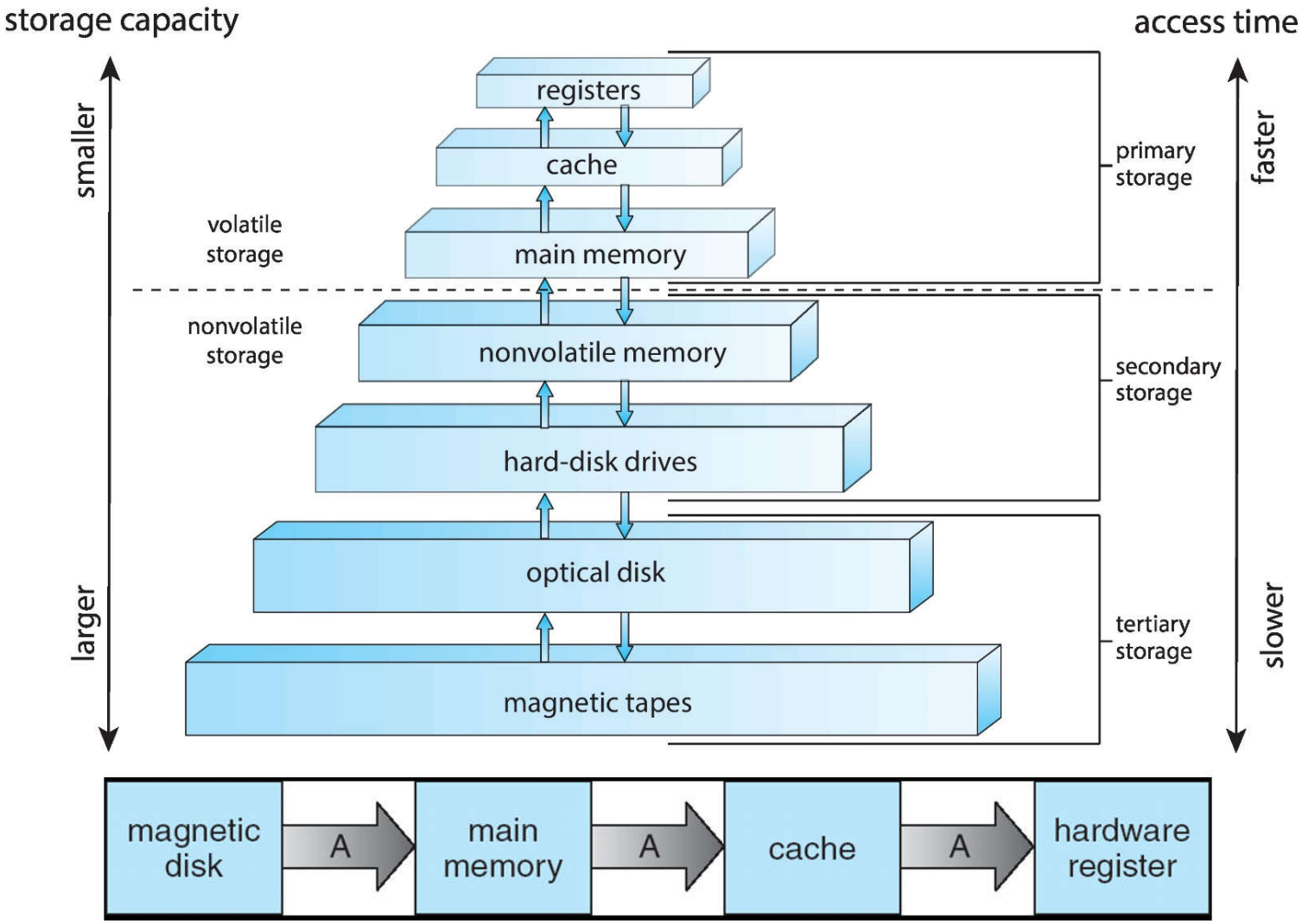
- Main memory
  - the only large storage media that the CPU can access directly
  - usually too small to store all needed programs and data permanently
  - volatile (易失的)
    - loses its contents when power is turned off or otherwise lost
- Secondary storage
  - large capacity
  - nonvolatile memory (NVM, 非易失存储) devices
- Hard disks
  - HDDs are the most common secondary-storage devices.
    - rigid metal or glass platters covered with magnetic recording material
    - Cylinders, tracks and sectors
  - The disk controller determines the logical interaction between the device and the mainframe.

Storage Structures

- Caching
  - Caching is an important principle performed at many levels in a computer
    - Information in use is copied from slower to faster storage temporarily.
    - hardware, operating system, apps.
  - Faster storage (cache) is checked first to determine if information is there.
    - If it is, information used directly from the cache.
      - It is faster.
    - Otherwise, data copied to cache and used there.
  - Cache is smaller than storage being cached.
    - Cache management is an important design problem.
    - Cache size and replacement policy matter
  - In most cases, cache is referred in particular to the SRAM within CPU.

Storage Structures

- Storage Hierarchy.



Storage Structures

- Performance of Various Levels of Storage.

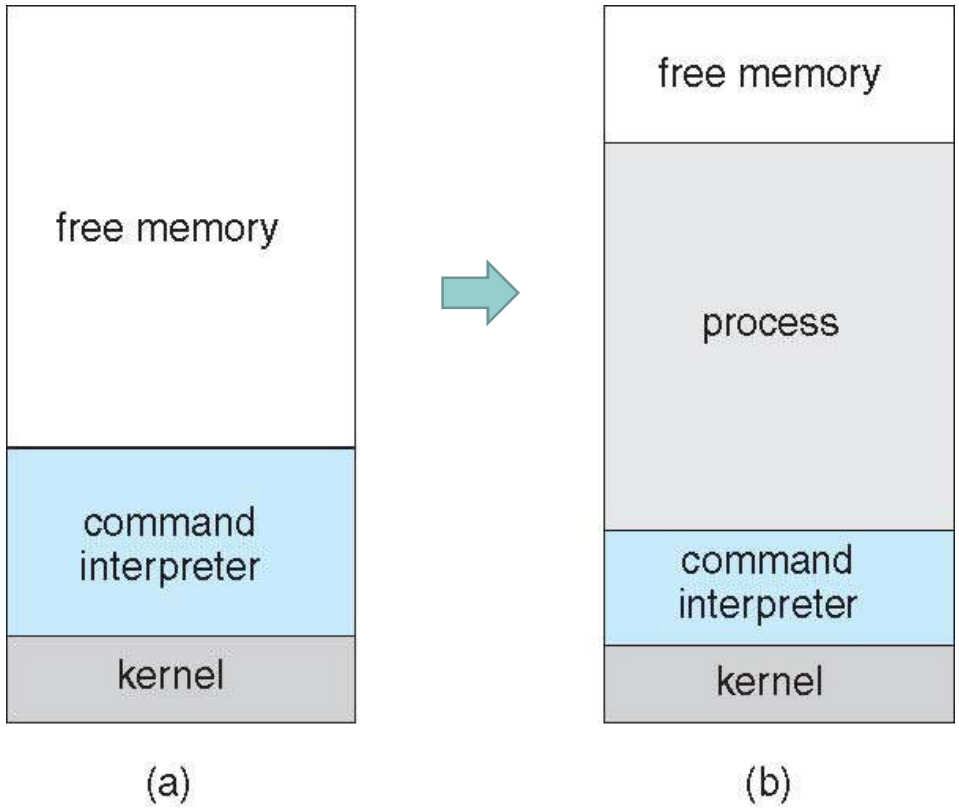
Level	1	2	3	4	5
Name	registers	cache	main memory	solid state disk	magnetic disk
Typical size	< 1 KB	< 16MB	< 64GB	< 1 TB	< 10 TB
Implementation technology	custom memory with multiple ports CMOS	on-chip or off-chip CMOS SRAM	CMOS SRAM	flash memory	magnetic disk
Access time (ns)	0.25 - 0.5	0.5 - 25	80 - 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

Main Memory Management

- Initial memory management techniques
  - Minimal Management
    - One program that manages memory for itself.
    - No memory protection problems here.
  - Memory Split
    - Resident Monitor (驻留程序) and User Job (User Program) split the memory between them.
  - Memory Division
    - The operating system and a few user jobs divide the available memory between them.

Main Memory Management

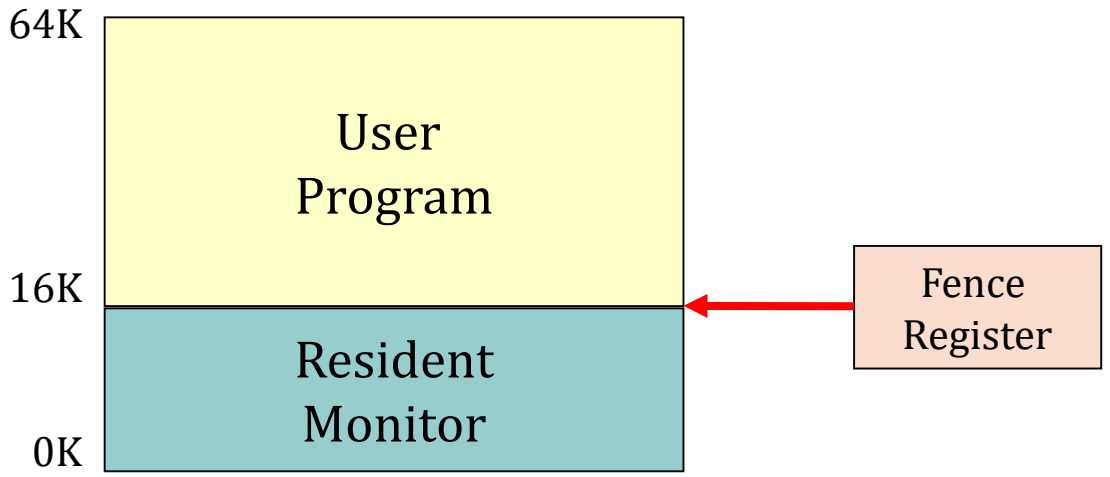
- Initial memory management techniques



MS-DOS Memory Split

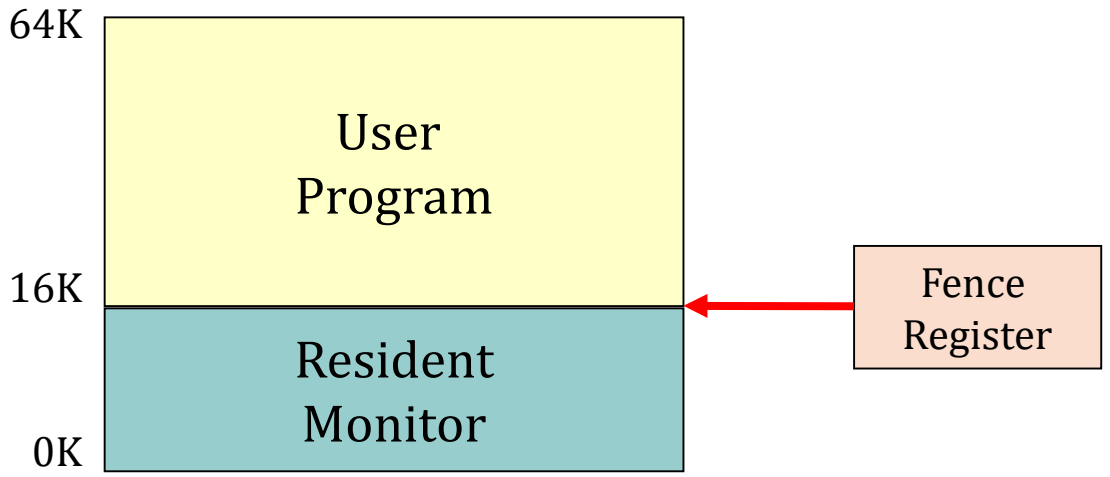
Main Memory Management

- Memory Management Dynamics
  - Sharing system resources requires the operating system to ensure that an incorrect program cannot cause other programs to execute incorrectly.
  - Resident Monitor is a “Trusted Program” but how to protect it from damage by the user program?
    - Solution: Use *Fence Register* (界限寄存器) and addressing access logic.



Main Memory Management

- Memory Management Dynamics
  - Fence Register is loaded with the base of the user program (which is also the limit of the Resident Monitor).
  - The user program can read any address but addressing access logic assures that it can write only to addresses that are larger than the Fence Register value.
  - The instruction to load the Fence Register has to be *privileged* (i.e., it can be executed only by the Resident Monitor)
    - but how to ensure that?



Main Memory Management

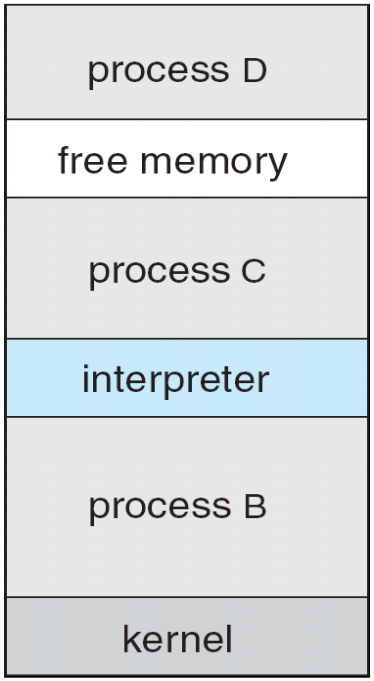
- Dual-Mode Operation
  - Hardware support is provided to differentiate between at least two modes of operations:
    - User Mode: execution done on behalf of a user.
    - kernel Mode: execution done on behalf of OS.
  - A user program could never gain control of the computer in kernel mode.
  - When any type of interrupt occurs, interrupt hardware switches to kernel mode, at the correct ISR in the kernel address space.
  - Privileged Instructions can be executed only in kernel mode.
    - Solution: A *mode bit* was added to computer hardware to indicate the current mode.
      - The mode bit is kept in *Status Register* (状态寄存器)
      - kernel/system/monitor/supervisor (0) or user (1).

Main Memory Management

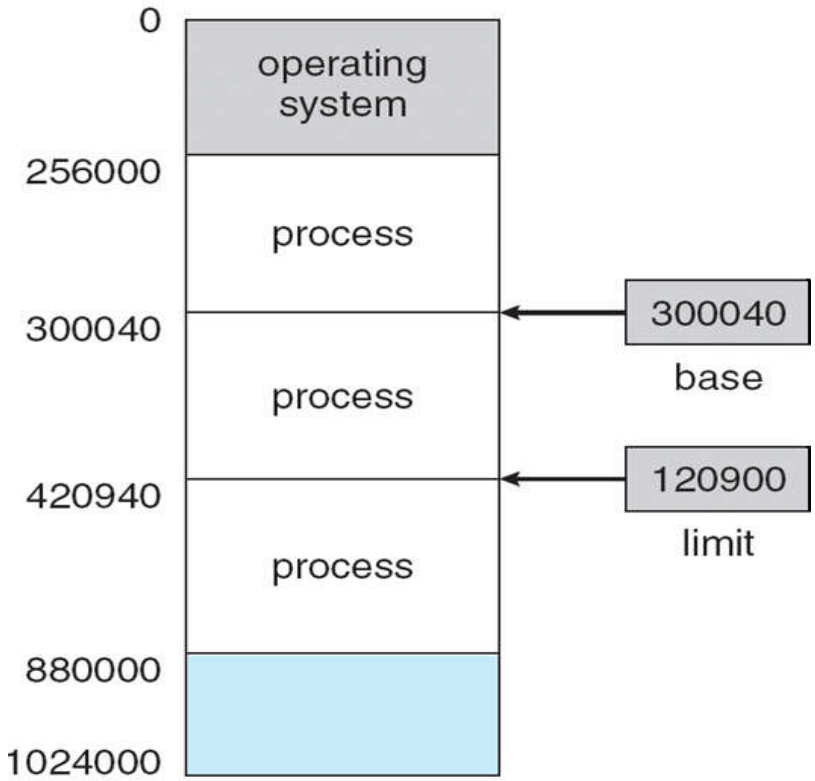
- Memory Division Protection
  - In order to have memory division protection, two registers are added to determine the range of legal addresses a program may access.
    - *Base Register* (基址寄存器)
      - It holds the smallest legal physical memory address of the program.
      - It is also called Lower Fence Registers.
    - *Limit Register* (限制寄存器)
      - It contains the *size* of the range.
      - It is also called Upper Fence Registers.
  - Memory outside the defined range is protected.

Main Memory Management

- Memory Division Protection.



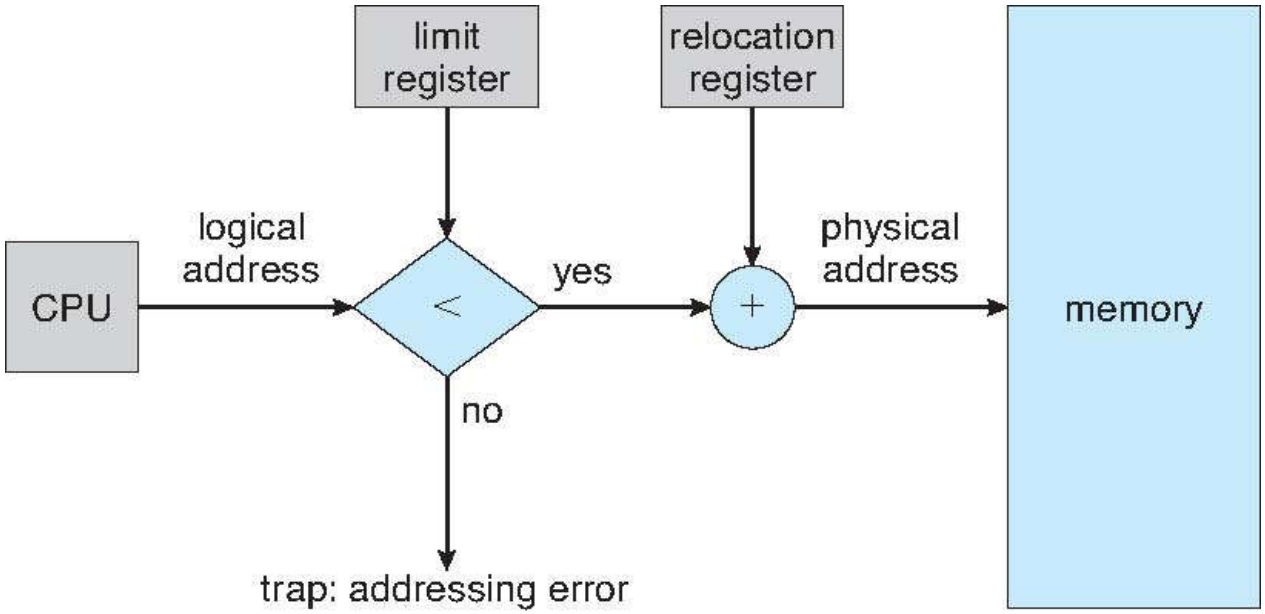
Unix Memory Division



Example of Base and Limit Registers

Main Memory Management

- Protection Hardware
  - When executing in kernel mode, the operating system has unrestricted access to both system and user's memory.
  - The load instructions for the base and limit registers are *privileged* instructions.
    - Privileged instructions can be issued only in kernel mode.
    - The read instructions for these registers need not be privileged.



Main Memory Management

- Traps
  - A trap/exception is a *software-generated* interrupt caused by an error of the program.
    - The trap uses the interrupt hardware to switch to kernel mode.
  - Example.
    - arithmetic overflow/underflow
    - division by zero
    - execute illegal instruction
    - reference outside user's memory space
  - A trap can also be initiated by an explicit trap instruction in the program.

  - *Trap, Fault and Programmable Exception
    - Trap: saving the next PC in EIP (e.g., a debugging point)
    - Fault: saving the current PC in EIP (e.g., a page fault)
    - Programmable Exception: system call (int 0x80), saving the next PC in EIP

Main Memory Management

- Memory Protection Summary
  - how to protect jobs in *memory* space?
    - use fence registers and addressing access logic
  - how to protect *fence registers*?
    - use privileged fence load instruction
  - how to ensure *privileged execution*?
    - use mode bit
  - but how to protect *mode bit*?
    - change to kernel mode only by interrupt hardware!

I/O Structure

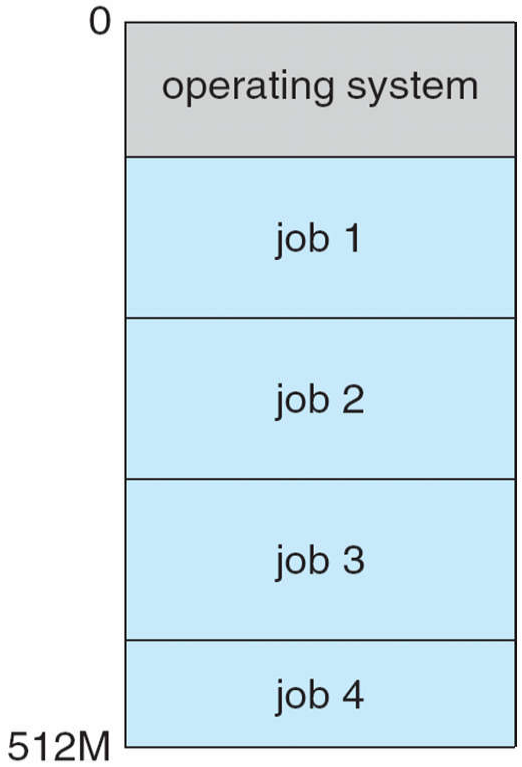
- A large portion of operating system code is dedicated to managing I/O.
  - It is important to the reliability and performance of a system.
  - It is also because of the varying nature of the devices.
- Interrupt-driven I/O
  - fine for moving small amounts of data (may be 1 or 64 bytes)
  - high overhead when used for bulk data movement
- Direct memory access (DMA)
  - After setting up buffers, pointers, and counters for the I/O device, the device controller transfers an entire block of data (may be 2^{16} bytes) directly to or from the device and main memory, with no intervention by the CPU.
  - Only *one interrupt* is generated *per block*, to tell the device driver that the operation has completed, rather than the one interrupt per byte generated for low-speed devices.
  - While the device controller is performing these operations, the CPU is available to accomplish other work.

Multiprogrammed Batch Systems

- Multiprogramming (aka Multitasking) is 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 of them is selected and run via job scheduling.
    - When the running job has to wait (e.g., for I/O), OS switches to another job.
- Batch multiprogramming (多道批处理) does not support interaction with users.

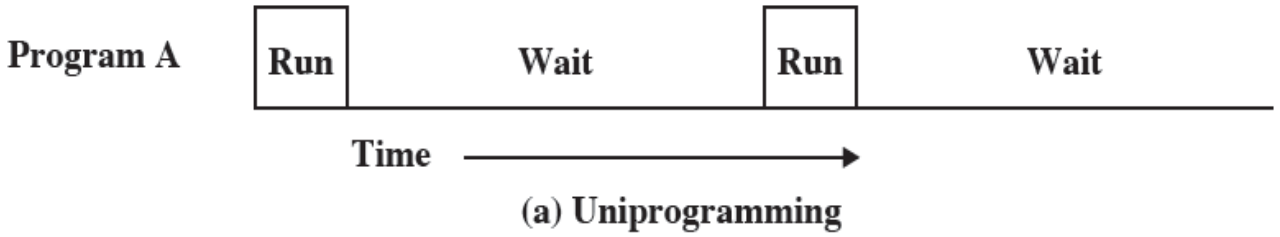
Multiprogrammed Batch Systems

- Memory Layout for Batch Multiprogramming
  - Several *jobs* are kept in main memory at the same time, and the CPU is multiplexed among them.

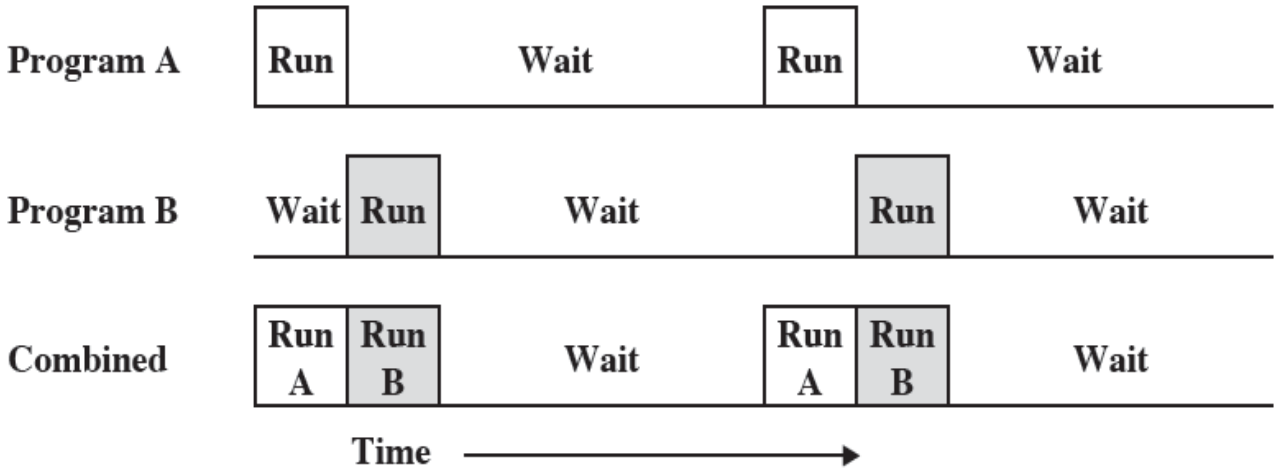


Multiprogrammed Batch Systems

- CPU efficiency of Batch Multiprogramming.



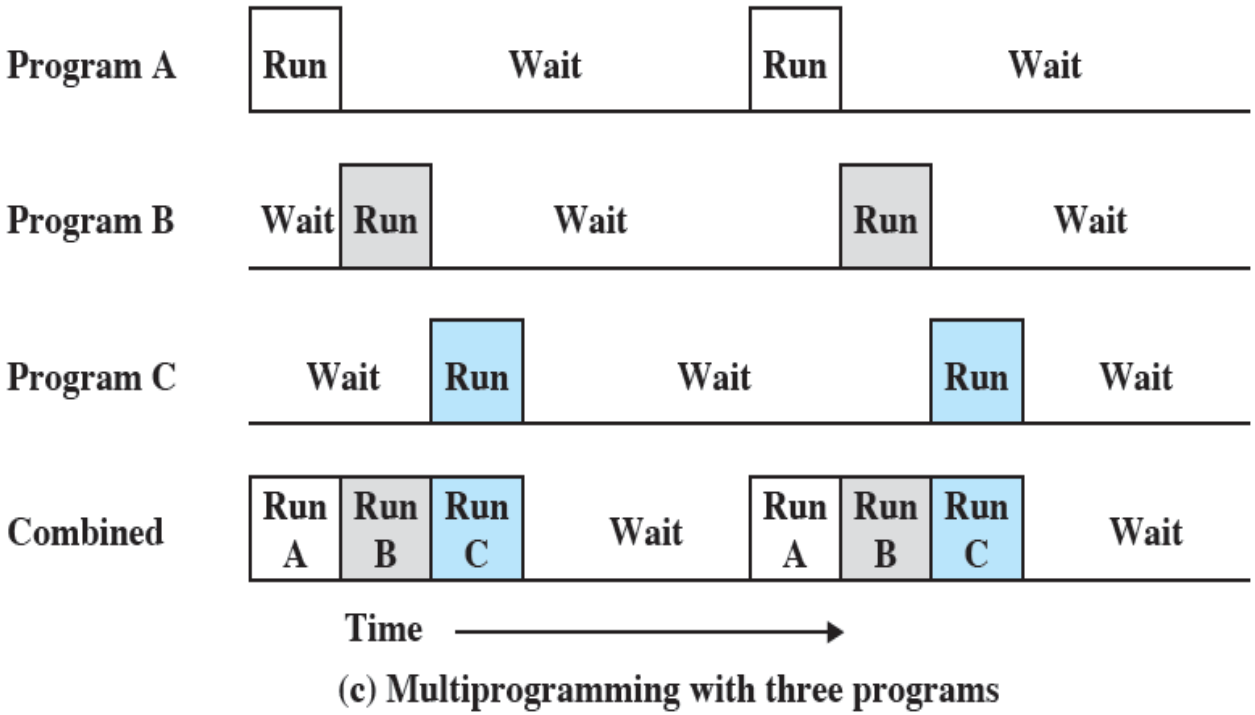
(a) Uniprogramming



(b) Multiprogramming with two programs

Multiprogrammed Batch Systems

- CPU efficiency of Batch Multiprogramming.



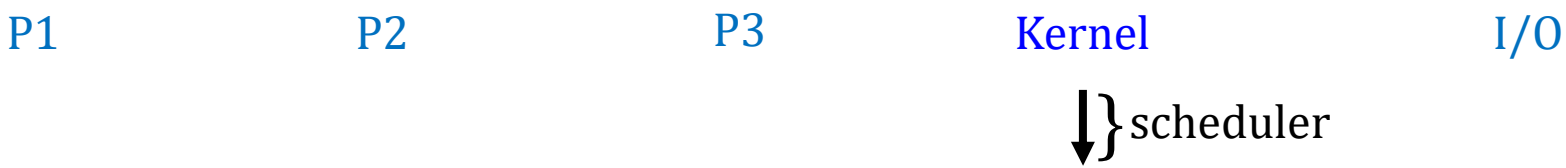
(c) Multiprogramming with three programs

Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.

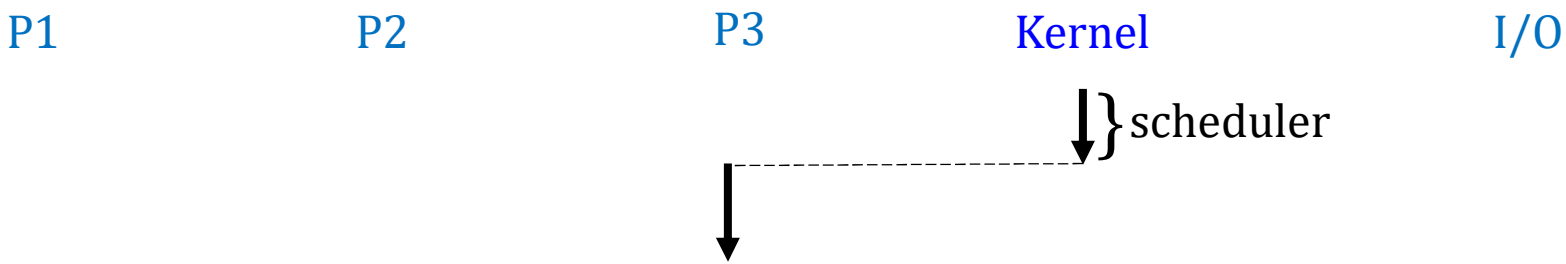
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



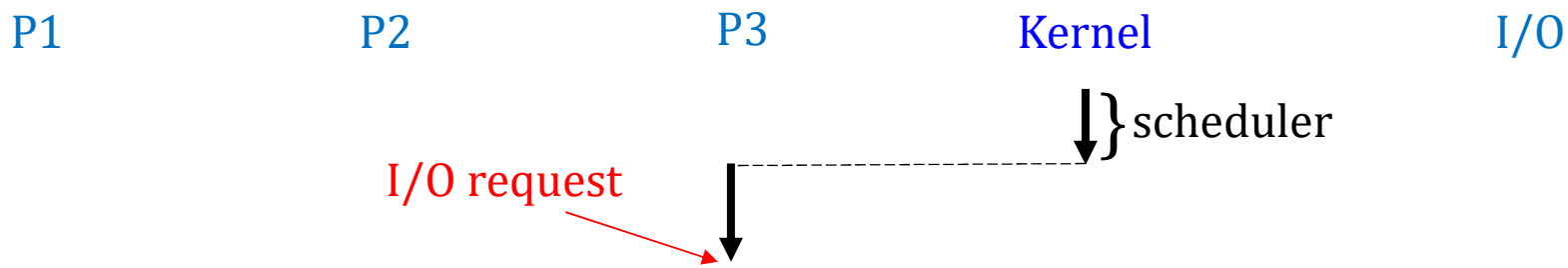
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



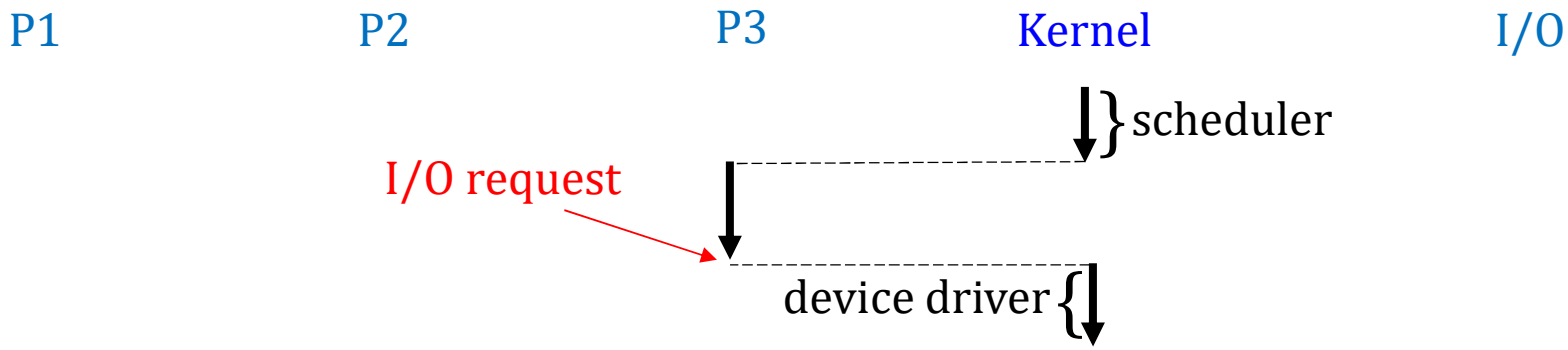
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



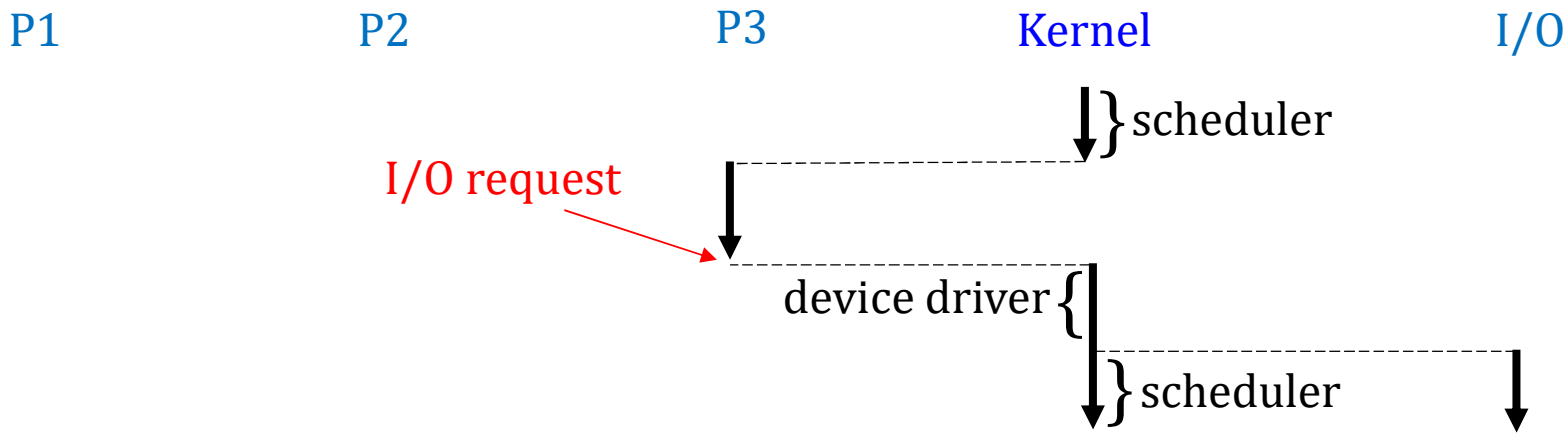
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



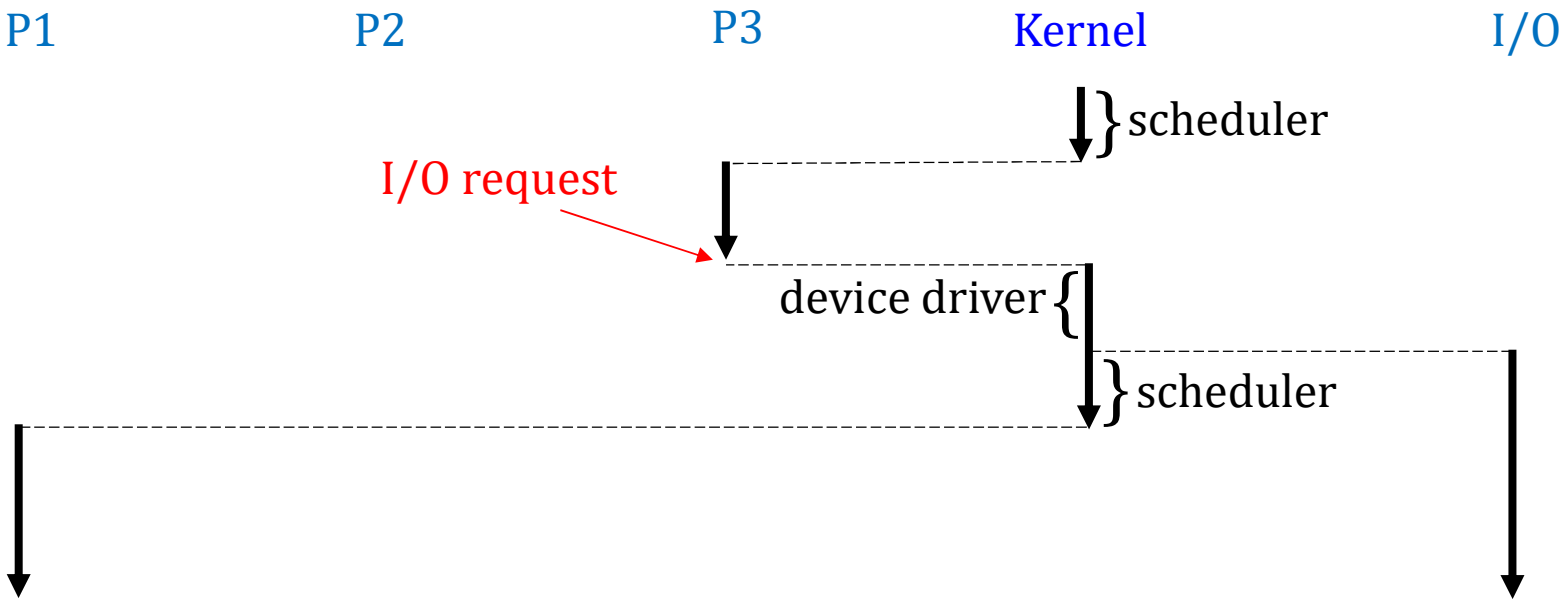
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



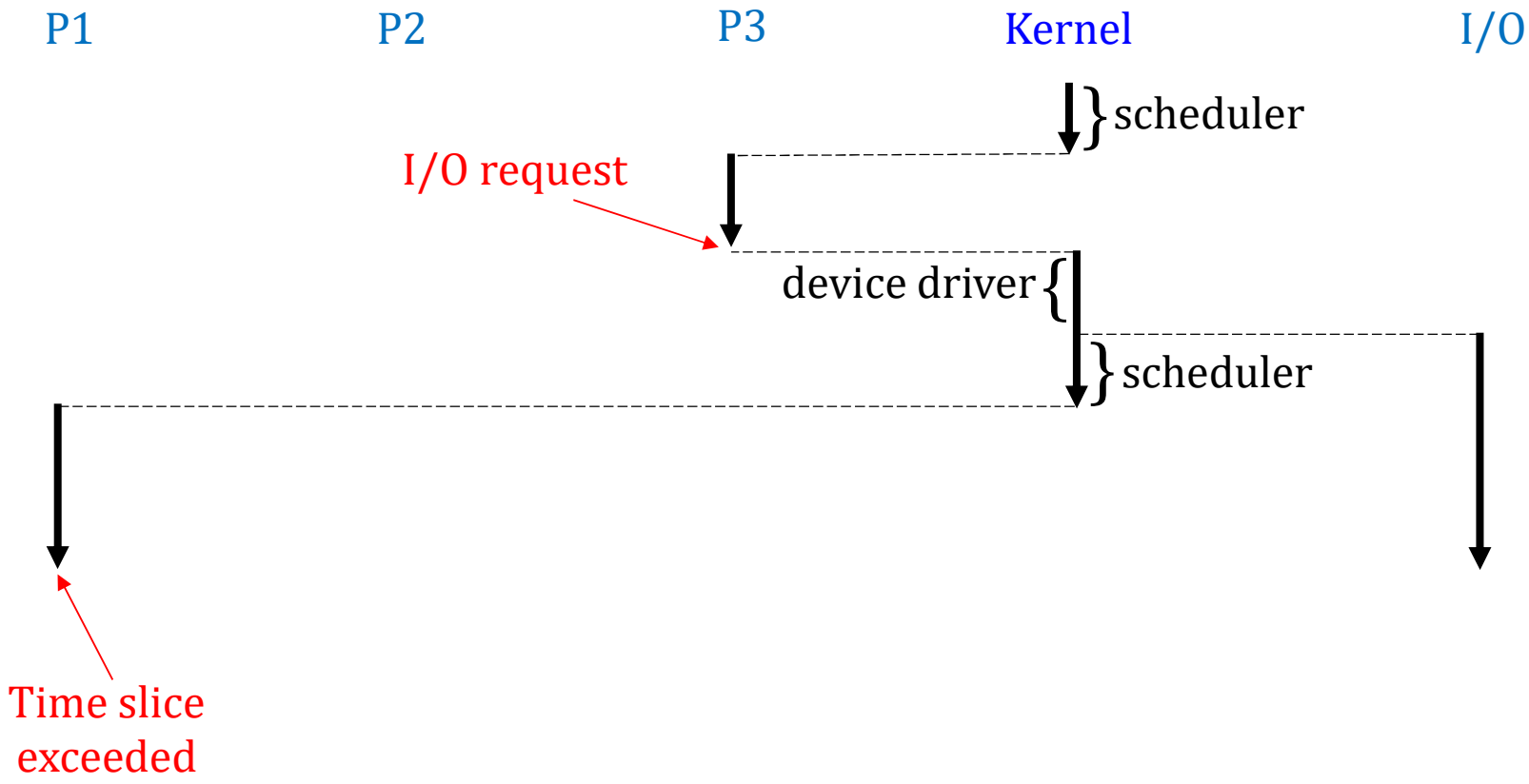
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



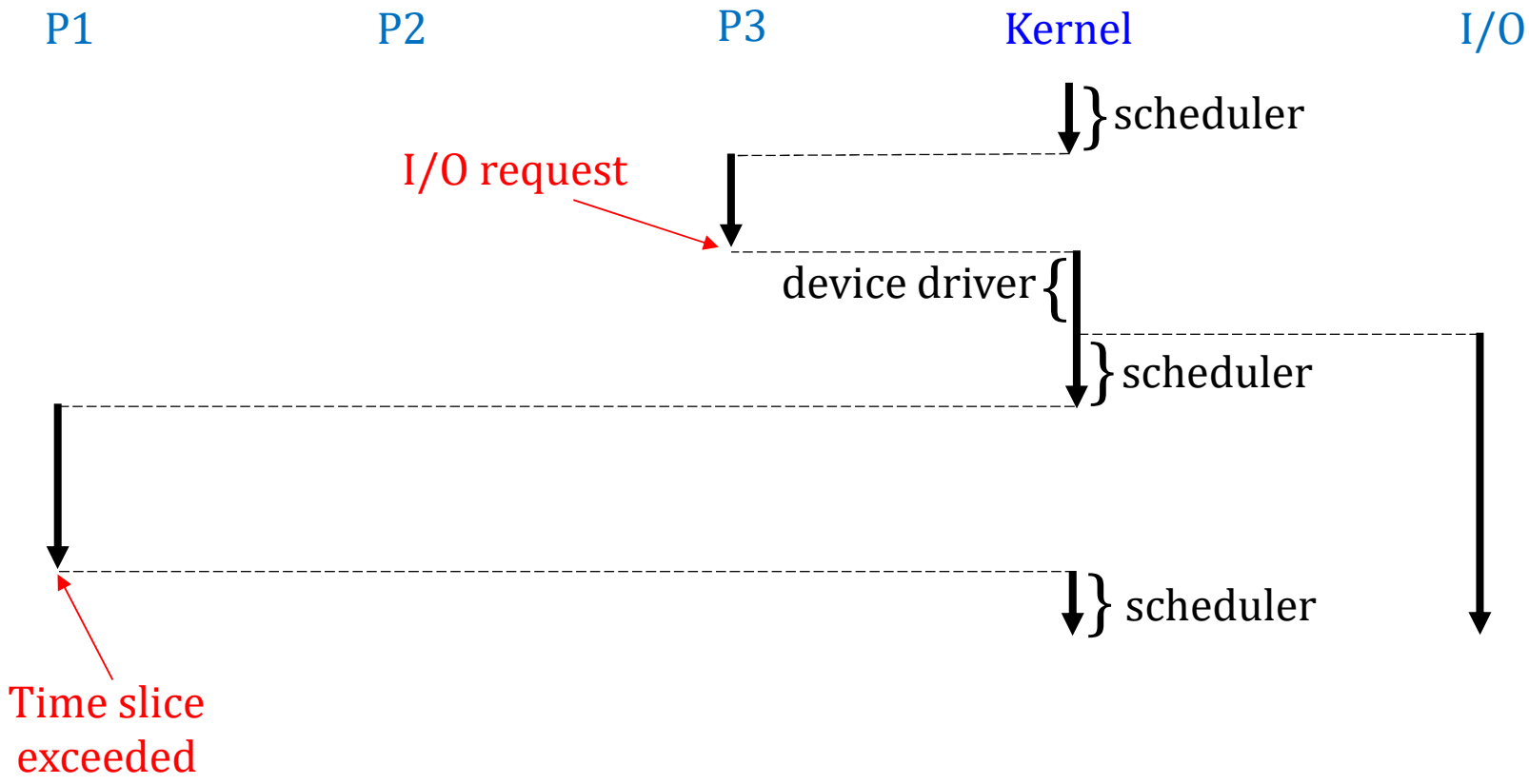
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



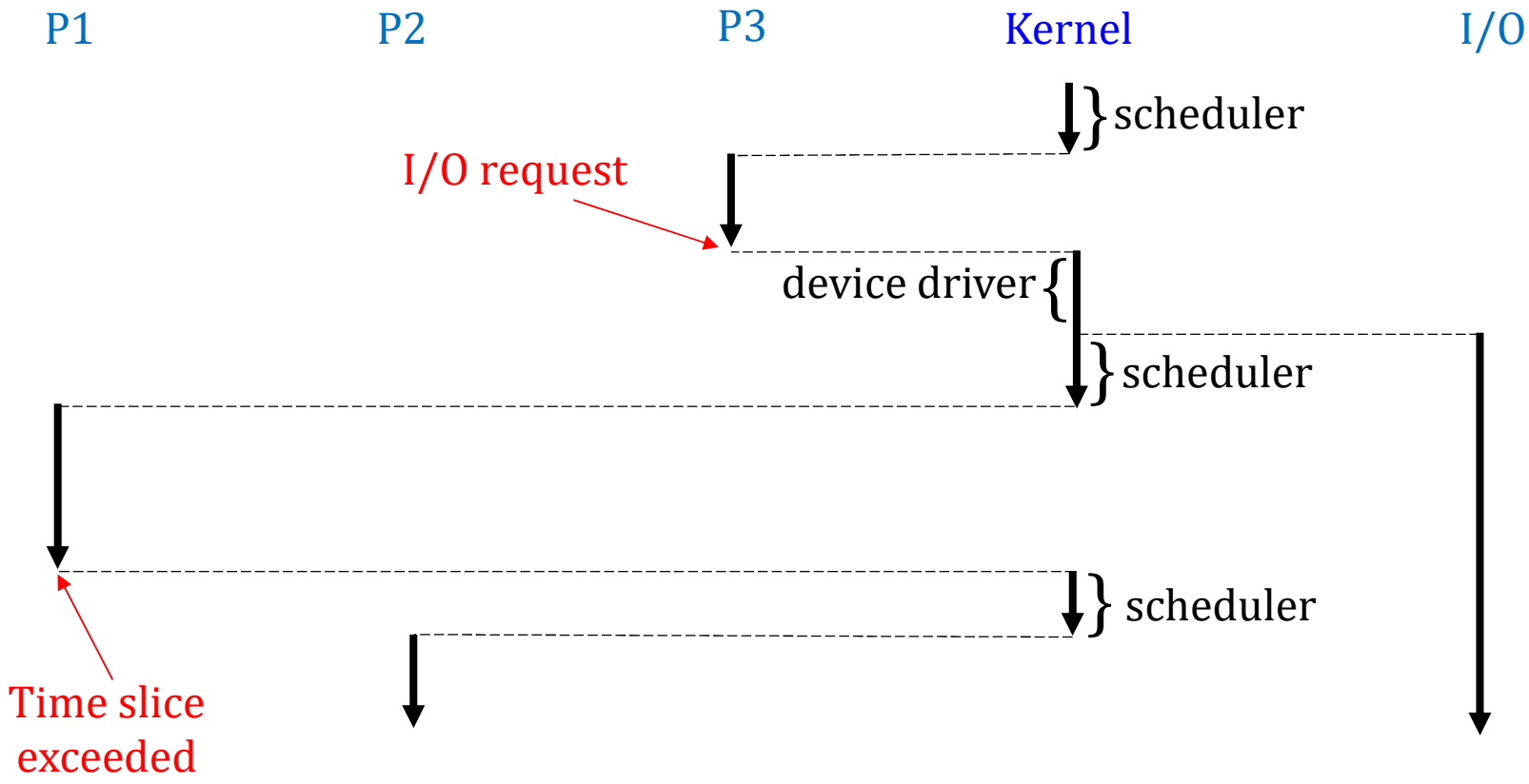
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



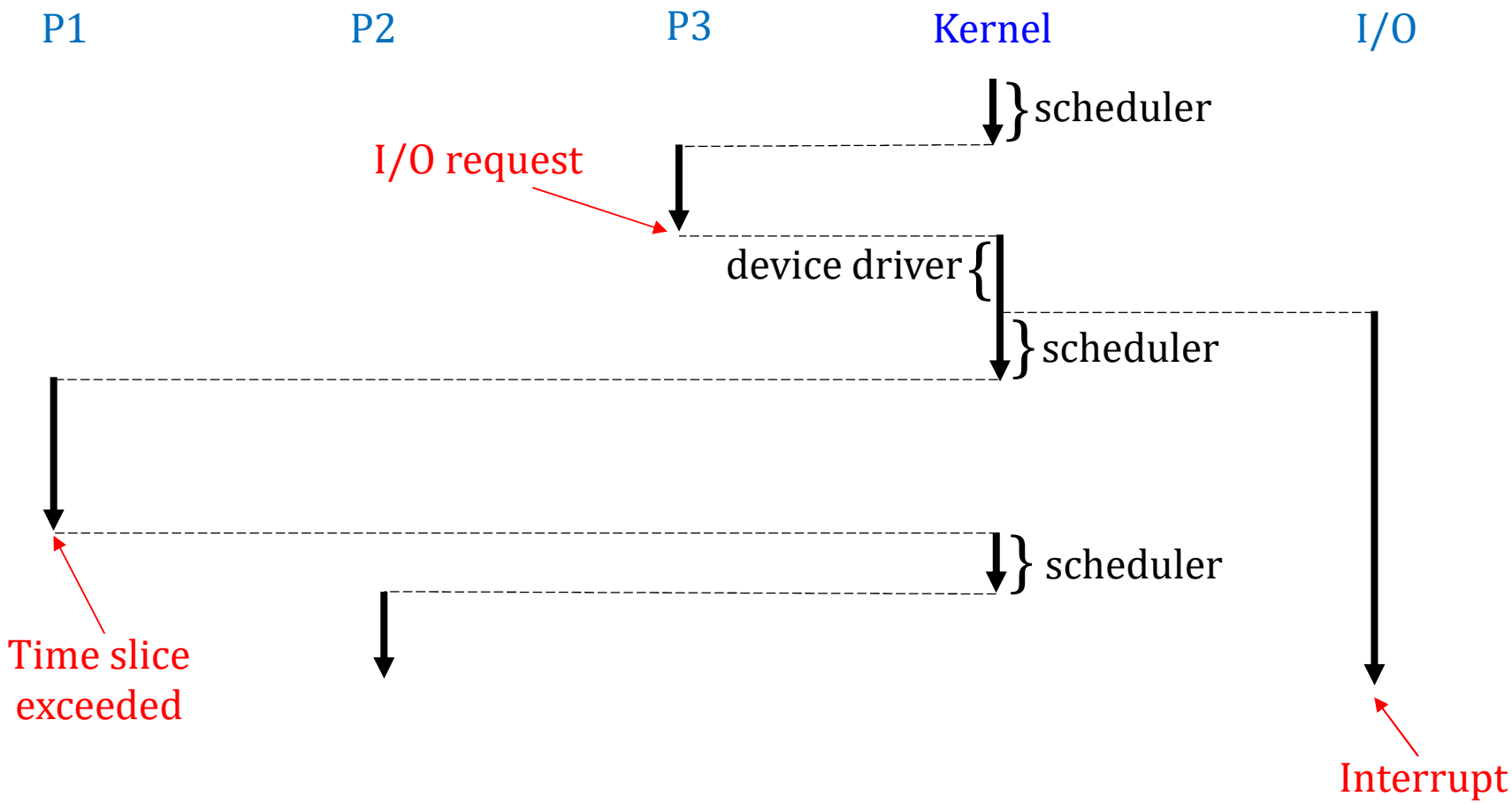
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



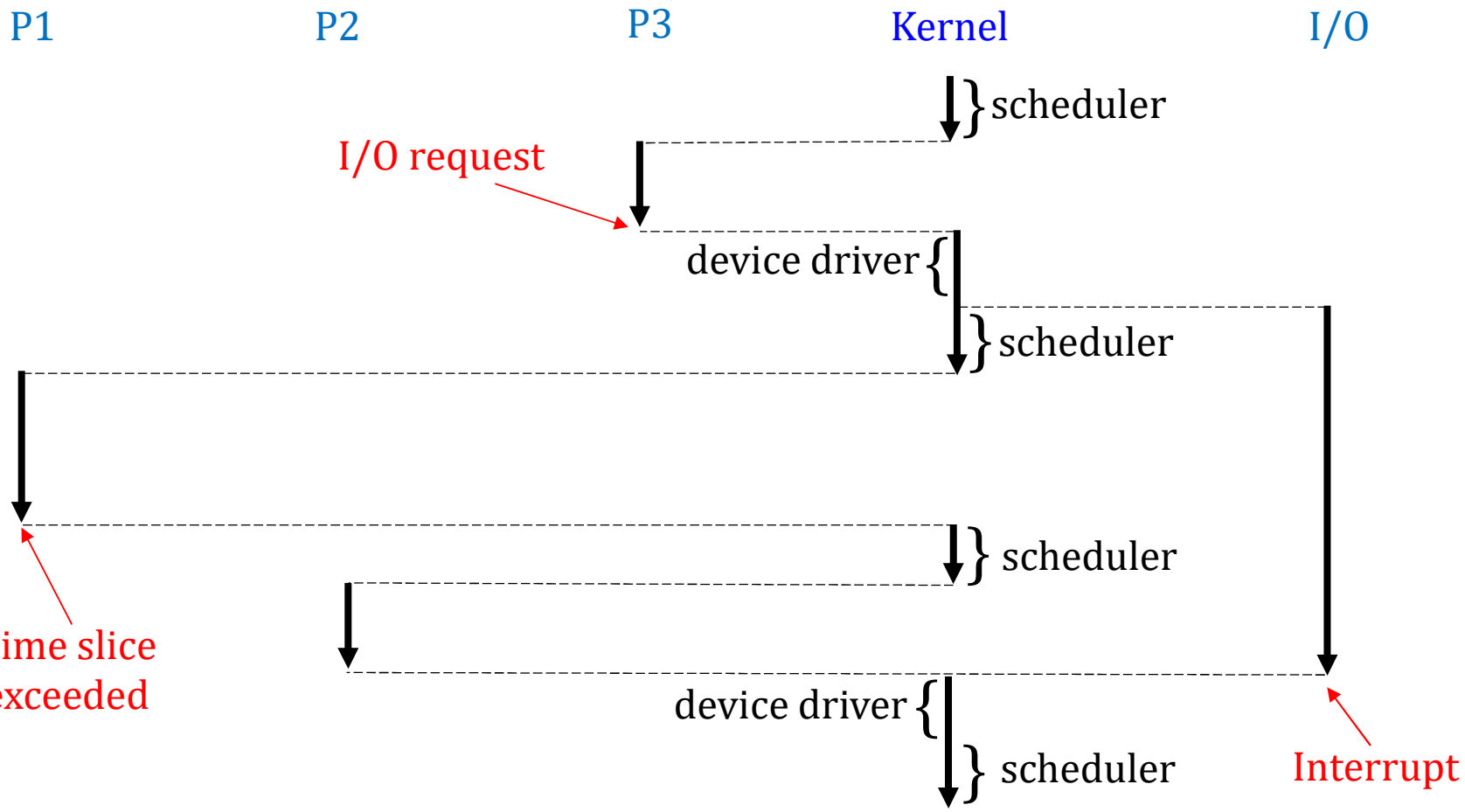
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



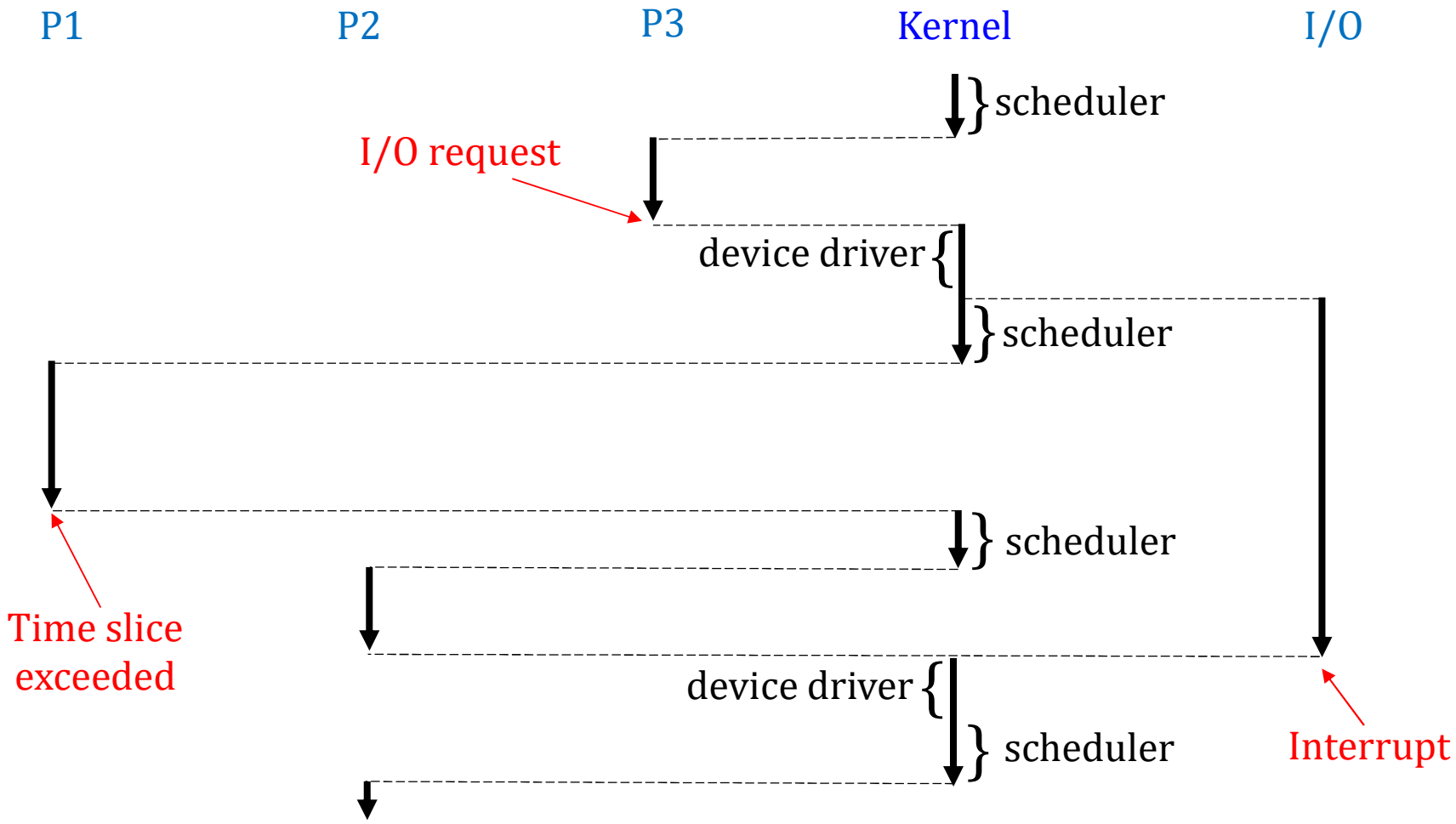
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



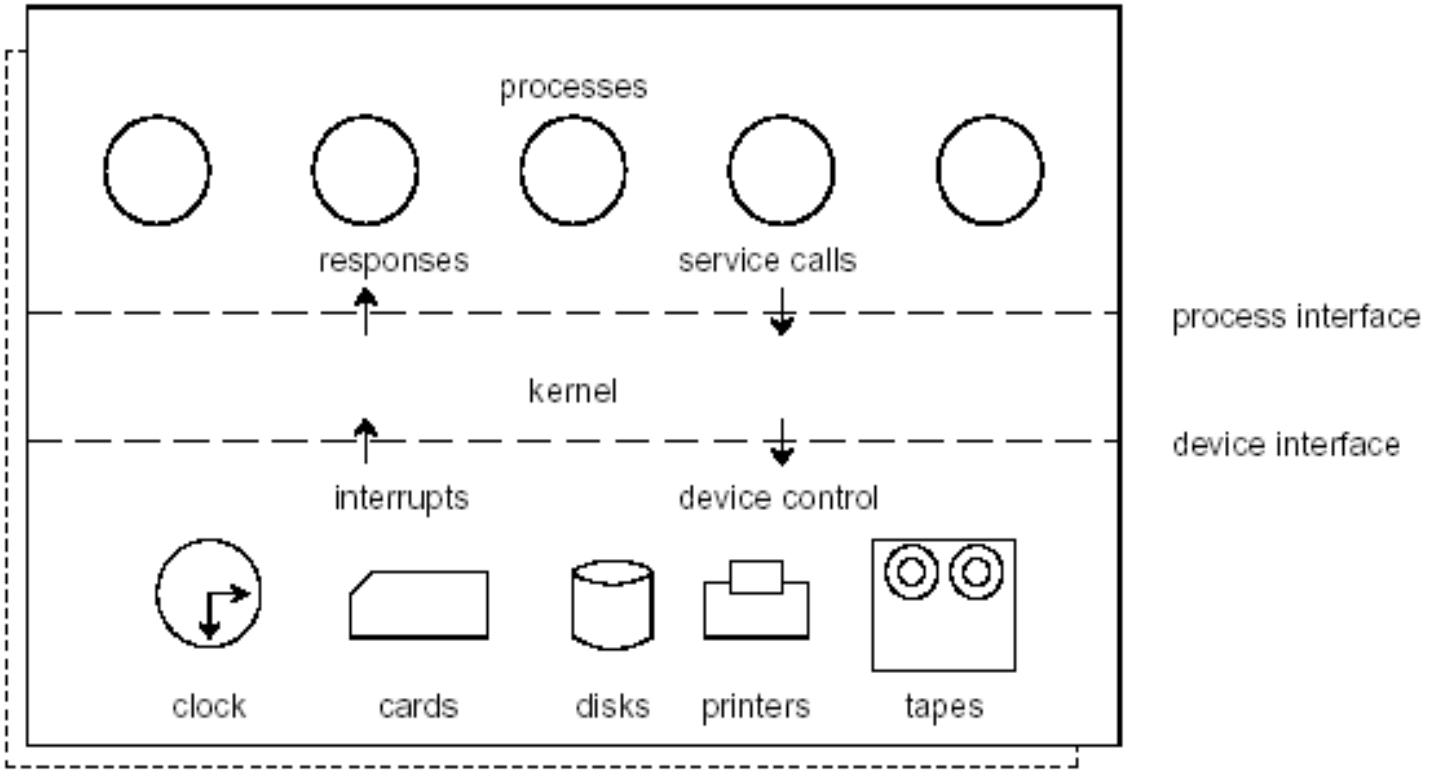
Multiprogrammed Batch Systems

- Example of Multiprogramming – a time sequence diagram.



Multiprogrammed Batch Systems

- Components of Multiprogramming.



Multiprogrammed Batch Systems

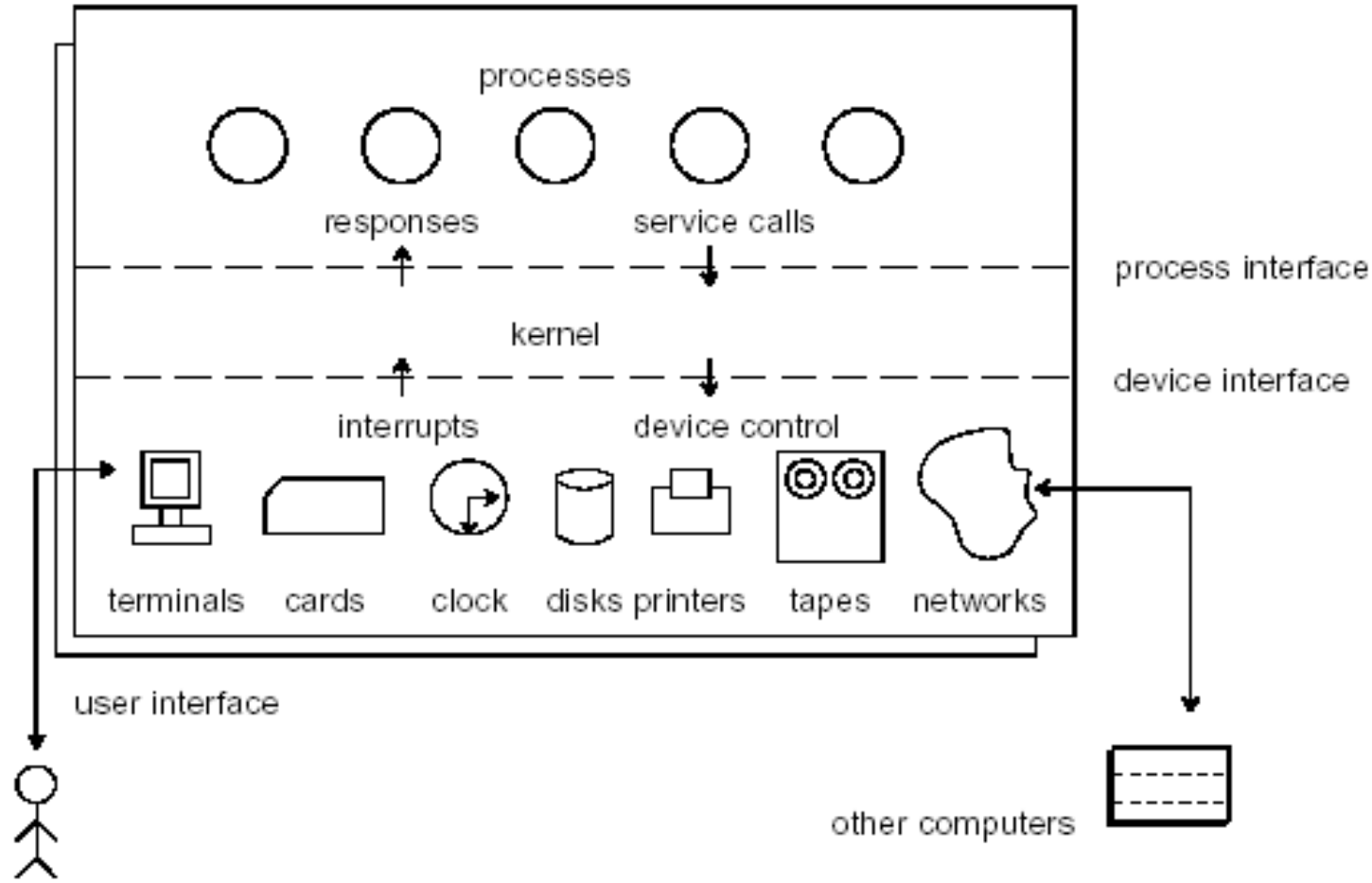
- Requirements for Multiprogramming
  - Hardware support
    - I/O interrupts and DMA controllers
      - in order to execute instructions while I/O device is busy.
    - Timer interrupts for CPU to gain control.
    - Memory management
      - several ready-to-run jobs must be kept in memory.
    - Memory protection (data and programs).
  - Software support from the OS
    - for scheduling (which program is to be run next).
    - to manage resource contention.

Time-Sharing Systems

- Time-sharing extends Batch Multiprogramming to handle multiple interactive jobs.
  - It's *Interactive Multiprogramming*.
  - Multiple users simultaneously access the system through commands entered at terminals.
  - Processor's time is shared among multiple users.
- In Time-sharing (multitasking) the 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 or less.
  - Each user has at least one program (process) executing in memory.
  - CPU scheduling supports several jobs ready to run at the same time.
  - 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.

Time-Sharing Systems

- Time-sharing Architecture.



Time-Sharing Systems

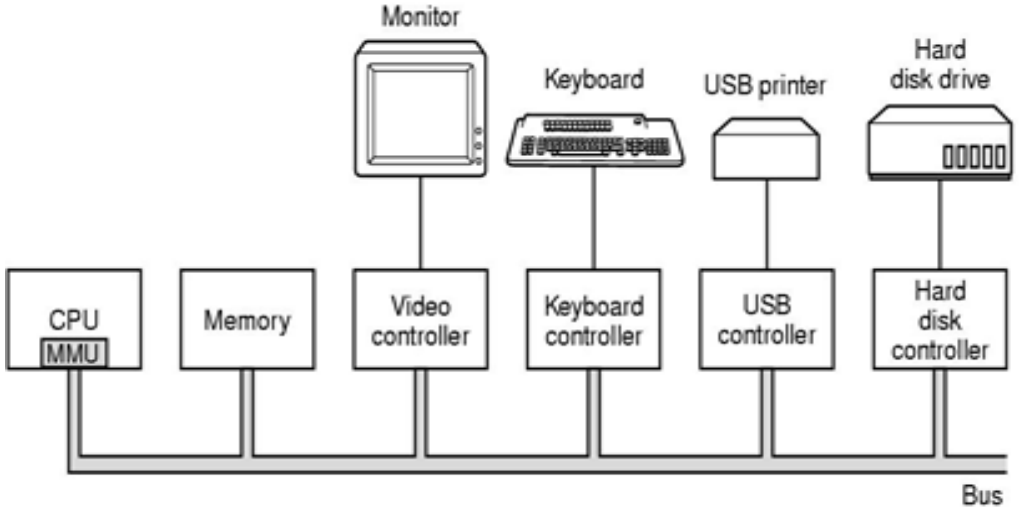
- Why did Time-sharing work?
  - Because of slow human reaction time
    - A typical user needs 2 seconds of processing time per minute.
    - Then many users should be able to share the same system without noticeable delay in the computer reaction time.
    - The user should get a good response time.
  - It fits the economic basis of the mainframe computer installation.

Real-Time Systems

- Real-Time (RT) systems are dedicated systems (not general-purpose) that need to adhere to *deadlines*, i.e., time constraints.
- *Correctness* of the computation depends not only on the logical result but also on the time at which the results are produced.
- Hard real-time
  - Hard real-time system *must* meet its deadline.
  - It conflicts with time-sharing systems, not supported by general-purpose OS.
  - Often used as a control device in a dedicated application:
    - Industrial control
    - Robotics
  - Secondary storage limited or absent, data/program is stored in short term memory, or Read-Only Memory (ROM).
- Soft real-time
  - Deadlines desirable but not mandatory
  - Limited utility in industrial control or robotics
  - Useful in modern applications (multimedia, video conference, virtual reality) requiring advanced operating-system features

Personal/Desktop Computers

- Personal computers – computer system dedicated to a single user.
- I/O devices
  - keyboards, mice, display screens, small printers
- User convenience and responsiveness
- Can adopt technology developed for larger operating system; often individuals have sole use of computer and do not need advanced CPU utilization of protection features
- May run several different types of operating systems
  - Windows, MacOS, UNIX, Linux



Components of a simple personal computer

Personal/Desktop Computers

- Office environment
  - PCs connected to LAN, terminals attached to mainframe or minicomputers providing batch and timesharing
  - Portals allowing networked and remote systems access to same resources
- Home networks
  - single system, modems
  - firewalled, networked

Multiprocessor Systems

- SISD and MIMD Computers
  - Single Instruction Single Data (SISD)
    - Single processor executes a single instruction sequence to operate on data stored in a single memory.
      - a Uniprocessor.
  - Multiple Instruction Multiple Data (MIMD)
    - A set of processors simultaneously execute different instruction sequences on different data sets.
      - a Multiprocessor.

Multiprocessor Systems

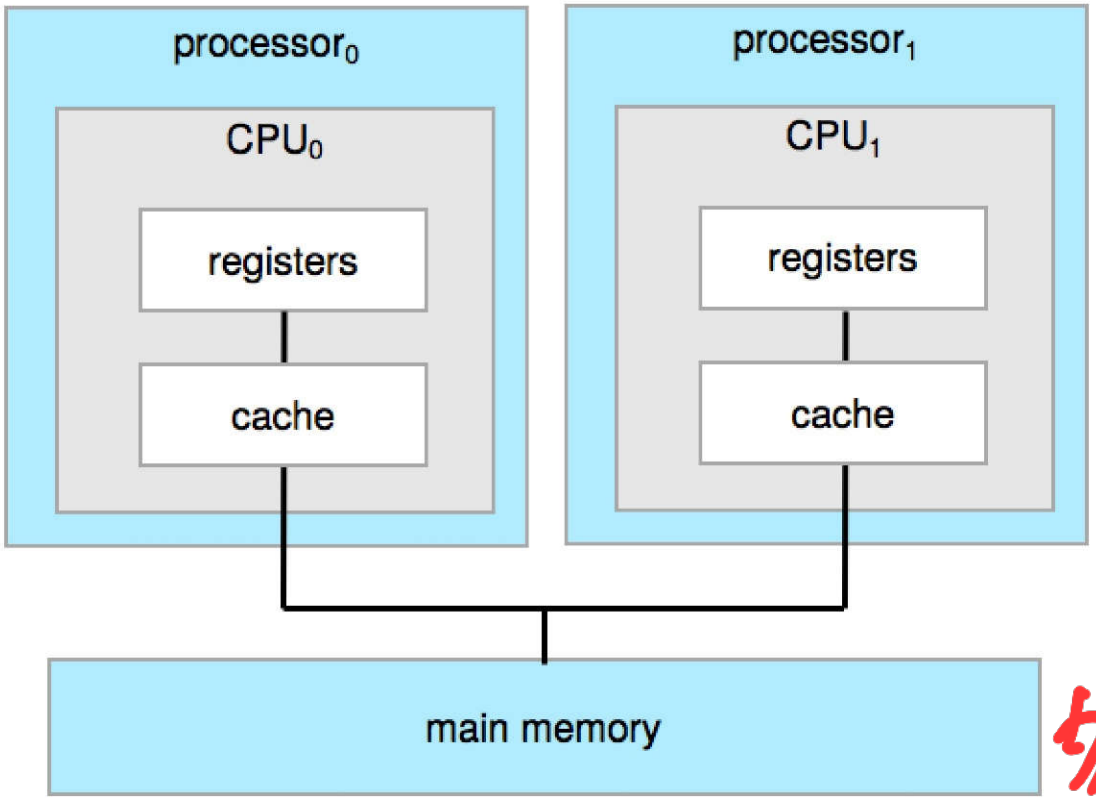
- Multiprocessor Systems
  - Traditionally, Multiprocessor Systems have two processors (or more), each with a *single-core* CPU.
    - The processors share the computer bus and sometimes the clock, memory, and peripheral devices.
    - Communication usually takes place through the shared memory.
    - aka parallel systems, tightly-coupled systems
  - Multiprocessor systems are growing in use and importance, dominating the landscape of computing.
    - Increased throughput
    - Economy of scale
    - Increased reliability
      - graceful degradation or fault tolerance (故障弱化或容错)
  - The speed-up ratio with N processors is less than N.
    - What lowers the expected gain from additional processors?
      - Overhead in multiple processors cooperation
      - Contention for shared resources

Multiprocessor Systems

- Asymmetric and Symmetric Multiprocessing Models
  - Asymmetric Multiprocessing
    - Master processor schedules and allocates specific work to slave processors.
  - Symmetric Multiprocessing (SMP)
    - SMP is the most common architecture.
      - Each peer CPU processor has its own set of registers.
      - All processors share physical memory over the system bus.
      - Each processor runs an identical copy of the OS and does self-scheduling from the pool of available processes.
      - Existence of multiple processors is transparent to the user.
    - Equilibration: OS schedules processes/threads across all the processors, lower the workload variance among the processors.
      - Multiple processes can run simultaneously without causing performance to deteriorate significantly.
    - Robustness: a single CPU failure does not halt the system, only the performance is reduced.
    - Incremental growth: just add another CPU!

Multiprocessor Systems

- Symmetric Multiprocessing (SMP).



Symmetric multiprocessing architecture

多核共享二级缓存

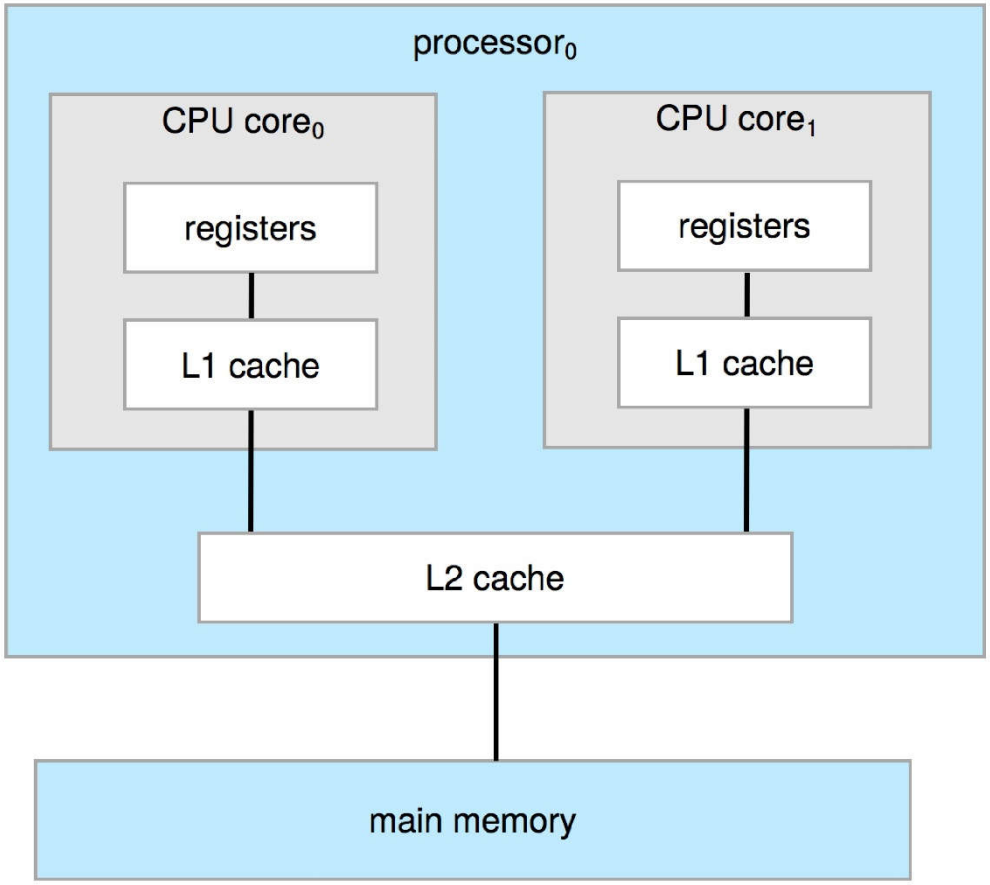
多CPU共享内存

Multiprocessor Systems

- Multicore Systems
  - The definition of multiprocessor now includes *multicore systems*, in which multiple computing cores reside on a single chip.
  - Multicore on a single chip can be *more efficient* than multiple chips each with a single core.
    - On-chip communication is faster.
    - Less power is required. This is an important issue for mobile devices as well as laptops.
  - The figure in next slide illustrates a dual-core design with two cores on the same processor chip, each core has its own register set and its own local cache (L1). L2 cache is local to the chip but is shared by the two processing cores.
    - Lower-level Local caches are generally smaller and faster than higher-level shared caches.
    - A multicore processor with N cores appears to the operating system as N standard CPUs.
  - Virtually all modern operating systems support multicore SMP.
    - E.g., Windows, macOS, Linux, Android, iOS mobile

Multiprocessor Systems

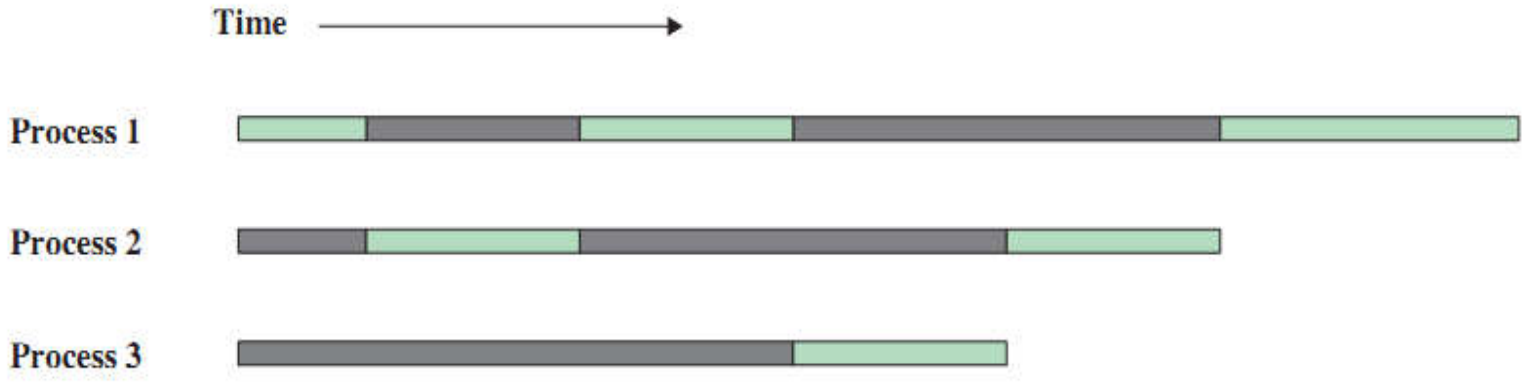
- Multicore Systems.



A dual-core design with two cores on the same chip

Multiprocessor Systems

- Multiprogramming vs. Multiprocessing.



(a) Interleaving (multiprogramming, one processor)

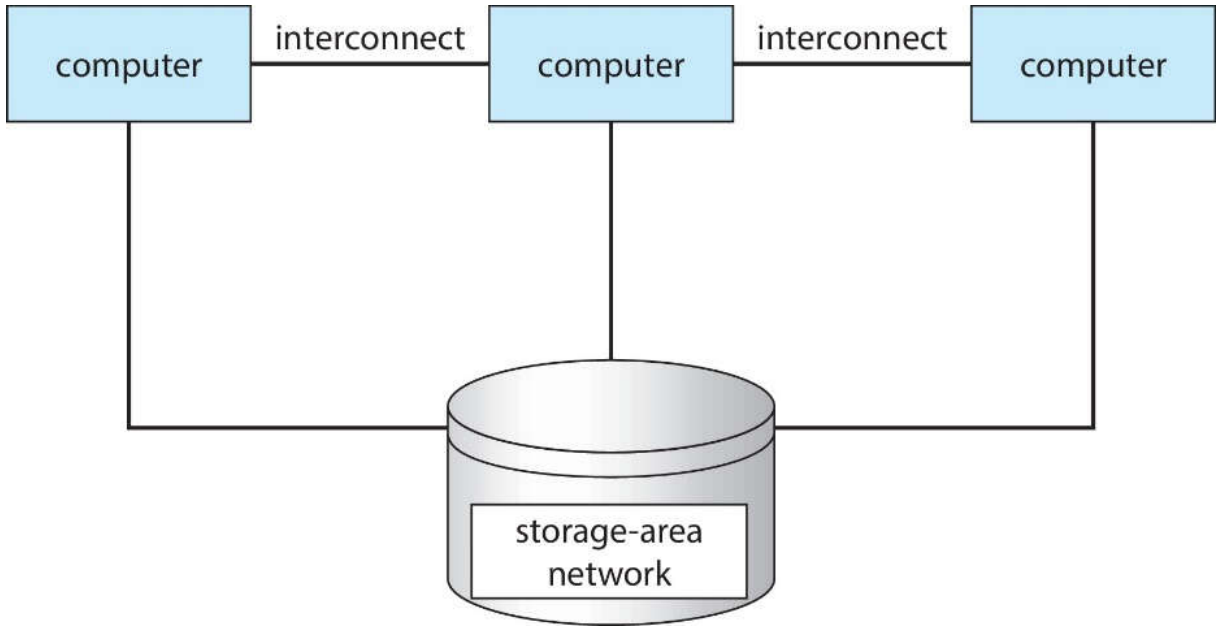
(b) Interleaving and overlapping (multiprocessing; two processors)

Clustered Systems

- A *Clustered System* (集群系统) is another type of multiprocessor system gathering together multiple CPUs. It is composed of two or more individual multicore systems.
  - considered loosely-coupled system (松耦合系统)
- Clustering allows individual systems to *share external storage* and balance CPU load:
  - Processors also have their own external memory.
  - Communication takes place through high-speed channels.
  - usually sharing storage via a Storage Area Network (SAN)
  - provides high-availability service which survives failures
    - graceful degradation or fault tolerant (容错)
- Asymmetric and Symmetric Clustering
  - Asymmetric clustering has one machine in hot-standby mode.
  - Symmetric clustering has multiple nodes running applications, monitoring each other.
- Some clusters are used for high-performance computing (HPC) where applications must be written to use parallelization.

Clustered Systems

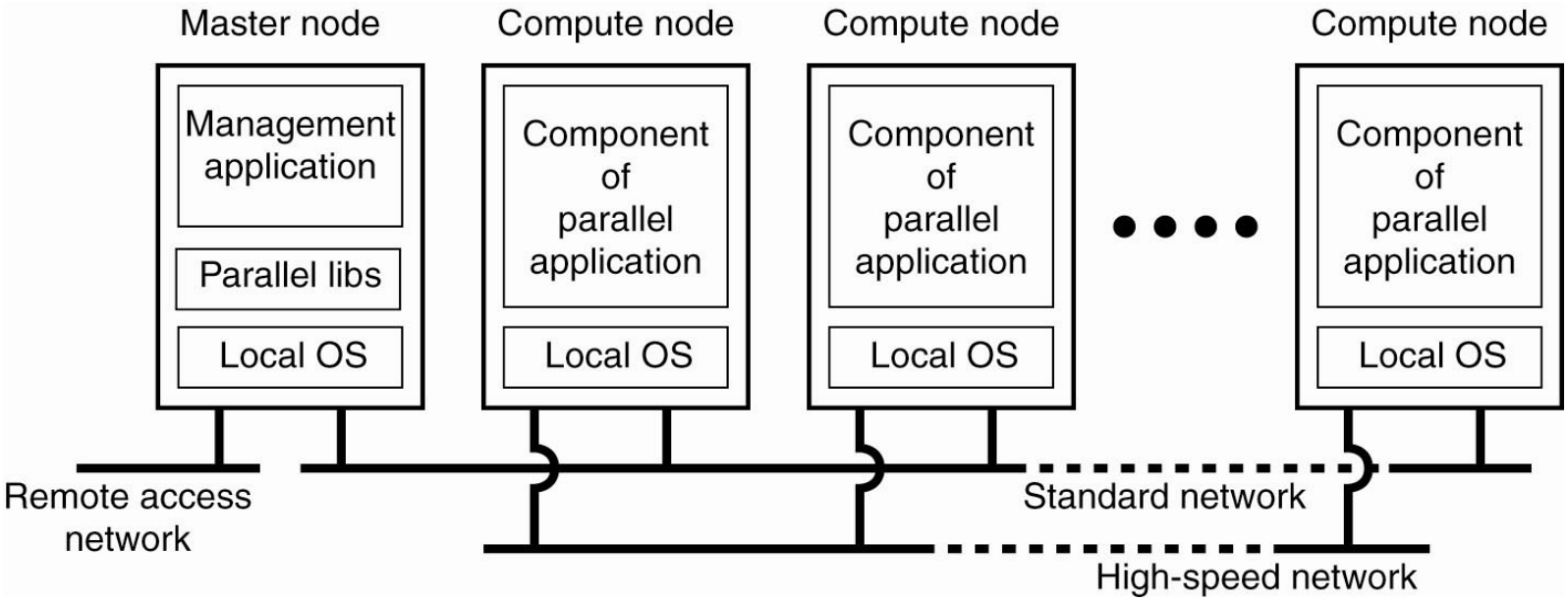
- Architecture of Clustered Systems.



General structure of a clustered system

Clustered Systems

- Layout of Clustered Systems.

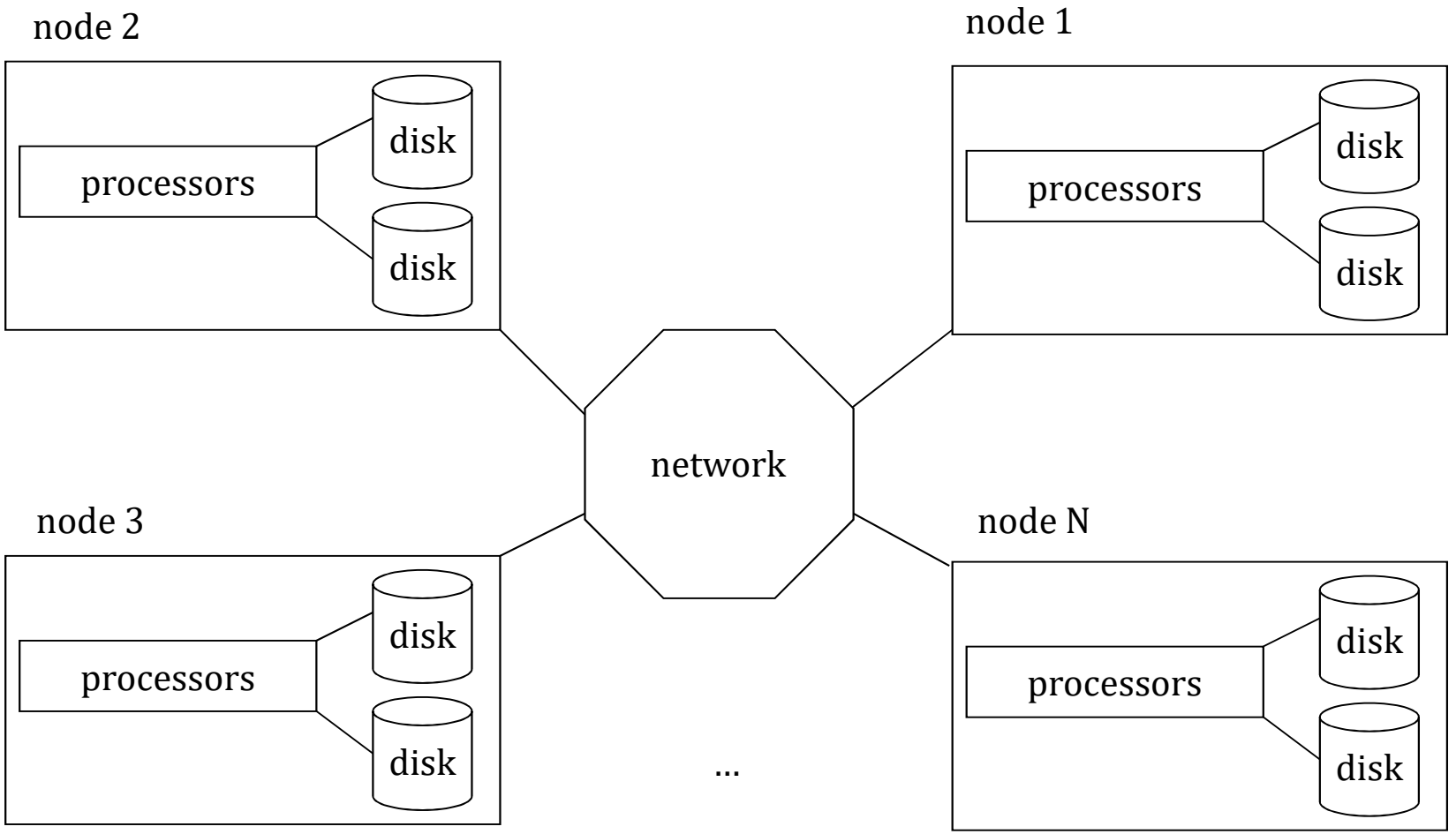


Networked Systems

- Network Systems distribute resources and computation among several physical processors.
  - Requires networking infrastructure
  - Most are Local Area Networks (LAN) or Wide Area Networks (WAN).
  - Also loosely coupled (松耦合系统)
    - Each processor has its own local memory.
    - Processors communicate with one another through various communications lines.
- Advantages:
  - Resources sharing
  - Computation speed up – load sharing
  - Reliability
- May be either Centralized Sever, Client/Server or Peer-to-Peer (P2P, 对等网络) systems

Networked Systems

- Layout of Networked Systems.

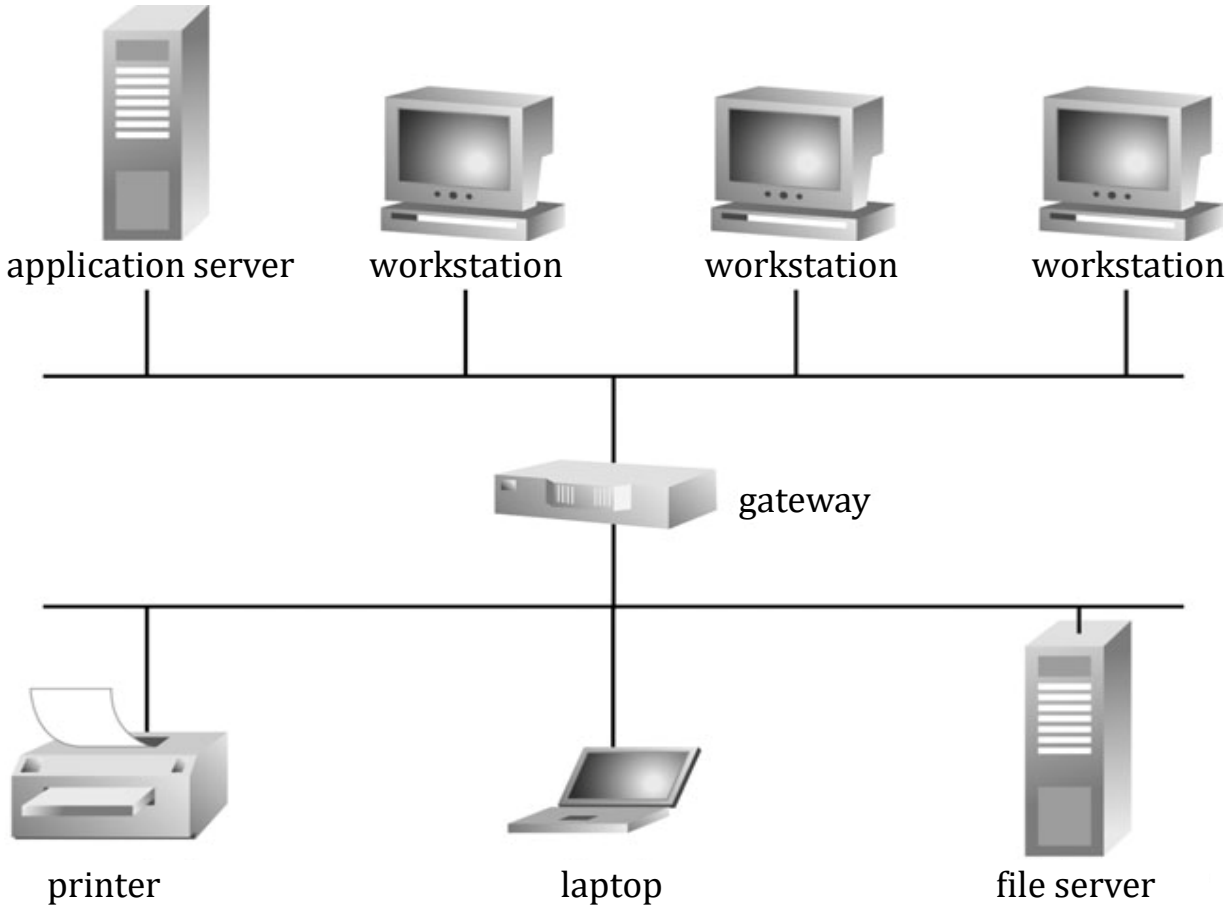


Networked Systems

- Local Area Network (LAN)
  - LAN designed to cover small geographical area
    - Multiaccess bus, Ring, or Star network
      - 多路总线、环形网络或星型网络
    - *Speed ≈ 10 – 100 Megabits/second.
    - Broadcast is fast and cheap.
    - Nodes:
      - usually workstations and/or personal computers
      - a few (usually one or two) mainframes

Networked Systems

- Local Area Network (LAN)
  - LAN designed to cover small geographical area.

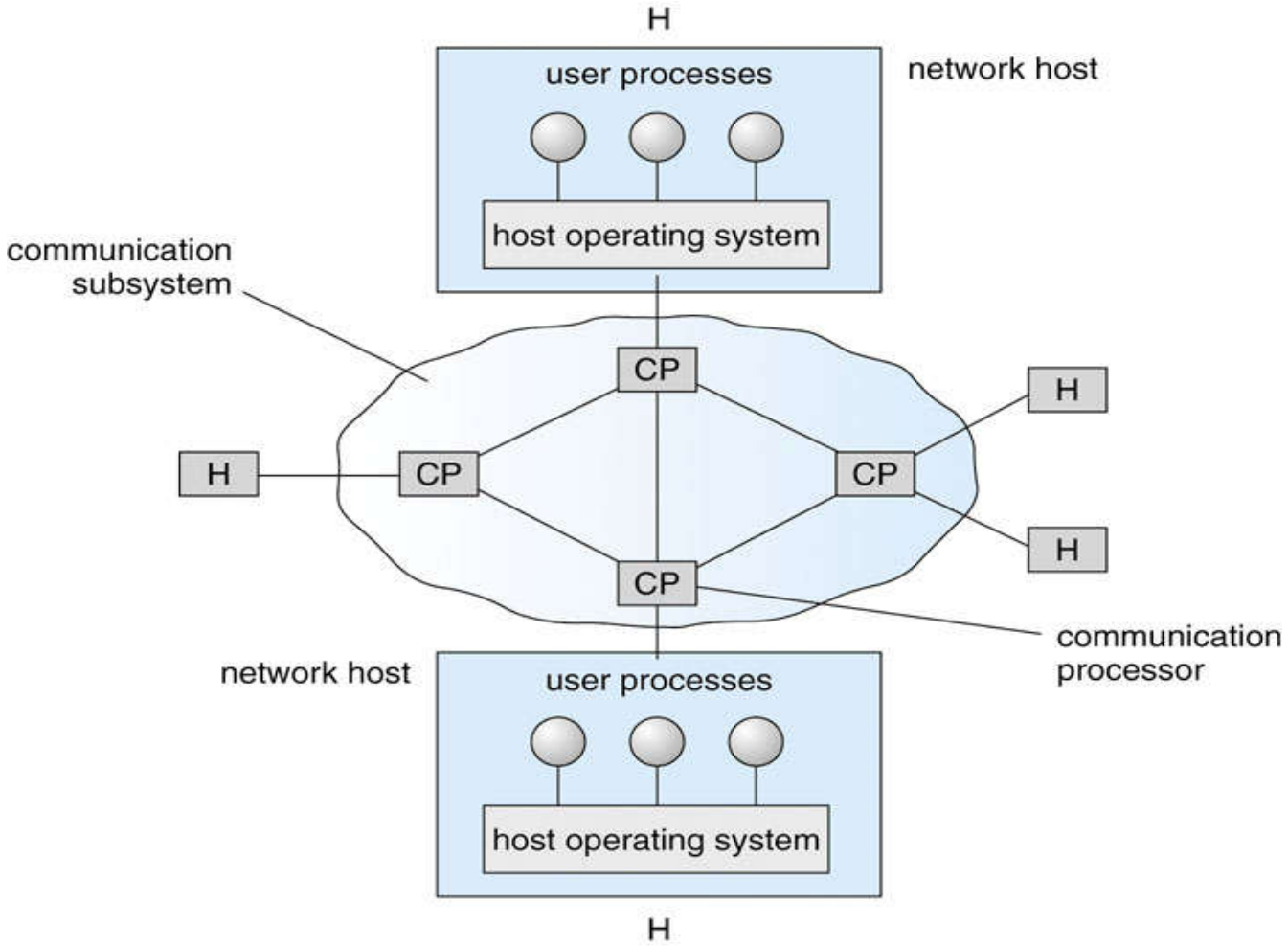


Networked Systems

- Wide Area Network (WAN)
  - Links geographically separated sites
  - Point-to-point connections over long-haul lines (长途线路)
    - e.g., leased from a phone company
  - *Speed ≈ 1.544 – 45 Megabits/second.
  - Broadcast usually requires multiple messages.
  - Nodes:
    - usually a high percentage of mainframes

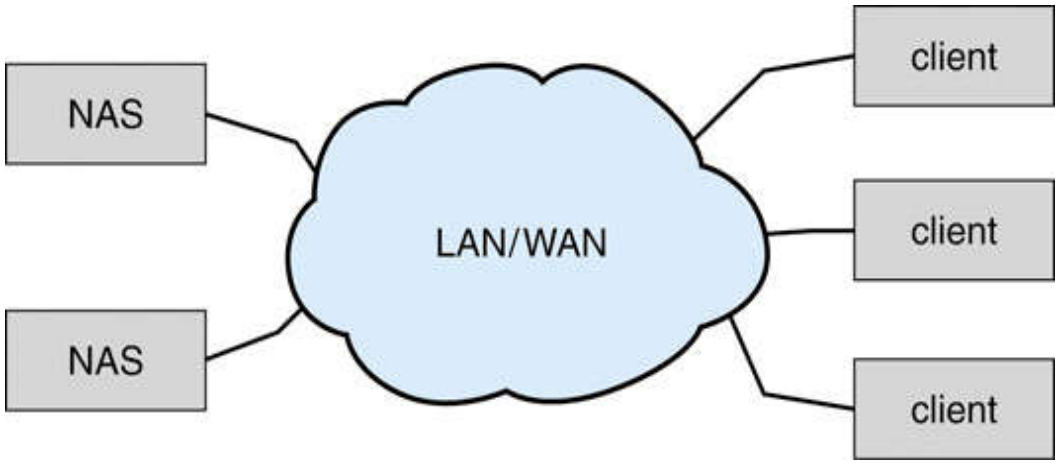
Networked Systems

- Wide Area Network (WAN)
  - Links geographically separated sites.



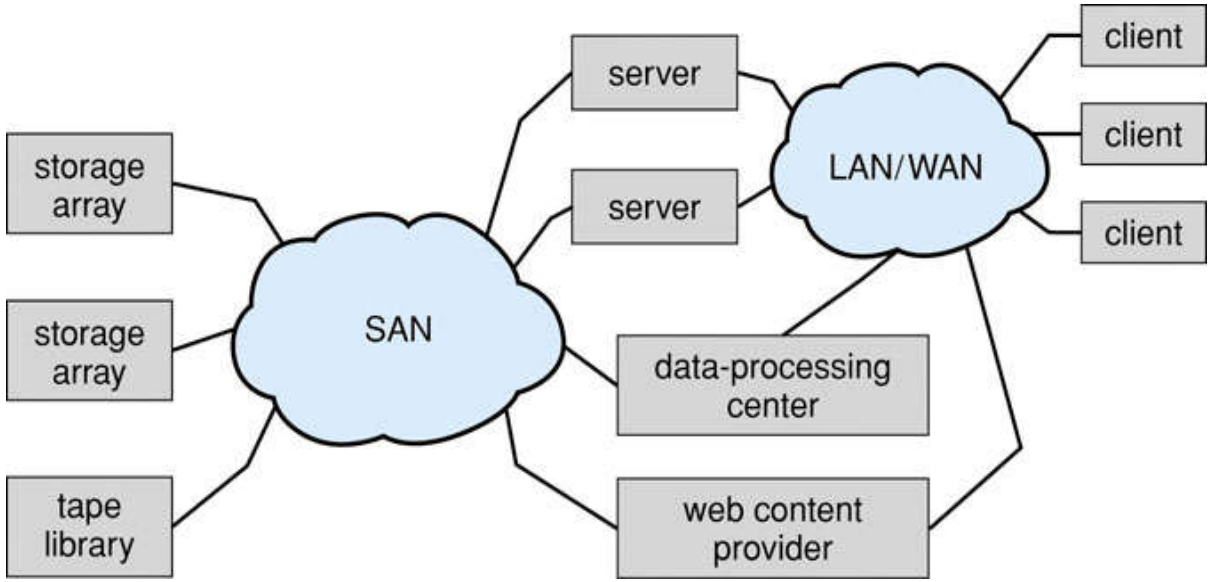
Networked Systems

- Network-attached Storage (NAS)
  - 网络附加存储.



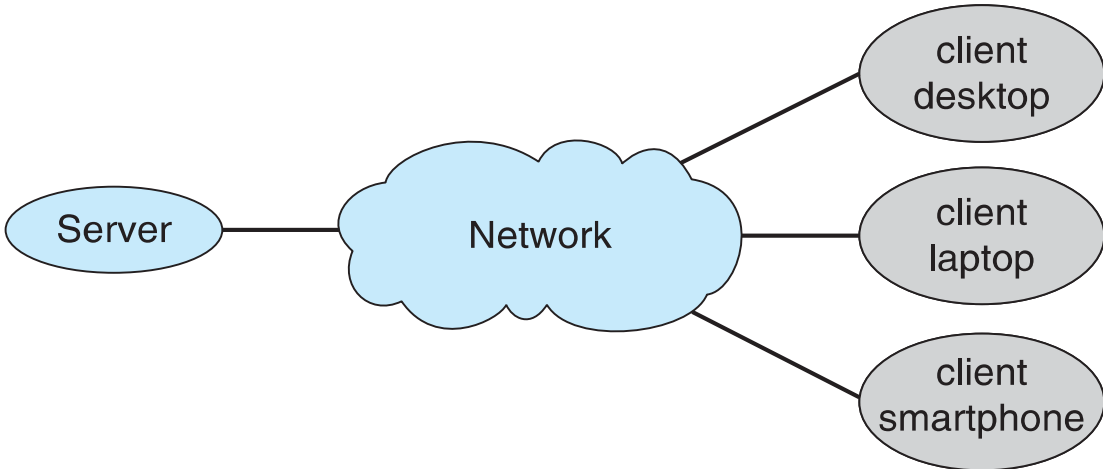
Networked Systems

- Storage-area Network (SAN)
  - 存储区域网络.



Networked Systems

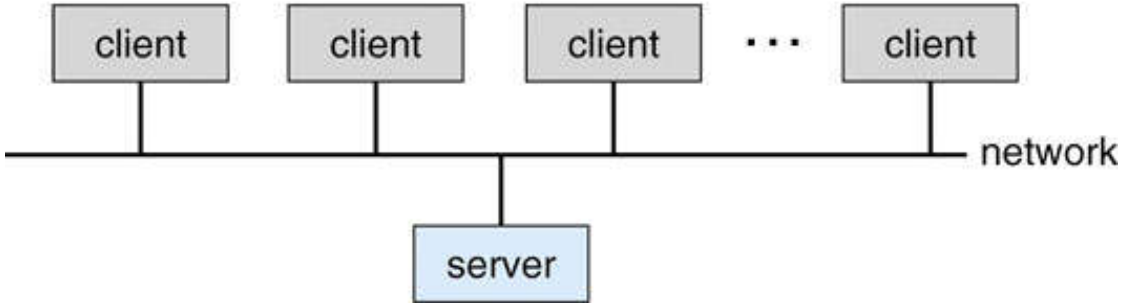
- Client/Server Computing
  - Dumb terminals supplanted by smart PCs (哑终端被智能PC取代)
  - Servers respond to requests generated by clients
    - A compute-server provides an interface for client to request services.
      - E.g., Database access
    - A file-server provides an interface for clients to store and retrieve files.

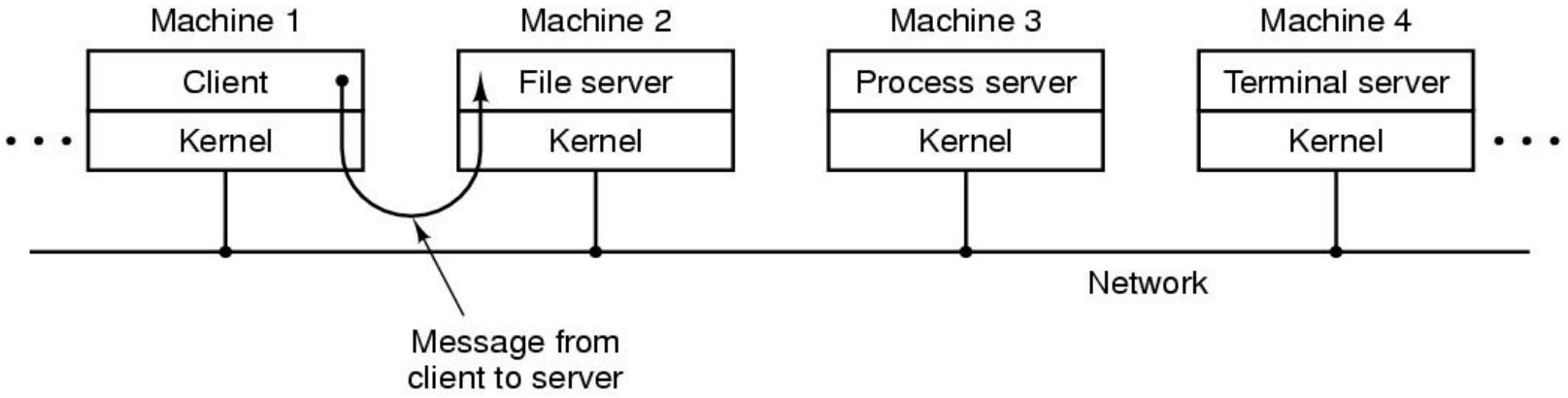


Networked Systems

Client/Server Computing

Layout of Client/Server Systems.





Networked Systems

- Peer-to-Peer (P2P) Systems
  - A P2P network (对等网络) does not distinguish clients and servers.
    - Instead all nodes are considered peers.
    - Each node may act as a client, a server or both.
  - A node joining a P2P network must:
    - register its service with central lookup service on network, or
    - broadcast request for service and responds to requests for service via discovery protocol.
  - Examples
    - Napster
    - Gnutella
    - VoIP such as Skype.

Networked Systems

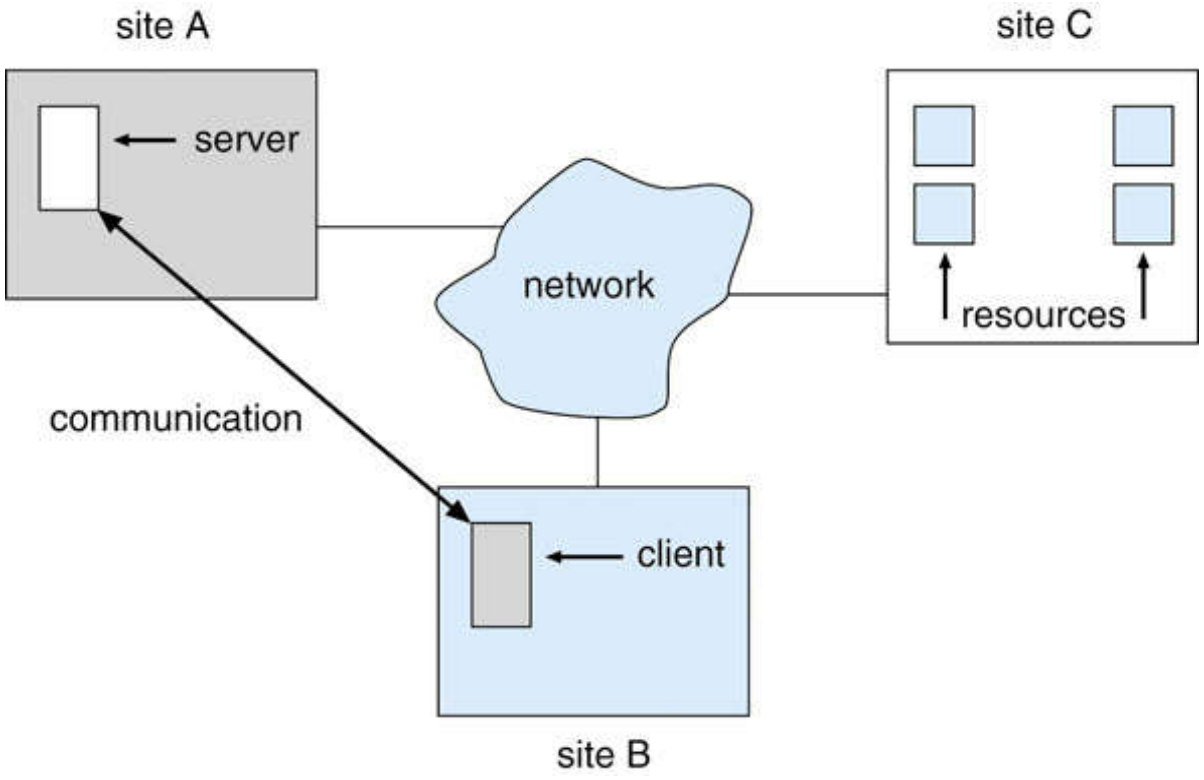
- Networked Operating Systems (NOS)
  - Each computer runs independently from other computers on the network.
    - Provides mainly file sharing
  - Users are aware of multiplicity of machines (用户需要考虑到机器的多样性).
  - Access to resources of various machines is done explicitly by:
    - remote logging into the appropriate remote machine
      - Telnet, SSH.
    - remote Desktop
      - Microsoft Windows.
    - transferring data from remote machines to local machines
      - File Transfer Protocol (FTP).

Distributed Systems

- Distributed System (分布式系统) is collection of loosely coupled processors interconnected by a communications network.
  - Processors variously called nodes, computers, machines, hosts.
- Reasons for distributed systems:
  - Resource sharing
    - sharing and printing files at remote sites
    - processing information in a distributed database
    - using remote specialized hardware devices
  - Computation speedup
    - load sharing
  - Reliability
    - detect and recover from site failure, function transfer, reintegrate failed site
  - Communication
    - message passing

Distributed Systems

- Layout of Distributed Systems.



Distributed Systems

- Distributed Operating Systems (DOS)
  - A DOS gives users the impression that there is a single operating system controlling the network.
    - Network is mostly transparent.
    - It's a powerful virtual machine.
  - Users are not aware of multiplicity of machines.
    - be different from that of NOS
    - access to remote resources similar to access to local ones
  - Data migration (数据迁移)
    - transfer data by transferring the entire file, or transferring only those portions of the file necessary for the immediate task
  - Computation migration (计算迁移)
    - transfer the computation, rather than the data, across the system
  - Process migration (进程迁移)
    - execute an entire process, or parts of it, at different sites

Web-based Systems

- Web has become ubiquitous (web 无处不在).
- PCs are most prevalent devices.
- more devices becoming networked to allow web access
- new category of devices to manage Web traffic among similar servers
  - Load Balancers
- basis for Grids/Cloud Computing

Web-based Systems

- Grid Computing Systems (网格计算系统)
  - Collection of computer resources, usually owned by multiple parties and in multiple locations, connected together such that users can share access to their combined power:
    - Can easily span a WAN (容易跨越广域网)
    - Heterogeneous environment (多样化环境)
    - Crosses administrative/geographic boundaries
    - Supports Virtual Organizations (VOs)
  - Examples.
    - EGEE
      - Enabling Grids for E-SciencE (Europe)
    - OSG
      - Open Science Grid (USA)

Web-based Systems

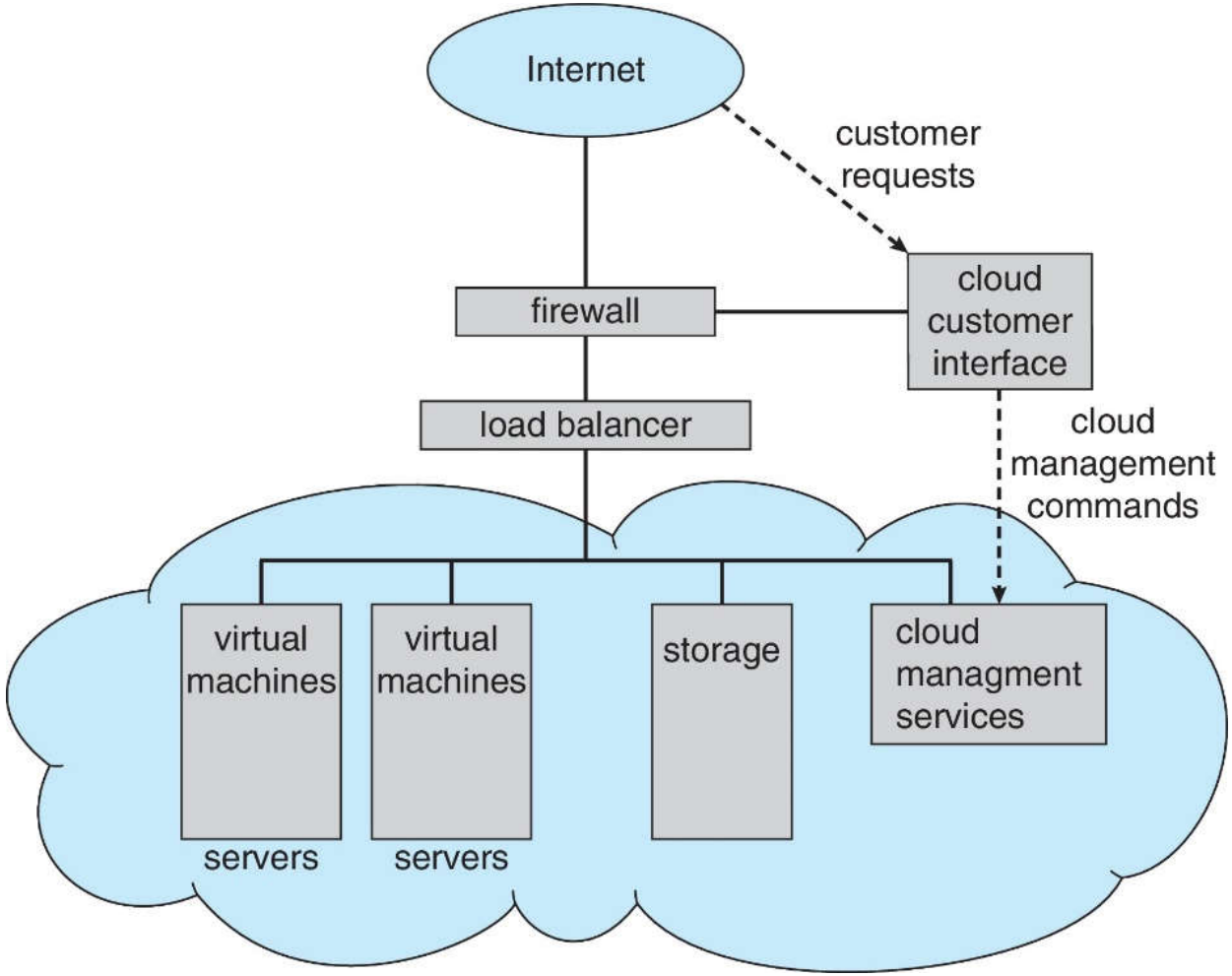
- Cloud Computing Systems
  - Collection of computer resources, usually owned by a single entity, connected together such that users can lease access to a share of their combined power
  - Characteristics of Cloud Computing Systems:
    - Location independence
      - The user can access the desired service from anywhere, using any device with any (supported) system.
    - Cost-effectiveness
      - The whole infrastructure is owned by the provider and requires no capital outlay (基础投资) by the user.
    - Reliability
      - enhanced by way of multiple redundant sites, though outages can occur, leaving users unable to remedy the situation (然而也会出现用户无法补救的服务中断状态)
    - Scalability
      - User needs can be tailored to available resources as demand dictates – cost benefit is obvious. (可伸缩性)

Web-based Systems

- Cloud Computing Systems
  - Characteristics of Cloud Computing Systems: (cont.)
    - Security
      - Low risk of data loss thanks to centralization, though problems with control over sensitive data need to be solved.
    - Readily consumable
      - The user usually does not need to do much deployment or customization, as the provided services are easy to adopt and ready-to-use. (用户通常不需要做太多的部署或定制工作，因为提供的服务容易获得且随时可用)
  - Examples
    - Amazon EC^2 (Elastic Compute Cloud)
    - Google App Engine
    - IBM Enterprise Data Center
    - MS Windows Azure
    - SUN Cloud Computing.

Web-based Systems

- Cloud Computing Systems.



Handheld Systems & Mobile Systems

- Handheld Systems
  - Handheld systems are also dedicated:
    - Personal Digital Assistants (PDAs)
    - Cellular telephones.
  - Issues
    - Limited memory
    - Slow processors
    - Small display screens
    - Support for multimedia (audio, video , images)
- Mobile Systems
  - handheld smartphones, tablets, etc.
  - use IEEE 802.11 wireless, or cellular data networks for connectivity
  - extra feature
    - more OS features like GPS, gyroscope
    - allows new types of apps like augmented reality
  - iOS and Google Android advance the mainstream development tendency.