Computer-System Architecture (cont.)

Symmetrically Constructed Clusters (cont.)

Advantages:

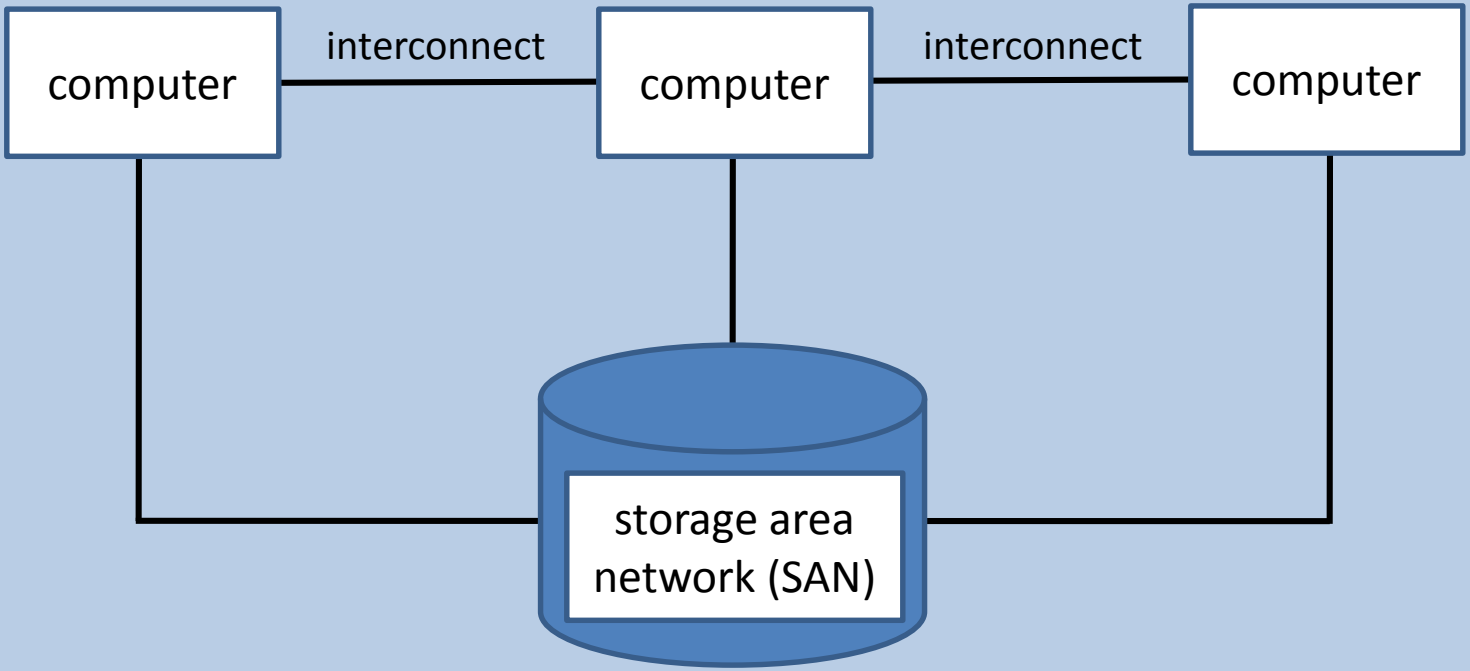
1. Greater computational power by running applications concurrently on all computers in the cluster.

Disadvantages:

1. Applications must be written specifically to take advantage of the cluster by using a technique known as parallelization (dividing a program into separate components that run in parallel on individual computers in the cluster).

Parallelization – dividing a program into separate components that run in parallel on individual computers in the cluster.

Computer-System Architecture (cont.)



General structure of a clustered system.

Operating-System Structure

The operating system provides the environment within which programs are executed.

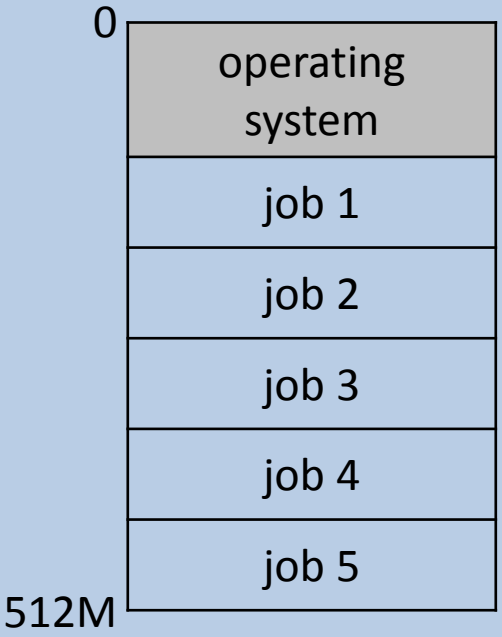
Operating systems must have the ability to multiprogram. (A single program cannot, in general, keep either the CPU or the I/O devices busy at all times.)

Single users frequently have multiple programs running. Multiprogramming increases CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

The operating system keeps several jobs in memory (on disk) simultaneously, which is called a job pool.

This pool consists of all processes residing on disk awaiting allocation of main memory. (to be executed)

Operating-System Structure



Memory layout for the multiprogramming system.

The set of jobs in memory can be a subset of the jobs kept in the job pool. The operating system picks and begins to execute one of the jobs in memory. Eventually, the job may have to wait for some task, such as an I/O operation to complete. (causes CPU to sit idle until process of needed job is complete)

Operating-System Structure

Multiprogrammed systems provide an environment which the various system resources (i.e. CPU, memory, and peripheral devices) are utilized effectively, but they do not provide for user interaction with the computer system.

In time sharing (or multitasking) systems, the CPU executes multiple jobs by switching among them, but the switch occurs so frequently that the user can interact with each program while it is running. Time sharing requires an interactive (or hands-on) computer system, which provides direct communication between the user and the system. The user gives instructions to the operating system or to a program directly, using an input device such as a keyboard or a mouse, and waits for immediate results on an output device. (*response time should be short*)

Operating-System Structure

A time-shared operating system allows many users to share the computer simultaneously. Since each action or command in a time-shared system tends to be short, only a little CPU time is needed for each user. As the system switches rapidly from one user to the next, each user is given the impression that the entire computer system is dedicated to his use, even though it is being shared among many users.

Time-shared operating systems uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer. Each user has at least one separate program in memory. A program loaded into memory and executing is called a process.

Operating-System Structure

The Execution of a process

- Executes for a short time.
- Either finishes or performs I/O.
  - Display output for user
  - User may respond using keyboard, mouse, etc.
    - Which compared to computer speed could be slow
    - Or if user leaves computer, CPU may be waiting on response

- Rather than sit idle, the operating system rapidly switch the CPU to the program of another user.

Operating-System Structure

Job Scheduling

Both time sharing and multiprogramming require that several jobs be kept simultaneously in memory. If several jobs are required to load and there is not enough

room in memory, the system must choose among them.

Making the decision is called job scheduling.

Jobs are selected and loaded into memory. If several jobs are ready to run at the same time, the system must choose among them. Making this decision is called CPU scheduling.

Several factors need to be considered when running multiple jobs concurrently. (i.e. *process scheduling, disk storage, and memory management*)

Operating-System Structure

In a time-sharing system, the operating system must ensure reasonable response time, that is sometimes accomplished through swapping (by keeping portions of the primary memory in secondary storage).

While multitasking and memory swapping are two completely unrelated techniques, they are very often used together, as swapping memory allows more tasks to be loaded at the same time.

Multitasking system allows another process to run when the running process hits a point where it has to wait for some portion of memory to be reloaded from secondary storage.

Operating-System Structure

Another way of insuring reasonable response time is virtual memory. This technique allows the execution of a process that is not completely in memory.

The main advantage of the virtual memory scheme is that it enable users to run programs that are larger than the actual physical memory. This allows programmers not to be concerned with memory-storage limitations.

Time-sharing systems must also provide a file system. The file system resides on a collection of disks; hence, disk management must be provided. Also time-sharing system provide a mechanism for protecting resources from inappropriate use. To ensure orderly execution, the system must provide mechanisms for job synchronization and communication, ensuring that jobs do not get stuck in a deadlock (forever waiting for one another).

Operating-System Operations

Modern operating systems are interrupt driven. If there are no process to execute, no I/O devices to service, and no users to who to respond, the operating system will sit quietly, waiting for something to happen.

Events are almost always signaled by the occurrence of an interrupt or a trap. A trap (or an exception) is a software-generated interrupt caused either by an error (division by zero or invalid memory access) or by a specific request from a user program that an operating system service be performed.

For each type of interrupt, separate segments of code in the operating system determine what action should be taken.

Operating-System Operations

Dual-Mode Operation

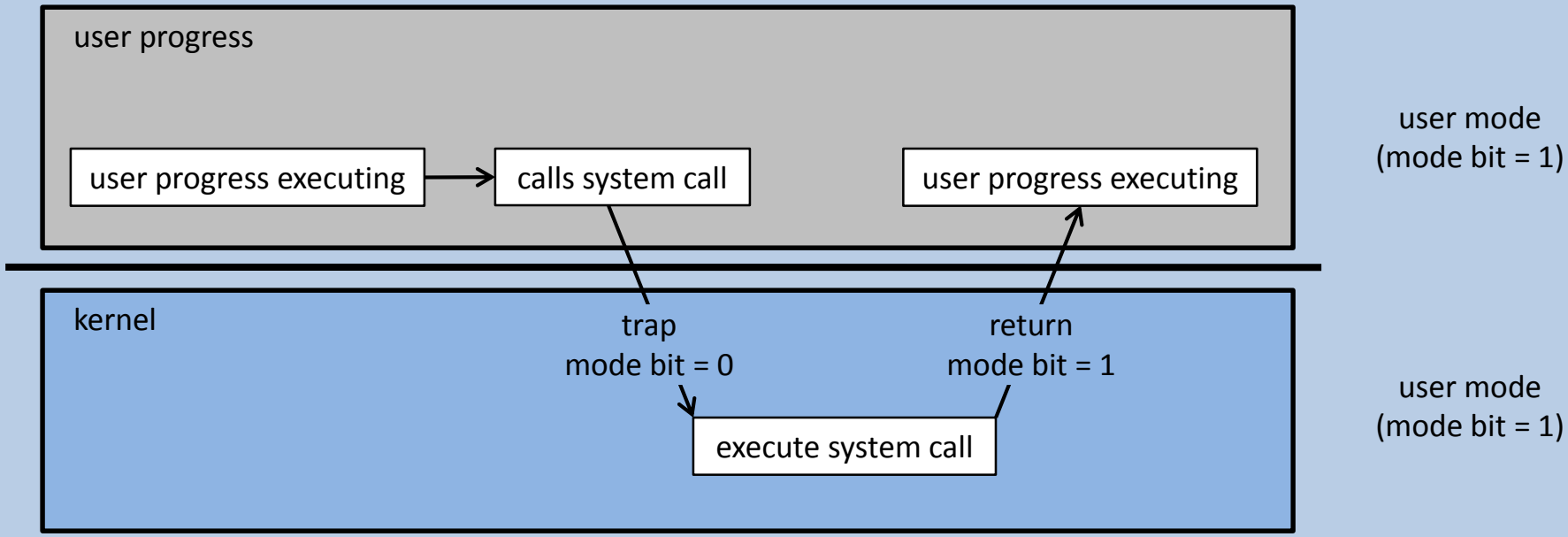
In order to ensure the proper execution of the operating system, we must be able to distinguish between the execution of operating-system code and user-defined code.

Computer must have two modes of operation.

Kernel mode (0 bit)	User mode (1 bit)
Supervisor mode	
System mode	
Privilege mode	

A bit, called the mode bit, is added to the hardware of the computer to distinguish between a task that is executed on behalf of the operating system and one that is executed on behalf of the user.

Operating-System Operations



Transition from user to kernel mode.

At system boot time, the hardware starts in kernel mode. The operating system is then loaded and starts user application in user mode. Whenever a trap or interrupt occurs, the hardware switches from user mode to kernel mode (changes mode bit to 0).

Process Management

The dual mode of operation provides the means for protecting the operating system from errant users and errant users from one another. This protection is accomplished by designating some of the machine instructions that may cause harm as privileged instructions.

Once hardware protection is in place, the OS detects errors that violate modes. (attempting to execute illegal instructions, or access memory not in user's address space)

Operating-System Operations

Timer

We must ensure that the operating system maintains control over the CPU. We cannot allow the user program to get stuck in an infinite loop or to fail to call system services and never return control to the operating system. To accomplish this goal, we can use a timer. A timer can be set to interrupt the computer after a specified time. (fixed or variable - i.e. 1 millisecond to 1 second)

The operating system ensures that the timer is set to interrupt. A program with a 7 minute time limit would have its counter initialized to 420. Every second, the timer interrupts and the counter is decremented by 1. As long as the counter is positive, control is returned to the user program. When the counter becomes negative, the operating system terminates the program for exceeding the assigned time limit.

Process Management

A program does nothing unless its instructions are executed by the CPU. A program in execution is a process. A time-shared user program such as a compiler is a process. A word-processing program being run by an individual user on a PC is a process. A system task, such as sending output to a printer can also be a process.

A process needs certain resources – including CPU time, memory, files, and I/O devices – to accomplish its task.

A single-thread process has one program counter specifying the next instruction to execute.

In short, a **thread of execution** is the smallest unit of processing that can be scheduled by an operating system.

Process Management

The operating system is responsible for the following activities in connection with process management:

- Scheduling processes and threads on the CPUs
- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication

Memory Management

The main memory is central to the operation of a modern computer system. Main memory is a large array of words or bytes, ranging in size from hundreds of thousands to billions. Each word or byte has its own address. Main memory is a repository of quickly accessible data shared by the CPU and I/O devices. The central processor reads instructions from main memory during the instruction-fetch cycle and both reads and write data from main memory during the data-fetch cycle (von Neumann architecture).

For the CPU to process data from disk, data must first be transferred to main memory by CPU-generated I/O calls.

Memory Management

To improve both the utilization of the CPU and the speed of the computer's response to its user, general-purpose computers must keep several programs in memory, creating a need for memory management.

The operating system is responsible for the following activities in connection with memory management:

- Keeping track of which parts of memory are currently being used and by whom
- Deciding which processes and data to move into and out of memory
- Allocating and deallocating memory space as needed

Storage Management

The operating system provides a uniform, logical view of information storage. The operating system abstracts from the physical properties of its storage devices to define a logical storage unit, the file. The operating system maps files onto physical media and access these files via the storage device.

File-System Management

Most Common Types of Physical Storage

- Magnetic disk
- Optical disk
- Magnetic tape