Multiprocessor and Real-Time Scheduling

Chapter 10

Classifications of Multiprocessor

- **Loosely coupled multiprocessor**
  - each processor has its own memory and I/O channels
- **Functionally specialized processors**
  - such as I/O processor
  - controlled by a master processor
- **Tightly coupled multiprocessing**
  - processors share main memory
  - controlled by operating system

Independent Parallelism

- **Separate processes running**
- **No synchronization**
- **Example is time sharing**
  - average response time to users is less

Very Coarse Parallelism

- **Distributed processing across network nodes to form a single computing environment**
- **Good when there is infrequent interaction among processes**
  - overhead of network would slow down communications

Coarse Parallelism

- Similar to running many processes on one processor except it is spread to more processors
- Multiprocessing

335

Medium Parallelism

- Parallel processing or multitasking within a single application
- Single application is a collection of threads
- Threads usually interact frequently

336

Process Scheduling

- Single queue for all processes
- Multiple queues are used for priorities
- All queues feed to the common pool of processors
- Specific scheduling disciplines is less important with more than on processor

Threads

- Executes separate from the rest of the process
- An application can be a set of threads that cooperate and execute concurrently in the same address space
- Threads running on separate processors yields a dramatic gain in performance

Multiprocessor Thread Scheduling

- **Load sharing**
  - processes are not assigned to a particular processor
- **Gang scheduling**
  - a set of related threads is scheduled to run on a set of processors at the same time

Multiprocessor Thread Scheduling

- **Dedicated processor assignment**
  - threads are assigned to a specific processor
- **Dynamic scheduling**
  - number of threads can be altered during course of execution

Load Sharing

- **Load is distributed evenly across the processors**
- **Assures no processor is idle**
- **No centralized scheduler required**
- **Use global queues**

Disadvantages of Load Sharing

- **Central queue needs mutual exclusion**
  - may be a bottleneck when more than one processor looks for work at the same time
- **Preemptive threads are unlikely to resume execution on the same processor**
  - cache use is less efficient
- **If all threads are in the global queue, all threads of a program will not gain access to the processors at the same time**

Gang Scheduling

- Simultaneous scheduling of threads that make up a single process
- Useful for applications where performance severely degrades when any part of the application is not running
- Threads often need to synchronize with each other

Dedicated Processor Assignment

- When application is scheduled, its threads are assigned to a processor
- Some processors may be idle
- Avoids process switching

Dynamic Scheduling

- **Number of threads in a process are altered dynamically by the application**
- **Operating system adjust the load to improve use**
  - assign idle processors
  - new arrivals may be assigned to a processor that is used by a job currently using more than one processor
  - hold request until processor is available
  - new arrivals will be given a processor before existing running applications

Real-Time Systems

- **Correctness of the system depends not only on the logical result of the computation but also on the time at which the results are produced**
- **Tasks or processes attempt to control or react to events that take place in the outside world**
- **These events occur in “real time” and process must be able to keep up with them**

Real-Time Systems

- **Control of laboratory experiments**
- **Process control plants**
- **Robotics**
- **Air traffic control**
- **Telecommunications**

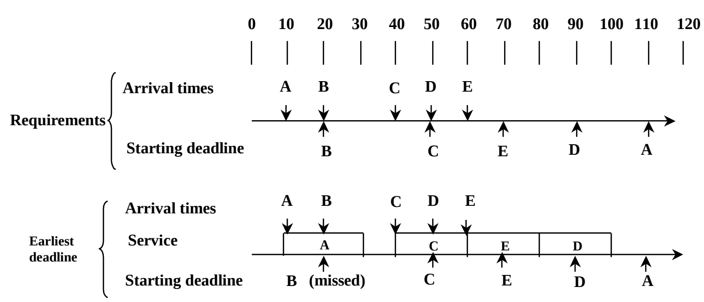
Real-Time Scheduling

- **Static table-driven**
  - determines at run time when a task begins execution
- **Static priority-driven preemptive**
  - traditional priority-driven scheduler is used
- **Dynamic planning-based**
- **Dynamic best effort**

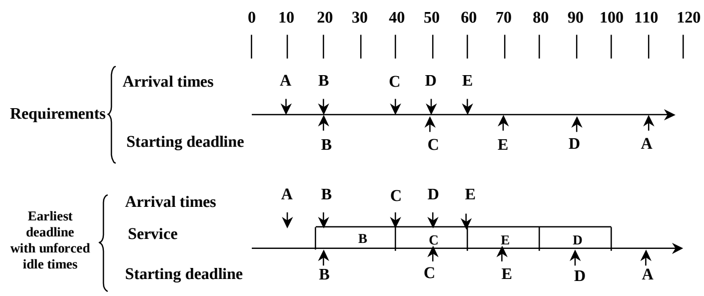
Deadline Scheduling

- **Real-time applications are not concerned with speed but with completing tasks**
- **Scheduling tasks with the earliest deadline minimized the fraction of tasks that miss their deadlines**
  - includes new tasks and amount of time needed for existing tasks

Scheduling of Real-Time Tasks



Scheduling of Real-Time Tasks



Scheduling of Real-Time Tasks