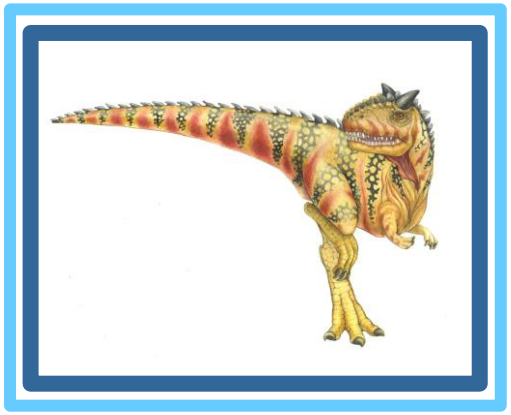
Chapter 5: CPU Scheduling



Scheduling Criteria

- **CPU utilization** – keep the CPU as busy as possible
  - CPU utilization range can be from 40 to 90 %
- **Throughput** – # of processes that complete their execution per time unit
- **Turnaround time** – amount of time to execute a particular process — from the time of submission to completion
- **Waiting time** – amount of time a process has been waiting in the ready queue
- **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

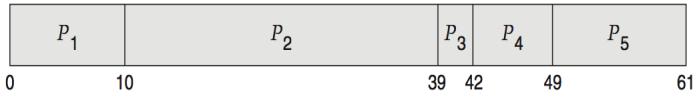
Algorithm Evaluation

- How to select CPU-scheduling algorithm for an OS?
- Determine criteria, then evaluate algorithms
- **Deterministic modeling**
  - Type of **analytic evaluation**
  - Takes a particular predetermined workload and defines the performance of each algorithm for that workload
- Consider 5 processes arriving at time 0:

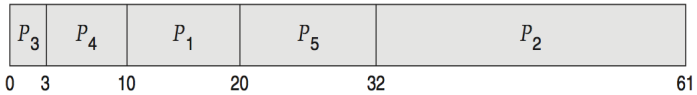
Process	Burst Time
P_1	10
P_2	29
P_3	3
P_4	7
P_5	12

Deterministic Evaluation

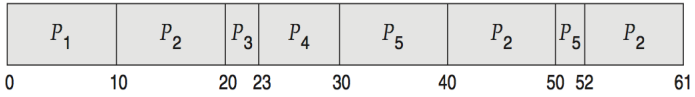
- For each algorithm, calculate minimum average waiting time
- Simple and fast, but requires exact numbers for input, applies only to those inputs
  - FCS is 28ms:



  - Non-preemptive SFJ is 13ms:



  - RR is 23ms:



Activity

- Consider P1, P2 and P3

Process ID	Arrival Time	Burst Time
P1	0	5
P2	1	7
P3	3	4

Priority is P3, P2, P1

- What is the completion order of P1, P2 and P3 for FCFS, SJF, Priority queue and RR2.

Which algorithm would you prefer for this set of processes and why?

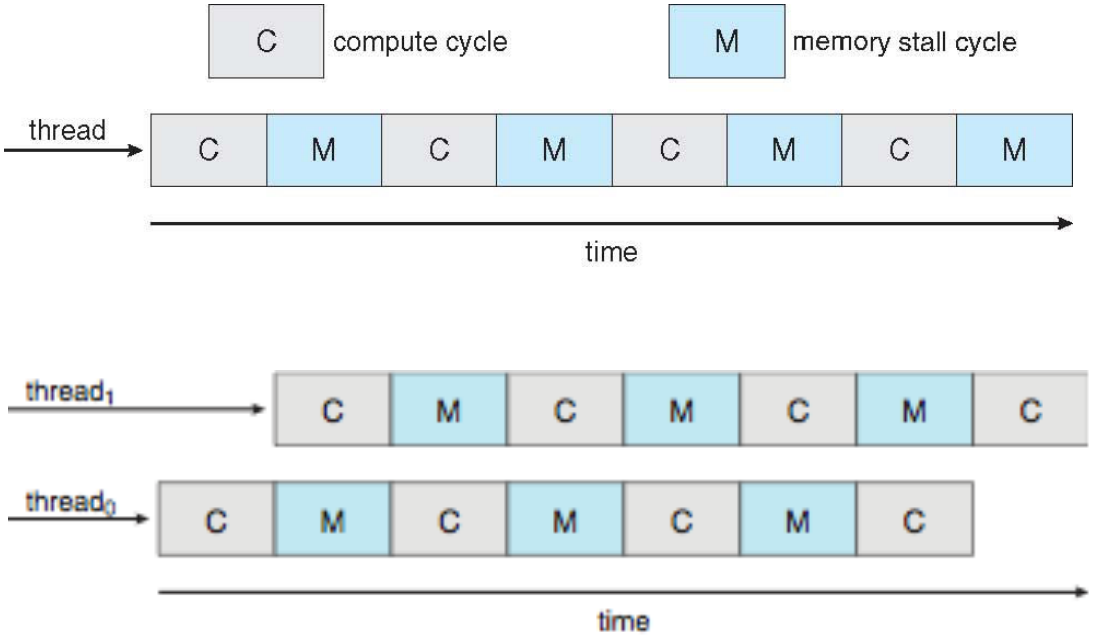
Multiple-Processor Scheduling

- Multiple processors?
- CPU scheduling more complex when multiple CPUs are available
- **Asymmetric multiprocessing** – only one processor accesses the system data structures, alleviating the need for data sharing
- **Symmetric multiprocessing (SMP)** – each processor is self-scheduling, all processes in common ready queue, or each has its own private queue of ready processes
  - Currently, most common

Multiple-Processor Scheduling – Load Balancing

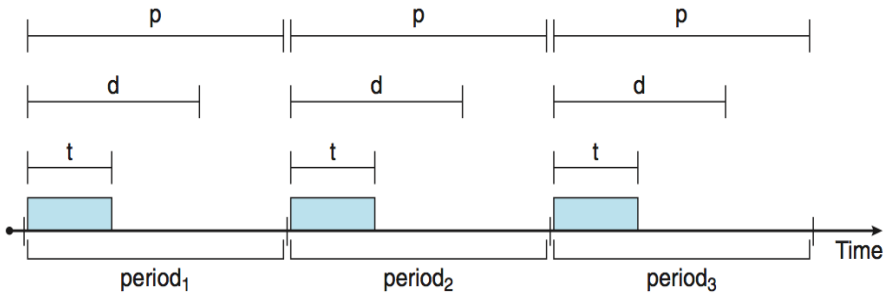
- If SMP, need to keep all CPUs loaded for efficiency
- **Load balancing** attempts to keep workload evenly distributed
  - **Push migration** – periodic task checks load on each processor, and if found pushes task from overloaded CPU to other CPUs
  - **Pull migration** – idle processors pulls waiting task from busy processor

Multithreaded Multicore System



Priority-based Scheduling

- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
  - But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: **periodic** ones require CPU at constant intervals
  - Has processing time *t*, deadline *d*, period *p*
  - $0 \le t \le d \le p$
  - **Rate** of periodic task is $1/p$



Rate Montonic Scheduling

- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority;
- Longer periods = lower priority
- Example with three processors

Process	Capacity	Period
P1	3	20
P2	2	5
P3	2	10

Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:

  the earlier the deadline, the higher the priority;

  the later the deadline, the lower the priority

Process	Capacity	Period	deadline
P1	3	20	7
P2	2	5	4
P3	2	10	8

Implementation

- Even simulations have limited accuracy
- Just implement new scheduler and test in real systems
  - High cost, high risk
  - Environments vary
- Most flexible schedulers can be modified per-site or per-system
- Or APIs to modify priorities
- But again environments vary

Activity:
Apply rate monotonic scheduling:

Process.	Capacity.	Period
P1	2	5
P2	1	8
P3.	4	10

End of Chapter 5