CSE 4/521 Introduction to Operating Systems

Lecture 8 – CPU Scheduling I

(Basic Concepts, Scheduling Criteria, Scheduling Algorithms)

Summer 2018

Overview

Objective:

1. To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
2. To describe various CPU-scheduling algorithms

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Recap

- Mutex Locks
  - Acquire() and release() a lock. Drawback: Busy-waiting
- Semaphores
  - Wait() and signal(). No busy-waiting
- Classic Problems of Synchronization
  - Bounded-buffer problem, Reader-Writer problem, Dining Philosopher's problem
- Synchronization Examples
  - ---

Questions

1. What are the 3 classic problems of synchronization? (Easy)

2. What are some of the synchronization primitives used in real-world? (Medium)

3. Does semaphores totally eliminate busy-waiting? If not, then why is it considered better than spinlocks? (Hard)

Overview

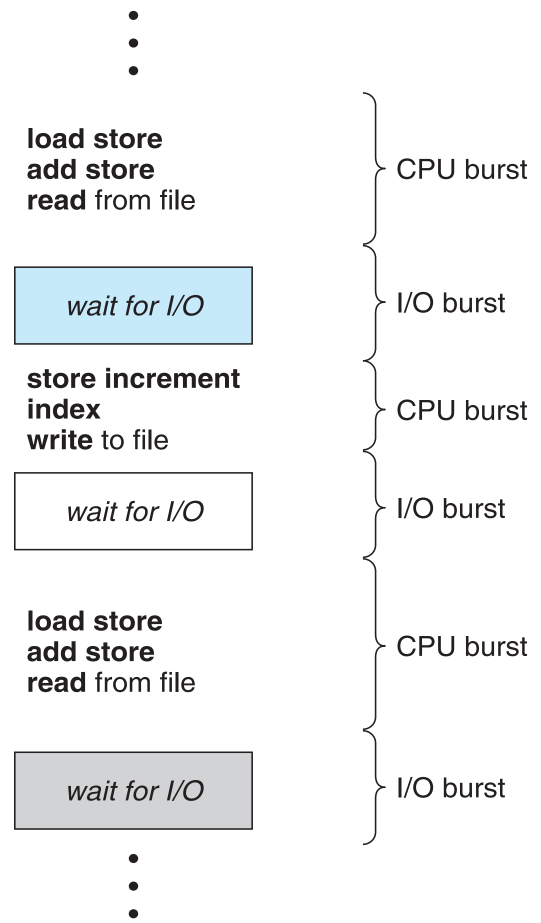
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Overview

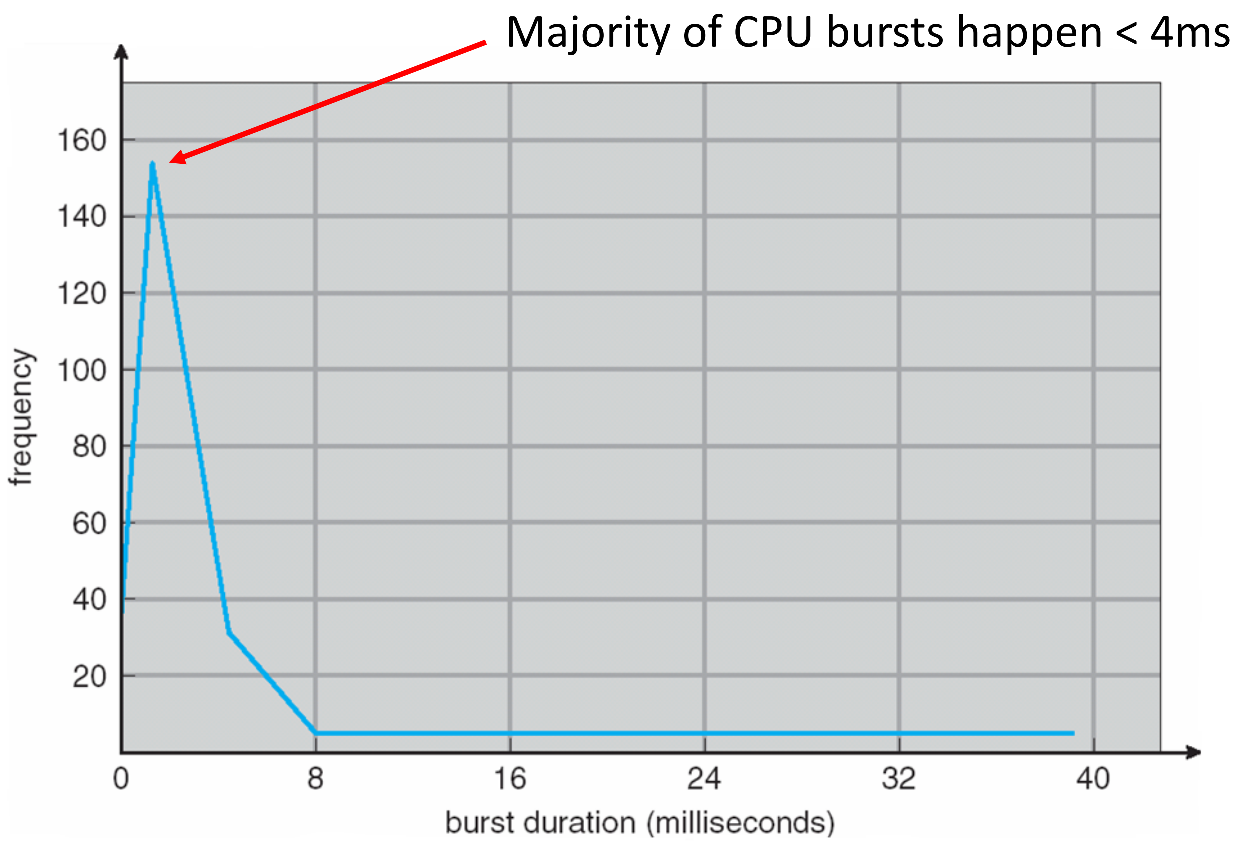
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait
- CPU burst followed by I/O burst
- CPU burst distribution is of main concern



Basic Concepts – Histogram of CPU-burst time



CPU Scheduler

- Short-term scheduler selects from among the processes in ready queue, and allocates the CPU to one of them
- CPU scheduling decisions may take place when a process:
  1. Switches from running to waiting state
  2. Switches from running to ready state
  3. Switches from waiting to ready
  4. Terminates

Overview

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Scheduling Criteria

1. CPU utilization – keep the CPU as busy as possible
2. Throughput – # of processes that complete their execution per time unit
3. Turnaround time – amount of time to execute a particular process
4. Waiting time – amount of time a process has been waiting in the ready queue
5. Response time – amount of time it takes from when a request was submitted until the first response is produced

Scheduling Critera

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time

Overview

- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms

Scheduling Algorithms - FCFS

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:

- Waiting time for P_1 = 0; P_2 = 24; P_3 = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Scheduling Algorithms – FCFS

Suppose that the processes arrive in the order:

$$P_2 , P_3 , P_1$$

- The Gantt chart for the schedule is:

- Waiting time for $P_1 = 6$; $P_2 = 0$, $P_3 = 3$
- Average waiting time: (6 + 0 + 3)/3 = 3
- Much better than previous case
- Convoy effect - short process behind long process

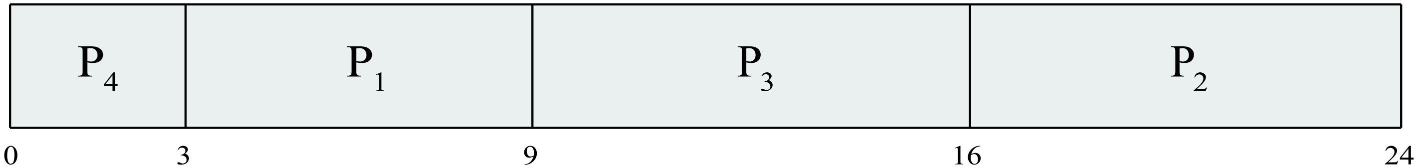
Scheduling Algorithms – Shortest-Job-First (SJF) Scheduling

- Associate with each process the length of its next CPU burst
  - Use these lengths to schedule the process with the shortest time
- SJF is optimal – gives minimum average waiting time for a given set of processes
  - The difficulty is knowing the length of the next CPU request
  - Could ask the user

Scheduling Algorithms – Shortest-Job-First (SJF) Scheduling

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart



- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Scheduling Algorithms – Determining Length of Next CPU Burst

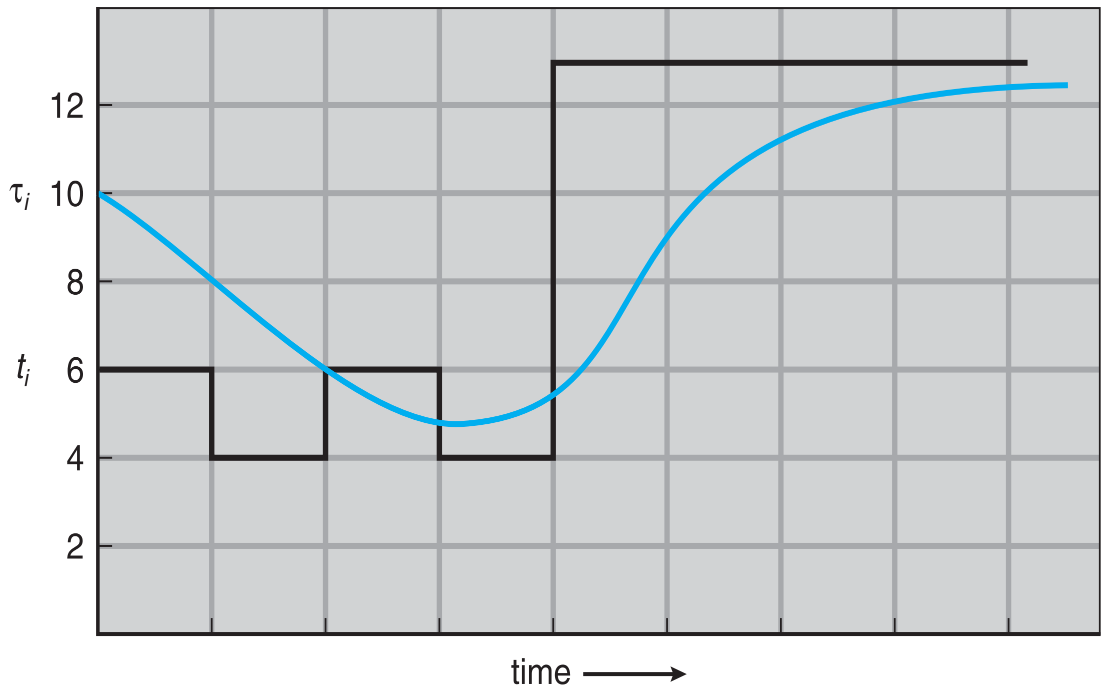
- Can only estimate the length – should be similar to the previous one
  - Then pick process with shortest predicted next CPU burst

- Can be done by using the length of previous CPU bursts, using exponential averaging

1. $t_n =$ actual length of n^{th} CPU burst
2. $\tau_{n+1} =$ predicted value for the next CPU burst
3. $\alpha, 0 \leq \alpha \leq 1$
4. Define: $\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$

- Commonly, α set to ½
- Preemptive version called shortest-remaining-time-first

Scheduling Algorithms – Prediction of the Length of the Next CPU Burst



CPU burst (t_i)		6	4	6	4	13	13	13 …
"guess" (τ_i)	10	8	6	6	5	9	11	12 …

Scheduling Algorithms – Examples of Exponential Averaging

- $\alpha = 0$
  - $\tau_{n+1} = \tau_n$
  - Recent history does not count
- $\alpha = 1$
  - $\tau_{n+1} = \alpha\, t_n$
  - Only the actual last CPU burst counts
- If we expand the formula, we get:

$$\tau_{n+1} = \alpha\, t_n + (1 - \alpha)\alpha\, t_{n-1} + \dots$$
$$+ (1 - \alpha)^j \alpha\, t_{n-j} + \dots$$
$$+ (1 - \alpha)^{n+1} \tau_0$$

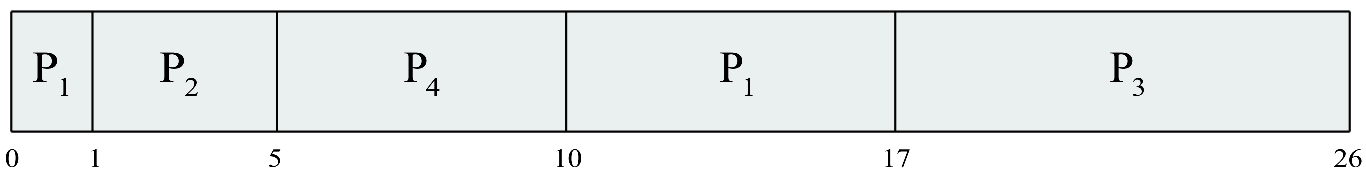
- Since both α and $(1 - \alpha)$ are less than or equal to 1, each successive term has less weight than its predecessor

Scheduling Algorithms – Examples of Shortest-remaining-time-first

- Now we add the concepts of varying arrival times and preemption to the analysis

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

- *Preemptive* SJF Gantt Chart



- Average waiting time = [(10-1)+(1-1)+(17-2)+5-3)]/4 = 26/4 = 6.5 msec

Scheduling Algorithms – Priority Scheduling

- A priority number (integer) is associated with each process

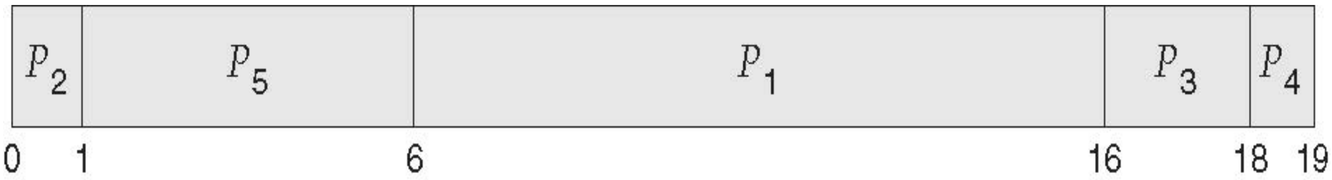
- The CPU is allocated to the process with the highest priority (smallest integer = highest priority)
  - Preemptive
  - Nonpreemptive

- Problem = Starvation – low priority processes may never execute
- Solution = Aging – as time progresses increase the priority of the process

Scheduling Algorithms – Example of Priority Scheduling

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart



- Average waiting time = 8.2 msec

Credits for slides

Silberschatz, Galvin and Gagne