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

Majority of CPU bursts happen < 4ms
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 Criteria

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

• Basic Concepts
• Scheduling Criteria
• Scheduling Algorithms
Scheduling Algorithms - FCFS

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>24</td>
</tr>
<tr>
<td>$P_2$</td>
<td>3</td>
</tr>
<tr>
<td>$P_3$</td>
<td>3</td>
</tr>
</tbody>
</table>

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

![Gantt Chart](chart)

- 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:

<table>
<thead>
<tr>
<th></th>
<th>P_2</th>
<th>P_3</th>
<th>P_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• 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

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>6</td>
</tr>
<tr>
<td>$P_2$</td>
<td>8</td>
</tr>
<tr>
<td>$P_3$</td>
<td>7</td>
</tr>
<tr>
<td>$P_4$</td>
<td>3</td>
</tr>
</tbody>
</table>

- 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, $\alpha$ set to $\frac{1}{2}$
• Preemptive version called shortest-remaining-time-first
Scheduling Algorithms – Prediction of the Length of the Next CPU Burst

\[ \tau_i \]

\[ t_i \]

CPU burst \( (t_i) \):
- 6
- 4
- 6
- 4
- 13
- 13
- 13
- ...

"guess" \( (\tau_i) \):
- 10
- 8
- 6
- 6
- 5
- 9
- 11
- 12
- ...

<table>
<thead>
<tr>
<th>t_i</th>
<th>\tau_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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:
  \[ \tau_{n+1} = \alpha t_n + (1 - \alpha)\alpha t_{n-1} + \ldots + (1 - \alpha)^j \alpha t_{n-j} + \ldots + (1 - \alpha)^{n+1} \tau_0 \]

- Since both $\alpha$ 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

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>$P_3$</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>$P_4$</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

• Preemptive SJF Gantt Chart

• Average waiting time = $\frac{[(10-1)+(1-1)+(17-2)+5-3]}{4} = \frac{26}{4} = 6.5$ 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

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>$P_2$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_3$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>$P_4$</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>$P_5$</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

- Priority scheduling Gantt Chart

- Average waiting time = 8.2 msec
Credits for slides

Silberschatz, Galvin and Gagne