CS 3305B

CPU Scheduling - I

Lecture 7

Jan 30 2017
When to Schedule

- Required on these occasions:
  - When a new process is created
  - When a process exits
  - When an I/O interrupt occurs
  - When a process blocks on I/O or a semaphore (more on this later)
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.
Alternating CPU And I/O Bursts

- CPU - I/O burst cycle:
  - Characterizes process execution
  - Alternates, between CPU and I/O activity.
  - CPU times are generally much shorter than I/O times.
Histogram of CPU-burst Times
CPU Schedular

- Selects from the Ready processes in memory

CPU scheduling decisions occur when process:
- A process switches from running to waiting state.
- A process switches from running to ready state.
- A process terminates.
When to Schedule

- **Non-preemptive**
  - Process runs until it voluntarily relinquishes CPU
    - Blocks on an event e.g., I/O or waiting on another process
  - Process terminates
When to Schedule

- **Preemptive**
  - Process runs for a maximum of some fixed time; or until
    - Process voluntarily relinquishes CPU
  - Requires a clock interrupt to occur at the end of the time interval to give control of the CPU back to the scheduler
Preemptive Scheduling

- Consider the case of two processes that share data.
- While a process is updating the data it is preempted e.g.,
  - \( X = X + 1 \) requires several machine level instructions
    - Load R1 X
    - ADD R1 1
    - Load X R1
  - What if the process is pre-empted after the second instruction?
- The second process now tries to read the data.
Preemptive Scheduling

- What if the OS pre-empts an OS process that is updating the state of process
  - E.g., updating the state from running to wait
- Most OS do not allow some of their OS processes to be pre-empted
- Other processes have to expect that they may be pre-empted - more later;
Scheduling Evaluation Metrics

- Many quantitative criteria for evaluating a scheduling algorithm:
  - CPU utilization: Percentage of time the CPU is not idle
  - Throughput: Completed processes per time unit
  - Turnaround time: Submission to completion
  - Waiting time: Time spent on the ready queue
  - Response time: Response latency
  - Predictability: Variance in any of these measures
  - Fairness: No Process suffers starvation
Scheduler Options

- First Come, First Served (FCFS)
- Last In First Out (LIFO)
- Shortest Job First
- Round Robin (RR)
- May use priorities to determine who runs next
  - Dynamic vs. Static algorithms
- Multilevel Queuing
- Multilevel Queuing with Feedback
First-Come, First-Served (FCFS) Scheduling

- The process that requests the CPU first is allocated the CPU first.
- The code for FCFS scheduling is simple to write and understand.
- We will illustrate the use of FCFS with three processes that are currently in a CPU burst phase.
- Two of the three processes are considered I/O bound since their CPU bursts are small.
First-Come, First-Served (FCFS) Scheduling

- Process | Burst Time
- P1      | 24
- P2      | 3
- P3      | 3

- Suppose that the processes arrive in the order: P1, P2, P3. The Gantt Chart for the schedule is:

```
<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

- Waiting time for P1 = 0; P2 = 24; P3 = 27
- Average waiting time: \( (0 + 24 + 27)/3 = 17 \)
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>P_2</th>
<th>P_3</th>
<th>P_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>6</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.
FCFS Scheduling

- Order of arrival was P1, P2, P3

- P1 gets the CPU first
- P2, P3 are in the ready queue
- The I/O queues are idle
- P1 finishes its current CPU burst and goes for I/O
- P2, P3 quickly finish their CPU bursts
- At this point P1, P2, P3 may be waiting for I/O leaving the CPU idle
FCFS Scheduling

- **Order of arrival was P2, P3, P1**

- P2 gets the CPU first
- P3, P1 are in the ready queue
- P2 finishes quickly as does P3
- P2 and P3 go for I/O while P1 is executing
  - Remember that I/O is slower than CPU
Scheduling Algorithms LIFO

- Last-In First-out (LIFO)
  - New processes are placed at head of ready queue
  - Improves response time for newly created processes

- Problem:
  - May lead to starvation - early processes may never get CPU