Lecture - V
CPU Scheduling - I

Tevfik Koşar

Louisiana State University
January 29th, 2008

Roadmap

- CPU Scheduling
  - Basic Concepts
  - Scheduling Criteria
  - Different Scheduling Algorithms
Basic Concepts

• Multiprogramming is needed for efficient CPU utilization
• CPU Scheduling: deciding which processes to execute when
• Process execution begins with a CPU burst, followed by an I/O burst
• CPU–I/O Burst Cycle - Process execution consists of a cycle of CPU execution and I/O wait
CPU Scheduler

- Selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them ➔ short-term scheduler
- 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
- Scheduling under 1 and 4 is nonpreemptive/cooperative
  - Once a process gets the CPU, keeps it until termination/switching to waiting state/release of the CPU
- All other scheduling is preemptive
  - Most OS use this
  - Cost associated with access to shared data
**Process State**

- As a process executes, it changes *state*
  - **new**: The process is being created
  - **ready**: The process is waiting to be assigned to a process
  - **running**: Instructions are being executed
  - **waiting**: The process is waiting for some event to occur
  - **terminated**: The process has finished execution

**Dispatcher**

- Dispatcher module gives control of the CPU to the process selected by the short-term scheduler;

  Its function involves:
  - switching context
  - switching to user mode
  - jumping to the proper location in the user program to restart that program

- **Dispatch latency** - time it takes for the dispatcher to stop one process and start another running
Scheduling Criteria

- **CPU utilization** - keep the CPU as busy as possible --> maximize
- **Throughput** - # of processes that complete their execution per time unit --> maximize
- **Turnaround time** - amount of time to execute a particular process --> minimize
- **Waiting time** - amount of time a process has been waiting in the ready queue --> minimize
- **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) --> minimize

Optimization Criteria

- Max CPU utilization
- Max throughput
- Min turnaround time
- Min waiting time
- Min response time
First-Come, First-Served (FCFS) Scheduling

<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:

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

**FCFS Scheduling (Cont.)**

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
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.
- Two schemes:
  - nonpreemptive - once CPU given to the process it cannot be preempted until completes its CPU burst
  - preemptive - if a new process arrives with CPU burst length less than remaining time of current executing process, preempt.
  This scheme is known as the Shortest-Remaining-Time-First (SRTF)
- SJF is optimal - gives minimum average waiting time for a given set of processes

Example of Non-Preemptive SJF

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0.0</td>
<td>7</td>
</tr>
<tr>
<td>P₂</td>
<td>2.0</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>P₄</td>
<td>5.0</td>
<td>4</td>
</tr>
</tbody>
</table>

- SJF (non-preemptive)

<table>
<thead>
<tr>
<th></th>
<th>P₁</th>
<th>P₃</th>
<th>P₂</th>
<th>P₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Average waiting time = (0 + 6 + 3 + 7)/4 = 4
Example of Preemptive SJF

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td>0.0</td>
<td>7</td>
</tr>
<tr>
<td>P₂</td>
<td>2.0</td>
<td>4</td>
</tr>
<tr>
<td>P₃</td>
<td>4.0</td>
<td>1</td>
</tr>
<tr>
<td>P₄</td>
<td>5.0</td>
<td>4</td>
</tr>
</tbody>
</table>

- SJF (preemptive)

<table>
<thead>
<tr>
<th>P₁</th>
<th>P₂</th>
<th>P₃</th>
<th>P₂</th>
<th>P₄</th>
<th>P₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Average waiting time = (9 + 1 + 0 + 2)/4 = 3

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
- SJF is a priority scheduling where priority is the predicted next CPU burst time
- Problem = Starvation - low priority processes may never execute
- Solution = Aging - as time progresses increase the priority of the process
Round Robin (RR)

- Each process gets a small unit of CPU time \((time \ quantum)\), usually 10-100 milliseconds. After this time has elapsed, the process is preempted and added to the end of the ready queue.
- If there are \(n\) processes in the ready queue and the time quantum is \(q\), then each process gets \(1/n\) of the CPU time in chunks of at most \(q\) time units at once. No process waits more than \((n-1)q\) time units.
- Performance
  - \(q\) large \(\Rightarrow\) FIFO

Example of RR with Time Quantum = 20

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1)</td>
<td>53</td>
</tr>
<tr>
<td>(P_2)</td>
<td>17</td>
</tr>
<tr>
<td>(P_3)</td>
<td>68</td>
</tr>
<tr>
<td>(P_4)</td>
<td>24</td>
</tr>
</tbody>
</table>

- The Gantt chart is:

<table>
<thead>
<tr>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>37</td>
<td>57</td>
<td>77</td>
<td>97</td>
<td>117</td>
<td>121</td>
<td>134</td>
<td>154</td>
</tr>
</tbody>
</table>

- Typically, higher average turnaround than SJF, but better response
Time Quantum and Context Switch Time

Turnaround Time Varies With The Time Quantum
Exercise

<table>
<thead>
<tr>
<th>Process ID</th>
<th>Arrival Time</th>
<th>Priority</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>B</td>
<td>5</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>C</td>
<td>10</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>D</td>
<td>15</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

- Draw gantt charts, find average turnaround and waiting times for above processes, considering:
  1) First Come First Server Scheduling
  2) Shortest Job First Scheduling (non-preemptive)
  3) Shortest Job First Scheduling (preemptive)
  4) Round-Robin Scheduling
  5) Priority Scheduling (non-preemptive)
  6) Priority Scheduling (preemptive)

Summary

- CPU Scheduling
  - Basic Concepts
  - Scheduling Criteria
  - Different Scheduling Algorithms

- Next Lecture: Project Overview
- Reading Assignment: Chapter 5 from Silberschatz.
Acknowledgements

- “Modern Operating Systems” book and supplementary material by A. Tanenbaum
- R. Doursat and M. Yuksel from UNR