CPU Scheduling

Disclaimer: some slides are adopted from Dr. Kulkarni’s and book authors’ slides with permission
Recap: CPU Scheduling

• CPU scheduling is a policy to decide
  – Which thread to run next?
  – When to schedule the next thread?
  – For how long?
Recap: CPU Scheduler

• When the scheduler runs?
  – The running thread finishes
  – The running thread voluntarily gives up the CPU
    • yield, block on I/O, ...
  – The OS **preempts** the current running thread
    • timer interrupt (quantum expire), higher priority thread wakes up
Recap: Performance Metrics for CPU Scheduling

• Waiting time
  – Time spent on waiting in the ready queue

• Turnaround time (Response time)
  – Time to complete a task (ready -> complete)

• Response time (Scheduling latency)
  – Time to schedule a task (ready -> first scheduled)
Quiz

- Q1. draw a FIFO schedule
- Q2. compute the average waiting time of all processes
Administrative

• Project 1 deadline
  – Mar 7. 2015

• Midterm
  – Mar. 11, 2015
  – Closed book, in-class
  – Review class: Mar. 9
Shortest Job First (SJF) Scheduling

- Order each process based on the length of its next CPU burst
- Allocate CPU to the process from the front of the list
  - shortest next CPU burst
- SJF is optimal
  - achieves **minimum average waiting time** for a given set of processes

- Problem?
Example of SJF

- SJF scheduling chart

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>P2</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>P3</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>P4</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

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

- How to know how long each thread will take in advance?
Estimate Length of Next CPU Burst

- Can only estimate the length
  - next CPU burst similar to previous CPU bursts?
  - how relevant is the history of previous CPU bursts?
- Calculated as an exponential moving average of the previous CPU bursts for the process
- Formula:

  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: $\tau_{n+1} = \alpha t_n + (1 - \alpha) \tau_n.$
Estimate Length of the Next CPU Burst

CPU burst ($t_i$)  |  6  |  4  |  6  |  4  | 13  | 13  | 13  | ... 
"guess" ($\tau_i$) | 10 |  8  |  6  |  6  |  5  |  9  | 11  | 12  | ...
Shorted Remaining Time First (SRTF)

- Preemptive version of SJF
- New *shorter* process can preempt *longer* current running process

<table>
<thead>
<tr>
<th>Process</th>
<th>Arrival Time</th>
<th>Burst Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P3</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>P4</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

**Shortest Remaining Time First scheduling chart:**

Averagewaiting time = 6.5
Round Robin Scheduling (RR)

- Round-Robin (= FCFS with preemption)
  - Schedule the first job in the queue *for one time-slice*
    Preempt job, add it to the end of the queue
  - Schedule the next job and continue...

- One time slice is called a *time quantum*

- simple, avoids starvation
Example of RR with Time Quantum = 4

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>24</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
</tr>
</tbody>
</table>

The Gantt scheduling chart is:

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P1</th>
<th>P1</th>
<th>P1</th>
<th>P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average waiting time = (6 + 4 + 7) / 3 = 5.66

Compare RR with FCFS

(+) Better for shorter jobs (P2, P3)

(--) Context switching overhead
Example of RR with Time Quantum = 2

The Gantt scheduling chart is:

<table>
<thead>
<tr>
<th>Process</th>
<th>Burst Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>24</td>
</tr>
<tr>
<td>P2</td>
<td>3</td>
</tr>
<tr>
<td>P3</td>
<td>3</td>
</tr>
</tbody>
</table>

Average waiting time = \((6 + 6 + 7)/3\) = 6.3

Compare RR(2) with RR(4)

(+ ) shorter response time

(-- ) more context switches, longer waiting time
Time Quantum and Context Switch Time

process time = 10

<table>
<thead>
<tr>
<th>quantum</th>
<th>context switches</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
Round Robin Scheduling

- How to choose the time quantum?
  - What if it is too big?
    - FCFS like behavior
  - What if it is too small?
    - Large context switch overhead

- In practice
  - Time slice ranges from 1 – 10 milliseconds
  - Context switch time is less than 10 microseconds
  - Less than 1% overhead is desired
Example

- Job A and B
  - CPU bound, run an hour
- Job C
  - I/O bound, repeat(1ms CPU, 9ms disk I/O)
- FCFS?
  - If A or B is scheduled first, C can begins an hour later
- RR and SRTF?
Example Timeline

RR with 100ms time quantum

RR with 1ms time quantum

SRTF
Summary

• First-Come, First-Served (FCFS)
  – Run to completion in order of arrival
  – Pros: simple, high throughput
  – Cons: short jobs can stuck behind the long ones

• Round-Robin (RR)
  – FCFS with preemption. Cycle after a fixed time quantum
  – Pros: better interactivity
  – Cons: performance is dependent on the quantum size

• Shortest Job First (SJF)/ Shorted Remaining Time First (SRTF)
  – Shorted job (or shortest remaining job) first
  – Pros: optimal average waiting time
  – Cons: you need to know the future, long jobs can be starved by short jobs
Discussion

• Comparison between FCFS, SRTF(SJF), and RR
  – What to choose for smallest average waiting time?
    • SRTF (SJF) is the optimal
  – What to choose for better interactivity?
    • RR with small time quantum (or SRTF)
  – What to choose for higher throughput?
    • FCFS
  – What if all jobs have the same size?
    • SRTF = FCFS
  – What if jobs have varying size?
    • SRTF (or RR): short jobs don’t stuck behind long ones