CSE 4/521
Introduction to Operating Systems
Lecture 10 – CPU Scheduling III
(Real-time CPU Scheduling, Operating Systems Examples)
Summer 2018
Overview

Objective:
1. To examine how real-time scheduling works
2. Glimpse of CPU scheduling in real operating systems

• Real-Time CPU Scheduling
• Operating Systems Examples
Recap

• Scheduling Algorithms
  • Priority Scheduling, Round-robin scheduling, Multi-level (Feedback) queues

• Thread Scheduling
  • PCS and SCS, Pthreads

• Real-time CPU Scheduling
  • Interrupt latency, Dispatch latency = (conflicts + dispatch time)
Questions

1. What is soft real-time and hard real-time CPU Scheduling? (Easy)

2. What is round-robin scheduling? In round-robin, what effect does the time quantum have on average throughput time? What’s the rule of thumb in deciding time quantum? (Medium)

3. Using round-robin scheduling, compute average throughput for:

<table>
<thead>
<tr>
<th>process</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>6</td>
</tr>
<tr>
<td>$P_2$</td>
<td>3</td>
</tr>
<tr>
<td>$P_3$</td>
<td>1</td>
</tr>
<tr>
<td>$P_4$</td>
<td>7</td>
</tr>
</tbody>
</table>

$q = 4$
Overview

- Real-Time CPU Scheduling
- Operating Systems Examples
Overview

• Real-Time CPU Scheduling
• Operating Systems Examples
Real-Time CPU Scheduling: Priority-based Scheduling

- For real-time scheduling, scheduler must support preemptive, priority-based scheduling
  - But only guarantees soft real-time
- For hard real-time must also provide ability to meet deadlines
- Processes have new characteristics: periodic ones require CPU at constant intervals
  - Has processing time $t$, deadline $d$, period $p$
  - $0 \leq t \leq d \leq p$
  - Rate of periodic task is $1/p$
Real-Time CPU Scheduling: Rate Monotonic Scheduling

- A priority is assigned based on the inverse of its period
- Shorter periods = higher priority;
- Longer periods = lower priority

$P_1$ is assigned a higher priority than $P_2$.

\[
\begin{align*}
P_1 &= 50 \\
P_2 &= 100 \\
t_1 &= 20 \\
t_2 &= 35
\end{align*}
\]

Deadline is to complete before next period.
Real-Time CPU Scheduling: Missed Deadlines with Rate Monotonic Scheduling

P1 = 50
P2 = 80
t1 = 20
t2 = 35
Deadline is to complete before next period

There is no pre-emption in Rate Monotonic Scheduling
Real-Time CPU Scheduling: Earliest Deadline First Scheduling (EDF)

- Priorities are assigned according to deadlines:
  - The earlier the deadline, the higher the priority
  - The later the deadline, the lower the priority

P1 = 50
P2 = 80
t1 = 25
t2 = 35
Deadline is to complete before next period
Overview

• Real-Time CPU Scheduling

• Operating Systems Examples
  • Linux Scheduling
  • Windows Scheduling
  • Solaris Scheduling
Operating Systems Examples – Linux Scheduling

• Version 2.5 moved to $O(1)$ scheduling time
  • Preemptive, priority based
  • Two priority ranges: time-sharing and real-time
  • Real-time range from 0 to 99 and nice value from 100 to 140
  • Higher priority gets larger q
  • All runnable tasks tracked in per-CPU runqueue data structure
    • Two priority arrays (active, expired)
    • Tasks indexed by priority
  • Worked well, but poor response times for interactive processes
The vruntime is the virtual runtime of a process which keeps track for how much time a process has run.
Operating Systems Examples – Linux Scheduling

- Real-time tasks have static priorities
- Real-time plus normal map into **global priority scheme**
- Nice value of -20 maps to global priority 100
- Nice value of +19 maps to priority 139
Windows uses priority-based preemptive scheduling

Highest-priority thread runs next

Thread runs until (1) blocks, (2) uses time slice, (3) preempted by higher-priority thread

Real-time threads can preempt non-real-time

32-level priority scheme

Variable class is 1-15, real-time class is 16-31

Priority 0 is memory-management thread

Queue for each priority

If no run-able thread, runs idle thread
Operating Systems Examples – Windows Scheduling

- Win32 API has several priority classes to which a process can belong
  - REALTIME_PRIORITY_CLASS, HIGH_PRIORITY_CLASS, ABOVE_NORMAL_PRIORITY_CLASS, NORMAL_PRIORITY_CLASS, BELOW_NORMAL_PRIORITY_CLASS, IDLE_PRIORITY_CLASS
  - All are variable except REALTIME
- A thread within a given priority class has a relative priority
  - TIME_CRITICAL, HIGHEST, ABOVE_NORMAL, NORMAL, BELOW_NORMAL, LOWEST, IDLE
- Priority class and relative priority combine to give numeric priority
- Base priority is NORMAL within the class
- If quantum expires, priority lowered, but never below base
Operating Systems Examples – Windows Scheduling

• If *wait* occurs, *priority boosted* depending on what was waited for
• Foreground window given 3x priority boost
• Windows 7 added *user-mode scheduling* (UMS)
  • Applications create and manage threads independent of kernel
  • For large number of threads, much more efficient
  • UMS schedulers come from programming language libraries like C++ *Concurrent Runtime* (ConcRT) framework
## Operating Systems Examples – Windows Scheduling

<table>
<thead>
<tr>
<th></th>
<th>real-time</th>
<th>high</th>
<th>above normal</th>
<th>normal</th>
<th>below normal</th>
<th>idle priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>time-critical</td>
<td>31</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>highest</td>
<td>26</td>
<td>15</td>
<td>12</td>
<td>10</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>above normal</td>
<td>25</td>
<td>14</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>normal</td>
<td>24</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>below normal</td>
<td>23</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>lowest</td>
<td>22</td>
<td>11</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>idle</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Operating Systems Examples – Solaris Scheduling

- Priority-based scheduling
- Six classes available
  - Time sharing (default) (TS)
  - Interactive (IA)
  - Real time (RT)
  - System (SYS)
  - Fair Share (FSS)
  - Fixed priority (FP)
- Given thread can be in one class at a time
- Each class has its own scheduling algorithm
- Time sharing is multi-level feedback queue
## Operating Systems Examples – Solaris Scheduling

<table>
<thead>
<tr>
<th>priority</th>
<th>time quantum</th>
<th>time quantum expired</th>
<th>return from sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>200</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>200</td>
<td>0</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>160</td>
<td>0</td>
<td>51</td>
</tr>
<tr>
<td>15</td>
<td>160</td>
<td>5</td>
<td>51</td>
</tr>
<tr>
<td>20</td>
<td>120</td>
<td>10</td>
<td>52</td>
</tr>
<tr>
<td>25</td>
<td>120</td>
<td>15</td>
<td>52</td>
</tr>
<tr>
<td>30</td>
<td>80</td>
<td>20</td>
<td>53</td>
</tr>
<tr>
<td>35</td>
<td>80</td>
<td>25</td>
<td>54</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>30</td>
<td>55</td>
</tr>
<tr>
<td>45</td>
<td>40</td>
<td>35</td>
<td>56</td>
</tr>
<tr>
<td>50</td>
<td>40</td>
<td>40</td>
<td>58</td>
</tr>
<tr>
<td>55</td>
<td>40</td>
<td>45</td>
<td>58</td>
</tr>
<tr>
<td>59</td>
<td>20</td>
<td>49</td>
<td>59</td>
</tr>
</tbody>
</table>
Operating Systems Examples – Solaris Scheduling

- Interrupt threads
- Realtime (RT) threads
- System (SYS) threads
- Fair share (FSS) threads
- Fixed priority (FX) threads
- Timeshare (TS) threads
- Interactive (IA) threads
• Scheduler converts class-specific priorities into a per-thread global priority
  • Thread with highest priority runs next
  • Runs until (1) blocks, (2) uses time slice, (3) preempted by higher-priority thread
  • Multiple threads at same priority selected via RR
Credits for slides

Silberschatz, Galvin and Gagne