CS 3305B
Operating System Examples - Scheduling

Lecture 9

Feb 7th 2017
Solaris
Solaris

- Solaris is a Unix Operating systems originally developed by Sun
- Solaris was developed as proprietary software
- 2005: Sun released most of its code base and thus Solaris became open source
- Sun was acquired by Oracle in 2010 and so Solaris is sometimes called Oracle Solaris
Solaris

- Solaris uses priority-based scheduling
- Each process belongs to one of six classes
  - Interrupt
  - Real time
  - System
  - Time sharing
  - Interactive
  - Fair share
  - Fixed priority
- Priority levels from 0 to 169
Priorities

- **global priority**
  - highest
    - 169: interrupt threads
    - 160
    - 159
  - 100
    - 99: realtime (RT) threads
  - 60
    - 59: system (SYS) threads
      - fair share (FSS) threads
      - fixed priority (FX) threads
      - timeshare (TS) threads
      - interactive (IA) threads
  - lowest
    - 0

- **scheduling order**
  - first
  - last
Priorities

- **Interrupt**
  - Highest priority

- **Real-time class**
  - High priority
  - Guaranteed response time
  - Kernel processes (very few) assigned to this class

- **System class**
  - Kernel processes are assigned to this class
  - Priorities are static
Priorities

- **Time-sharing**
  - Processes can have their priorities adjusted

- **Fixed-priority**
  - No dynamic adjustment of priorities

- **Fair-share**
  - Uses CPU *shares* instead of priorities

- **Interactive processes (e.g., window managers)**
  - Typically have higher priority (good response time)
Windows 7
Windows

- Windows refers to a collection of graphical operating systems developed by Microsoft
- There are recent versions of Windows for PCs, server computers, smartphones and embedded devices
- There is a specialized version of Windows that runs the Xbox One game console
- Windows was originally designed for desktop machines
Scheduling

- Priority Scheduling
- The scheduler is called a dispatcher
- 32 priorities
- Priorities are divided into two classes:
  - User class: priorities 1 to 15
  - Real-time class: priorities 16 to 31
- Priority 0 is used for memory management processes
- There is a queue for each priority
- Round Robin (time quantum) for each queue
Adjusting Priority

- If process was in user class
  - Time quantum expires:
    - If process is in the user class the priority is lowered
  - Process switches from blocked to running:
    - Priority is increased
    - The amount depends on what the process was doing
    - Keyboard I/O gets a large increase while disk I/O gets a moderate increase

- Some processes always have a low priority e.g., disk fragmenter
Adjusting Priority

- The priority of a process cannot be lowered passed the base priority (lower threshold value) of the process.
- Windows 7 distinguishes between the foreground process that is currently selected on the screen and the background processes that are not currently selected.
  - Tends to give good response times to interactive processes that are using the mouse and windows.
Linux
Linux

- As of Dec 2015
  - Webservers: W3Cook reports that 96.5% of web servers run Linux (1.5% run Windows)
  - Desktops/laptops: 1.5% use Linux
  - Mobile devices: Android (based on Linux kernel) is used in 80% of all mobile devices
  - Platform of choice for film industry
- Linux is free and open-source
- Please note that Linux uses the term “task”
Linux Scheduler

- **Current:** Linux version 2.6
- **Priority Scheduling**
- **Linux has these scheduling classes:**
  - Real-time (RT) classes
  - Completely fair scheduler (CFS) class
- **Tasks in RT have higher precedence than tasks in the CFS**

- We will first start with a discussion of nice values which is related to CFS.
CFS - move

- The Completely Fair Scheduler (CFS) is a significant departure from the traditional UNIX process scheduler.
- Tries to guarantee fairness (CPU Usage)
Nice Values

- A **nice value** is assigned to each task
  - Nice values range from -20 to +19
  - Lower nice value indicates a higher relative priority
  - Tasks with lower nice values receive a higher proportion of CPU processing time than tasks with higher nice values
  - The default nice value is 0
CFS- Time Slice Calculation

- The general idea is that
  - Every process that changes nice value up by one level gets 10% less CPU power
  - Every process that changes nice value down by one gets 10% more CPU power
CFS - Picking the next process

- Pick process with the weighted minimum runtime so far
- The virtual run time (vruntime) of a task is the actual runtime weighted by its niceness
- The value of vruntime is used by the scheduler to determine the next process to run
  - Process with the smallest vruntime is selected to run next
CFS - Virtual Runtime

- High nice values should result in less CPU time allocated to a process

- This implies that vruntime cannot be the same as the real runtime
CFS – Virtual Runtime

- Example: Assume a process runs for 200 milliseconds
  - Nice value is 0: \texttt{vruntime} will be 200 milliseconds
  - Nice value < 0: \texttt{vruntime} will be less than 200 milliseconds
  - Nice value > 0: \texttt{vruntime} will be greater than 200 milliseconds
Linux Scheduler

- Linux has **these scheduling classes:**
  - Real-time (RT) class
  - Completely Fair Scheduler (CFS) class
- We discussed CFS in detail
- Tasks in the RT class have higher precedence than tasks in the CFS
Linux Scheduler

- 140 priority levels:
  - RT: There are 100 priority lists ranging from 0 to 99.
  - User tasks: 101-140 (nice values -20 to 19)
- Within RT there are are two scheduling policies
  - `SCHED_RR`: round-robin
  - `SCHED_FIFO`: FIFO
- Round-robin has precedence over FIFO
## System Calls Related to Scheduling

<table>
<thead>
<tr>
<th>System Calls</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>nice</td>
<td>Change the priority of a CFS task</td>
</tr>
<tr>
<td>getpriority</td>
<td>Get the priority of a process</td>
</tr>
<tr>
<td>setpriority</td>
<td>Set priority of a process</td>
</tr>
<tr>
<td>sched_getscheduler</td>
<td>Get the scheduling policy of a process</td>
</tr>
<tr>
<td>sched_setscheduler</td>
<td>Set the scheduling policy and priority of a process</td>
</tr>
<tr>
<td>sched_rr_get_interval</td>
<td>Get the time quantum value for the RR policy</td>
</tr>
</tbody>
</table>
Other
MAC OS X

- Based on MACH and Unix BSD
- Priorities are categorized into priority bands
  - Normal: Applications
  - System high priority: Processes with higher priority than Normal
  - Kernel mode: Reserved for kernel processes
  - Real-time: For processes that must be guaranteed a slice of CPU time by a particular deadline
Priorities change dynamically
- Based on wait time and amount of time that the process has had the processor
- Stay within the same priority band

Reschedules every tenth of a second and recomputes priorities once every second

Process will relinquish CPU after time quantum or when it must wait for an I/O completion

Dynamic priority prevents starvation
Android

- For mobile devices
- Today it is the most commonly used platform
- Uses Linux for device managers, memory management, process management
We have examined scheduling in several contemporary operating systems.