CMPSCI 377 Operating Systems **Fall 2005**

Lecture 6: October 4

Lecturer: Emery Berger *Scribe: Dennis Gove, Ivan Ordonez*

Today:

- Scheduling

6.1 Scheduling

6.1.1 Last Time

- There is no perfect scheduling algorithm
- We would like to minimize response time, maximize throughput, and optimize fairness
- FIFO (FCFS) is a basic first-in first-out scheduler that works just like a queue, first job in runs till completion, then next job, etc...

6.1.2 Round Robin (RR)

- RR works by picking the first item on the ready queue, running for a quantum, moving item to back of queue, then repeating
  - A varient of this is used in most systems
- This algorithm is totally dependent on the quantum length
  - Large quantum leads to a larger response time
    - As quantum goes toward infinity, RR looks more like FCFS
  - Small quantum leads to a decresed throughput
    - As quantum goes toward 0, context switch overhead dominates

6.1.3 Shortest Job First (SJF)

- Great algorithm if the amount of time a job requires is known
- The job with the shortest completion time is run first
- Wait times are as small as possible minimizing the starvation of other jobs
- Advantages
  - Minimizes completion time
  - Optimal with respect to wait time (minimal average)

  - Works for both pre-emptive and non-pre-emptive schedulers
    * Pre-emptive SJF = SRTF (**S**hortest **R**emaining **T**ime **L**eft)
  - I/O bound jobs get priority over CPU bound jobs
    * IO bound jobs theoretically have a lower completion time if you think about before and after I/O as seperate jobs

- Disadvantages
  - Impossible to predict CPU usage time a job has left (in general)
  - Long running CPU bound jobs can starve

6.1.4 Multi Level Feedback Queues

- Uses past job behavior as predictor for future behavior
  - If the job stopped on IO once then it will most likely stop on IO again
  - If the job ran until quantum completion then it will most likely run until quantum completion again
- The scheduler will favor jobs which use less CPU time
  - Jobs using IO will get higher priority than jobs using CPU
  - Scheduler is adaptive because a change in job behavior leads to a change in scheduling decisions
- *Structure*
  1. Multiple priority queues where IO has highest priority
     - IO jobs (priority 1)
     - mixture (heavily IO) (priority 2)
     - mixture (heavily CPU) (priority 3)
     - CPU jobs(priority 4)
  2. Run highest priority queue first (priority 1)
     - While in queue, use round robin
  3. Change the quantum length for each priority level (priority 1 gets quantum 1 unit, priority 4 gets quantum 4 units)
  4. Move jobs from queue to queue based on their behavior
     - If quantum time expires then CPU bound so move job to lower priority queue
     - If quantum time does not expire then IO bound so move job to higher priority queue

6.1.5 Lottery Scheduling

- Non-hack, ellegant way of solving some of the issues associated with schedulers
- *Structure*
  1. Every job gets a lottery ticket (high priority jobs could get multiple tickets, but each job gets at least one)
     - The algorithm degrades gracefully as load gets higher
  2. At each quantum, pick a random winner and run its job for one quantum - repeat
- Example on slides 38 - 44 (http://www.cs.umass.edu/ emery/cmpsci377/cmpsci377-06.ppt)