Team Name: Forkbomb

Members: Patrick Roteman, Joshua Filstrup, Tim Stullich

CS 149

Assignment 2

2/24/2014

We implemented our scheduler in c++11. The c++11 flag must be set for it to compile. To compile it under gcc 4.6 or 4.7, use -std=c++0x. For gcc 4.8 or 4.9 use -std=c++11. We completed the extra credit, so Highest Priority First Preemptive and Highest Priority First Nonpreemptive both take advantage of process aging. With how many processes we were generating, starvation was not a significant issue so the process aging did not alter the results significantly from when we had it turned on.

Our analysis of each algorithm's performance follows:

- Shortest Job First: SJF's main advantage, beyond being simple to implement, was a relatively short wait time and response time, that were balanced pretty well. It didn't learn very far towards either of the two, while also maintaining a solid turnaround time.
- Shortest Remaining Time: SRT showed significant improvements over SJF in everything except throughput, which took a significant hit. In cases where throughput is not a large concern, SRT seems to be directly superior to SJF, though that may be pretty niche circumstances.
- First Come First Serve: FCFS seems to be a mediocre scheduling algorithm all around. It's wait time was second worst, with the worst response time of all algorithms. It also had the second worst turnaround time. It did have the best throughput, but that is at least partially the result of how we truncated the simulation as soon as we finished all started processes after quantum 99. The main thing that our simulation doesn't show is the number of context switches. This algorithm has relatively few, which is a point in its favor not shown by our simulation.
- Round Robin: RR had a more obvious skew towards one metric than any other algorithm. It led to fantastic response times that completely blew away every other algorithm, but the wait time and turnaround times were the worst of all the algorithms. It would be best suited in a situation where processes finishing isn't particularly important compared to latency.
- Highest Priority First Nonpreemptive: HPFN had pretty average metrics in every category, while also allowing for process prioritization. Because of the lack of

severe hits in any of the other metrics, this algorithm seems to be a solid all around algorithm when prioritization is important.

- Highest Priority First preemptive: Similarly to Round Robin, this algorithm was skewed more heavily towards response times than most of the rest, but the fact that the round robin occurred within smaller queues meant that the average response time was not as stellar as regular Round Robin, while the performance hits that regular Round Robin takes were also not as severe. It would be an effective algorithm when response time is the main focus on a subset of processes.