Yilin Shi<shiy9> Zhihao Wang<wangz35>

Project Analysis

1. Of the four simulated algorithms, which algorithm is the "best" algorithm for CPU-bound processes? Which algorithm is best-suited for I/O-bound processes?

Of four simulated algorithms, SRT is the best algorithm for it has lowest wait time for CPU-bound processes.

For I/O-bound processes, fcfs is the best algorithm.

2. For the SJF and SRT algorithms, what value of α produced the "best" results?

from our trials, alpha close to either 1 or 0 produces best reults in terms of high cpu usage and low waiting time.

3.For the SJF and SRT algorithms, how does changing from a non-preemptive algorithm to a preemptive algorithm impact your results?

For SJF specifically, preemption will lead to lower waiting time and consequently better performance than SRT. On the contrastm, it does have a draw back of more context switches than that in SJF. Overall, the performance of SRT is better than SJF as the scrafice on context switches is relavtively low.

4.Describe at least three limitations of your simulation, in particular how the project specifications could be expanded to better model a real-world operating system.

1. Currently, the simulation we got has lower efficiency as it only can deal with one process while more process will be operated in real situation.

2. Our simulator can save limit amount of process, while cpu in real time will be able to record nearly unlimited process. We may also have to improve the output algorithm.

3. We may need to have a flexible alpha parameter generator to have different parameter thta suitable for different CPU bursts.

5.Describe a priority scheduling algorithm of your own design (i.e., how could you calculate priority?). What are its advantages and disadvantages?

One possibility could be SRT with penalty level. From our observation, some times exponential averaging does not produce the best estimates of CPU bursts. For those "unreliable" process, we could therefore mark it as unpredictable by lowering its priority and use the average of CPU bursts time as an estimate for priority sorting. This way we keep the preemption mechanism allowing lower waiting time but also ensures that every process will not face unexpected long wait.