Assignment 3

CS 301 - Operating Systems

Due date: 28th October, 2021

1 Introduction

This assignment is a little different than the others in this course: instead of a systems-level exercise, it's a Jupyter notebook! Yes, we will be coding in Python instead of C in this assignment. The aim of this assignment is to understand the various scheduling algorithms. Check out an online tutorial if you want to familiarize yourself with Jupyter notebooks.

Try to run this notebook on your personal machine, instead of the VM, to not run into unexpected surprises. You'll need to make sure that you have installed Python3 as well as the Jupyter software that will run the notebook. Finally, you will need to install the Python packages NumPy and Matplotlib. Submit your finished Jupyter notebook on git along with the log outputs separately in a pdf.

2 Implementing Scheduling Algorithms

For these problems, we would like you to implement the schedulers in the IPython notebook and then run the simulation to obtain the CPU log output.

2.1 Shortest Remaining Time First

Implement the SRTF scheduler. Whenever two threads have the same time remaining until completion, prioritize by their order of arrival in FIFO fashion. Run it on workload3 in the IPython notebook with a quantum of 2; when it is complete, report the CPU log output.

2.2 Multilevel Feedback Queue

Implement the MLFQ scheduler. Use two queues, one high-priority queue for interactive tasks and one low-priority queue for CPU-bound tasks; within each queue tasks are scheduled in a round-robin fashion. Each task begins in the high-priority queue; if its quantum expires before the CPU burst time ends, it is demoted to the low-priority queue. Run it on workload3 in the IPython notebook, using a quantum of 4 in the low-priority queue and a quantum of 2 in the high-priority queue, and produce the CPU log output here.

3 Approaching 100% Utilization

Consider a sequence J_i of CPU bursts, where each CPU burst has a fixed length T. The first burst, J_0, arrives at time t = 0 (i.e., ArrivalTime(J_0) = 0). For $i \geq 1$, the arrival time of J_i is given by ArrivalTime(J_i) = ArrivalTime($J_i - 1$) $+X_i$, where the X_i are independent and identically distributed exponentially distributed random variables with parameter λ.

Set up a simulation in the IPython notebook to model the system with a large number of CPU bursts. Fix some value for T. Vary the arrival rate (i.e, λ) starting at a small number, increasing it to approach a value λ_M where the mean time between the arrivals is equal to T. Run the simulation at various points along the way, with multiple trials for each point, making sure to choose some points where λ is very close to the value λ_M. What is the value of λ_M? What value of λ should we choose, such that the system runs at 50% utilization on average?

As you vary λ, what happens to CPU utilization? Show a line plot with the arrival rate on the x axis and CPU utilization on the y axis, depicting the results.

As you vary λ, what happens to the response time for each CPU burst? Show a line plot with the arrival rate on the x axis and response time on the y axis, depicting the results.