CS3360: Homework #4

Due on Mar, 24, 2023 at 11:55PM (Firm)

Mina Guirguis

PLEASE READ: You may discuss this problem set with other students as long as you do not share/verify answers. However, you must also write the names of other students you discussed any problem with. You must *write up your answers on your own showing your work.* Each problem has a different weight. Please state any assumptions you are making in solving a given problem. Late assignments will not be accepted with prior arrangements. By submitting this assignment, you acknowledge that you have read the course syllabus and abiding to the copyright notice (e.g., no posting) and the Honor Code (e.g., not using note sharing sites such as Chegg, OneClass, CourseHero, etc. as well as using chatGPT) requirements.

Problem 1

Overview: In this project, we are going to build a discrete-time event simulator for a First-Come First-Served (FCFS) CPU scheduling algorithm. The goal of this project is to assess the impact of different workloads on the following performance metrics:

- The average turnaround time of processes
- The total throughput (number of processes done per unit time)
- The CPU utilization
- The average number of processes in the Ready queue

The simulator needs to generate a list of processes. For each process, we need to generate its *arrival time* and its requested *service time*. We can assume that processes arrive with an average rate λ that follows a Poisson process (hence exponential inter-arrival times). The service times are generated according to an exponential distribution. We will vary λ to simulate different loads while keeping the average service time fixed. The simulator should stop after 10,000 processes complete, then it should output the metrics above.

Events (e.g., process arrival, process departure) that occur causes the simulator to update its current state (e.g., cpu busy/idle, contents of the Ready queue). Events are to be kept in a priority queue called Event Queue that describes the future events and is kept sorted by the time of each event. The simulator keeps a clock variable the represents the current time which initially takes the time of the first event in the Event Queue. As the simulator runs and events are handled, additional future events may be added to the Event Queue. For example, when a process gets serviced by the CPU and completes, another process can start executing (if one is waiting in the Ready queue) and under FCFS, we know exactly when this process would finish (since FCFS is non-preemptive), so we can schedule a departure event in the future and place it in the Event queue. Notice that time hops between events, so you would need to update your simulator clock accordingly.

Your scheduler needs to maintain the Ready queue that holds processes that had arrived but waiting their turn on the CPU. The Ready queue is not to be confused with the Event Queue.

The Runs: The simulator should take 2 command-line arguments: The first is the average arrival rate and the second is the average service time. We will vary the average arrival rate, λ, of processes from 10 process per second to 30 processes per second (based on a Poisson process). The service time is chosen according to an exponential distribution with an average service time of 0.04 sec. For each run, you would need to output the four metrics above.

The Output and submission details: An output report would include a brief description of the results and show a single plot for each one of the above metrics (with λ on the x-axis). Submissions are done through Canvas. Submissions will include the code, how to compile and run the simulator on one of the CS servers, along with a report containing the results and their interpretation.

You can write your simulator in any of these languages (C, C++, Python or Java), however, it is your responsibility to ensure it runs under the CS Linux servers with a command line – nothing graphical. Please indicate clearly how to compile and run your simulator.

Grading breakdown: 30% of the grade is on developing the correct design and data structures (e.g., Event queue, Ready queue, etc.) for the simulator. 60% of the grade is on obtaining the correct results (i.e., the metrics above) for the schedulers. 10% of the grade is on proper documentation (i.e., explanation of the results, providing the compile and run command lines, etc.). This project is worth 100 points.

Important note: This project builds on the workload generation problem in previous homework and its structure will be used in future assignments. The goal of this project is not to get the metrics above with any program but rather develop a discrete-time event simulation engine with an event queue that we would expand on over the semester (e.g., comparing different schedulers and/or considering multiple CPUs).