**Program 4 Monte Carlo Simulation**

**Problem Statement**

The assigned task is to write a C/C++ program which implements 4 scheduling algorithms, First-Come-First-Serve, Round Robin, Highest Response Ratio Next, and Feedback. The program needs to perform the 4 scheduling algorithms on one thousand processes one thousand times. The process service times are randomly generated as a normal distribution with a mean of 10 and a standard deviation of 5. The importance of the project is to learn about scheduling algorithms and determine their effectiveness on the data set.

**Approach**

The approach I took for this program was to generate a set of normally distributed service times and implement each scheduling algorithm in C++ by assigning the service times to processes.

To generate a set of normally distributed service times I used the normal\_distribution<> object and default\_random\_engine class included in the random library. To enforce a lower bound of one I regenerated a number that fell below one.

The first scheduling algorithm I implemented was first-come-first-serve. In this scheduling algorithm the processes are executed in a FIFO manner, so the first process is executed and then the second etc. FCFS is an easy scheduling algorithm to implement using a queue data structure because a new process enters the queue at the tail and the scheduler selects the next process to execute from the head of the queue. FCFS is a non-preemptive decision mode scheduling algorithm.

I implemented the round robin scheduling algorithm that uses time slicing to execute processes. Each process is given a slice of time before being preempted, with the assigned time slices being one unit of time in this implementation. The round robin scheduling algorithm is relatively simple to implement because you execute a standard time slice on each process currently waiting. Round robin is a preemptive decision mode scheduling algorithm.

I implemented the highest response ratio next scheduling algorithm. This scheduling algorithm selects the next process to execute based on a response ratio that is calculated as shown in figure 1.

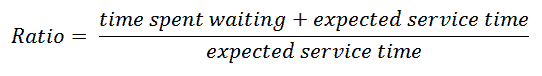


Figure 1

The algorithm selects the process with the greatest ratio. This scheduling implementation is considered an optimal algorithm because it accounts for the age of a process. While shorter jobs are favored, aging without service increases the ratio so that a longer process will eventually get past competing shorter jobs. This algorithm was difficult to implement because I needed to keep track of the response ratio at each unit of time. Highest response ratio next is a non-preemptive decision mode scheduling algorithm.

I implemented the feedback queue scheduling algorithm with three queues. The priority of execution between queues is highest in queue 1 and decreases with each subsequent queue. Queue 1 and 2 follow round robin with a time slice of 1 and queue 3 follows first-come-first-served. A process first enters queue 1 where it is executed in round robin style, if the process does not complete in queue 1 then it is demoted to queue 2. Queue 2 acts in the same manner as queue 1. When a process is demoted to queue 3 it has the lowest priority and will be executed when all other queues are empty. This algorithm was difficult to implement because I needed to handle three queues with differing priorities. Feedback is a preemptive scheduling algorithm.

**Solution**

The assigned task is to write a C/C++ program which implements 4 scheduling algorithms, First-Come-First-Served, Round Robin, Highest Response Ratio Next, and Feedback. To implement the solution, I used C++. I implemented the scheduling algorithms and passed my data sets to them.

My program prints out a counter for each test and when 1000 tests are finished the metrics are calculated and printed out.

In the FCFS algorithm there is minimal overhead because I just take the service times and executed them linearly. Computing is switched at process completion, meaning that processes can be stored in one structure.

In the round robin scheduling algorithm, the overhead is derived from the constant swapping of processes being executed. Processes are computed one time slice at a time so many processes don’t complete for multiple cycles throughout all current processes.

In highest response ratio next scheduling algorithm, overhead exists when calculating the response ratio and determining which process has the highest response ratio. Also, I created a separate array to contain the current service times for the processes so overhead exists when calculating times.

In feedback scheduling algorithm, overhead exists due to swapping processes between queues and swapping between queues. I also implemented an index to determine where the current process to execute is in a queue so overhead exists when calculating the processes index.

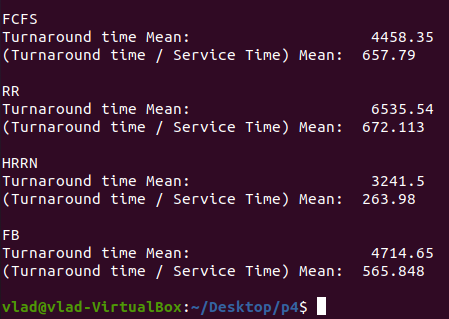


Figure 2

Figure 2 shows the collected statistics from 1000 processes in 1000 tests. The first statistic is the average turnaround time from all 1000 tests. Turnaround time is the time spent by the process from coming in ready state for the first time to its completion. The lowest turn around time is the highest response ratio next scheduling algorithm meaning that full processes complete fastest with this method. Turnaround time / service time is a metric that uses both times to compute a better measurement that also considers abnormalities in service times. Highest response ratio next also has the lowest Turnaround time / service time value.

First come first serve performed average compared to the rest of the scheduling algorithms. FCFS is limited by not prioritizing different service times. Round robin was the worst performing scheduling algorithm. This is because full process completion is not prioritized so then processes have a big turnaround time. Highest response ratio next scheduling algorithm performed the best in both metrics. This is because it prioritizes smaller processes, which decreases the average turnaround time because processes complete faster. The feedback scheduling algorithm performed as the second lowest and third lowest turnaround time and Turnaround time / service time respectively. Feedback will complete shorter processes faster due to its priority queue implementation. However, it is still running round robin within all the queues, except for the last, so turnaround time is impacted by other processes in the queue.

Based on all the information that I collected in this assignment, I would recommend the highest response ratio next scheduling algorithm to my manager because it is able to perform full processes at a faster rate. HRRN performed the best in both metrics of my project because of its prioritizing of processes.