Project 1 Report

Files included are README, pr1.cpp and this report. README tells you how to compile and run pr1.cpp to create a run of the simulator. “pr1.cpp” is the workload simulator that runs up to 10000 processes using a given lambda for arrival times, and given service times for, which then prints out the Average Turnaround Time, Throughput, CPU Utilization, Disk Utilization and average processes in both CPU ready queue and Disk queue. Each completion of “pr1.cpp” counts as a run for this project. This report will share the results of 30 runs of “pr1.cpp” with lambda ranging from 1-30 with CPU and Disk service time being 0.02 and 0.06s respectively.

ASSUMPTIONS: When writing this program, I was under the assumption that both disk and ready queue were individually FCFS. Meaning that I let new processes enter the cpu when processes were still going through the disk. This will be affect the results to some extent.

The program “pr1.cpp” uses a linked list to create an event queue that tracks events such as arrivals and departures. Upon each event the program saves the state of the current process and creates a new event depending on which event occurred. The event is then deleted from the queue and the next event arrives. Upon each arrival event the arrival time and the id are stored into the ready queue and the next arrival event is created if the server is not idle. If the server is idle, then it schedules the next departure event. For each departure event a process has a 60% chance to leave the system where it calculates the turnaround for the process and removes the process from the ready queue. The departure also creates the next departure event for the cpu unless the ready queue count is 0, or empty. The other 40% will add the process into the disk queue and schedule an arrival event to the cpu. If a process possesses an id from the disk queue it will then create a new arrival event from the disk queue if the disk queue is not empty. The process states are then averaged out at the departure of 10000 events, resulting in the following output:

Text

Description automatically generated

This is based on lambda = 12.

The program also uses a seed to make the results of the experiment replicable.

Results of the 30 runs are as follows:

What these results show is that the disk queue becomes overloaded at lambda = 25, but since I let in more processes to the cpu the other statistics did not respond in the expected way at 25. However, I did notice that while the throughput did not equal 25, it did start to fall further and further from the inputted the lambda the further it got from 25. Other stats affected include the ready and disk queue as the disk queue was supposed to be higher based on the example given on canvas, but since new arrivals never stopped coming the ready queue was higher until the disk queue overloaded. Since the system overloaded, processes that entered the disk queue took longer which resulted in a higher average turnaround time.