Elizabeth Roberts

Schedular Dispatch Simulation Report

**System Design**

Diagram

Description automatically generated with low confidence

**Queue Configuration**

Timeline

Description automatically generated

**Assumptions**

An assumption made in designing this system is implementing the timing as discrete events, instead of using a continuous interval clock module. These discrete events include events such as CPU bursts, I/O bursts, and process shuffling. Process queues are modeled using deque containers, so the elements can be pushed and popped from both sides and for iterator functionality. In the model, the process at the front of the queue is the process that is currently executing.

Upon arrival time, processes are moved to the CPU queue which holds processes that are ready for CPU execution. Processes in execution will run for their designated burst time. Some processes require an I/O operation. To model an I/O operation, when the I/O burst time is higher than the remaining CPU burst time, the process is moved to the I/O queue where it waits for I/O operation. Once I/O is completed, the process is moved back to the CPU queue where it is again ready for CPU execution.

**Modeled Algorithms**

RR1 – Round Robin with Time Quantum q = 1, Single Processor

RR4 – Round Robin with Time Quantum q = 4, Single Processor

RR8 – Round Robin with Time Quantum q = 8, Single Processor

FCFS – First Come First Serve, Single Processor

FCFS.MULT – First Come First Serve, Dual Processor, processes split evenly between processors

FCFS.MULT.DIST – First Come First Serve, Dual Processor, processes split by if I/O operation is required or not

SPN – Shortest Process Next, Single Processor

SPN.MULT – Shortest Process Next, Dual Processor, processes split evenly between processors

**Graph – Average Wait Times**

**Graph – Average Response Times**

**Graph – Average Turnaround Times**

**Graph – Throughput Times**

**Discussion of Findings and System Assessment**

This schedular dispatch simulation tests different scheduling algorithms and CPU configurations to collect data on average wait time, average response time, average turnaround time, and throughput time for each. For this model, five test runs were performed where thirty processes were run through eight different configurations. For Round Robin, tests were run with time quantums of one, four, and eight. For First Come First Serve, tests were run with a single CPU, a dual CPU with the processes evenly split between processors, and a dual CPU with processes requiring I/O going to one processor, and processes not requiring I/O going to the other processor.

A counter variable was implemented that represents the clock. Deques were used to represent the CPU queue and I/O queue, because of their iterator functionality. All processes start in a ‘starting’ deque and upon their arrival time, processes are moved to the CPU queue which holds processes that are ready for CPU execution. Processes in execution run for their designated burst time. Some processes in the model require I/O operation and to simulate this, the process is moved to the I/O queue when the I/O burst time is higher than the remaining CPU burst time. Once an I/O operation is completed, the process is moved back to the CPU queue where it is again ready for CPU execution. The scheduling algorithm being used by the CPU determines the next process that will be in execution.

Regarding average wait time, the RR algorithm had the highest overall wait time. Increasing the time quantum for this algorithm demonstrated a lower average wait time; RR with a time quantum of eight had a lower wait time than RR with a time quantum of one. Of the single processor configuration tests, SPN had the lowest overall wait time and FCFS was higher than SPN but lower than RR. Of the dual CPU configurations, SPN still had a lower wait time than FCFS. SPN was overall the most efficient regarding wait times.

Regarding average response time, FCFS had the highest overall response time. RR with a time quantum of eight and SPN were very similar in their response time and were the next highest for response time. FCFS with dual processor configuration had the next highest response time, with no notable difference between process distribution configurations. SPN with dual process configuration and RR with a time quantum of one had the most efficient response times. One notable finding was the lower the time quantum of an RR algorithm, the quicker response time it would have.

Turnaround times had a very obvious split between single CPU and dual CPU configurations, with single taking approximately twice as long as dual configurations. Throughput times also demonstrated a similar pattern, demonstrating a significant speedup with the addition of a second processor.

One thing that was tested was the distribution of processes based on requirements for I/O operation. One processor handled processes with no I/O operations and the other handling processes that required I/O operations. This model demonstrated worse performance for algorithms that split processes based on the presence of a required operation vs. algorithms that split the processes evenly between processors.

Overall, the most notable finding was the addition of multiple processors in scheduling algorithms greatly increased the efficiency of running processes and across the board, reduced wait times, response times, and turnaround times. Of the algorithms themselves, SPN was the most efficient the shortest wait time and turnaround time. Its response time was also low but was slightly worse than RR with a time quantum of eight.

After analyzing this system, one weakness that can be found is potentially inaccurate modeling of how processors actually handle processes. This system was modeled as close to a process schedular as possible and demonstrates results that make sense but may not be giving the most accurate results it could be giving. The way the processes are handled as having one CPU burst and potentially one I/O burst isn’t an accurate model, as a normal process would most likely have several of these bursts.

This project was completed individually, so I was the sole developer of this simulation.