Computer Systems Engineering-1

Assignment 5

Dhruv Sharma - 2018101095

Scheduling Policies in xv6 Performance Report

OVERVIEW

We test a benchmark program which runs for more than 20 seconds to compare the performance of different scheduling policies - Round Robin Scheduling Policy, Priority Based Scheduling Policy, First Come First Serve Policy, and Multi Level Feedback Queue Scheduling Policy.

The Benchmark Program

The program creates **identical** child processes and then wastes cpu time by useless computations inside a large ‘for’ loop in the child process.

Performance Analysis

- Round Robin Scheduling Policy

  On running the time command to run benchmark, so that it returns the running and waiting time of the whole process, the waiting time, which depends on machine to machine, was found to be average, around 2291 ticks on my machine.

- First Come First Serve Scheduling Policy

  On running the time command to run benchmark, so that it returns the running and waiting time of the whole process, the waiting time, which depends on machine to machine, was found to be the least, around 2094 on my machine, which should be expected, because identical processes are being created and FCFS does not allow pre-emption. So the extra overhead due to context switches is not there, and therefore, the waiting time is the least.

- Priority Based Scheduling Policy

  On running the time command to run benchmark, so that it returns the running and waiting time of the whole process, the waiting time, which depends on machine to machine, was found to be very similar to the Round Robin Policy, around 2481 on my machine, which is very much expected because we are setting the default priority to be 60, and priorities are not being changed in between. Round Robin Policy is applied for processes having equal priorities, which explains the output.

- Multi Level Feedback Queue Scheduling

  On running the time command to run benchmark, so that it returns the running and waiting time of the whole process, the waiting time, which depends on machine to machine, was found to be less than Round Robin Policy, around 1945 on my machine, which is around expected because it uses pre-emption and aging to prevent starvation.

Possible Exploitation of MLFQ Policy by a Process

If a process voluntarily relinquishes control of the CPU, it leaves the queuing network, and when the process becomes ready again after the I/O, it is inserted at the tail of the same queue, from which it is relinquished earlier. This can be exploited by a process, as just when the time-slice is about to expire, the process can voluntarily relinquish control of the CPU, and get inserted in the same queue again. If it ran as normal, then due to time-slice getting expired, it would have been preempted to a lower priority queue. The process, after exploitation, will remain in the higher priority queue, so that it can run again sooner that it should have.