OS-Assignment-3: CS3510

(Implementing Rate-Monotonic Scheduling & Earliest Deadline First Scheduling through Discrete Event Simulation)

Name - C.Asish

Roll number – CS20BTECH11010

Low-Level design:-

- This program implements two programs that can perform rate monotonic and earliest deadline first algorithms
- The program takes a input file and outputs stats and log file
- The input file contains no of process to be executed on the first line(i.e. n) and the following n lines with metadata of these programs (P_id, burst time, period, no of times this program to be executed)
- We collect all this information in a struct array named process and this struct additionally has variables like in_time, status of the program, time_left for easier programming
- Till all the processes are either terminated or completed, a while loop runs by increasing time by 1.(time variable)
- In the while loop, the process having the highest priority at that particular time will either execute, preempt, resume, terminated or get pre-empted by other process. And their respective statements are printed in log file
- If the highest priority process has to be terminated. Then it prints statement in log file and it is given a status ‘t’. While time variable does not add up.
- If the highest priority process has to be continued executing, then wait times of all remaining processes increases while time variable increases by 1
- If the highest priority process changes and the existing process is to be preempted a sentence regarding it will be printed in log file while new process executes and time variable increases by 1.(avg wait times of all other active process increases by 1)

- We use prev_index to get previous process, using previous process we get whether new process is preempted or resumed or previous process is finished and if it is finished we print a finish statement in log file
- After all process are either terminated or completed, the stats like no of process, no of processes terminated, no of processes successfully run, and average waiting time of each process are printed in stats file.

Complication that arose:-

- As preemption can happen at any time, we need to check all the process to get highest priority process at random times. So, we check at least possible time, in this case it is 1 sec.

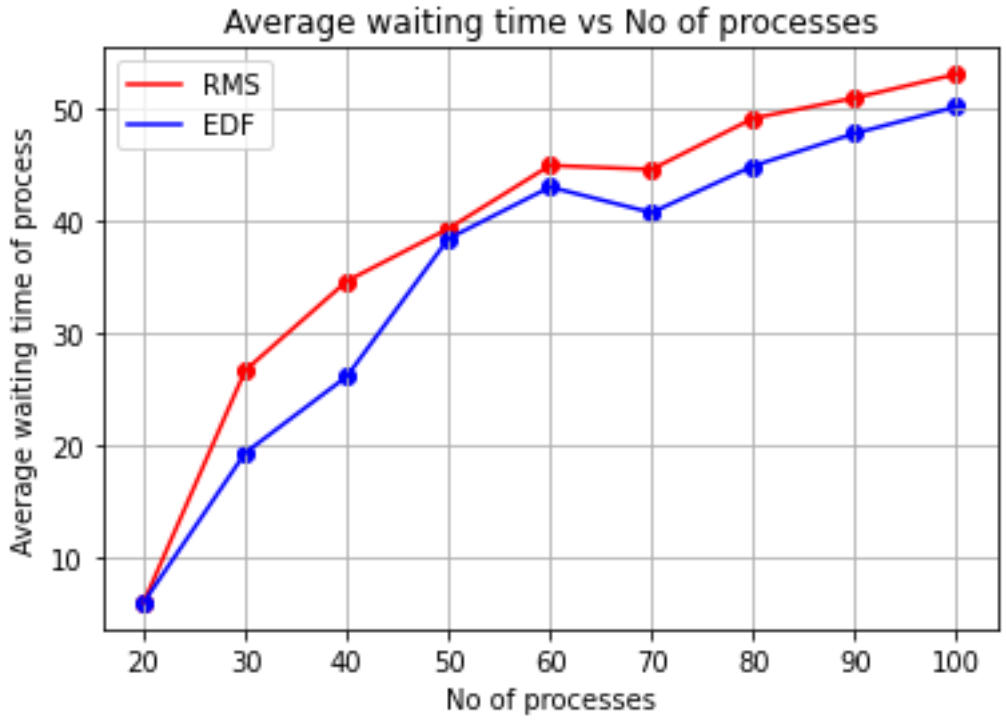
comparison of the performance of RM & EDF algorithms.

1. Deadline missed vs. No of processes

Graph 1: Analysis

- Unlike the RMS which uses a static priority policy, EDF scheduling uses assigns priority dynamically.
- On theory EDF scheduling is optimal. It can schedule processes so that each process can meet its deadline if possible.(most process)
- So, we can see that no of deadlines in case of EDF scheduling is always less than are equal to RMS scheduling
- With increasing number of processes there is an increase in number of missed deadline processes in both the algorithms due to the increase in demand of CPU allocation
- So EDF algorithm is efficient if you want to miss fewer deadlines.

2. Deadline missed vs. No of processes



Graph 2: Analysis

- Theoretically EDF ensures 100% utilization of CPU(but it is not possible in practise due to cost of context switching between the processes)
- By above graph, we can say EDF is more efficient but it can't ensure the average waiting to be better than RMS always
- As the no of process increase, the average waiting time of process increases as efficiency of algorithms decreases and more process will wait for more time
- In general the average waiting times for EDF will be less. But the average waiting time could be less for RMS also.