New Scheduling Algorithms/Strategies

Mitul Agrawal (B20AI021)
Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur
Rajasthan, India
agrawal.17@iitj.ac.in

Navlika Singh (B20AI025)
Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur
Rajasthan, India
singh.119@iitj.ac.in

Mitul Vashista (B20AI022)
Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur
Rajasthan, India
vashista.1@iitj.ac.in

Abstract

In the following paper, we made new scheduling algorithms and scheduling using Reinforcement Learning (reward based machine learning) and compared/visualized the results.

Index Terms

Operation Scheduling, Reinforcement Learning.

I. Introduction

Operation scheduling (OS) is a vital undertaking in the high-stage synthesis technique. Irrelevant scheduling of the operations can fail to make the most of the full potential of the system. We ambition to accumulate statistics on available scheduling algorithms, and observe and examine them. We brainstormed ideas and put in force modifications that might be nevertheless required to get advanced performance. We explored and performed machine learning-based methods for scheduling.

II. Literature Review

Sr. No.	Title	Method	Conclusion
1	Applying Machine Learning Techniques to improve Linux Process Scheduling	They use machine learning to discover most important static and dynamic attributes for minimum turnaround time	1.4 - 5.8 % reduction in turnaround time
2	A Machine Learning Approach for Improving Process Scheduling: A Survey	It discusses various ML approaches used for scheduling. (1: Interactive/Non-Interactive; 2: grouping processes with similar behavior; 3: semantic cognitive scheduling; 4: predicting burst time; 5: variable time slice for different processes; 6: assigning best time slice to reduce context switches	Machine learning techniques can be efficiently integrated into existing operating systems to deliver a seamless user experience.

3	Integration of Machine Learning into Operating Systems : A Survey	1 : 'Learned Operating System'; 2: Above survey; 3: dedicated small computer for artificial intelligence to process ml algorithms and store results	Analyzed 3 research papers related to machine learning in operating systems
4	Multiple Resource Management and Burst Time Prediction using Deep Reinforcement Learning	They use SchedQRM to classify burst time of jobs based on their signature and Deep Q-Network algorithm to find an optimal solution for any arbitrary job set.	Their scheduler SchedQRM outperforms the ad-hoc heuristics. Their agent is unable to choose two jobs of the same burst time together
5	Reinforcement Learning for Scheduling Threads on a Multi-Core Processor	They use Q-Learning state S - determined by the core utilization reward R - fraction of time each thread was scheduled vs not-scheduled action - limited to moving a thread from one core to another or do nothing	The RL algorithm ran every one second and it could get only ~80 samples per test. The downward spikes increase over time indicating that the algorithm made better scheduling decisions as time progressed

III. Approach

A. Customary Scheduling Algorithms -

We first implemented the customary scheduling algorithms namely: First-Come, First-Served Scheduling (FCFS) algorithm, Shortest-Job-First Scheduling Technique, Priority Scheduling, Round-Robin Scheduling, and Multilevel Queue Scheduling. First Come First Serve (FCFS) was easy to understand and implement. However, it gave a poor performance as the average waiting time was high. Shortest Job First (SJF) was the best approach to minimize waiting time. It was easy to implement in batch systems where CPU time is known in advance. But it is impossible to implement in interactive systems where the required CPU time is not known. Priority-based scheduling executed the processes based on their priority, and processes with the same priority were executed on a first come first serve basis. Priority was decided based on memory requirements. Round Robin scheduling required context switching to save states of preempted processes.

B. Round Robin with Varying Time Quantum -

We made 3 new Round Robin based scheduling algorithms -
i) Time Quantum is More for Higher Priority
ii) Time Quantum is More for Higher Burst Time
iii) Time Quantum is More for Lower Burst Time

C. Scheduling Using Reinforcement Learning -

Performed Scheduling Using a Technique of Machine Learning - Reinforcement Learning, where we made an environment (processes arrival and ready queue) and an agent (scheduler), where the scheduler gives a score to all of the processes in the ready queue and the process with the highest score gets chosen. The agent learns how to give this 'score' as we give feedback (reward) to it based on various factors like awt, att, priority. We had chosen different reward mechanisms and compared the awt with fcfs & sfs scheduling algorithms.

States (Inputs to Neural Network) -

1) Total Waiting Time since Process entered Ready Queue the First Time
2) Waiting Time since last execution of the Process
3) Total Execution Time
4) Execution Time Remaining
5) Priority

[States 1-4 are passed through Sigmoid Function to make them between 0-1, State 5 is already between 0-1, where 0 denotes maximum priority and 1 denotes least priority]

Actions (Outputs of Neural Network) -

1) Score [0-1] for each Process in the Ready Queue

Rewards (Feedback for Neural Network) -

1) + [1-Priority] for the process that got selected by the scheduler
2) - [Wait Time / 100] for the processes that didn't get selected
3) + 1 when a Process completes its burst

[This way a good balance between Priority and Wait Time can be achieved]

Environment (Rules of Processes Simulations) -

- Took Number of Processes in each Simulation as - 5,10,15,25
- Arrival - Random Difference with exponential distribution
- Burst - Random Integer between 1 & 20
- Priority - Random Uniform between 0 & 1

The Code -

model.py -

- Code for the Model (Pytorch)
- Took 1 Input Layer (5 Neurons), 1 Hidden Layer (32 Neurons), 1 Output Layer (1 Neuron)

plotter.py -

- To help with the plotting of the results

agent.py -

- ❖ Made function for tanh, sigmoid
- ❖ Made a ‘Ready Queue’ Class to implement the ready queue
- ❖ Made a ‘table’ class to generate the table as per the rules
- ❖ Made a ‘algos’ class to implement FCFS & SJF for Comparison
- ❖ Made a ‘sim’ class where scheduler picks the process and gets the reward
- ❖ Made a ‘Agent’ class where there are functions for the model to learn and predict
- ❖ def train : The main function to run above classes and perform the training

Final Model - model.pth

IV. Evaluation Technique

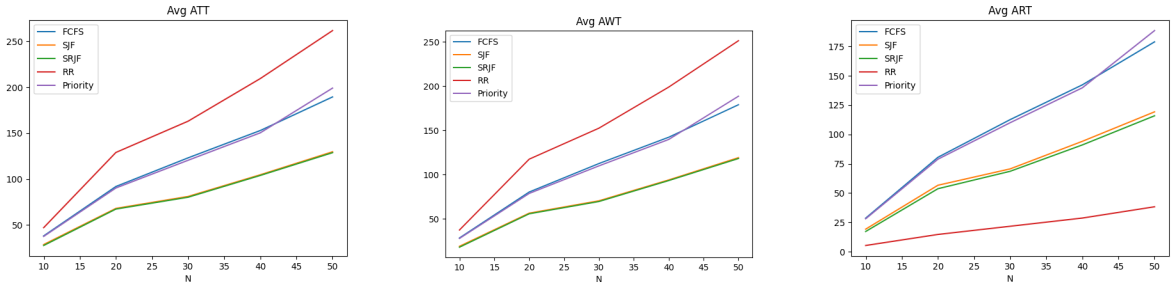
1) Average Waiting Time (Approach A & B & C)
   [It is Average of the Total Time each Process had to Wait in the Ready Queue.]

2) Average Turnaround Time (Approach A)
   [It is Average of the Total Time it took to Complete Process’ Request]

3) Average Response Time (Approach A & B)
   [It is Average of the Time each Process had to Wait before First Execution.]

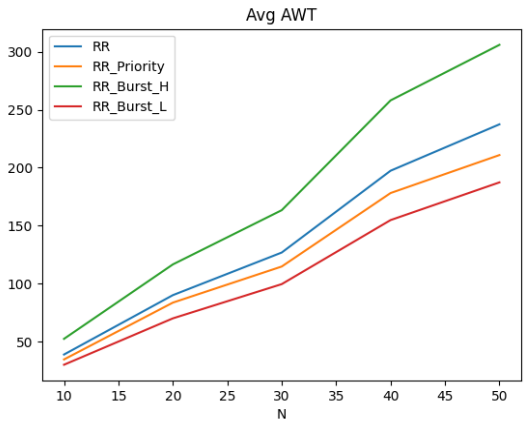
V. Results

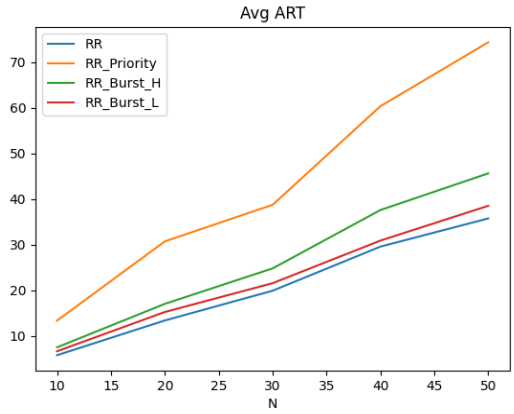
A. Customary Scheduling Algorithms -



RR has high AWT/ATT and low ART whereas SJF/SRJF has low AWT/ATT. Priority has similar times compared to FCFS while giving importance to processes with more priority.

B. Round Robin with Varying Time Quantum -



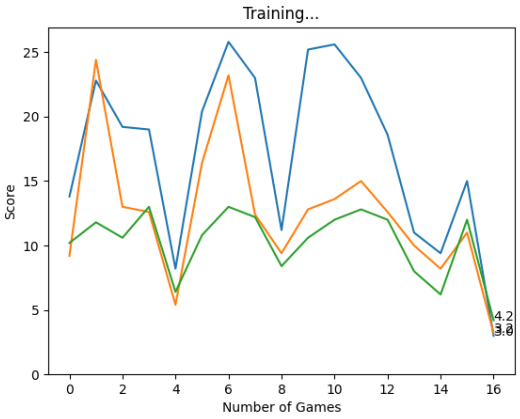


Round Robin where time quantum is more for less burst time has lowest AWT, Normal Round Robin (Quantum=2) has lowest ART.

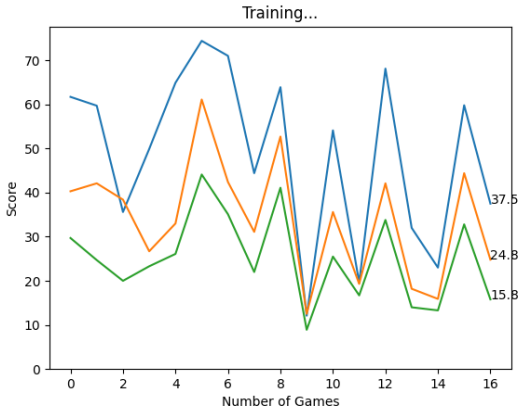
C. <u>Scheduling Using Reinforcement Learning</u> -

(Blue - Reinforcement Learning Scheduler | Orange - FCFS Algorithm | Green - SJF Algorithm)

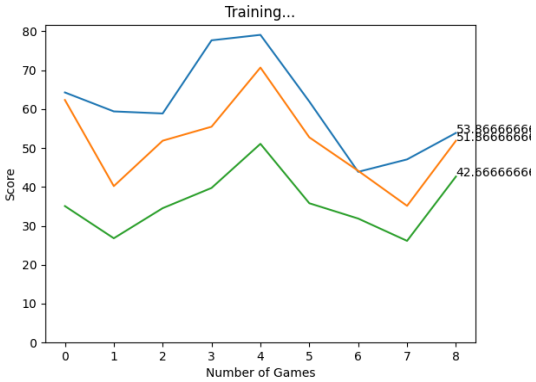
N (Process) = 5 :



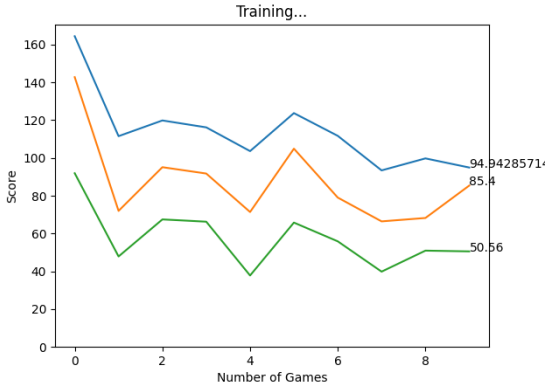
N (Process) = 10 :



N (Process) = 15 :



N (Process) = 25 :



[Average Waiting Time]
Each time the model ran for around 500 simulations.

We can see the Gaps closing in between the model and FCFS as the number of simulations increases. It is able to maintain a good balance between wait time and priority while choosing processes. The Gap is less for lesser N as that enables the model to learn faster and larger N enables the model to handle complex situations.

VI. Contribution

All authors have contributed equally to the project. We met frequently and did the work together.

Mitul Agrawal :
Initial Research, Common Algorithms, RR variations, Reinforcement Learning Approach, Making of the Report

Mitul Vashista :
Initial Research, Common Algorithms, RR variations, Reinforcement Learning Approach, Making of the Report

Navlika Singh :
Initial Research, Common Algorithms, RR variations, Reinforcement Learning Approach, Making of the Report