An Efficient Dynamic Round Robin Algorithm

1st Darsh Badodariya
B.Tech Information and Communication Technology
DA-IICT, Gandhinagar, India
201801049@daiict.ac.in

2nd Nishit Jagetia
B.Tech Information and Communication Technology
DA-IICT, Gandhinagar, India
201801027@daiict.ac.in

***Abstract*—The process scheduling algorithms of the operating system determines the efficiency of the system. Hence a better CPU scheduling algorithm results in faster OS performance using less resources for a small time. Out of the various scheduling algorithms, Round robin algorithm is mainly used. But the efficiency of the Round robin algorithm mainly depends on the chosen Time Quantum. Different types of approaches are available for determining time quantum related to round robin. These approaches help us to reduce waiting time, turnaround time and context switches of the processes eventually resulting in better efficiency of the systems. Lastly in this paper, we will propose our algorithm based on improvements made on round robin scheduling algorithms and compare with all the other existing algorithms based on implementation and testing done by us.**

***Index Terms*—Process Scheduling, Time Quantum, Waiting Time, Turn Around Time, Priority**

I. Introduction

For the multitasking systems, scheduling algorithms is one of the important functionalities of the Operating systems that should be executed. The goal of the scheduling algorithms is to increase the efficiency of the CPU. Here efficiency refers to lower average waiting time (the time process waits in the ready queue), lower average turnaround time (time between completion and starting of the process), lower the number of context switches and maximizing the throughput (the number of processes completed per unit time).

These scheduling algorithms are mainly divided into 2 types. One is non-preemptive algorithms where CPU is allocated to a process till it gets executed completely while the other is preemptive scheduling algorithms where CPU gets switched among the processes present in ready according to the allocated time quantum. Now the Most widely used scheduling algorithm is the Round Robin scheduling algorithm with some improvements made to it. But the issue with round robin scheduling algorithms is that if we keep the time quantum larger than the average waiting time and average turnaround time increases while if we keep small time quantum then context switches get increased resulting in low performance of the CPU. Hence there is need to make some improvements to this algorithm in order to improve the performance of the CPU. And at last we will propose a new algorithm and also will compare all the improved algorithms with the help of graphs.

II. Related Work

A. Improved Round Robin Algorithm (IRR)

In the Improved Round Robin Algorithm, the process is executed till one time quantum (static time quantum given by user). After the completion of one Time Quantum it's remaining time is checked. Now if the remaining time of the process is less than one time quantum then it is executed again and the process gets completed and is removed from the ready queue or else if the remaining time of the process is greater than one time quantum than it is pushed at the end of the ready queue. Hence it reduces the average waiting time compared to normal round robin algorithm.

B. Improved Round Robin Algorithm with Dynamic Mean Time Quantum (DABRR)

In this algorithm (DABRR) the process is same as Improved Round Robin algorithm but here the time quantum is calculated dynamically rather than provided by the user. Time Quantum of the process is equal to the mean of the remaining time of all the processes present in the ready queue. (Implemented With and Without Priority)

C. Improved Round Robin Algorithm with Dynamic (Mean + maximum / 2) Time Quantum (AMRR)

In this algorithm (AMRR) the process is same as Improved Round Robin algorithm but here the time quantum is calculated dynamically. Time Quantum of the process is equal to the (mean of the remaining time + maximum remaining time)/2 of the processes in the ready queue.

D. Improved Round Robin Algorithm with Dynamic Minimum Time Quantum (IRRVQ)

In this algorithm (IRRVQ) the process is same as Improved Round Robin algorithm but here the time quantum is calculated dynamically and it is equal to the minimum remaining time of the processes present in the ready queue. (Implemented With and Without Priority)

E. Improved Round Robin Algorithm with shortest job first (IRR_SJF)

In this algorithm (IRR_SJF) the process is same as Improved Round Robin algorithm but the ready queue is dynamically sorted according to the shortest remaining time first and the time quantum is taken as input (static time quantum) from the user. (Implemented With and Without Priority)

F. Adaptive Round Robin Algorithm (ARR_SJF_DTQ)

In Adaptive Round Robin Algorithm (ARR_SJF_DTQ), the ready queue is sorted according to the remaining time first of the processes and if the size of the ready queue is even then the time quantum is equal to the mean of the remaining time of all the processes else if the size is odd then the time quantum is equal to the remaining time of the middle process.

III. Implementation

A. Input and Output Files

We have implemented different files for input and output of the processes so that we can see the input and output data correctly and can also compare different terms for each algorithm. Also for comparison of different processes, we made an output file for graphs.

B. Real Time Input

We have taken 10,000 processes as input so that we can have accurate results of each algorithm and it will match almost to the real time scenario.

Also we have given inputs of different processes in such a manner that it quite resembles the real time scenario, as in like a user opens a system till he closes it, what may be the number of processes, what can be the arrival time and burst time of each process and so on.

C. Distribution of Arrival Time

Considering the situation, like initially when the user opens the system there are less number of processes and as he works and opens more applications the number of processes increases and when finally he completes his work, he closes all the applications and shuts down the system, so this results in the decrease of the processes, which can be seen by looking into the lowered ratio (number of process/time interval of arrival time) as compared to above ratios.

- First 100 process $\rightarrow$ at between 1 to 100
- Processes in between n/10 $\rightarrow$ at between 50 to 1000
- Process in between 2*n/10 $\rightarrow$ 1000 to 3000
- Process in between 4*n/10 $\rightarrow$ 3000 to 6000
- Process in between 2*n/10 $\rightarrow$ 6000 to 8000
- Process in between n/10 $\rightarrow$ 8000 to 10000

D. Distribution of Burst Time

Considering the situation, initially all the processes related to opening the system will occur and this will have less burst time and when the user opens and uses the applications the burst time of those processes will be greater compared to the initial process.

- First 100 process $\rightarrow$ between 1 to 50
- Remaining process $\rightarrow$ between 1 to 300

E. Distribution of Priorities

In real case, there might be some high level processes which are known as system process and their priority is the highest and their execution needs to be done as soon as possible otherwise they might cause some serious issues, then there are some medium level processes also known as interactive process, and then there are low level processes which are also known as batch process which are usually done in background. Considering all these scenarios and dividing all the the different levels of priorities into further sub levels we have maintained different priorities for all the processes.

- First 100 $\rightarrow$ priority between 1 to 5
- Process in between $\rightarrow$ priority between 1 to 50

F. Changes in levels of priority

After each dynamic/static time quantum, if the process doesn't get executed then if it is not of the highest priority its priority will increase by 1 unit automatically, this will help to solve the problem of starvation for lower priority processes and it will also help to reduce average waiting time and average turnaround time.

IV. Proposed Algorithm

A. Algorithm

- Our algorithm is based on dynamic time quantum.
- It is a combination of Improved Round Robin with Shortest Job First and time quantum to be calculated dynamically. (Name of the algorithm in the code implemented :- **irr_sjf_dtq** with and without priority).
- Our algorithm will dynamically calculate the time quantum each time and will execute for that particular time for each process in the ready queue.
- Time Quantum for our algorithm will be the mean of all the processes which are present in the ready queue added with the highest remaining time of a process in the ready queue and divided by 2.
- TQ = (Mean + Max_Remaining_Time) / 2.
- Then we will sort the process in the ready queue according to the remaining time so that it will reduce the average waiting and average turnaround time.
- As we are using the dynamic approach, we will not have the issue of response time. (in case any large process arrives first).
- We will implement the above algorithm based on considering the priorities and also not considering the priorities.
- (Note - For priority based implementation the time quantum of our algorithm will be dynamically calculated for that particular priority only, i.e. each process with the same priority will have the same time quantum).

B. *Pseudo-code*

Below is the pseudo code and some terms related to the algorithm.
TQ : Time Quantum
P : Priority
R : Range of priority
Remain : number of processes with remaining time greater than zero

1) First sort the processes according to their arrival time
2) while (remain $>$ 0) repeat step 2 to 6
3) Sort the processes present in ready queue according to their remaining time
4) Calculate the sum of remaining time and maximum remaining time of ready processes
5) Calculate the time quantum as (mean of remaining time + max of remaining time) / 2
6) Run all the ready processes with the above calculated time quantum.
7) At last, calculate the total average waiting time and total turnaround time for all the processes.

Note :- In case of priority based implementation, some changes will be as follows:-

- In step 1, first sort according to priorities and then sort according to arrival time.
- There will be R different time quantums related to R different priorities. Hence the time quantum for particular priority will be (mean + max)/2 of the ready processes having that particular priority.
- While executing the processes, if the priority is not 1 than the priority of that process will get increased by 1 after every execution until it gets completed.

V. Experimental Findings

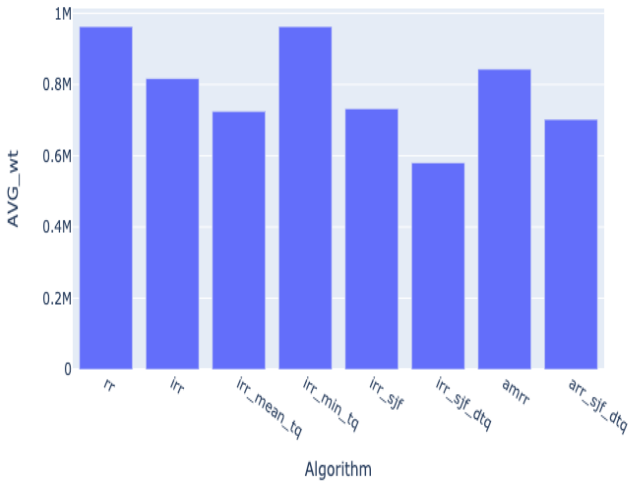

Fig. 1. Average waiting time of processes without priorities.

We have executed each algorithm on 10000 processes and we have distributed the arrival time, burst time and priorities for the processes in such a manner that it corresponds to some real time scenario.

We will compare the results based on the average waiting time and average turnaround time of all the processes in a particular algorithm.

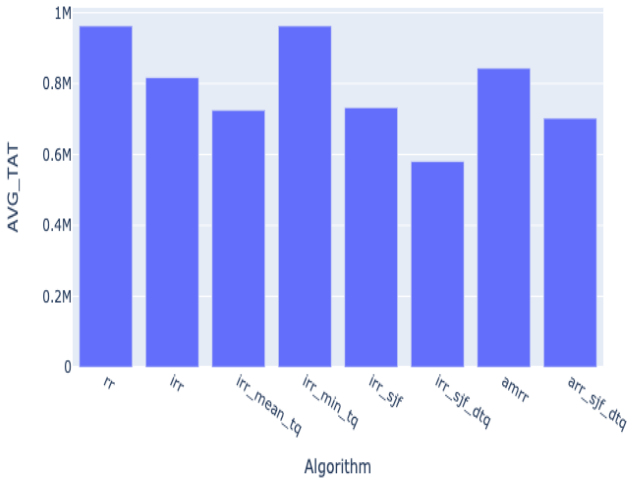

Fig. 2. Average turn-around time of processes without priorities.

As you can see in the results above (figure - 1 and figure - 2) the algorithms without using the priorities of processes, our algorithm (irr_sjf_dtq, i.e. Improved Round Robin Shortest Job first with Dynamic Time Quantum) is performing the best.

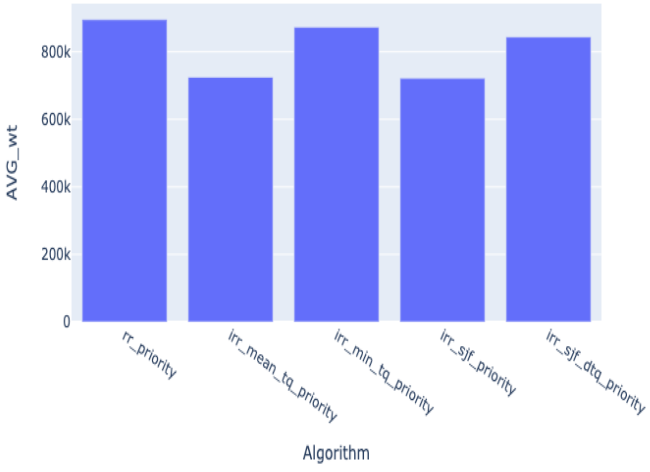

Fig. 3. Average waiting time of processes with priorities.

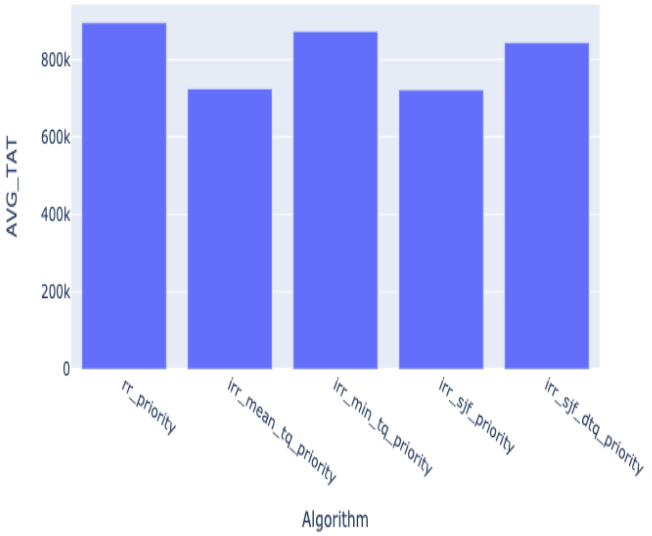

Fig. 4. Average turn-around time of processes with priorities.

As you can see in the results above (figure - 3 and figure - 4) the algorithms with using the priorities of processes, our algorithm(irr_sjf_dtq, i.e. Improved Round Robin Shortest Job first with Dynamic Time Quantum) is performing better than the normal round robin algorithm.

VI. Conclusion

CPU scheduling is a great challenge indeed to enrich and make the best and significant utilization of CPU in the operating system. As we can see that the proposed algorithm is quite more efficient than the normal round robin algorithm when compared with the average waiting time and average turnaround time, but we have presented a model that is quite similar to the real case scenario, while in real case there might be some limitations or drawbacks of this algorithm.

Nonetheless, implementing it at a kernel level and deploying all its features will only tell us how perfect the algorithm is and what changes can be made to it to optimize it further. Our future work is to deploy the algorithm at the kernel level and further improve the performance and the utility of the CPU.

References

[1] Abd Elkader, Afaf & Eldahshan, Kamal. (2017). Round Robin based Scheduling Algorithms, A Comparative Study.

[2] Alsheikhy, Ahmed & Ammar, Reda & ElFouly, Raafat. (2015). An Improved Dynamic Round Robin Scheduling Algorithm Based on a Variant Quantum Time. 10.1109/ICENCO.2015.7416332.

[3] Farooq, Muhammad & Shakoor, Aamna & Siddique, Abu. (2017). An Efficient Dynamic Round Robin Algorithm for CPU scheduling. 10.1109/C-CODE.2017.7918936.

[4] Khatri, Jayanti. (2016). An Improved Dynamic Round Robin CPU Scheduling Algorithm Based on Variant Time Quantum. IOSR Journal of Computer Engineering. 18. 35-40. 10.9790/0661-1806043540.

[5] Igbaoreto, Azeez & Omotehinwa, Temidayo Oluwatosin & Sunday Samuel, Olofintuyi. (2019). A Simplified Improved Dynamic Round Robin (SIDRR) CPU scheduling Algorithm.

[6] Forhad, Md & Das, Mrinmoy & Hossain, Md. (2018). An Improvement of Round Robin Scheduling Algorithm.

[7] A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon and Q. Mao, "Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling," 2019 SoutheastCon, 2019, pp. 1-5, doi: 10.1109/SoutheastCon42311.2019.9020439.

[8] Biswas, Dipto & Samsuddoha, Md. (2019). Determining Proficient Time Quantum to Improve the Performance of Round Robin Scheduling Algorithm. International Journal of Modern Education and Computer Science. 11. 33-40. 10.5815/ijmecs.2019.10.04.

[9] Mohanty, Rakesh & Das, Manas & Prasanna, M. & Sudhashree,. (2011). Design and Performance Evaluation of A New Proposed Fittest Job First Dynamic Round Robin(FJFDRR) Scheduling Algorithm.