**Optimized Round Robin CPU Scheduling Algorithm Using Priority Technique**

**Abstract:**

*One of the most essential and critical part of a computer is Operating system. Scheduling is the most important function involved in working of operating system. Operating system with uni-processor were generally divided into two catagories. i.e uni-programming operating system and multi-programming operating system.Uni- programming performs a single task at a time, while multiprogramming system is able to manage various jobs simultaneously. Use of resources is the basic goal of operating system. Multiprogramming operating system has many scheduling algorithms. But in the view of optimization, our target is to develop a novel scheduling algorithm for multi-programming system. We have developed an algorithm that gives output according to the prescriptive and precise task scheduling e.g. in the form of experimental results. Round Robin (RR) is the programming algorithm mostly used in timeshared systems. Quantum time is the basic factor to define the efficiency of Round Robin algorithm, if the time slice is small, there will be overheads of more context switches and if the quantity time is high, then provided algorithm will work as First Come First Served (FCFS) in which the risk of starvation is greater.*

**Keywords:**

*Operating system, uni-processor, uni programming, multi-programming, priority, round robin, etc.*

***Introduction:***

*Scheduling is the core of any system of computer as it includes the decision to allocate resources among possible processes. When we share computer resources among different components is also called scheduling. The CPU is called as one of the primary computer resources, therefore its scheduling is necessary for the design of an operating system. Efficient utilization of resources is achieved through the sharing of system resources among multiple users and system processes. Optimum sharing of information depends on the processor's efficient scheduling of competing users and device processes, which makes process scheduling an essential feature of a multi - programming operating system. Since the processor is called as the most important resource so, process scheduling, becomes very important in achieving the above-mentioned goals. One of the reason for using multi-programming is that the operating system itself is implemented as one or more processes, so there must be a way to share the CPU with the operating system and the application processes. In proposed algorithm, we have improved the average turnaround time and average waiting time of round robin scheduling algorithm by using the scheduling technique. The processes were divided into two categories i.e. high priority scheduling and low priority scheduling.*

***Literature Review:***

Thegoal of this paper is to strengthen the current Round Robin Scheduling by reducing the person waiting time as a result and also overall waiting and turnaround time. In this paper we also compare the existing round robin scheduling algorithm with the proposed algorithm, using Gantt chart. The suggested methodology will help in minimizing a number of performance parameters such as average waiting time and average turnaround time[1]. The goal of this paper is to strengthen the current Round Robin Scheduling by reducing the person waiting time as a result and also the overall waiting and turnaround time. In this paper we also compare the existing round robin scheduling algorithm with the proposed algorithm, using Gantt. The suggested methodology will help in minimizing a number of performance parameters such as average waiting time and average turnaround time. This paper calls for a new algorithm which enhanced further on the enhanced Round Robin CPU (IRR) Manish implementation planning and AbdulKadir. Application of the proposed algorithm and Measured against 5 other factor analysis model in the Read more Literature. The proposed algorithm as compared to the other Algorithms, yield minimum average waiting time (AWT), Average turnaround time (ATAT), and the interpretation number Turns (NCS). The proposed findings are based on those results. Algorithm should be preferred to other timetables. Systems algorithm is proposed which adopt RR CPU scheduling.[2]. A new CPU scheduling algorithm, named DABRR (Dynamic Average Burst) was presented in this thesis Robin Round). That vector images Dynamic Time rather than Quantum Static Time used in RR. The efficiency of the proposed controller is evaluated experimentally with conventional RR and some current ones RR-variants. The results of our approach outlined in this paper show improved performance in Average processing time, average turnaround time, and background switching conditions.[3].The following algorithm categorizes the process into High and low priority process. Average waiting time of processes having high priority has been reduced completely and in some cases of low priority. The overall waiting time of all the processes has been reduced which is a positive factor for round robin scheduling algorithm.[4]. The quantum is the main factor for the execution of processes, if it is too large then process have to wait for long and if too small then a lot of context switches occur during the process. In this paper, a new technique has been developed using Integar Programming that decides a price that is neither too large nor too small so that each process will get affordable quantum time to execute while throughput of the system would not be reduced during unnecessary context switching.[5]

**Steps for Good Scheduling:**

• **Utilization/Efficiency:**

It keeps the CPU busy with useful work 100 per cent of the time.

**• Throughput**: maximize throughput by increasing the number of jobs handled in unit time

• **Turnaround time:**

The time on which a process enters in the ready queue to the time at which it executes completely is called Turnaround time. We have to minimize the Turnaround time for good scheduling.

• **Waiting time:**

It is the sum of time spend in the waiting queue by a process for its execution. We have to minimize the waiting time of processes.

**• Response Time**:

There is a difference between first execution time to the arrival time.

**Scheduling Algorithms:**

We will start with five commonly used scheduling algorithms.

**First Come First Serve (FCFS):**

FCFS is non-preemptive CPU scheduling algorithm. The process that arrives first in the ready queue will be executed first. Disadvantage of scheduling FCFS is that short processes have to wait a little bit longer to execute.

**Shortest Job First (SJF):**

It is non-preemptive CPU scheduling. The processes with small burst time will execute first. Disadvantage of SJF is that there will be a chance that process with large burst time will starvate.

**Priority Scheduling Algorithm:**

In priority scheduling, priority has been considered as a factor for execution of processes. The process with high priority will be given the priority to execute. It has also disadvantage that process with low priority has a chance to starvate.

**DRAWBACKS OF ROUND ROBIN SCHEDULING ALGORITHMS:**

Round Robin scheduling algorithm has many disadvantages, which are as following:

* High Average Waiting Time.
* Throughput is very low.
* Context switch.
* Response time is high.
* Turnaround time is very high.

**Experimental Work**

**Improved Round Robin:**

In improved Round Robin algorithm, the number of processes were divided into two categories i.e High priority processes and Low priority processes queues. In this proposed algorithm, the process with priority value 1 are considered as High priority processes and the processes with priority value 0 are considered as Low priority processes. In the given chart, we have categorized P1 and P2 as High priority processes and P3, P4 and P5 as Low priority processes. There will be two separate queues for the processes to execute. The processes present in High priority queue will execute first and then processes with Low priority will execute.

**Algorithm:**

* Collect the processes with their burst time, quantum time and priority value
* Divide the processes into two ready queues i.e High priority queue and Low priority queue
* Arrange all the process in ascending order according to burst time in two ready queues.
* Check if there is any process in high priority queue, then execute it first. Else execute the processes from low priority queue
* Calculate the average turnaround time and average waiting time.

|  |  |  |
| --- | --- | --- |
| Process | Priority | burst time |
| P1 | 1 | 2 |
| P2 | 1 | 6 |
| P3 | 0 | 7 |
| P4 | 0 | 12 |
| P5 | 0 | 14 |

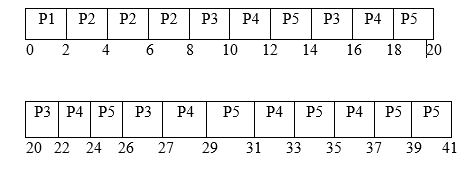
Quantum=2

**Table with High Priority**

|  |  |  |
| --- | --- | --- |
| Process | Priority | Burst Time |
| P1 | 1 | 2 |
| P2 | 1 | 6 |

**Table with Low Priority**

|  |  |  |
| --- | --- | --- |
| Process | Priority | Burst Time |
| P3 | 0 | 7 |
| P4 | 0 | 12 |
| P5 | 0 | 14 |

Gantt Chart:

Turn Around Time=Completion Time –Arrival Time

As all the processes arrives at the same time, Arrival time of all the processes would be 0

TAT(1)=2

TAT(2)=8

TAT(3)=27

TAT(4)=37

TAT(5)=41

Avg TAT= 2+8+27+37+41 /5

Avg TAT= 115/5

Avg TAT=23

Waiting Time = Turn Around Time – Burst Time

WT(1)= 2-2 = 0

WT(2)= 8-6 = 2

WT(3)=27-7 = 20

WT(4)=37-12 = 25

WT(5)=41-14 = 27

Avg WT= 0+2+20+25+27 /5

Avg WT= 74/5

Avg WT=14.8

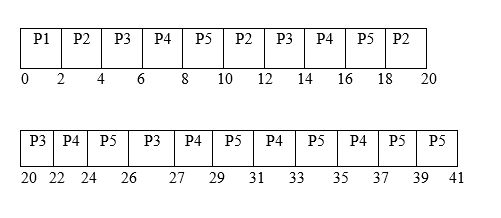
**Comparison with Round Robin Scheduling Algorithm**

Round Robin is a preemptive(CPU can be snatched from running process) CPU Scheduling algorithm. In round robin scheduling algorithm, each process is provided with a fix time to execute, called Quantum time. Once a process execute for a specific period, then CPU is preempted and given to next process waiting in the ready queue.

**Round Robin Scheduling Algorithm**

|  |  |
| --- | --- |
| Process | Burst Time |
| P1 | 2 |
| P2 | 6 |
| P3 | 7 |
| P4 | 12 |
| P5 | 14 |

Quantum=2

Gantt chart:

Turnaround Time=CompletionTime- Arrival Time

As all the processes arrives at the same time, Arrival time of all the processes would be 0

TAT(1)=2

TAT(2)=20

TAT(3)=27

TAT(4)=37

TAT(5)=41

Avg TAT= 2+20+27+37+41 /5

Avg TAT= 127/5

Avg TAT=25.4

Waiting Time = Turn Around Time – Burst Time

WT(1)= 2-2 = 0

WT(2)= 20-6 = 14

WT(3)=27-7 = 20

WT(4)=37-12 = 25

WT(5)=41-14 = 27

Avg WT= 0+14+20+25+27 /5

Avg WT= 80/5

Avg WT=17.2

**Comparison Table**

|  |  |
| --- | --- |
| Round Robin  Scheduling | Improved Round Robin Scheduling |
| Average TAT= 25.4 | Average TAT= 23 |
| Average WT= 17.2 | Average WT= 14.8 |

From the above calculations, it has been proved that Average Turnaround Time and Average Waiting Time has been improved in the proposed round robin scheduling algorithm.

**Conclusion**

This whole paper concludes the comparative studies of Round Robin Algorithm alongwith the new, improved round robin algorithm. By introducing priority scheduling technique in round robin scheduling algorithm turnaround time and waiting time has been improved which would reduce the process overhead and save memory space

**References:**

[1] M. Laxmijeevani, T. S. P. Madhuri, and Y. S. Devi, “Improvised Round Robin Scheduling Algorithm and Comparison with Existing Round Robin CPU Scheduling Algorithm,” *IOSR J. Comput. Eng.*, vol. 20, no. 3, pp. 1–4, 2018, doi: 10.9790/0661-2003010104.

[2] A. Abdulrahim, S. E. Abdullahi, and J. B. Sahalu, “A New Improved Round Robin (NIRR) CPU Scheduling Algorithm,” *Int. J. Comput. Appl.*, vol. 90, no. 4, pp. 27–33, 2014, doi: 10.5120/15563-4277.

[3] A. R. Dash, S. kumar Sahu, and S. K. Samantra, “An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum,” *Int. J. Comput. Sci. Eng. Inf. Technol.*, vol. 5, no. 1, pp. 07–26, 2015, doi: 10.5121/ijcseit.2015.5102.

[4] . M. K. M. R., “an Improved Approach To Minimize Context Switching in Round Robin Scheduling Algorithm Using Optimization Techniques,” *Int. J. Res. Eng. Technol.*, vol. 03, no. 04, pp. 804–808, 2014, doi: 10.15623/ijret.2014.0304141.

[5] D. P. Kumar, T. S. Reddy, and A. Y. Reddy, “Finding Best Time Quantum for Round Robin Scheduling Algorithm to avoid Frequent Context Switch,” vol. 5, no. 5, pp. 6750–6754, 2014.