International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-9, Issue-1, January 2022
https://doi.org/10.55524/ijircst.2022.10.1.10
Article ID IJIRD-1154, Pages 59-65
www.ijircst.org

Improvisation of Round Robin Algorithm

Aishna Gupta[1], and Patil Darshan Rajkumar[2]

[1,2] Student, Department of Computer Science and Engineering. Vellore Institute of Technology, Vellore, India

Correspondence should be addressed to Aishna Gupta; darshofficial18@gmail.com

ABSTRACT- We are trying to improvise the existing Round Robin Algorithm and test its performance across host OS and VM as well. We would be testing the performance of the improvised algorithm on the Cent OS running on Virtual Box..
We have worked on the scheduling algorithm specifically Round Robin Scheduling in our lab classes. Basically, we will be finding out the best way to properly select the "Time Quantum" so as to get Minimum 'Waiting Time' and 'Turnaround Time'.
After looking into so many research papers, we have found that, by sorting the processes in increasing order of their burst time and taking median of all the burst time, we can minimize the waiting and turnaround time.

KEYWORDS- Round Robin, Cent OS, Waiting Time, Time Quantum, Turnaround Time.

I. INTRODUCTION

When we look around and count how many electronic devices we can see from our current point of view, we probably have numerous devices just within our current eyesight. In order for many of those devices to work, they have some sort of operating system (OS) that allows them to boot up and provide whatever experiences they have to offer[1]. In fact, if you are reading the digital version of this report, the very device you are now holding has an operating system of its own.

II. SCHEDULING ALGORITHM

The Central Processing Unit (CPU) should be utilized efficiently as it is a core part of Computers and for this reason, CPU scheduling is very necessary. CPU Scheduling is an important concept in Operating System. Sharing of computer resources between multiple processes is called scheduling. A process is an instance of a program running on a computer[2]. It includes the current values of the program counters, all the registers, and also the variables. The processes waiting to be assigned to a processor are put in a queue called the ready queue. Burst time is amount of time for which a process is being held by the CPU. When a process arrives at the ready queue it is the arrival time. From the time a process is submitted, the time it is completed is called the turnaround time. When a process waits in the ready queue, the time is called the waiting time. The number of times the CPU gets switched from one process to another, it is called context switching[3]. The optimal and the best scheduling algorithm will have less waiting time, less turnaround time and a smaller number of context switches.

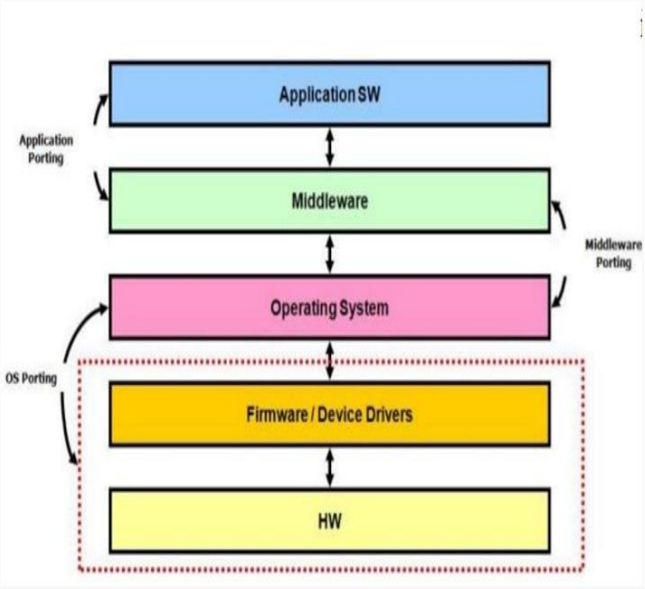

Figure 1: Operating System Framework

In Figure 1, Hardware is a framework comprising of electronic gadgets, intended to cooperate as a solitary unit. Firmware is a program that is explicitly intended to work with specific equipment and it lives in non-unstable memory such a blaze and it is executed straightforwardly from it. An operating system is a program that abstracts the hidden programming determined to work on the effectiveness and usability both for the end-clients and application software engineers. Middleware is a PC program that interfaces 2 programming together. The 2 programming that requirement to interface can be in a similar machine or in 2 machines in a similar room or it tends to be in 2 corners of the world. Software is a program that can chip away at a wide variety of equipment and they are generally replicated from non-unpredictable memory (like hard-plate or SSD) onto unstable memory (like SRAM and DRAM) before they can begin executing.

Objectives of Process Scheduling algorithm:

- Maximum CPU utilization.
- Fair allocation of CPU.
- Achieve the maximum throughput (Number of processes that complete their execution per time unit).
- Minimize the turnaround time (Time taken by a process to finish execution).

- Minimize the waiting time (Time a process waits in ready queue).
- Minimize the response time (Time when a process produces first response).

III. LITERATURE SURVEY

Authors	Name of the Paper	Outcomes
Emami Ale Agha et al.[6]	A New Method to Improve Round Robin Scheduling Algorithm with QuantumTime Based on HarmonicArithmetic Mean	Quantum Time Based on Harmonic-Arithmetic Mean.
Mohan et al.[7]	Robust Quantum Based Low-power Switching Technique to improve System Performance	Low-power Switching Technique
Helmy et al.[8]	Burst Round Robin as a Proportional-Share Scheduling Algorithm	Proportional-share CPU scheduling, Quality of Services and Performance Management.
Manoj Kumar Srivastav et al. [9]	Fair Priority Round Robin with Dynamic Time Quantum: FPRRDQ	Fair value of time quantum to each process according to the priority and burst time of that process.
Behera et al.[10]	A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time quantum for Real Time Systems	Context Switches, Waiting Time, Turnaround time

IV. PROJECT RESOURCE REQUIREMENTS

A. Software Requirements

- Any version of Windows, preferably Windows 11.
- C-compiler.
- Command Prompt.
- Jupyter Labs.
- Ubuntu on VirtualBox host.

B. Hardware Requirements

- Laptop/Computer.
- 1.8GHz CPU.
- I3 Processor.
- 4GB RAM.

V. ROUND ROBIN ALGORITHM

In Round Robin Algorithm, each process is allowed to use the CPU for a given amount of time and if it does not finish within the allotted time, it is pre-empted and then moved at the back of the line so that the next process in line is able to use the CPU for the same amount of time.[4]

A. Algorithm for Round Robin Algorithm

- Start.
- Enter the number of process and Time Quantum.
- Enter the Arrival Time and Burst Time for all the process.
- Using for loop parse through all the process.
- If(burst-time<quantum-time), then execute the process.
- Else, execute the process till the quantum-time and continue to the next process.
- Now calculate the Average waiting time, average turnaround time and total Number of context switches required for the process.
- End.

B. Schematic Representation of Round Robin Algorithm

From the Fig. 2, we have calculated the following values:
Time Quantum: 3ms
Average Waiting Time=11ms
Average Turnaround Time=15.2ms
Context switch: 9

0 P1 3	3 P2 6	6 P3 8	8 P4 11	11 P5 14	14 P1 16	16 P2 17	17 P4 20

Figure 2: Gantt Chart

VI. FLOWCHART

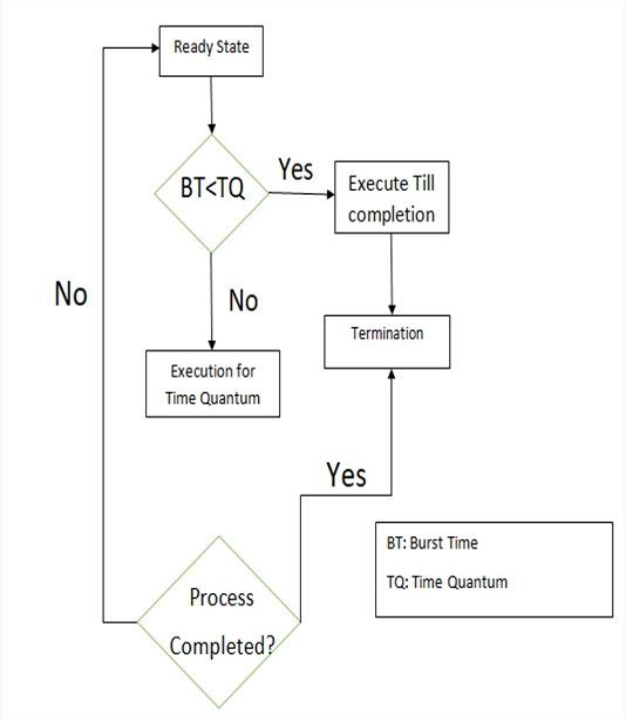

Figure 3: Flowchart of working of Round Robin Algorithm

The Figure.3 explains the working of round robin algorithm. Once the user enters the number of processes, quantum time, arrival time and burst time, if BT<TQ then execute the process till completion else execute it for the till quantum time and continue to the next process. Then calculate the required entities and terminate the process.

VII. CODE

```
#include<stdio.h>
int main()
{
   int count, j, n, time, remain, flag=0, time_quantum, x=0;
   int wait_time=0, turnaround_time=0, at[25], bt[25], -
   rt[25];

   printf("Enter Total Process:\t ");
   scanf("%d",&n);

  remain=n;
 for(count=0;count<n;count++)
 {
    printf("Enter Burst Time for Process Process Number
            %d :",count+1);
    scanf("%d",&bt[count]);
   at[count]=0;
  rt[count]=bt[count];
 }
          printf("Enter Time Quantum:\t");
          scanf("%d",&time_quantum);
          printf("\n\nProcess\t|TurnaroundTime|Waiting
          Time\n\n");
for(time=0,count=0;remain!=0;)
 {
  if(rt[count]<=time_quantum&& rt[count]>0)
  {
   time+=rt[count];
   rt[count]=0;
   flag=1;
  }
  else if(rt[count]>0)
  {
   rt[count]-=time_quantum;
   time+=time_quantum;
   x++;
  }
  if(rt[count]==0 && flag==1)
  {
   remain--;
    printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-
    at[count],time-at[count]-bt[count]);
    wait_time+=time-at[count]-bt[count];
    turnaround_time+=time-at[count];
   flag=0;
          x++;
  }
  if(count==n-1)
   count=0;
      else if(at[count+1]<=time)
   count++;
      else
        count=0;
 }
printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
printf("Average Turnaround Time =
%f",turnaround_time*1.0/n);
printf("\nContext Switching=%d",x);
 return 0;
}
```

VIII. OUTPUT ON JUPYTER LABS

```
Enter Total Process:     5
Enter Burst Time for Process Process Number 1 :5
Enter Burst Time for Process Process Number 2 :4
Enter Burst Time for Process Process Number 3 :2
Enter Burst Time for Process Process Number 4 :7
Enter Burst Time for Process Process Number 5 :3
Enter Time Quantum:     3


Process |Turnaround Time|Waiting Time

P[3]    |       8       |       6
P[5]    |       14      |       11
P[1]    |       16      |       11
P[2]    |       17      |       13
P[4]    |       21      |       14

Average Waiting Time= 11.000000
Average Turnaround Time = 15.200000
```

Figure 4: Output of Round Robin Algorithm

In Figure.4 we obtained the output of Round Robin Algorithm by running the code in Jupyter Labs. User has entered 5 processes with its burst time and time quantum

value. Our code has calculated the waiting time, and turnaround time.

IX. DRAWBACKS

- The efficiency of the system is decreased if the quantum value is low as frequent switching takes place.
- The system may become unresponsive if the quantum value is high.

X. IMPROVISED VERSION OF ROUND ROBIN ALGORITHM

We are trying to improvise the classical Round Robin Algorithm. In this, we will sort the processes in increasing order of their Burst Times. For selection of Time Quantum, we will use the formula TQ=ceil (sqrt (median*highest Burst Time)) [5]. By using this Time Quantum, we can achieve Minimum Waiting and Turnaround Time.

A. Algorithm for Improvised version Round Robin Algorithm

- Start.
- Sort all the processes in increasing order, according to their burst time.
- While (ready queue! = NULL) Find Time-Quantum, where Time-Quantum=Ceil (sqrt (median * highest Burst time)).
- Assign Time-Quantum to process.
- Now, follow the same steps as followed in round robin method.
- Now calculate the Average waiting time, average turnaround time and total Number of context switches required for the process.
- End.

B. Schematic Representation of Improvised Version of Round Robin Algorithm

Table 1: Representation of processes and its Burst time

Process	Burst Time
P3	2
P5	3
P2	4
P1	5
P4	7

In Table 1, we have sorted number of processes in increasing order of their burst time.

C. Gantt Chart

Table 2: Gantt Chart

0 P3 2	2 P5 5	5 P2 9	9P1 14	14 P420	20 P421

Table 2 represents the Gantt chart of the inputs and by using the above Gantt chart we have calculated following values:

Time Quantum=6ms
Average Waiting Time=6ms
Average Turnaround Time=10.2ms
Context switch: 6

D. Flowchart of Improvised version of Round Robin Algorithm

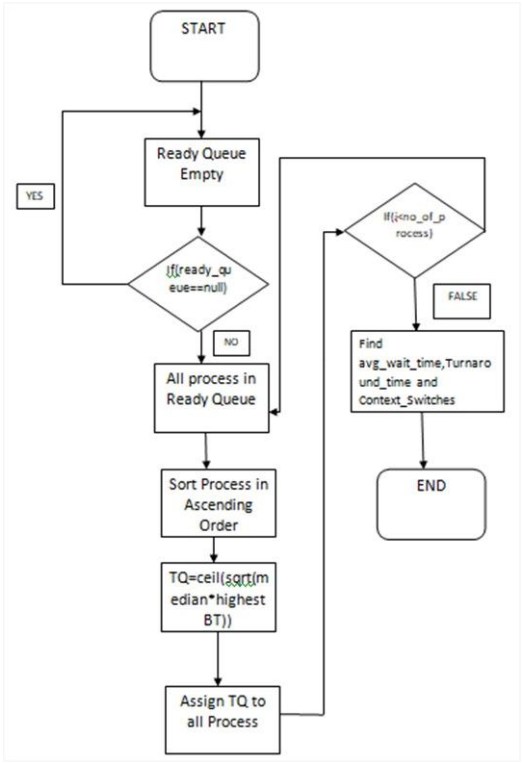

Figure 5: Flowchart of Improvised version of Round Robin Algorithm

In the Figure 5, we have summarized the working of Improvised version of Round Robin Algorithm. Once the process starts, sorting of processes are done in increasing order of their burst time and then time quantum value is calculated by using the special formula. Once time quantum value is assigned to the processes, average waiting time, turnaround time, and context switch value is calculated.

E. Code

```
#include<stdio.h>
#include<math.h>
//Iniatilization
int st[10],x=0;
//Function for Finding Time Quantum
int get_tq(int b[] , int s)
{
        int i,j,maxbt,tmp,hbt,median;
        float k,l,m;
//Sorting the process according to process Burst Times
for(i=0;i<s;i++)
{
        for(j=i+1;j<s;j++)
{
if(b[i]>b[j])
{
        tmp=b[i];
        b[i]=b[j];
        b[j]=tmp;
}
```

```
}
}
hbt=b[i-1];
median=b[i/2];
   for(i=0;i<s;i++)
   st[i]=b[i];
   l=(float)hbt;
   m=(float)median;
  k=sqrt((l*m));
    return(ceil(k));
}
//Main Function
int main()
{
      int bt[10],wt[10],tat[10],n,tq;
      int i,count=0,swt=0,stat=0,temp,sq=0;
      float awt=0.0,atat=0.0;
      printf("Enter the Number of Processes: ");
      scanf("%d",&n);

//Getting Burst Time for all sequences

     printf("Enter the Burst Time for all the Sequences:\n");
       for(i=0; i<n; i++)
        {
            scanf("%d",&bt[i]);
            st[i]=bt[i];
        }

tq=get_tq(st, n);
printf("\nTime quantum is ceil ((Highest Burst Time + Median)/2) = %d\n",tq);
//Working like Round Robin Algorithm
while(1)
{
    for(i=0, count=0; i<n; i++)
{
temp=tq;
if(st[i]==0)
{
    count++;
continue;
x++;
}
  if(st[i]>tq)
   {
     st[i]=st[i]-tq;
     x++;
   }
else
if(st[i]>=0)
{
            temp=st[i];
            st[i]=0;
            x++;
}
 sq=sq+temp;
  tat[i]=sq;
}
  if(n==count)
   break;
}

for(i=0; i<n; i++)
{
            wt[i]=tat[i]-bt[i];
            swt=swt+wt[i];
            stat=stat+tat[i];
}
            awt=(float)swt/n;
            atat=(float)stat/n;
            printf("\nAverage Waiting Time is %f\n",awt);
            printf("\nAverageTurnAround Time is %f\n",atat);
            printf("\nContext Switching=%d",x);
}
```

F. Output on Jupyter lab

```
jovyan@jupyter-jupyterlab-2djupyterlab-2ddemo-2dnpr4tbuk:~$ gcc Imprrs.c -o Imprrs -lm
jovyan@jupyter-jupyterlab-2djupyterlab-2ddemo-2dnpr4tbuk:~$ ./Imprrs
Enter the Number of Processes: 5
Enter the Burst Time for all the Sequences:
5
4
2
7
3

Time quantum is ceil ((Highest Burst Time + Median)/2) = 6

Average Waiting Time is 6.000000

Average TurnAround Time is 10.200000
```

Figure. 6: Output of Improvised version of Round Robin Algorithm

In Figure 6, we have obtained the output of Improvised version of Round Robin Algorithm by running the code in Jupyter Labs. User have entered 5 processes with its burst time. Our code has first calculated the Time Quantum value by using the formula as printed on Figure. 6, and then waiting time, and turnaround time are calculated.

G. Improvement in Various Parameters

Table 3: Improvements in Various Parameters

Set	Number of Processes	Burst Times	Time Quantum in Round Robin	Time Quantum in improvised Round Robin	% Difference between TQ	Average Waiting Time in Round Robin	Average Waiting Time in Improvised Round Robin	% Improvement in Avg Waiting Time	Average Turnaround Time in Round Robin	Average Turnaround Time in Improvised Round Robin	% Improvement in Avg Turnaround Time	Context Switching in Round Robin	Context Switching in Improvised Round Robin	% Improvement in Context Switching
1	3	5 1 2	3	4	75.00%	3.33	1.33	60.06 %	6.00	4.00	33.33 %	4	4	0%
2	5	5 4 2 7 3	3	6	50.00%	11.00	6.00	45.45 %	15.20	10.20	32.89 %	9	6	33.33 %
3	6	15 18 10 6 25 30	12	24	50.00%	52.00	33.33	35.90 %	69.33	50.66	26.9 %	12	8	33.33 %
4	8	10 6 25 30 45 20 18	20	30	66.67%	80.75	49.25	39.00 %	102.00	70.50	30.88 %	12	9	25.00 %

From Table 3 we can conclude that:

- If the % change in Time quantum is greater than 35% but less than 45%, then there is great improvement.
- If the % change in Time quantum is greater than 45% but less than 70%, then there is less improvement.
- If the % change in Time quantum is greater than 70% then there is no improvement.

H. Executing Round Robin Algorithm on Ubuntu using Virtual Box

```
DARSHAN-UBUNTU$gcc -o RRSwithoutArrival RRSwithoutArrival.c
DARSHAN-UBUNTU$./RRSwithoutArrival
Enter Total Process:     5
Enter Burst Time for Process Process Number 1 :5
Enter Burst Time for Process Process Number 2 :4
Enter Burst Time for Process Process Number 3 :2
Enter Burst Time for Process Process Number 4 :7
Enter Burst Time for Process Process Number 5 :3
Enter Time Quantum:     3

Process |Turnaround Time|Waiting Time

P[3]    |       8       |       6
P[5]    |       14      |       11
P[1]    |       16      |       11
P[2]    |       17      |       13
P[4]    |       21      |       14

Average Waiting Time= 11.000000
Average Turnaround Time = 15.200000DARSHAN-UBUNTU$
```

Figure. 7: Output of Round Robin Algorithm on Ubuntu

In Figure. 7, we have printed the output of Round Robin Algorithm on Ubuntu to just make sure that our code works

on Ubuntu environment too.

I. Executing Improvised version of Round Robin Algorithm

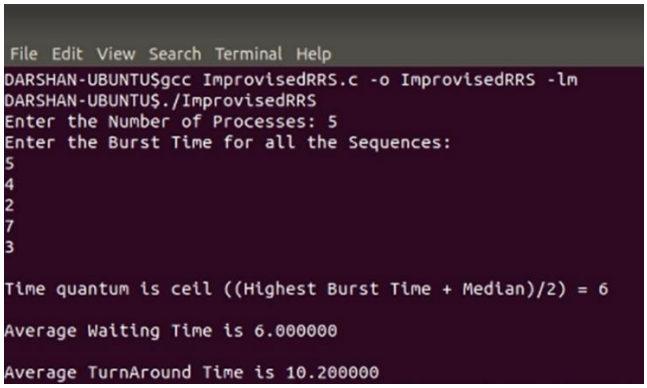

Figure 8: Output of Improvised version of Round Robin Algorithm on Ubuntu

In Figure. 8, we have printed the output of Improvised version of Round Robin Algorithm on Ubuntu to just make sure that our code works on Ubuntu environment too.

XI. CONCLUSION

Classical Round Robin Algorithm will be improvised using the above techniques. We used C-programming to improve the Round Robin Algorithm. We require the basics of Data Structures and Algorithm.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

ACKNOWLEDGMENT

This paper and the research behind it would not have been possible without the exceptional support of each other. Our enthusiasm, knowledge and exacting attention to detail have been an inspiration and kept our work on track to the final draft of this paper.

REFERENCES

[1] Rami J. Matarneh. Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of Now Running Processes, American J. of Applied Sciences 6(10): 1831-1837, 2009.

[2] H.S. Behera, Rakesh Mohanty, Debashree Nayak. A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis , International Journal of Computer Applications, Vol. 5, no. 5, August 2010.

[3] Helmy, T. and A. Dekdouk, 2007. Burst Round Robin as a Proportional-share Scheduling Algorithm, IEEEGCC, http://eprints.kfupm.edu.sa/1462.

[4] Pallab banerjee, probal banerjee, shweta sonali dhal. Comparative Performance Analysis of Average Max Round Robin Scheduling Algorithm (AMRR) using Dynamic Time Quantum with Round Robin Scheduling Algorithm using static Time Quantum, IJITEE,ISSN: 2278-3075, Volume-1, Issue-3, August 2012.

[5] BC Carlson. Algorithms involving arithmetic and geometric means, Amer. Math. Monthly 78 (1971), 496–505. MR 0283246.

[6] Emami Ale Agha, Ashkan & Jafarali Jassbi, Somayyeh. (2013). A New Method to Improve Round Robin Scheduling Algorithm with Quantum Time Based on Harmonic-Arithmetic Mean (HARM). International Journal of Information Technology and Computer Science. 5. 10.5815/ijitcs.2013.07.07.

[7] Mohan, Lavanya & Siva, Saravanan. (2013). Robust Quantum Based Low-power Switching Technique to improve System Performance. International Journal of Engineering and Technology. 5. 3634-3638.

[8] Helmy, Tarek & Dekdouk, Abdelkader. (2007). Burst round robin as a proportional-share scheduling algorithm.

[9] Manoj Kumar Srivastav, Sanjay Pandey, Indresh Gahoi and Neelesh Kumar Namdev. “Fair Priority Round Robin with Dynamic Time Quantum : FPRRDQ.” (2012).

[10] Behera, Himanshu Sekhar, P Jajnaseni, Dipanwita Thakur and Subasini Sahoo. “A New Proposed Two Processor Based CPU Scheduling Algorithm with Varying Time quantum for Real Time Systems.” Journal of Global Research in Computer Sciences 2 (2011): 81-87.

ABOUT THE AUTHORS

Aishna Gupta, Bachelor of Technology in Computer Science and Engineering with specialization in Bioinformatics in Vellore Institute of Technology, Vellore.

She has published 4 papers in International Journals

Patil Darshan Rajkumar, Bachelor of Technology in Computer Science and Engineering with specialization in Bioinformatics in Vellore Institute of Technology, Vellore.

He has published 3 review papers in International Journals and 1 review paper published in conference proceedings.