International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 03 | Mar-2016 www.irjet.net p-ISSN: 2395-0072

IMPROVED ROUND ROBIN SCHEDULING ALGORITHM WITH BEST POSSIBLE TIME QUANTUM AND COMPARISON AND ANALYSIS WITH THE RR ALGORITHM

Wasim Firuj Ahmed[1], Sahana Parvin Muquit[2]

[1]*B.Tech Student, Computer Science Engineering, VIT University , Vellore , Tamil Nadu ,India*
[2]*B.Tech Student, Electronics and Communication Engineering , Amity University , Noida , UP , India*

Abstract - *Round Robin (RR) Method is an old, simple scheduling algorithms in Operating system,mainly for time-sharing systems. Each process will be having equal priority and is having a time quantum after which the process is preempted. It performs in optimum way in timeshared systems as each and every process will be having an equal amount of static time quantum. Based on the Time quantum the effectiveness of RR algorithm is determined A comprehensive study and analysis of RR algorithm is done and discussed the new and improved version of Round Robin by assigning the processor to processes with shortest remaining burst in round robin manner using the best possible time quantum . Computation of the quantum is done by median and highest burst time. The experimental analysis also shows that this algorithm performs better than RR algorithm by reducing number of context switches, reducing average wait time and also the average turnaround time.*

Key Words: Scheduling Algorithm, Round Robin, Context switch, Waiting time, Turnaround time.

1.INTRODUCTION

The Central Processing Unit (CPU) should be utilized efficiently as it is the core part of Computers. For this reason CPU scheduling is very necessary. CPU Scheduling is a important concept in Operating System. Sharing of computer resources between multiple processes is called scheduling.

A process is an instance of a program running in a computer. 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 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 to the time it is completed is the turnaround time. When a process waits in the ready queue that time is the waiting time. The number of times CPU gets switched from a process to another one is called context switching . The optimal and the best scheduling algorithm will have less waiting time, less turnaround time and less number of context switches.

2. LITERATURE REVIEW

Many CPU scheduling algorithms are in use like First Come First served scheduling (FCFS), shortest job First scheduling (SJF), Priority Scheduling etc. The scheduling algorithms are: -

1. FCFS (First Come, First Serve) CPU Scheduling: In this scheduling the process that request the CPU first is allocated to CPU first. FCFS is simple and is similar to FIFO queue. Unfortunately, however, FCFS can have very long and high average waiting times, mainly if the first process to get there takes a long time.

2. SJF (Shortest Job First) CPU Scheduling: - In this scheduling the process having shortest CPU burst time will be firstly allocated to CPU. SJF can be proven to be the fastest scheduling algorithm, but it suffers from one important problem: the amount of time the next CPU burst is going to take is unknown. A more practical approach will be to predict the length of the next burst, based on some of the historical measurement of recent burst times for this process. *Exponential average* is one which is simple, fast, and relatively much more accurate method .

3. Priority Scheduling: - In this scheduling the process with high priority is allocated to CPU first. Priority scheduling can be either preemptive or non-preemptive. Priorities can be assigned in two ways : either it can be done internally or it can be done externally. Internal priorities are done by OS using many criterias like finding average burst time, ratio of

CPU to that of I/O activity, system resources use, and other factors that are available to the kernel.

Priority scheduling can have a big issue called as ***indefinite blocking***, or ***starvation***, in which a task with low priority may have to wait forever as there will always be some jobs around that have more priority. For that common solution will be aging implies as time progresses increase the priority of the process.

4. Round Robin Scheduling: - It is basically same as FCFS scheduling but pre-emption is also present for switching between processes. A Time Quantum (TQ) that is static is used in this CPU Scheduling. In this algorithm , a small unit of time called time quantum or time slice is assigned to each process . Once time quantum gets expired, the CPU switching is done from one to another process. So we can say performance mainly depend on Round Robin.

The various scheduling parameters are:

1. Context Switch: A context switch is basically storing and restoring context or state of a pre-empted process, so that at a later point of time , it can be started from same point once the execution is stopped. So the goal of CPU scheduling algorithms is to optimize only these switches.

2. Throughput: Throughput is defined as number of processes completed in a period of time. Context switching and Throughput are inversely proportional to each other.

3. CPU Utilization: This is the fraction of time when CPU is in use. Usually, to maximize the CPU utilization is the goal of the CPU scheduling

4. Turnaround Time: This is the total time which is required to spend to complete the whole process and amount of time it takes to execute that process.

5. Waiting Time: Waiting time is defined as the total amount of time a process that waits in ready queue.

6. Response Time: For responding to a particular system the amount of time used by the system.

The characteristic of good scheduling algorithm are:

Minimum context switches, Maximum CPU utilization, Maximum throughput,Minimum turnaround time , Minimum waiting time

3. EXISTING APPROACH

Round Robin (RR) Method is a old, simple scheduling algorithms in Operating system,mainly for time-sharing systems. Each process will be having equal priority and is having a time quantum and after that the process will be under preemption. The Operating Systems using RR, will first take the first process from the ready queue, will set a timer for interrupting after one time quantum and gives the processor to that process. Now it will compare processor burst time and time quantum ,If the processor burst time is smaller than that of time quantum, then either it will release the processor voluntarily, or it will terminate by issuing an I/O request. The OS starts with the next process which will be in the ready queue. On the other hand, if the processor burst time of a process exceeds the time quantum, then the timer will go off after there is an expiry of one Time Quantum, and it interrupts (preempts) the current process and puts its PCB to the end of the ready queue.

For example, Suppose there are 5 different processes as given below:

Table 1

PROCESSES	BURST TIME
P1	12
P2	15
P3	23
P4	37
P5	21

Let us Suppose for the above processes ,the time quantum to be 10 ns then the Gantt Chart for Round robin scheduling is:-

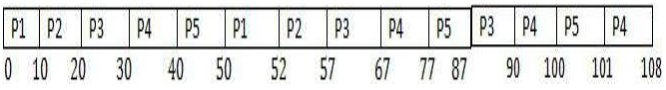

Chart:1 Gantt Chart

Total Wait time = [52-12] +[57-15] +[90-23] +[108-37] +[101-21] = 300

Now Average Waiting time will be: 300/5 = 60

Turn around time = 52 + 57 + 90 + 108 + 101 =408

Average turn around time = 408/5 =81

The main disadvantage/drawback in this RR is that Average turn around time and Average Waiting time is more and mainly context switching is too much which makes the algorithm very inefficient.

2. PROPOSED MODEL

2.1 Approach of proposed method:

In order to get rid of the drawbacks faced in previous round robin scheduling algorithm , a new method is proposed to find the best possible time quantum and make the algorithm an efficient one.

The proposed algorithm is as follows:

1. Start

2. Sort all the processes in increasing order those which are present in ready queue.

3. While (ready queue! = NULL)

Find TQ, where TQ Ceil (sqrt (median * highest Burst time))

4. Assign TQ to process

5. go to step 4 if (i<n)

6. Now update the counter n and go to step 2 , If a new process is arrived.

End of while

7. Now calculate the Average waiting time, average turnaround time and total Number of context switches required for the process.

8. End

2.2 Flowchart:

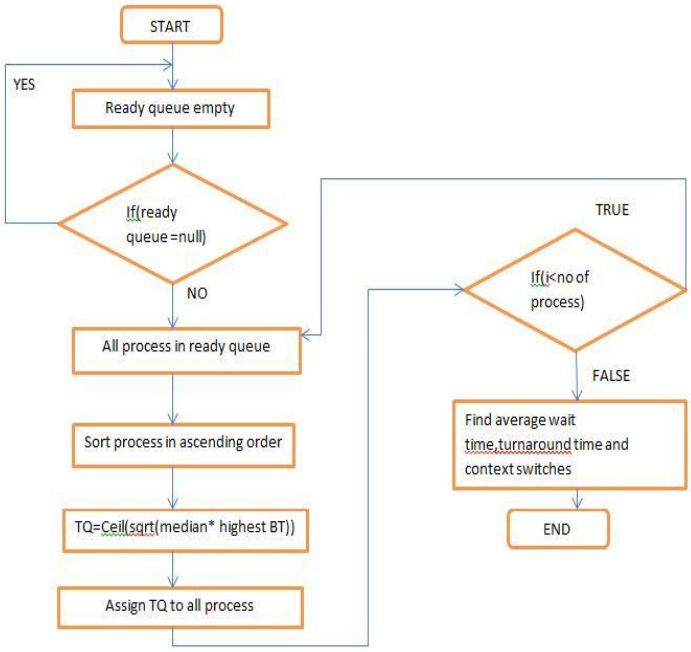

Chart 1: Flow chart for the algorithm

Illustration--Let us assume five processes with their burst time as given below:

Process	Burst time
P1	13
P2	35
P3	46
P4	63
P5	97

Table 2: Processes burst time chart

Now, as per the algorithm Time Quantum is calculated as follows

Time Quantum (TQ) = Ceil (sqrt (median * highest Burst time))

TQ = Ceil (sqrt(46 * 97)) = 67

So the Gantt chart is:

A. P1	B. P2	C. P3	D. P4	E. P5	F. P5

0 13 48 94 157 224 254

Chart 2: Gantt Chart

Here, Number of Context Switches = 5

Average Waiting Time = (0+13+48+94+157) / 5 = 62.4

Average Turn around Time = (13+48+94+157+254) / 5 = 113.2

Now if we implement the same illustration by round robin algorithm then in this manner time quantum is 25 in above case and hence context switch comes out to be 11 and average waiting time also becomes 97.4 and average turnaround time comes out to be 148.2

3. Simulation and screenshots:

ospaper.cpp

```
#include<iostream>
#include<math.h>
using namespace std;
int st[10];
int get_tq(int b[],int s)
{
int i,j,maxbt,tmp,hbt,median;
float k,l,m;
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));
}
```

Fig 1: Screenshot of code

```
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;

cout<<"Enter the no  of processes:";
cin>>n;
cout<<" burst time for all sequences:";
for(i=0;i<n;i++)
{
cin>>bt[i];
st[i]=bt[i];
}
tq=get_tq(st,n);
cout<<"\ntime quantum is  ceil((highestbt+Median)/2) = "<<tq;
while(1)
{
for(i=0,count=0;i<n;i++)
{
temp=tq;
if(st[i]==0)
{
count++;
```

Fig 2: Screenshot of code

```
continue;
}
if(st[i]>tq)
st[i]=st[i]-tq;
else
if(st[i]>=0)
{
temp=st[i];
st[i]=0;
}
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("Process_no\t Burst time\t Wait time\t Turn around time\t");
//for(i=0;i<n;i++)
//printf("%d\t %d\t %d\t %d\t",i+1,bt[i],wt[i],tat[i]);
cout<<"\nAvg waiting time is  "<<awt<<endl;
cout<<"\nAvg turn around time is"<<atat;

}
```

Fig 3: Screenshot of code

```
D:\hackerrank\codesns\ospaper.exe
Enter the no  of processes:5
 burst time for all sequences:13
35
46
63
97

time quantum is  ceil((highestbt+Median)/2) = 67
Avg waiting time is  62.4

Avg turn around time is113.2
--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Fig 4:Screenshot of output

4. RESULTS AND DISCUSSIONS

We observed from the above example, that in the newly proposed algorithm the number of context switches has been reduced from the general round robin algorithm. The average waiting time and turnaround time also have been reduced as compared to Round Robin algorithm which makes the algorithm an efficient one as we know the main goal of CPU scheduling is maximum CPU utilization, minimum waiting time , minimum turnaround time. Thus, we can say that if we calculate time quantum based on this algorithm then we can have a best possible time quantum for scheduling to make the CPU scheduling more efficient one.

5. CONCLUSIONS

From the above discussions , clearly it can be concluded that the proposed algorithm will perform better than the

static Round Robin Algorithm in terms of all measures like average waiting time, average turnaround time and number of context switches. For the future work, processes at different arrival of time can be considered for the proposed algorithm.

6. REFERENCES

1) http://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/
2) http://www.jgrcs.info/index.php/jgrcs/article/viewFile/110/109
3) http://www.tutorialspoint.com/operating_system/os_process_scheduling.htm
4) http://research.ijcaonline.org/volume98/number3/pxc3897235.pdf

7. BIOGRAPHIES

Wasim Firuj Ahmed: Pursuing my **B.tech** in Computer Science Engineering from VIT University, Vellore. At present I am in 3rd year of my Btech degree. I have lots of interest in Operating Systems, Embedded Systems and also in developing new Algorithms.

Sahana Parvin Muquit: I am pursuing **B.tech** from Amity University,Noida in Electronics and Communications. I have special interest in Embedded systems, Computer Architecture and also in Operating Systems.