International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.4, July-Aug 2012 pp-1548-1551 ISSN: 2249-6645

Shortest Time Quantum Scheduling Algorithm

ABDULLA SHAIK
Department Of IT, RTMNU,
Nuva College Of Engineering & Technology,
Nagpur. Maharashtra 440015, India

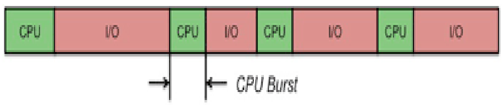
Abstract: Scheduling is the method by which threads, processes or data flows are given access to system resources. This is usually done to load balance a system effectively or achieve a target quality of service. Different CPU scheduling algorithms have different properties, and the choice of a particular algorithm may favor one class of processes over another. A typical scheduler is designed to select one or more primary performance criteria and rank them in order of importance. One problem in selecting a set of performance criteria is that they often conflict with each other. For example, increased processor utilization is usually achieved by increasing the number of active processes, but then response time decreases. It is desirable to maximize CPU utilization and throughput and to minimize turnaround time, waiting time, and response time. In most cases, we optimize the average measure. However, under some circumstances, it is desirable to optimize the minimum or maximum values rather than the average.

Round robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called time quantum. The performance of RR is sensitive to the time quantum selected. If the quantum is large enough, then RR reduces to the FCFS algorithm; If it is very small, then each process gets 1/nth of the processor time and share the CPU equally. The major problem in RR scheduling is that how the time quantum is decides? If any process require more time then the problem arrives that process must wait for long time to complete the execution. To overcome this problem in RR scheduling algorithm we come across with idea that the time quantum is decides based on the burst time needed for process. The percentage of CPU resource is decided based on burst time and the reaming process is similar to RR scheduling but difference is that allocation will be done based on newly calculated time quantum of process.

Keywords: Burst time, scheduling, time quantum, waiting time, execution time.

I. Introduction

If you look at any process, you'll notice that it spends some time executing instructions (computing) and then makes some I/O request, for example to read or write data to a file or to get input from a user. After that, it executes more instructions and then, again, waits on I/O. The period of computation between I/O requests is called the CPU burst.



Compute-intensive processes, conversely, spend more time running instructions and less time on I/O. Most interactive processes, in fact, spend the vast bulk of their existence doing nothing but waiting on data. If you consider Mac OS system, In which 44 processes running. This includes a few browser windows, a word processor, spreadsheet, several shell windows, Photoshop, iTunes, and various monitors and utilities. Most of the time, these processes collectively are using less than 3% of the CPU. This is not surprising since most of these programs are waiting for user input, a network message, or sleeping and waking up periodically to check some state.

The base idea in multiprogramming is that kept CPU always busy. For this we use scheduling. This is the method by which threads, processes or data flows are given access to system resources. This is usually done to load balance a system effectively or achieve a target quality of service. The need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking and multiplexing.

RR scheduling involves extensive overhead, especially with a small time unit. Balanced throughput between FCFS and SJF, shorter jobs are completed faster than in FCFS and longer processes are completed faster than in SJF. Poor average response time, waiting time is dependent on number of processes, and not average process length. Because of high waiting times, deadlines are rarely met in a pure RR system. Starvation can never occur, since no priority is given. Order of time unit allocation is based upon process arrival time, similar to FCFS. Round robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called time quantum. When a process is given the CPU, a timer is set for whatever value has been set for a time quantum. If the process finishes its burst before the time quantum timer expires, then it is swapped out of the CPU just like the normal FCFS algorithm. If the timer goes off first, then the process is swapped out of the CPU and moved to the back end of the ready queue. The ready queue is maintained as a circular queue, so when all processes have had a turn, then the scheduler gives the first process another turn, and so on. Most modern systems use time quantum between 10 and 100 milliseconds, and context switch times on the order of 10 microseconds, so the overhead is small relative to the time quantum. The major problem in RR scheduling is that how the time quantum is decides? If any process require more time then the problem arrives that process must wait for long time to complete the execution. To overcome this problem in RR scheduling algorithm we come across with idea that the time quantum is decides based on the burst time needed for process. The percentage of CPU resource is decided based on burst time and the reaming process is similar to RR scheduling but difference is that allocation will be done based on newly calculated time quantum of process. The processing is similar to SJF with RR scheduling model.

II. Shortest Time Quantum Scheduling Algorithm

The degree of multiprogramming is decided based on number of process/programs running simultaneously at a time. This can be improve are maintain by proper scheduling of multiple process in CPU. For that we are having different scheduling schemes one of them is FCFS the alternative of the FCFS is given in RR scheduling algorithm. Round robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called time quantum. When a process is given the CPU, a timer is set for whatever value has been set for a time quantum. . Most modern systems use time quantum between 10 and 100 milliseconds, and context switch times on the order of 10 microseconds, so the overhead is small relative to the time quantum. The major problem in RR scheduling is that how the time quantum is fixed for all the process. If the process is having burst time is low or high but the time quantum is same for all. As per RR algorithm low burst time process completes its execution first. To overcome this problem in RR scheduling algorithm we come across with idea that the time quantum is decides based on the burst time. It mean that based on time burst time quantum is decided so the time quantum is not fix for all process. The percentage of CPU resource is decided based on burst time and the reaming process is similar to RR scheduling but difference is that allocation will be done based on newly calculated time quantum of process.

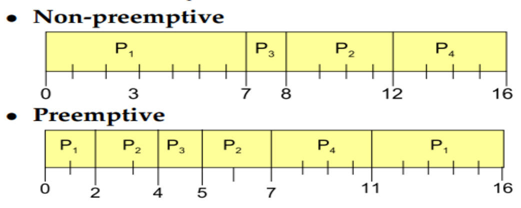
III. Processing

Round Robin scheduling is similar to FCFS scheduling, except that CPU bursts are assigned with limits called time quantum. When a process is given the CPU, a timer is set for whatever value has been set for a time quantum. Let first see how the shortest job scheduling is work. The Shortest job first scheduling algorithm gives minimum average waiting time for a given set of processes.

Examples

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- **Non-preemptive**
- **Preemptive**



The problems in SJF are: Only optimal if all jobs/process are available simultaneously. Usually run times are not known.

First-Come-First-Served algorithm is the simplest scheduling algorithm is the simplest scheduling algorithm. Processes are dispatched according to their arrival time on the ready queue. Being a no preemptive discipline, once a process has a CPU, it runs to completion.

Let consider example:

Process	Burst time
P1	53
P2	17
P3	68
P4	24

P1	P2	P3	P4

0 53 70 138 162

As per the FCFS algorithm the order of executing process is based on arrival. And once the process starts execution it is not suspended.

To overcome this problem in FCFS the Round Robin Scheduling algorithm is proposed. Each process gets a small unit of CPU time (time quantum). Then put back in Ready queue. The *RR Scheduling algorithm* is Preemptive. When interrupted, go to end of FIFO queue. Good for multi-user time-sharing - fast response time. Sometimes, OS just says "carry on". Sometimes don't even need 1 time quantum since process leaves voluntarily.

Example:

Process	Burst time
P1	53
P2	17
P3	68
P4	24

Say time quantum = 20:

P1	P2	P3	P4	P1	P3	P4	P1	P3	P3

0 20 37 57 77 97 117 121 134 154 162

Setting the time quantum size

If quantum too small, too much admin, not enough work.

As quantum goes to infinity, this goes to straight FIFO non-preemptive.

Want:

context switch << time slice

average CPU burst <= time slice

(some, but not too many > time slice)

Shortest Time Quantum Scheduling Algorithm:

By considering the above example the process P2 is having burst time =17 and p3 is having burst time =68. But both are having the time quantum 20ms for process P1 and P3. But the process P3 is having much more burst time then compare to P2. It need more time to execute quickly.

Example:

Process	Burst time
P1	53
P2	17
P3	68
P4	24

The total burst time needed for all the process The total execution time=P1+P2+P3+P4

=53+17+68+24

= 162

We fix minimum time quantum is 0.5 for increase the processing.

For each unit of CPU process need

= CPU %/ total execution time

=100/162

=0.617(>0.5)

The smallest Bust time require by process P2=17
Now we calculate the time quantum for each process
Time quantum of P1=51*0.617=32ms (ceil)
Time quantum of P2=17*0.617=11ms (ceil)
Time quantum of P3=68*0.617=42ms (ceil)
Time quantum of P4=24*0.617=15ms (ceil)

P1	P2	P3	P4	P1	P2	P3	P4

0 32 43 85 100 119 125 149 162

According to the Traditional RR scheduling based on time quantum. But here we are applying the way is shortest time quantum.

Now the actual way of processing is done as per ***shortest quantum***
So
Time quantum of P1=51*0.617=32ms (ceil)
Time quantum of P2=17*0.617=11ms (ceil)
Time quantum of P3=68*0.617=42ms (ceil)
Time quantum of P4=24*0.617=15ms (ceil)

P2	P4	P1	P3	P2	P4	P1	P3

0 11 26 58 100 106 115 138 162

Example:

Process	Burst time
P1	5
P2	10
P3	5

The total burst time needed for all the process The total execution time=P1+P2+P3
=5+10+5
= 20

For each unit of CPU process need
=20/100
=0.2(<0.5)
Here we fix minimum time quantum is 0.5 for increase the processing according to that the time quantum for each process
Time quantum of P1=5*0.5=3ms (ceil)
Time quantum of P2=10*0.5=5ms (ceil)
Time quantum of P3=5*0.5=3ms (ceil)

P1	P3	P2	P1	P3	P2

0 3 6 11 13 15 20

IV. Compression with other algorithms

The efficiency will be calculated based on average waiting time.
As per the examples consider

Process	Burst time
P1	53
P2	17
P3	68
P4	24

As per ***FCFS*** Scheduling Algorithm processing is done in order

P1	P2	P3	P4

0 53 70 138 162
Avg. waiting time= (0+53+70+138)/4 = 65.25ms

As per ***SJF*** Scheduling Algorithm processing is done in order

P2	P4	P1	P3

0 17 41 94 162
Avg. waiting time= (0+17+41+94)/4 = 38ms

As per ***RR*** Scheduling Algorithm processing is done in order
Say time quantum = 20:

P1	P2	P3	P4	P1	P3	P4	P1	P3	P3

0 20 37 57 77 97 117 121 134 154
162
Waiting time of p1= 121-(20+20)= 81ms
Waiting time of p2= 20ms
Waiting time of p3= 134-(20+20)= 94ms
Waiting time of p4= 117-20= 97ms
Avg. waiting time= (81+20+94+97)/4 = 72.25ms

As per **STQ** Scheduling Algorithm processing is done in order we fix minimum time quantum is 0.5 for increase the processing.
For each unit of CPU process need
= CPU %/ total execution time
=100/162
=0.617(>0.5)

Now we calculate the time quantum for each process
Time quantum of P1=51*0.617=32ms (ceil)
Time quantum of P2=17*0.617=11ms (ceil)
Time quantum of P3=68*0.617=42ms (ceil)
Time quantum of P4=24*0.617=15ms (ceil)

P2	P4	P1	P3	P2	P4	P1	P3

0 11 26 58 100 106 115 138 162
Waiting time of p1= 115-32= 83ms
Waiting time of p2= 100-11=89ms
Waiting time of p3= 138-42= 96ms
Waiting time of p4= 106-15= 91ms
Avg. waiting time= (83+89+96+91)/4 = 89.75ms

The average waiting time is much more compare to the other algorithms but the CPU resources are given to all the process besed on the brust time. So that the high brust time process like IO bound process are also complete the execution similar to the CPU bound process. And this algorithm never enters in to the CPU bound process. The major advantage is that the waiting time of individual are the average waiting time is much more nearer to the Round Rabin algorithm.

V. Conclusion

The degree of multiprogramming is decided based on number of process/programs running simultaneously at a time. This can be improve are maintain by proper scheduling of multiple process in CPU. For that we are having different scheduling schemes one of them is FCFS the alternative of the FCFS is given in RR scheduling algorithm. RR Scheduling algorithm is work based on fixed time quantum. Most modern systems use time quantum between 10 and 100 milliseconds, and context switch times on the order of 10 microseconds. The major problem in RR scheduling is that how the time quantum is fixed for all the process. If the process is having burst time is low or high but the time quantum is same for all. As per RR algorithm low burst time process completes its execution first. To overcome this problem in RR scheduling algorithm we come across with idea that the different process having different time quantum and minimum is 0.5% of CPU. Time quantum is decides based on the burst time. The percentage of CPU resource allocated on the basis of shortest time quantum first. The processing is similar to SJF with RR scheduling model. This combination will give the new kind of processing of programs.

Acknowledgments

I am grateful to My Brothers Dr. Khaleel Basha Shaik and Dr. Nazer Shaik , Dr Ajay Shenkar Singh and P. Manoj Kumar for giving guideline their valuable time to me. Finally last but not the least Mr. A. Hasheem Babu, Mr. Mohammad Shaik and Mr.John Saida other my friends for their insightful comments.

References

[1] Advanced Concepts In Operating Systems :Distributed Database And Multiprocessor Operating Systems, Singhal Mukesh, Shivaratri Niranjan G, Tata Mcgraw Hill Publishing Co. Ltd, 2008.
[2] Modern Operating Systems, Tanenbaum Andrew S, 3rd Edition, Pearson Education, 2008.
[3] Operating Systems Deitel H M , Deitel P J, Choffnesd R, 4th edition, Pearson Education, 2009
[4] Operating Systems: Internals And Design Principles, Stallings William, 6th Edition, Pearson Education, 2009.
[5] Design Of The Unix Operating System, Bach Maurice J, Phi Learning Pvt Ltd. 2008
[6] Minix Book Operating Systems: Design And Implementation, Tanenbaum Andrew S, Woodhull Albert S, 3rd Edition, Pearson Education, 2009
[7] Operating System Principles Silberschatz Abraham, Galvin Peter Baer, Gagne Greg, 7th Edition, Wiley India, 2008.
[8] Operating Systems, Godbole Achyut S, 2nd edition, Tata Mcgraw Hill Publishing Co. Ltd. 2009
[9] Operating Systems : A Concept-Based Approach, Dhamdhere D M, 2nd edition, Tata Mcgraw Hill Publishing Co. Ltd. 2006.
[10] Operating Systens, Madnick Stuart E,Donovan John J, Tata Mcgraw Hill Publishing Co. Ltd. 2008
[11] Systems Programming And Operating Systems, Dhamdhere D M, 2nd edition, Tata Mcgraw Hill Publishing Co. Ltd. 2008
[12] Inside the Linux 2.6 Completely Fair Scheduler: Providing fair access to CPUs since 2.6.23, M. Tim Jones, IBM developerWorks, December 15, 2009
[13] Token-ordered LRU: an effective page replacement policy and its implementation in Linux systems,Los Alamos National Laboratory, Los Alamos, NM 87545, USA
[14] CFLRU: A Replacement Algorithm for Flash Memory,seon-yeong park,dawoon jung,CASES,October 23-35,2006
[15] A Comparison of Page Replacement Algorithms,Amit S. Chavan, Kartik R. Nayak, Keval D. Vora, Manish D. Purohit and Pramila M. Chawan,IACSIT International Journal of Engineering and Technology, Vol.3, No.2, April 2011
[16]Generalized page replacement algorithms in a relational data base, R. G. Casey, I. Osman ,SIGFIDET '74 Proceedings of the 1974 ACM SIGFIDET ,Pages 101 - 124

Abdulla Shaik working as Assistant professor in Department of MCA, Nuva College of Engineering and Technology. Previously worked as lecturer in Joginapally B R Engineering College, Hyderabad. He completed his MCA in 2008, M.Tech(CSE) in 2010 from Acharya Nagarjuna university, Guntur with distinction. His interested in research he has publish Five international papers in area of software engineering, Operating Systems, Data Structures and the Software Testing Methodologies. And he is contributed as assistant for Prof. Dr. Ajay senkar Sing and P.Manoj Kumar to complete their research papers.