ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 3, March 2013

DETERMINING THE VARIABLE QUANTUM TIME (VQT) IN ROUND ROBIN AND IT'S IMPORTANCE OVER AVERAGE QUANTUM TIME METHOD

Yashasvini Sharma[1]

Abstract— **The process scheduling, is one of the most important tasks of the operating system. One of the most common scheduling algorithms used by the most operating systems is the Round Robin method in which, the ready processes waiting in ready queue, seize the processor for a short period of time known as the quantum (or time slice) circularly. The present work is a theory to find best quantum time for any process. Actually it is a work in which the method is providing a tool to control multiprogramming rate. Generally average is thought to be as the best option to find quantum time, that's true but in this paper use of average quantum time is in a different form. This method gives an analyst, an appropriate choice for declaring multiprogramming rate. This theory will give freedom to person to declare his best multiprogramming rate i.e. he can increase or decrease number of processes executing in given period of time , can change value of time slicing and thus can increase or decrease waiting time of processes, and can change quantum time in accordance to his needs, which will always gives a controlled execution as the value of the quantum parameter is selected in 10-100 milliseconds range.**

Index Terms— ***Time Quantum, Round Robin, Burst time, Arrival time, FCFS***

I. INTRODUCTION

The operating system (OS), as the most important program needed for starting up and using the hardware, has different managing tasks each of which, is performed by one of the management units of the OS. The process management, as one of these management units, allocates the processor to the processes using several allocating algorithms.

One of the most common algorithms in processor allocation is the Round Robin (RR) algorithm in which, the ready processes waiting in ready queue, are dispatched sequentially and allocate the processor for certain period of time known as time quantum (q) or time slice. If a process is finished during its time quantum, releases the processor, otherwise the processor is pre-empted by the OS and is allocated to the next ready process waiting in front of the ready queue and the current process will be moved toe the end of the this queue. The algorithm continues until all the processes are terminated, or the system is switched off. This method is known as best dispatching algorithm in time sharing systems.

The most important issue in RR algorithm, which highly affects its efficiency, is determining the amount of the time quantum. Assigning very small and very large values for this parameter, will lead to performance decrease due to increasing context switching overhead, and degradation of the RR method to FCFS algorithm consequently. Actually the value of the quantum parameter is selected in 10-100 milliseconds (ms) range.

So in round robin we have to find a suitable quantum time for processes, so that they can execute showing good multiprogramming rate. As explained above the value of the quantum parameter is selected in 10-100 milliseconds range, this means any burst time less than 10ms will execute in just one stroke while burst time greater than 100ms will definitely get chopped, otherwise a single process will take much larger time to get solved.

But a problem is there- that suppose 5 processes arrive at a same time having different burst time, now a question arise how we will decide quantum time for all these, the best option is finding average of all burst time and the answer of it will be our quantum time for all 5 processes, this way really is good in numerous aspects but a problem is here that is starvation means those process that come after these 5 processes will have to wait for a long which is explained later. So this theory basically is for having control over multiprogramming and uniprogramming in general and reducing the waiting time for the other processes.

This theory directly shows its importance over average quantum time selection. When we use average quantum time selection we get an average value as quantum for all processes, here the multiprogramming rate depends upon the value of average quantum, which we find.

Following example shows that

Let there are 5 processes P1, P2, P3, P4, P5

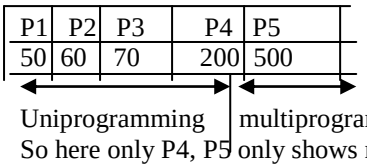
And let arrival time is same

Process name	Burst Time (ms)
P1	50
P2	60
P3	70
P4	200
P5	500

Average is 176

We can take average as 100 ms.

Now following diagram make it clear that how this average decides the multiprogramming rate



So here only P4, P5 only shows multiprogramming but P1, P2, P3 will show uniprogramming.

IR. Yashasvini Sharma**,** *BE Student of 6th Semester, Computer Science, Gyan Ganga Institute of Technology and Sciences Jabalpur, Madhya Pradesh, India*

All Rights Reserved © 2013 IJSETR

Thus first problem is
The rate of multiprogramming cannot be increased or decreased up to a level.
Dependency on upcoming values of burst time is indispensable over here, but in this new theory (VQT) if we ignore the selected value of the quantum parameter, that is 10-100 milliseconds range means the quantum value could be anything then we could find a perfect number (say Y) such that number of turns for executing all the process would always less than Y. In actual practice we can't ignore this, so we will get a perfect number (say Y) only for process whose burst time and remaining burst time is greater than 10 and less than 100ms.
To overcome these problems, a new way of using average quantum time is explained here.

II. Literature Review

2.1 Terminology

In the operating system design the issue that is considered in this research has an important role. This section is containing, the more frequently used terms in this literature. Some of these terms, are used as the efficiency criteria by some researchers.

- Process: The program which is loaded into memory to be executed.
- Ready Queue: A waiting list holding ready processes according to incoming order. These processes are waiting for dispatch.
- Scheduling: Making decision about allocating policies of the recourses to the processes.
- Burst time: A time period which a process needs for completion.
- Context Switching: Switching the processor from the current process to next ready includes storing and retrieving the values of flags and some important registers.
- Waiting time: A time period which a process waits to allocate a resource in the system.
- Throughput: The number of completed(finished) jobs in a specific time period

2.2 Goals of Round Robin Scheduling and scope of VQT

The main goal criteria can be listed as below:-

- Acceptable response time.
- Performing the task (process) during the user pre-defined time period.
- Increasing the CPU utilization.
- Increasing the utilization of other system resources.
- Decreasing overall overhead.
- Decreasing the user waiting time.
- Decreasing turnaround time.
- Increasing the system throughput

Obviously there may be numerous way of achieving it, but for different person some of these factors may have higher priority that other in accordance to their need. For example one person wants increase in CPU utilization, so that waiting time per process decrease; obviously he wants a good multiprogramming rate. A second person may cares for reduced overall overhead and thus keeping its priority more than that of waiting or utilization. It means every body surely needs all the above to happen at same time but may have their different opinion about their magnitude. This is very simple to explain by one more example say media player-One person loves to hear slow music and second fast and third normal, each of them is interested in same music but magnitude of speed may vary as all wants to hear best music, for first slow music is best, for second fast and third normal, so what does a media player provider will do! Simply he will provide a navigating or sliding tool that will provide music in all speed and user will just select or scroll or whatever, according to his needs. It may difficult to find its importance for user as very few of them are interested in concepts of operating systems but researchers, analyst and students of this field will surely get benefited by this.

2.3 Previous Works

This is one of those fields on which hundreds of researches have been done and still it is one of the favorite researching fields for researchers-as again for different researchers the criteria of defining a good quantum time is different.
Shahram Saeidi and Hakimeh Alemi Baktash [1] proposed paper; a non-linear programming mathematical model is developed to determine the optimum value of the time quantum, in order to minimize the average waiting time of the processes.
Hiranwal & Roy [2] proposed a priority driven scheduling algorithm based on the burst time of the processes. They showed that the use of this method increases the performance and stability of the operating system and supports building a self-adaptive OS.
Albielmona [6] did an overall review over many scheduling algorithms. Proportional share scheduling algorithm proposed in [7] combines the small overhead of the RR method by protecting the short processes. The capability of re-adjusting the weights, enables the algorithm to have a more fair behaviour.All upon this number of traditional way of finding quantum time are FCFS, SJF, Priority, average etc based quantum time scheduling.

All Rights Reserved © 2013 IJSETR

III. The Proposed Method

3.1 Assumptions

All the below are explained in programming language terms (C++)

Firstly we will have 4 structures each structure will have one char and one int (integer) type of data type. We will make their arrays. The 4 structures will be S1, S2, S3, and S4 respectively.

The starting of accepting process starts from S2. Arrays will be of 1-D and it can be of any number of subscripts (blocks). This means we can select the number of blocks in the arrays, more the number of blocks, and more the multiprogramming rate.

S2 is used for holding the pattern of incoming process. S3 is used for ordering them in a particular manner and finding the quantum. S4 is used for again arranging all processes in their correct manner, but instead of holding burst time in int data type, it will store processes quantum time, which will used to execute process in S2.In between this, other processes will start entering in S1, which after execution of all processes in S2, will be transferred to S2.

3.2 The Working Model explanation with an Example

Let's take an example and then compare both the ways of finding quantum time in round robin. First way is VQT and other is average quantum time.

Let's take total no of processes to be 5. Their burst time and arrival time are listed below-

Process	burst time (ms)	arrival time (ms)
P1	20	0
P2	41	4
P3	30	6
P4	60	6
P5	50	10

Firstly by the theory (VQT) 4 structures having 5 blocks each:-

1st step in S2

For S2

P1	P2	P3	P4	P5
0-4	4-6	6	6-10	10
16	39	30	56	50

→ name of processes arriving one by one executing and waiting for next process until all processes do not enter, but processes can't be greater than 5

↓ remaining burst time

1st step in S3

For S3

P4 P5 P2 P3 P1 → processes arranged in descending order, according to their remaining burst time

56 50 39 30 16 → burst time arranged in descending order

Av1=16+30+39+50+56/5=38 ms (Av=Average)
Av2=16+30+39+50/4=34 ms app
Av3=16+30+39/3=28 ms app
Av4=16+30/2=23 ms app
Av5=16/1=16ms

2nd step in S3

S3

Rewrite

P4	P5	P2	P3	P1
38	34	28	23	16

→ the average value, that we have find above

1st step in S4

S4

P1 P2 P3 P4 P5 → rearranging according to the process in S2

16 28 23 34 38 → accordingly average value (these are the variable quantum for each processes)

Allow above process with predefined quantum will go to scheduler one by one

1st step in S1

After executing processes, the processes that still need to get execute are arranging in structure S1 array

S1

All Rights Reserved © 2013 IJSETR

P2 P3 P4 P5 → upcoming processes are assembled in S1 according to their arrival time

11 7 22 12 → burst time (here remaining)

S1 passes all value to S2

2nd step in S2

S2

P2 P3 P4 P5 → Again the same process…

11 7 22 12

So that's how we can find suitable quantum for each process

IF WE WERE USING SIMPLE AVERAGE METHOD OF FINDING QUANTUM TIME IN ROUND ROBIN THEN
For above example we would get an average and a same quantum for every process that is:-
Av=38 ms
So in S2
For S2

P1	P2	P3	P4	P5
0-4	4-6	6	6-10	10
16	39	30	56	50

→ Name of processes arriving one by one

→ Executing and waiting for next process until all processes do not enter, but processes can't be greater than 5

↓ Remaining burst time

Up to this step every thing is same, but after this we get quantum time=38 ms
So it would be like

P1	P2	P3	P4	P5
0	1	0	18	12

→ remaining burst time

3.3 RESULTS

If we will use simple average method of finding quantum time in round robin then
Quantum time=38 ms then, only P4, P5, P2 will show multiprogramming and also, after 16+38+30+38+38+1=161ms P4 will get its turn back to execute rest of its 18ms.
If we will use VQT method of finding quantum time in round robin then
Quantum time is variable, so P4 will get its turn again after 16+28+23+34+38+10+7=156ms which is less than 161ms, this difference will be greater if we consider it for P2 as
If we will use simple average method of finding quantum time in round robin then
P2 will get its turn again after 16+38+30+38+38=160ms.

If we will use VQT method of finding quantum time in round robin then
P4 will get its turn again after 16+28+23+34+38=139ms, which is much less than 160ms.
This proves, for same number of blocks in array in both the theory VQT shows much better way of applying multiprogramming approach.
Secondly we will prove that by VQT we can control the multiprogramming rate, means we can increase or decrease the multiprogramming rate without the interference of which type of process are coming.
If we are provided with a due number of processes then for any value of burst time, the arrays of structures containing more number of blocks will either show greater or equal multiprogramming rate than that of the arrays of structures containing less number of blocks.
The reason is quite simple if we have 6 process and we are taking 3 blocks then 1(having the least burst time) out of 3 (if quantum is less than 100 ms) processes will surely totally ends up or execute completely and we will left with just 5 processes, in next turn we will again loose 1 process out of 3 process and left with 4 process in just 2 steps, so here when we traverse through 6 processes we solve 2 of these by uniprogramming approach. If we were using 6 blocks then 1 out of 6 processes would ends up, so here when we would traverse 6 processes then we could solve only one by uniprogramming approach.
This gives result that
The arrays of structures containing more number of blocks will show greater multiprogramming rate than that of the arrays of structures containing less number of blocks.
It may happen that all the n processes have burst time less than 10 ms then there is no need of multiprogramming, also if 1st process arrives at 0 and of burst time of 20 and second process comes at 20ms then again no need of doing multiprogramming. So here result is
The arrays of structures containing more number of blocks will show equal multiprogramming rate than that of the arrays of structures containing less number of blocks.

3.4 The Mathematical Model

Following example shows how it works but also how it increases multiprogramming rate and decreases the waiting time for other process with respect to average quantum time
To check how with increase in the number of blocks in the array the multiprogramming rate increases, we have a formula

All Rights Reserved © 2013 IJSETR

If we are given a definite number of processes and all their bursts time and remaining burst time can be any value then

Lets say there are P processes and number of blocks that are selected in an array is N then the number of at most number of turns (say T) taken by all P processes to complete their execution is given by-

T= ((P-(N-1))*N) + (N-1) + (N-2) +…..+ (N-N) (1)

That is number of turns are always equal to T.

More precisely

T=Q+R (2)

Where Q= ((P-(N-1))*N) (3)

R= (N-1) + (N-2) +…..+ (N-N) (4)

So if there are 5 processes (P) arriving at the same time and number of blocks are 4 (N) then number of turns=14 turns

If there are 5 processes (P) arriving at the same time and number of blocks are 5(N) then number of turns=15 turns

This shows the array having more blocks will show higher rate of multiprogramming, but if we increase number of block to 6 then the result will be same for 5 number of processes as was in array of 5 blocks.

This gives us conclusion if number of blocks is greater than number of processes then multiprogramming rate does not increase, but actually it will freeze.

So always we are going to get a perfect value, but it may happen that a process of very large burst time appears and the quantum time calculated by VQT is also too large, then it will surely cause starvation to other processes in queue. Also if the burst time of any process is very small and the quantum time calculated is even much smaller than burst time than it is foolishness to go for its quantum time execution

as we can prefer to execute this process as a whole otherwise it may cause context switching overhead. To avoid this value of the quantum parameter is selected in 10-100 milliseconds range.

Rule 1

If we are given a definite number of processes and some of their bursts time and remaining burst time are lesser than 10 and less than 100ms then the formula describes below-

Lets say there are P processes and no. of blocks that are selected in an array is N then the number of at most number of turns (say T) taken by processes to complete their execution is given by-

T= ((P-(N-1))*N) + (N-1) + (N-2) +…..+ (N-N) (1)

That is number of turns are always less than T.

More precisely

T=Q+R (2)

Where Q= ((P-(N-1))*N) (3)

R= (N-1) + (N-2) +…..+ (N-N) (4)

Rule 2

If we are given a definite number of processes and some of their bursts time and remaining burst time are greater than 10 and greater than 100ms then the formula describes below-

Lets say there are P processes and no. of blocks that are selected in an array is N then the number of at most number of turns (say T) taken by processes to complete their execution is given by-

T= ((P-(N-1))*N) + (N-1) + (N-2) +…..+ (N-N) (1)

That is number of turns are always greater than T.

More precisely

T=Q+R (2)

Where Q= ((P-(N-1))*N) (3)

R= (N-1) + (N-2) +…..+ (N-N) (4)

Rule 3

If we are given a definite number of processes and some of their bursts time and remaining burst time are less than 10 and greater than 100ms then the formula describes below-

Lets say there are P processes and no. of blocks that are selected in an array is N then the number of at most number of turns (say T) taken by processes to complete their execution is given by-

T= ((P-(N-1))*N) + (N-1) + (N-2) +…..+ (N-N) (1)

That is number of turns can be greater or smaller than T according to the value of burst time.

More precisely

T=Q+R (2)

Where Q= ((P-(N-1))*N) (3)

R= (N-1) + (N-2) +…..+ (N-N) (4)

From all above discussion the value that does not change is T, it remains fixed. For any value of burst time we will rotate around this value T.

So what is the significance of this T? The answer is quite simple, earlier any process with any burst time when come and execute we had no criteria to measure whether the process is having quantum greater than or less than 100millisecond (by taking rule 2). For checking point of view one can set this value to 200,300 or anything else and can get his answer. There may be other ways of finding all this but by this quantum time selection criteria one can also determine these results also.

IV. Conclusions

4.1 CONCLSION

1. By this rule we can conclude- this process finds a new way of rolling between multiprogramming and uniprogramming approach. If person wants to enjoy high rate of multiprogramming then he can consider large number of blocks, if he wants to reduce this approach he can consider lesser number of blocks in array of structures.

2. For same number of blocks in array in the theory, average quantum time and VQT, VQT shows much better way of applying multiprogramming approach.

3. We proved that by VQT we can control the multiprogramming rate, means we can increase or decrease the multiprogramming rate without the interference of which type of process are coming.

4. We can calculate a fix number T which will decide what burst time of processes that are under execution that will

All Rights Reserved © 2013 IJSETR

give benefit to researchers or programmers of operating system.

5. VQT usage gives a way to avoid starvation problem, which was a big problem for average quantum time theory.

6. Many other theory that deals with varying quantum may use complex calculation, but VQT calculation is very simple and much generalized as we are only changing the style of using average quantum time theory, which itself is one of the best theories of finding quantum in round robin approach.

REFERENCES

[1]Round-robin scheduling From Wikipedia, the free encyclopedia

[2] Shahram Saeidi, Hakimeh Alemi Baktash "Determining the Optimum Time Quantum Value in Round Robin Process Scheduling Method" , *I.J. Information Technology and Computer Science*, 2012

[3] Hiranwal & Roy, "Adaptive Round Robin scheduling using shortest burst approach, based on smart time slice", *International Journal of computer Science and Communication*, 2(2), 219-326, 2011

4] W. Stallings, *Operating Systems: Internals and Design Principles.* 6th Edition, Prentice Hall, 2008.

[5] M. Fahimi, *Operating Systems*, vol. 1, 1st Edition, Tehran: Jelve Publishing, 1992.

[6] R. Abielmona, "Scheduling Algorithmic Research", Department of Electrical and Computer Engineering, Ottawa-Carleton Institute, 2000.

[7] T. Helmy, and A. Dekdouk, "Burst Round Robin: As a Proportional-Share Scheduling Algorithm", *IEEE, Proceedings of the fourth IEEE-GCC Conference on towards Techno-Industrial Innovations*, pp. 424-428, 2007.

Author

Yashasvini Sharma, student of BE computer science branch, sixth semester, pursuing from Gyan Ganga Institute of Technology and Sciences, Jabalpur, Madhya Pradesh, India.

All Rights Reserved © 2013 IJSETR