International Journal of Engineering Research and General Science Volume 4, Issue 1, January-February, 2016
ISSN 2091-2730

Improvising Round – Robin Process Scheduling through Dynamic Time Quantum Estimation

Mr. Nischaykumar Hegde[1], Mr. Pramod Kumar P M[2]

Department of Computer Science, Vivekananda College of Engineering & Technology, Puttur (D.K), India,

Email:meetnischay@gmail.com

Department of Computer Science, Vivekananda College of Engineering & Technology, Puttur (D.K), India,

Email:pramodkumar24a@gmail.com

Abstract— Round Robin, considered as the most widely adopted CPU scheduling algorithm, undergoes severe problems directly related to quantum size. If time quantum chosen is too large, the response time of the processes is considered too high. On the other hand, if this quantum is too small, it increases the overhead of the CPU due to frequent context switches. In this paper, we propose a new algorithm based on a new approach called dynamic-time-quantum estimation intending to reduce the number of context switches among processes and offering optimal response time. The idea of this approach is to make the operating systems adjust the time quantum according to the second maximum (S_MAX) burst time of the set of waiting processes in the ready queue. A register is maintained for storing S_MAX, named as Second Max. Register (SMR). Based on the simulations and experiments, we show that the proposed algorithm solves the fixed time quantum problem and improvises the performance of Round Robin.

Keywords— *Round Robin, Time Quantum, Context Switch, Response Time, CPU Scheduling Algorithm, Operating Systems, S_MAX, SMR*

1. INTRODUCTION

Modern Operating Systems are moving towards multitasking environments which mainly depend on the CPU scheduling algorithm since CPU is the most effective or essential part of the computer. Round Robin is considered the most widely used scheduling algorithm in CPU scheduling [8, 9], also used for flow passing scheduling through a network device [1]. CPU Scheduling is an essential operating system task, which is the process of allocating the CPU to a specific process for a time slice. Scheduling requires careful attention to ensure fairness and avoid process starvation in the CPU. This allocation is carried out by software known as scheduler and dispatcher [8, 9]. Operating systems may feature up to 3 distinct types of a long-term scheduler (also known as an admission scheduler or high-level scheduler), a mid-term or medium term scheduler and a short-term scheduler (fig1). The dispatcher is the module that gives control of the CPU to the process selected by the short-term scheduler [8].

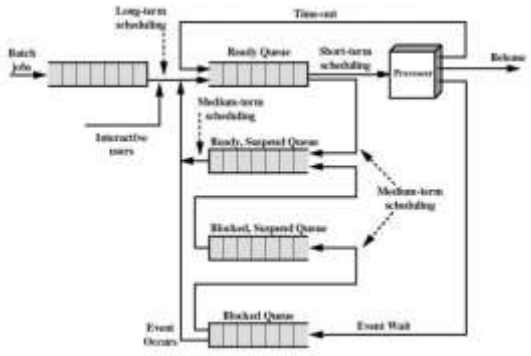

Figure 1.Queuing diagram for scheduling

There are many different scheduling algorithms which varies in efficiency according to the holding environments, which means what we consider a good scheduling algorithm in some cases which is not so in others, and vice versa. The Criteria for a good scheduling algorithm depends, among others, on the following measures [8]:

-Fairness: all processes get fair share of the CPU,

-Efficiency: keep CPU busy 100% of time,

-Response Time: minimize response time,

-Turnaround Time: minimize the time batch users must wait for output,

-Throughput: maximize number of jobs per hour.

Moreover, we should distinguish between the two schemes of scheduling: preemptive and non preemptive algorithms. Preemptive algorithms are those where the burst time of a process being in execution is preempted when a higher priority process arrives. Non preemptive algorithms are used where the process runs to complete its burst time even a higher priority process arrives during its execution time. First-Come-First-Served (FCFS) [8, 9] is the simplest scheduling algorithm, it simply queues processes in the order that they arrive in the ready queue. Processes are dispatched according to their arrival time on the ready queue. Being a non preemptive discipline, once a process has a CPU, it runs to completion. The FCFS scheduling is fair in the formal sense or human sense of fairness but it is unfair in the sense that long jobs make short jobs wait and unimportant jobs make important jobs wait [8, 9]. Shortest Job First (SJF) [8, 9] is the strategy of arranging processes with the least estimated processing time remaining to be next in the queue. It works under the two schemes (preemptive and non-preemptive). It's provably optimal since it minimizes the average turnaround time and the average waiting time. The main problem with this discipline is the necessity of the previous knowledge about the time required for a process to complete. Also, it undergoes a starvation issue especially in a busy system with many small processes being run [8, 9, 10]. Round Robin (RR) [8, 9]which is the main concern of this research is one of the oldest, simplest and fairest and most widely used scheduling algorithms, designed especially for time-sharing systems. It's designed to give a better response but the worst turnaround and waiting time due to the fixed time quantum concept. The scheduler assigns a fixed time unit (quantum) per process usually 10-100 milliseconds, and cycles through them. RR is similar to FCFS except that preemption is added to switch between processes [2, 3, 8]. In this paper, we propose a new algorithm to solve the constant time quantum problem. The algorithm is based on dynamic time quantum approach where the system adjusts the time quantum according to the second maximum (S_MAX) burst time of processes found in the ready queue. The second section states some of previous works done in this field. Section III describes the proposed method in details. Section IV discusses the simulation done in this method, before concluding this paper in the last section.

2. PREVIOUS WORKS

Round Robin becomes one of the most widely used scheduling disciplines despite of its severe problem which rose due to the concept of a fixed pre-determined time quantum [2, 3, 4, 5, 6, and 7]. Since RR is used in almost every operating system (windows, BSD, UNIX and UNIX based etc…), many researchers have tried to fill this gap, but still much less than needs. Matarneh [2] found that an optimal time quantum could be calculated by the median of burst times for the set of processes in ready queue, unless if this median is less than 25ms. In such case, the quantum value must be modified to 25ms to avoid the overhead of context switch time [2]. David B. Stewart and Pradeep K. Khosla proposed the maximum-urgency-first algorithm, which can be used to predictably schedule dynamically changing systems [11].Other works [7], have also used the median approach, and have obtained good results. Helmy et al. [3] propose a new weighting technique for Round-Robin CPU scheduling algorithm, as an attempt to combine the low scheduling overhead of round robin algorithms and favor short jobs. Higher process weights means relatively higher time quantum; shorter jobs will be given more time, so that they will be removed earlier from the ready queue [3]. Other works have used mathematical

approaches, giving new procedures using mathematical theorems [4]. Mohanty and others also developed other algorithms in order to improve the scheduling algorithms performance [5, 6, 7]. One of them is constructed as a combination of priority algorithm and RR [5] while the other algorithm is much similar to a combination between SJF and RR [6].

3. PROPOSED METHODOLOGY

In this paper, we present a solution to the fixed time quantum problem of RR by making the operating system adjust the time quantum according to the second maximum (S_MAX) burst time among the set of processes in the ready queue. When operating system boots for the first time, it begins with time quantum equals to the burst time of first dispatched process, which is subject to change after the end of the first time quantum. So, we assume that the system will immediately take advantage of this method. The determined time quantum represents real and optimal value because it is based on real burst time unlike the other methods, which depends on fixed time quantum value. Our methodology performs an estimation to decide the value for the subsequent time quantum i.e., when a new process is loaded into the ready queue in order to be executed, the operating system identifies the second maximum (S_MAX) of the burst times among the processes in the ready queue including the new arrival process. This method needs a register to store the identified second maximum (S_MAX) burst time, we have named it as SMR i.e., Second Maximum Register. When a process in execution finishes its time slice or its burst time, the ready queue and the registers will be updated to store the new data values.

The algorithm described in the previous section can be formally described by pseudo code and flow chart like follows:

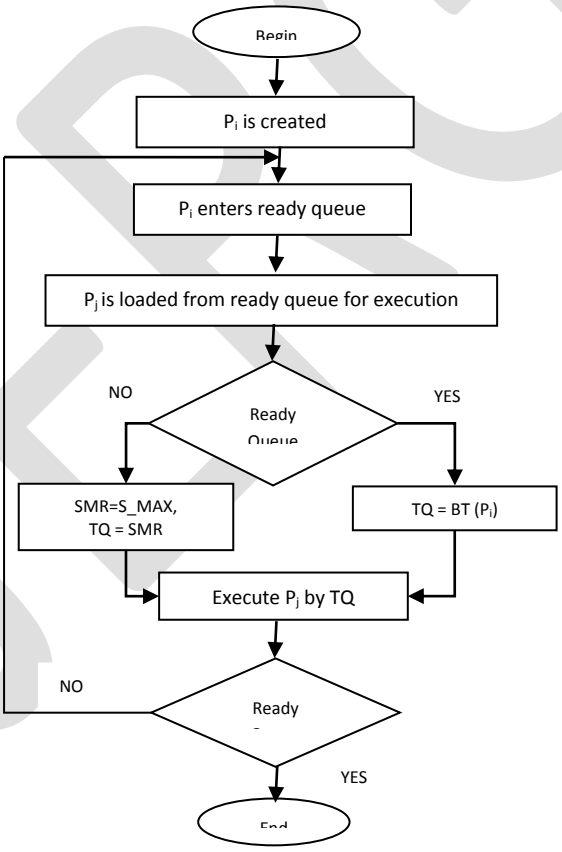

Figure 2.Flowchart of the proposed work

Pseudo code:

```
New process P arrives
P Enters ready queue
Update SMR with its BT
Process P is loaded from ready queue into the CPU to be executed
IF (Ready Queue is Empty)
```

```
S_MAX <- BT (P)
SMR <- S_MAX
TQ <- SMR
End if
IF (Ready Queue is not empty)
Identify second maximum (S_MAX) burst time
SMR <- S_MAX
TQ <- SMR
End if
CPU executes P by TQ time
IF (P is terminated)
Update SMR
End if
IF (P is not terminated)
Return P to the ready queue with its updated burst time
Update SMR
End if
```

SIMULATIONS

In order to validate our algorithm over the existing Round Robin, we have used PSSAV 201007282301, a simulator built using Java Net Beans, presents the user data and solutions after fetching in a graphical representation which is not found in most other languages. This simulator calculates the average waiting time and the average turnaround time of the whole system consisting of N processes according to the traditional Round Robin algorithm. We interrupt the process to edit TQ according to the proposed methodology. Following section deals with the results and observations.

4. RESULTS AND OBSERVATIONS

As a result of the simulation and hand solved examples we've reached to a conclusion that our proposed methodology could improve the efficiency of Round Robin by changing the idea of fixed time quantum to dynamically estimated one.

To evaluate our proposed method, we have considered different scenarios with random burst, in fact the number of processes does not change the result because the algorithm works effectively even if it is used with a very large number of processes. For each case, we will compare the result of our developed method with the traditional Round Robin approach (with fixed TQ = 10ms) and with the method proposed. We have observed the performance of our proposed work considering 5 different cases along with randomly taken burst and arrival times. Values shown in the below tables and subsequent graphs would justify our work.

CASE 1: Consider all processes arrive at time 0, Burst times P1 = 2ms, P2 = 4ms, P3 =8ms, P4 =10ms.

	Round Robin	Proposed Work
AWT	8.5	5.5
ATAT	14.5	11.5
CS	3	3

CASE 2: Consider all processes arrive at time 0, Burst times P1 = 10ms, P2 = 20ms, P3 =30ms, P4 =40ms.

	Round Robin	Proposed Work
AWT	33.5	25.0
ATAT	58.5	50.0

CS	8	3

CASE 3: Consider all processes arrive at time 0, Burst times P1 = 5ms, P2 = 10ms, P3 =15ms, P4 =20ms.

	Round Robin	Proposed Work
AWT	13.5	12.5
ATAT	31.0	25.0
CS	8	3

CASE 4: Consider all processes arrive at different time as shown below,

Process	Arrival Time (ms)	Burst Time (ms)
P1	0	22
P2	1	44
P3	2	66
P4	3	88

	Round Robin	Proposed Work
AWT	103.5	53.25
ATAT	158.5	108.5
CS	13	3

CASE 5: Consider all processes arrive at different time as shown below,

Process	Arrival Time (ms)	Burst Time (ms)
P1	0	15
P2	0	25
P3	29	27
P4	29	29

	Round Robin	Proposed Work
AWT	28.5	13.5
ATAT	52.5	51.0

CS	11	5

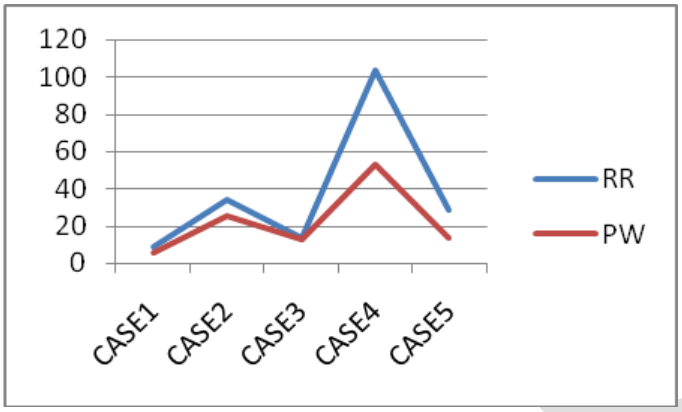

Figure 3 Performance comparison of the proposed work over RR w.r.t Average Waiting Time

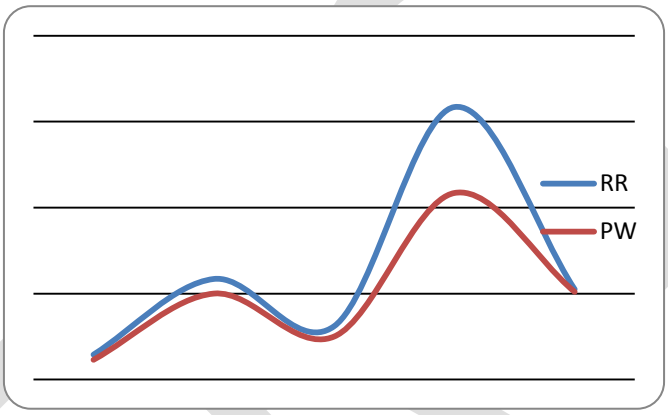

Figure 4 Performance comparison of the proposed work over RR w.r.t Average Turnaround Time

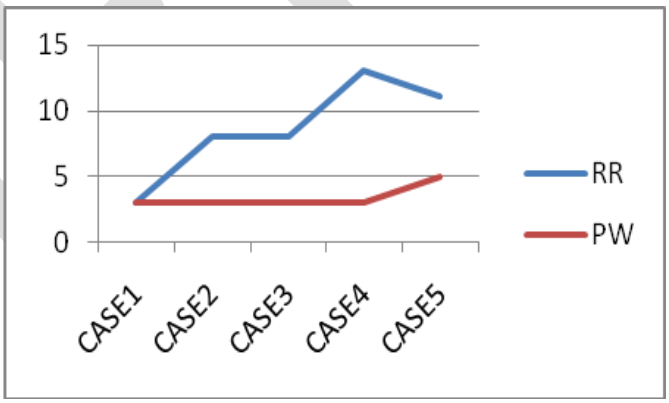

Figure 5 Performance comparison of the proposed work over RR w.r.t Context Switches

5. CONCLUSIONS

Time quantum is the bottleneck facing round robin algorithm and was more frequently asked question: What is the optimal time quantum to be used in Round Robin algorithm? In light of the effectiveness and the efficiency of the RR algorithm, this paper provides an answer to this question by using dynamic time quantum instead of fixed time quantum, where the operating system itself finds the optimal time quantum without user intervention. In this paper, we have discussed our proposed methodology that could be a simple step for a huge aim in obtaining an optimal scheduling algorithm. It will need much more efforts and researches to score a goal.

REFERENCES:

[1] Weiming Tong, Jing Zhao, "Quantum Varying Deficit Round Robin Scheduling Over Priority Queues", International Conference on Computational Intelligence and Security. pp. 252- 256, China, 2007.

[2] Rami J. Matarneh, "Self-Adjustment Time Quantum in Round Robin Algorithm Depending on Burst Time of the Now Running Processes", American Journal of Applied Sciences, Vol 6, No. 10, 2009.

[3] Tarek Helmy,Abdelkader Dekdouk, "Burst Round Robin as a Proportional-Share Scheduling Algorithm", In Proceedings of The fourth IEEE-GCC Conference on Towards Techno-Industrial Innovations, pp. 424-428, Bahrain, 2007.

[4] Samih M. Mostafa, S. Z. Rida, Safwat H. Hamad, "Finding Time Quantum Of Round Robin Cpu Scheduling Algorithm In General Computing Systems Using Integer Programming", International Journal of Research and Reviews in Applied Sciences (IJRRAS), Vol 5, Issue 1, 2010.

[5] Rakesh Mohanty, H. S. Beheram Khusbu Patwarim Monisha Dash, M. Lakshmi Prasanna , "Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems", (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.2, February 2011.

[6] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, "Design and Performance Evaluation of a New Proposed Shortest Remaining Burst Round Robin (SRBRR) Scheduling Algorithm", In Proceedings of International Symposium on Computer Engineering & Technology (ISCET), Vol 17, 2010.

[7] Rakesh Mohanty, H. S. Behera, Debashree Nayak, "A New Proposed Dynamic Quantum with Re-Adjusted Round Robin Scheduling Algorithm and Its Performance Analysis", International Journal of Computer Applications (0975 – 8887), Volume 5– No.5, August 2010.

[8] Silberschatz ,Galvin and Gagne, Operating systems concepts, 8th edition, Wiley, 2009.

[9] Lingyun Yang, Jennifer M. Schopf and Ian Foster, "Conservative Scheduling: Using predictive variance to improve scheduling decisions in Dynamic Environments", SuperComputing 2003, November 15-21, Phoenix, AZ, USA.

[10] C. Yaashuwanth and R. Ramesh, : A New Scheduling Algorithm for Real Time System, International Journal of Computer and Electrical Engineering (IJCEE), Vol. 2, No. 6, pp 1104-1106, December, 2010.

[11] David B. Stewart and Pradeep K. Khosla: Real-Time Scheduling of Dynamically Reconfigurable Systems, Proceedings of the IEEE International Conference on Systems Engineering, pp 139-142, August, 1991