J. of university of anbar for pure science : Vol.6:NO.2 : 2012

ISSN: 1991-8941

Improving Efficiency of Round Robin Scheduling Using Ascending Quantum And Minumim-Maxumum Burst Time

Ali Jbaeer Dawood
*** University of Anbar - College of Computer**

Abstract : Round Robin (RR) is a kind of process algorithms, where the time quantum is fixed along the processes execution. In the other hand it depending on the First Come First Serve (FCFS) algorithm. Also RR performs in timesharing system by given each process static Time Quantum (TQ). In this paper, The TQ studied to improve the efficiency of RR and performs the degrades with respect to Context Switching (CS), Average Wait Time (AWT) and Average Turned Around Time (ATAT) that an overhead on the system. Thus, the new approach was proposed to calculate the TQ, known as Ascending Quantum and Minumim-Maxumum Round Robin (AQMMRR). The processes were ascending with shortest remaining burst time and calculate the TQ from multiply the summation of minimum and maximum BT by (80) percentage. The experimental result shows that AQMMRR performs better than RR and comparing with other two related works.

Keywords-component: Round Robin; Time Quantum; Context Switching; Average Wait Time; Average Turned Around Time.

Introduction

Scheduling is central to operating system design. In case of multi-programmed operation system CPU scheduling plays a fundamental role by switching the CPU among various processes. The intention of an Operating system should allow process many as possible running at all times in order to maximize the CPU utilization. In a multi-programmed operating system a process is executed until it must wait for the completion of some I/O request. In this case the time has been used proficiently. A number of processes are kept in memory simultaneously and while one process occupies the CPU selected by the Operating. [1]
CPU scheduling algorithms decides which of the processes in the Ready Queue (RQ) is to be allocated to the CPU. There are many different CPU scheduling algorithms, out of those algorithms.

The processes are assigned to a processor are put in a queue called Ready Queue. CPU Utilization is the percentage of time that the processor is busy. It generally ranges from 0 to 100 percent. Throughput means how many processes are finished by the CPU with in a time period. The time interval between the submission of the process and time of the completion is the Turnaround time. Waiting time is the amount of time a process is waiting in the RQ, waiting in I/O and waiting in CPU. The number of times CPU switches from one process to another is called as the number of context switches. There are well known CPU scheduling algorithms that has been developed such as First Come First Serve (FCFS) algorithm, Shortest Job First (SJF) algorithm, Shortest Remaining Time Next (SRTN) algorithm, Round Robin (RR) algorithm and Priority Scheduling algorithm. RR and SRTN are preemptive in nature. RR is most suitable for time sharing systems. But its average output parameters (turn-around time, waiting time, etc.) are not feasible enough to be employed in real-time systems. [2]

RR is the oldest, simplest and most widely used proportional share scheduling algorithm [2, 3, and 4]. It's designed to give a better responsive 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 [1, 5, and 6].

A. Scheduling Criteria.

Many criteria have been suggested for comparing CPU scheduling algorithms. Which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following: [9]

- CPU Utilization. We want to keep the CPU as busy as possible.
- Throughput. If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called throughput. For long processes, this rate may be one process per hour; for short transactions, it may be 10 processes per second.
- Turnaround time. From the point of view of a particular process, the important criterion is how long it takes to execute that process. The interval from the time of submission of a process to the time of completion is the turnaround time. Turnaround time is the sum of the periods spent waiting to get into memory, waiting in the ready queue, executing on the CPU, and doing I/O.
- Waiting time. The CPU scheduling algorithm does not affect the amount of the time during which a process executes or does I/O; it affects only the amount of time that a process spends waiting in the ready queue. Waiting time is the sums of periods spend waiting in the ready queue.
- Response time. In an interactive system, turnaround time may not be the best criterion. Often, a process can produce some output fairly early and can continue computing new results while previous results are being output to the user. Thus, another measure is the time from the submission of a request until the first response is produced. This measure, called response time, is the time it takes to start responding, not the time it takes to output the response. The turnaround time is generally limited by the speed of the output device.

B. Motivation

In traditional RR the context switching is the number of process in each round, its high when comparing with other scheduling algorithms. In other hand the givers large averages waiting time and turnaround time. Thus motivates us to improving the traditional RR to overcome the above limitation.

C. Related works

In last few years many researcher studied RR to increase the performance of RR scheduling in different ways. This is done by CPU utilization, throughput, turnaround time, waiting time and number of context switching.

SARR algorithm [5] is based on a new approach called dynamic time quantum, in which time quantum is repeatedly adjusted according to the burst time of the running processes. In MMRR [2] time quantum is taken as the range of the CPU burst time of all the processes. The range of the processes is the difference between the largest (maximum) and smallest (minimum) values. The authors [10] calculate the time slice for the tasks based on their Priority and these tasks are arranged based on Priority execute in the main Processor with their individual time slices. Time slice calculation for this architecture using the rang, as shown below:

$$\text{Timeslice} = \frac{\text{Range (R) X Total no. of Process in the system (N)}}{\text{Priority (Pr) X total no. of Priority in the system (P)}} \quad \ldots.(2)$$

$$\text{Range} = \frac{\text{Maximum cpu burst + minimum cpu burst}}{2} \quad \ldots\ldots(3)$$

Dynamic Quantum with Re-adjusted Round Robin (DQRRR) [7] algorithm is based on a TQ, in which TQ is calculated as median of the existed set of processes. SRBRR algorithm [8] the time quantum is taken as the median of the increasingly sorted burst time of all the processes.

Proposed Approach

In our work, TQ calculating by ascending the TQ, sum the maximum and minimum CPU burst time and multiply the result by (80) percentage. The (80) percentage is chosen depending to two reasons: First, if the TQ calculated depending only on the summation the algorithm is become as the Short Job First (SJF). Second, the rule of thumb is that 80 percent of the CPU bursts should be shorter than the time quantum. Lastly, improving to the MMRR algorithm, where if the result of subtracting of the rang (MaximumBT-MinimumBT) is less than (25), the new TQ is (25), in this state this algorithm become as traditional RR [2], as shown in the bellow equation.

$$TQ = \begin{cases} M, \text{ if } \geq 25 \\ 25, \text{ if } M < 25 \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (1)$$

This led to the uniqueness of our approach.

A. Proposed Algorithm

When the processes are in the ready queue, the number of processes (n) and the arrival time are accepted as input:

Input: (n=number of processes, BT= burst time, Arrival Time)
Output: (Gantt Chart, AWT, ATAT and CS)

1. All the processes present in ready queue are sorted in ascending order.
2. While(Ready Queue!= NULL)
   MaxMin = MaximumBT + MinimumBT
   TQ_{new}= MaxMin * 0.8
3. //Assign TQ_{new} to (1 to n) process
   For i = 1 to n assign TQnew to all the available processes
4. Calculate the remaining burst time of the processes
5. Remove all processes from RQ if it's BT = 0 //BTPi=0
6. If new process is arrived add to the Ready Queue
7. // begin another round
   go to step 2
8. Draw the Gantt Chart, Calculate AWT, ATAT and CS
9. End

Figure 1. AQMMRR proposed algorithm

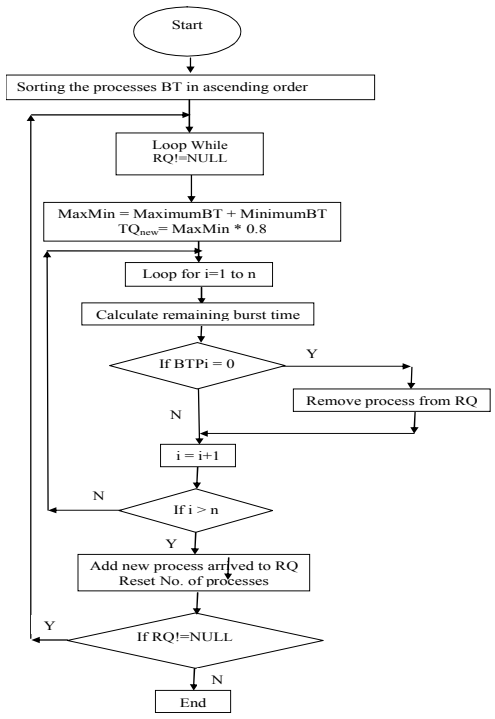

Figure 2. The flowchart of AQMMRR proposed algorithm

B. Illustration

Suppose five processes arriving at zero time, and CPU Burst Time comes as the following order (P1=28, P2=92, P3=40, P4=30, P5=10). First step arrange the BT in ascending order. The processes become in the following order (P5, P1, P4, P3 and P2). Calculate the TQ by summing the minimum BT with the maximum BT and multiply the summed by 0.8 (10+92)*0.8=81.6≈82). After first round, remove the processes (P1, P3, P4, P5), because it's remaining BT=0. In the second round only P2 has BT=(92-82)=10. Thus, the TQnew=10, in this case not need to calculate, because the TQnew=(10+10)*0.8=16, all most less than 10. Lastly, calculate AWT=44.8, ATAT=84.8 and CS=5.

Simulations and Result Analysis

The simulation of our proposed algorithm, the set of five processes was taken in two cases. The evaluation done by comparing the results of traditional RR, SRBRR and MMRR with result of our algorithm (AQMMRR). In traditional RR the fixed TQ=25.

Case 1: Suppose there are five processes arriving at time=0, TQ for RR is 25, with BT as shown in (table 1) below:

Table 1. Snap sheet for case 1

Process	Arrival time	Burse Time
P_1	0	13
P_2	0	35
P_3	0	40
P_4	0	63
P_5	0	97

Table 2 below shows the comparison result among RR, SRBRR, MMRR and our algorithm AQMMRR

Algorithm	TQ	AWT	ATAT	CS
RR	25	97.4	148.2	11
SRBRR	46, 34, 17	71.6	122.4	7
MMRR	84, 15	62.4	113.2	5
AQMMRR	88, 14	62.4	113.2	5

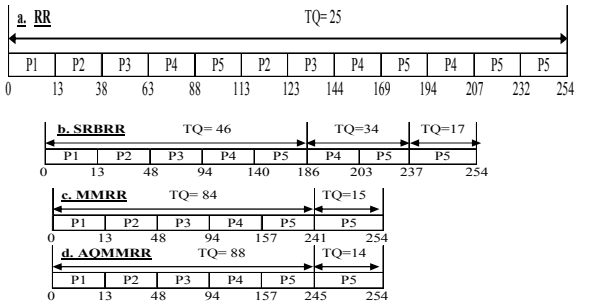

Figure 3. Gantt Chart for (a. RR b. SRBRR c. MMRR d. AQMMRR)

From table 2 shows MMRR and AQMMRR are the same results. When, the different between maximum and minimum is less than 25, the behavior of MMRR as in RR. But in our algorithm AQMM not affected and resisted to this consideration.

Case 2: Suppose there are five processes arriving at time=0, fixed TQ=25, with BT as shown in table 3 below:

Table 3. Snap sheet for case 1

Process	**Arrival time**	**Burse Time**
P_1	0	20
P_2	0	22
P_3	0	25
P_4	0	30
P_5	0	40

Table 4. Below shows the comparison result among RR, SRBRR, MMRR and our algorithm AQMMRR

Algorithm	TQ	AWT	ATAT	CS
RR	25	50.2	77.6	7
SRBRR	25, 10, 5	50.2	77.6	7
MMRR	25, 10, 5	50.2	77.6	7
AQMMRR	48	45.2	72.6	5

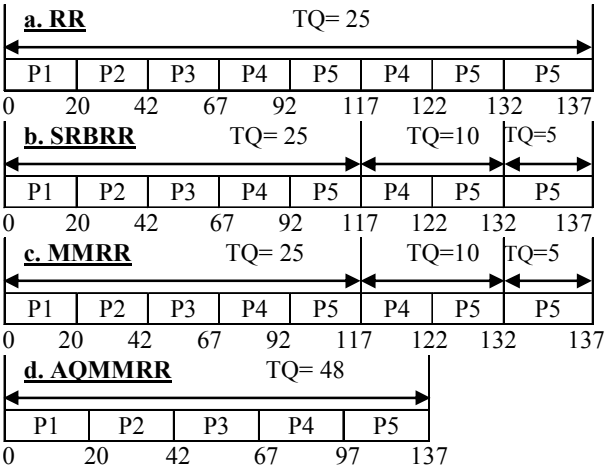

Figure 4. Gantt Chart for (a. RR b. SRBRR c. MMRR d. AQMMRR)

Conclusion

From the comparisons in above section, the results was concluded in tables 2 and 4 that our algorithm AQMMRR is better than the others, in terms of AWT, ATAT and CS.

REFERENCES

[1] Silberchatz, Galvin and Gagne, (2003) , "operating systems concepts",(6th edn, John Wiley and Sons).

[2] S. K. Panda, S. K. Bhoi, (January 2012), "An Effective Round Robin Algorithm using Min-Max Dispersion Measure". International Journal on Computer Science and Engineering (IJCSE) ISSN : 0975-3397 Vol. 4 No. 01.

[3] J. Nieh, C. Vaill and H. Zhong, (June 2001), "Virtual-Time Round-Robin: An O(1) Proportional Share Scheduler", Proceedings of the USENIX Annual Technical Conference, Boston, Massachusetts, USA, pp. 25-30.

[4] S. M. Mostafa, S. Z. Rida and S. H. Hamad, (October 2010), "Finding Time Quantum of Round Robin CPU Scheduling Algorithm in General Computing Systems using Integer Programming", IJRRAS 5 (1), pp.64-71.

[5] Rami J. Matarneh, (2009), "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.

[6] Tarek Helmy,Abdelkader Dekdouk, (2007), "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.

[7] H. S. Behera, R. Mohanty, D. Nayak, (August 2010), "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.

[8] Rakesh Mohanty, H. S. Behera, Khusbu Patwari, Monisha Dash, (2010), "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.

[9] R. K. Yadav, A. K. Mishra, N. Prakash and H. Sharma, (2010), "An Improved Round Robin Scheduling Algorithm for CPU scheduling", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 04, 1064-1066.

[10] Yaashuwanth .C IEEE Member, Dr.R. Ramesh, (March 2010), "Design of Real Time scheduler simulator and Development of Modified Round Robin architecture", International Journal of Computer Science and Network Security, VOL.10 No.3.

تحسين كفاءة خوارزمية الجدولة راوند روبن باستخدام الكم التصاعدي وادنى–اعلى وقت للتنفيذ

علي جبير داود
dralijd@yahoo.com

الخلاصة:

تعتبر خوارزمية راوند روبن من خوارزميات الجدولة التي تستخدم كما وقتيا ثابتا خلال مدة التنفيذ. وفي المقابل فهي تعتمد على خدمة من يأتي اولا. وهي تؤدي بشكل جيد في انظمة التشارك في الوقت من خلال اعطاء العمليات كما وقتيا مستقرا. في هذا البحث تم دراسة كم الوقت لتحسين اداء خوارزمية راوند روبن وتحسين اخفاقاتها في التبديل الضمني ومعدل وقت الانتظار ومعدل وقت الانتهاء التي عادة ماتثقل النظام. لهذا تم اقتراح اسلوب جديد لحساب كم الوقت تم تسميته الكم التصاعدي وادنى–اعلى راوند روبن. وكانت العمليات تصاعدية مع اقل وقت متبقي للتنفيذ وتم احتسابه بالضرب بنسبة 80% من مجموع ادنى–اعلى وقت. وقد اثبتت النتائج ان الخوارزمية المقترحة افضل اداء من الخوارزمية الاصلية والاعمال السابقة.