Australian Journal of Basic and Applied Sciences, 9(21) Special 2015, Pages: 39-42

ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

DTQS: Dynamic Time Quantum Scheduling Algorithm for CPU Scheduler in Real Time Environment

[1]Dr.Shyamala R. and [2]Chandrasekar S.

[1] Assistant Professor, Department of Information Technology, University College of Engineering Tindivanam, Melpakkam 604 001.
[2]Teaching Fellow, Department of Information Technology, University College of Engineering Tindivanam, Melpakkam 604 001.

ARTICLE INFO

Article history:
Article Received : 12 January 2015
Revised: 1 May 2015
Accepted: 8 May 2015

Keywords:
turnaround time, waiting time, response time, context switching.

ABSTRACT

Central Processing Unit (CPU) scheduling algorithm is the most significant part of operating system (OS) in real time systems. The CPU scheduling algorithm in real time systems is fair to all processes, starvation free and waiting time is minimum. But the existing algorithms suffer from some drawbacks such as more number of context switches, long waiting and long turnaround time. The main objective of this paper is to improve existing CPU algorithm by calculating dynamic time quantum (DTQ) in real time for processes in ready queue. The proposed algorithm in this paper finds DTQ by calculating average burst time of processes in the ready queue. The result of experimental study also shows that the proposed DTQ scheduling algorithm performs better than the conventional algorithms. The performance metrics such as number of context switches, average waiting, and turnaround time is calculated and compared.

To Cite This Article: Dr.Shyamala R. and Chandrasekar S., DTQS: Dynamic Time Quantum Scheduling Algorithm for CPU Scheduler in Real Time Environment. ***Aust. J. Basic & Appl. Sci.,*** *9(21): 39-42, 2015*

INTRODUCTION

Processor is one of the most vital computer resources. The operating system should be designed to use this resource in a most efficient way through CPU scheduler. In OS environment there are several processes waiting in ready queue to be executed. The CPU scheduler determines which process among these needs to be given control of CPU by scheduling algorithm. An improved selection of a particular scheduling algorithm needs to be made as quality of service provided to users and performance of computer system depends on it. There are several scheduling algorithms through which various processes can be allocated CPU depending on their need. Some of these algorithms are described below;

1.1 First Come First Serve (FCFS):

Processes present in ready queue are allocated CPU in the same order in which they come. The process that requests the CPU first is allocated the CPU first hence it is FCFS and it is popularly used in batch system. In real life buying a ticket is analogous to FCFS. It is implemented using queue data structure. New process enters at the tail of the queue. The scheduler selects the processor from the head of the queue. The FCFS is a non-preemptive algorithm and it can't utilize the resource in parallel. If the CPU bound process arrives at first, then it waits all I/O process until completes. Hence I/O devices idle until the CPU bound process completed thus, results in convoy effect.

1.2 Shortest Job First (SJF):

A process which has the shortest expected execution time is given the processor first i.e., it schedules the process with shortest burst time first. It is used for batch processing system. The burst time needs to be known in advance. In some scenarios a job may wait forever and results in starvation.

1.3 Round Robin (RR):

All processes are executed in FCFS order for only a specific time quantum assigned by the system in a cyclic order. This cycle continues until every process executes completely. Round robin is a simple and commonly used scheduling algorithm. The selection of process is based on round robin fashion. It does not consider the priority. The important drawback of round robin is that the numbers of context switches are more.

1.4 Priority Scheduling (PS):

A process with the highest priority is executed first. Each job is assigned with burst time and priority. The job with highest priority is selected for processing. The overhead of this algorithm is how to set the priority and it also results in starvation.

Corresponding Author: Dr.Shyamala R., Assistant Professor, Department of Information Technology, University College of Engineering Tindivanam, Melpakkam 604 001.
E-mail: vasuchaaru@gmail.com

2. Related Works:

The authors (Datta, A.K. and R. Patel, 2014) developed two priority-based CPU scheduling algorithms, Algorithm Cache Miss Priority CPU Scheduler (CM-PCS) and Algorithm Context Switch Priority CPU Scheduler (CS-PCS), which take advantage of dynamic performance data and reduced power consumption by 20%. Seehwan *et al* (2014) used quantization overhead and developed a new scheduling algorithm called SH-quantization that provides accurate and efficient parameterization of a real-time virtual machine and reduces the bandwidth upto 60%. Korotaev *et al* (2005] proposed a CPU scheduler based on parameter for limiting CPU a user can get even if there are spare CPU resources. This proposed QoS parameters provide better control over CPU apportioning and allow limiting user to some CPU shares and limits users CPU consumption and CPUs power. Jawad at al (2014) designed a rule-based scheduling algorithm, in which a fuzzy-based decision maker has been proposed to compute a new priority of all CPU processes according to the process pre-priority and its execution time and the performance closer to SJF algorithm. The authors (Chandra, A., 2001) presented Deadline Fair Scheduling (DFS) for multiprocessor servers which improve proportionate allocation, performance isolation and work-conserving behavior with the scheduling overhead. The work (Chahar, V. and S. Raheja, 2013) proposed a new fuzzy based multilevel queue CPU scheduling algorithm that divided work ready queue into two sub-queues that contains I/O bound and CPU bound processes. CPU time is allocated dynamically to each queue using two fuzzy inference systems. The first inference, dynamically allocate the CPU time to two queues and the second calculated the time quantum to schedule the I/O bound processes. This scheduling improved the starvation problem and response time. The hierarchical weight readjustment algorithm (Chandra, A. and P. Shenoy, 2008) improves CPU bandwidth in multiprocessor system. Sindhu at al (2010) designed CPU scheduler which can handle all types of process with optimum scheduling criteria. Saleem *et al* developed a tool which runs a simulation in real time, and generates useful data to be used for evaluation of CPU scheduling algorithms[9]. Kamalapur *et al* (2006) proposed a genetic based algorithm that improves response time but consumes more computation time.

3. Proposed Approach:

In conventional CPU scheduling algorithm the system assigns a time quantum that does not change at all. In this paper a hybrid scheduling by integrating conventional CPU algorithms have been proposed so that the time quantum of those processes is dynamic to some extent. In our approach DTQ is calculated from the relative difference between minimum and maximum burst time with respect to mean time quantum. All the process is not preempted in DTQ time and if it is not completed then it finished in the successive DTQ.

3.1 Symbols used in Proposed Algorithm:

The following mathematical symbols are used in the DTQS algorithm.

P_i : i^{th} Process where i= 1,2,3……N
N : Total number of process in ready queue.
TQ : Time Quantum
DTQ : Dynamic Time Quantum
bt[i] : Burst time of ith process
α : The minimum burst time.
β : The maximum burst time.
μ : The mean value of burst time.
SBT : Sum of Burst Time

3.2 Proposed DTQ Scheduling Algorithm:

Input: Ready Queue consisting of various Processes.

Procedure DTQS (int bt[1..n], int n)

1. {
2. Initialize DTQ=SBT=α=μ=0,β=∞;
3. If (Ready_Queue != NULL)
4. {
5. Sort the 'n' process in Ready_Queue;
6. α=bt[0];
7. β=bt[n];
8. SBT=0;
9. for(i=0; i<n; i++)
10. SBT+=bt[i];
11. Calculate μ=SBT/n;
12. Calculate DTQ = SQRT((β-α)*μ) ;
13. }
14. Initialize i=1;
15. while(Ready_Queue != NULL)
16. {
17. Select the process at front of ready queue
18. if (bt[i]<= DTQ)
19. {
20. Execute the Process P_i.
21. Remove the process P[i] from ready queue
22. }
23. else if (bt[i]>DTQ)
24. {
25. Execute the Process P_i.
26. Calculate bt[i]= bt[i]-DTQ
27. Remove process P_i from front end of ready queue and add it to the rear.
28. }
29. }// End While
30. }// End.

The procedure for calculating dynamic quantum time (DTQ) is described in this section. In the first step the processes in the ready queue were sorted with respect to burst time. The minimum (min) is set to the first value and the maximum (max) is assigned

as last value from the sorted set. The sum and mean of burst time is calculated in step 10 and step 11. The DTQ is calculated using the formula (1). For each process in the ready queue, the burst time is compared with the DTQ. If the burst time is less than the DTQ then time quantum is set as burst time otherwise processing time is set as DTQ and burst time is recalculated and added in the ready queue for the next cycle.

$$DTQ = \sqrt{(\beta - \alpha) * \mu} \quad (1)$$

4. *Experimental analysis:*

For evaluating the performance it is assumed that the environment where all the experiments are performed is a single processor and the burst time for all processes is known prior to submitting of process to the scheduler. Moreover all the processes do not have priority. The following performance metrics; number of context switches, average waiting time and turnaround time is calculated and compared. As an example, five processes in three different scenario1, 2 and 3 with burst time as shown in Table 1.

Table 1: Processes Specification.

Process Name (P)	Burst Time(BT) ms	Burst Time(BT) ms	Burst Time(BT) ms
	Scenario 1	Scenario 2	Scenario 3
P1	2	1	26
P2	4	9	5
P3	6	16	12
P4	8	15	10
P5	10	21	5

The Figure 1 shows the Gantt chart for round robin scheduling algorithm with time quantum of 4 ms, 7 ms and 5 ms. The turnaround time for the process is calculated as time of submission of a process to the time of completion of the process is obtained through Gantt chart for RR scheduling. The turnaround time for process P1, P2, P3, P4 and P5 is derived as 2, 6, 20, 24 & 30. The average turnaround time is 16.4 ms. Waiting time for the process is calculated as time taken by the process to wait in the ready queue is observed from Gantt chart for RR scheduling. Waiting time for process P1, P2, P3, P4 and P5 is accomplished as 0, 2, 14, 16 & 20 respectively and average waiting time is 10.4 ms. Likewise, the turnaround time and waiting time is calculated for all other scenario and summarized in Table II. From the above results it is observed that Round Robin (RR) is suitable for time sharing systems.

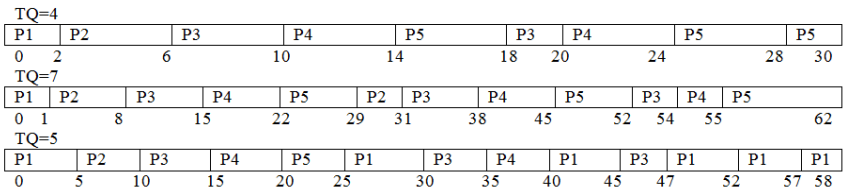

Fig. 1: Gantt chart for RR with TQ =7 ms, 16 ms, 16ms.

The Figure 2 shows the Gantt chart for our proposed scheduling algorithm in which time quantum is dynamically calculated. The turnaround time for process P1, P2, P3, P4 and P5 is derived as 2, 6, 12, 27 & 30. The average turnaround time is 15.4 ms. Waiting time for process P1, P2, P3, P4 and P5 is calculated as 0, 2, 6, 19 & 20 respectively and average waiting time is 9.4 ms. Likewise, the turnaround time and waiting time is calculated for all other scenario and summarized in Table 2.

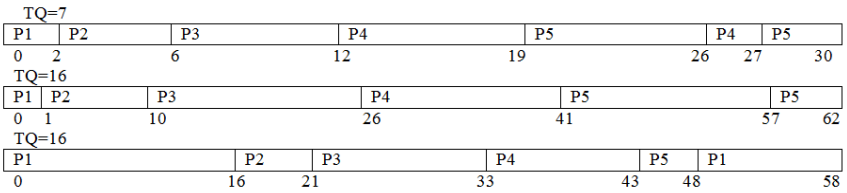

Fig. 2: Gantt chart for Dynamic Time Quantum Scheduling Algorithm.

5. *Performance comparison:*

Performance of RR and DTQS is discussed in section 5. The performance metrics such as average waiting time, average turnaround time, and number of context switches were measured. Table 2 shows the experimental results of the proposed algorithm and the comparisons with the conventional CPU scheduling algorithms. From the results it is observed

that our proposed scheduling improved the performance of RR with minimum context switches and average waiting time.

Table 2: Performance Comparison of RR and DTQS.

Performance Metrics	RR			DTQS		
	Scenario 1	Scenario 2	Scenario 3	Scenario 1	Scenario 2	Scenario 3
No. of Context Switches	9	12	13	7	6	6
Avg. Waiting Time	10.4	28.2	22.4	9.4	15.4	13.4
Avg. Turnaround Time	16.4	40.6	36.0	15.4	27.8	25

6. Conclusions:

Time quantum plays a very important role in CPU scheduling. In this paper a hybrid version of CPU scheduling algorithm is proposed. This approach assigns a mean time quantum for the processes than the allocated fixed time quantum. From the experimental results it was proved that the number of context switch is decreased in the proposed algorithm which is almost equivalent to best/worst case of conventional CPU Scheduling algorithm. Experimental results also show a significant improvement in results of proposed algorithm over conventional round robin scheduling algorithm without much affecting the response time.

REFERENCES

Datta, A.K. and R. Patel, 2014. "CPU Scheduling for Power/Energy Management on Multicore Processors Using Cache Miss and Context Switch Data," *IEEE Transactions on Parallel and Distributed Systems,* 25(5): 1190-1199.

Seehwan Yoo and C. Yoo, 2014. "Real-Time Scheduling for Xen-ARM Virtual Machines," *IEEE Transactions on Mobile Computing,* 13(8): 1857-1867.
doi: 10.1109/TMC.2013.109

Korotaev, K., 2005. "Hierarchical CPU Schedulers for Multiprocessor Systems, Fair CPU Scheduling and Processes Isolation," *IEEE journal of International Cluster Computing,*1-1. doi: 10.1109/CLUSTR, 347085

Jawad, S., 2014. "Design and evaluation of a neurofuzzy CPU scheduling algorithm," *IEEE 11th International Conference on Networking, Sensing and Control (ICNSC),* 445-450: 7-9. doi: 10.1109/ICNSC.2014.6819667

Chandra, A., M. Adler, P. Shenoy, 2001. "Deadline fair scheduling: bridging the theory and practice of proportionate pair scheduling in multiprocessor systems," In *Proceedings of Seventh IEEE Real-Time Technology and Applications Symposium,* 3(14).

Chahar, V. and S. Raheja, 2013. "Fuzzy based multilevel queue scheduling algorithm," *International Conference on Advances in Computing, Communications and Informatics (ICACCI),*115-120: 22-25.

Chandra, A. and P. Shenoy, 2008. "Hierarchical Scheduling for Symmetric Multiprocessors," *IEEE Transactions on Parallel and Distributed Systems,* 19(3): 418-431. doi: 10.1109/TPDS.2007.70755

Sindhu, M.R., P. Rajkamal and Vigneshwaran, 2010. "An Optimum Multilevel CPU Scheduling Algorithm," *International Conference on Advances in Computer Engineering (ACE),*90-94: 20-21.

Saleem, U. and M.Y. Javed, 2000. "Simulation of CPU scheduling algorithms," In *Proceedings of TENCON,* 2: 562-567-2. doi: 10.1109/TENCON.2000.888801.

Kamalapur, S. and N. Deshpande, 2006. "Efficient CPU Scheduling: A Genetic Algorithm based Approach," *Ad Hoc and Ubiquitous Computing, ISAUHC '06. International Symposium on,* 206-207: 20-23.

Silberschatz, Galvin and Gagne, 2009. "Operating System Concepts", 9th edition, Wiley, J Archer Harris, "Operating System", Schaum Outline.