IJETST- Vol.||04||Issue||09||Pages 6016-6021||September||ISSN 2348-9480 2017

Open access Journal International Journal of Emerging Trends in Science and Technology
IC Value: 76.89 (Index Copernicus) Impact Factor: 4.219 DOI: https://dx.doi.org/10.18535/ijetst/v4i9.23

Enhanced Round Robin Technique with Variant Time Quantum for Task Scheduling In Grid Computing

Authors
Dr. T. Kokilavani[1]
Assistant Professor, Department of Computer Science
St. Joseph's College, Tiruchirappalli – 2, India

Abstract
Grid computing is a form of distributed architecture in which large number of computers are connected to solve a complex problem. In the grid computing model, geographically distributed resources run independent tasks and are loosely linked by the Internet or low-speed networks. Efficient scheduling systems are needed to improve the performance of the Grid. Round robin scheduling algorithm (RR) is designed especially for time sharing system.In real-time embedded systems, scheduling policy is considered as one of the main factors that affect their performance. It helps to choose the best resource for executing a task. Round Robin (RR) scheduling algorithm is widely used and its performance highly depends on a Quantum time Qt, which is a predefined amount of time assigned by CPU to every task to be executed. However, the performance degrades with respect to an average waiting time (AWT), an average turnaround time (ATT) and a number of context switches (NCS). This paper presents an Enhanced Round Robin Technique with variant time quantum to reduce the average waiting time, turnaround time and the number of context switches in order to improve the overall Grid performance. It also presents a comparative analysis between existing Round Robin algorithm and the proposed technique based on the average waiting time, average turnaround time, average response time and number of context switches.
Keywords-Grid Computing, Task Scheduling, Round Robin, Time Quantum

1. Introduction

Grid computing is a distributed architecture where large number of computers are connected to solve a single complex problem. In the grid computing model, the unused processor cycles of many heterogeneous computers are used to create a pool of large computing services. In the recent years, grid computing has emerged as an alternative for computation intensive jobs [1]. The Computers in Grid may be connected directly or via scheduling systems. Grid computing combines computers from multiple administrative domains to reach a common goal, to solve a single task, and may then disappear just as quickly [2]. One of the main strategies of grid computing is to use middleware to divide and apportion pieces of a program among several computers, sometimes up to many thousands. Grid computing involves computation in a distributed fashion, which may also involve the aggregation of large-scale clusters. The size of a grid may vary from small confined system to a network of computer workstations within a corporation, for example to large, public collaborations across many companies and networks. Grids are a form of distributed computing whereby a "super virtual computer" is composed of many networked loosely coupled computers acting together to perform very large tasks. This technology has been applied to computationally intensive scientific, mathematical, and academic problems through volunteer computing, and it is used in commercial enterprises for such diverse applications as drug discovery, economic forecasting, seismic analysis, and back office data tasking in support for e-commerce and Web services.

Grid computing tries to bring under one umbrella all the work being done in high-performance, cluster, peer-to-peer and internet computing [3]. Coordinating applications on Grids can be a complex task, especially when coordinating the flow of information across distributed computing resources. Grids started off in the mid-90s to address large-scale computation problems using a network of resource-sharing commodity machines that deliver the computation power affordable only by supercomputers and large dedicated clusters at that time. The major motivation was that these high performance computing resources were expensive and hard to get access to, so the starting point was to use federated resources that could comprise compute, storage and network resources from multiple geographically distributed institutions, and such resources are generally heterogeneous and dynamic.

The main purpose of scheduling policy is to ensure completely fairness between different tasks in the ready queue, maximizing the throughput, minimizing the average waiting and turnaround times and the overhead that occurs due to context switches, and makes sure no starvation happens at all. Two types of scheduling algorithms are followed in Grid computing namely, Preemptive algorithms – where a task can be blocked by a higher priority task and Non preemptive algorithms – where the task completes its execution time even if a higher priority task has arrived [4]. The factors used to determine whether a scheduling policy is good or not are:

(i)Waiting time: It is the time between the task arrival and the first time of its execution.

(ii) Resource Utilization: The percentage of the Resource being busy.

(iii) Turnaround time: The summation of waiting and execution time for each task.
(iv) Fairness: Dividing the resource time equally among all available jobs.

2. Literature Review

Scheduling maps the set of tasks to the available resources based on user constraints. There are two types of Grid scheduling, namely, independent task scheduling and dependent task scheduling. Independent task scheduling is done to reduce the completion time of tasks. Each independent task is scheduled to the available suitable resources. When tasks are dependent on each other, then the tasks require the result of previous tasks to start their execution. First Come First Serve (FCFS) is adequate for dependent task [5].

Ajit et al [6] developed an algorithm which reduces the number of context switching, average waiting time and average turnaround time. This algorithm performs by allocating the CPU to every task in Round Robin fashion with an initial time quantum (say k units). After completing first cycle, it doubles the initial time quantum (2k units); selects the shortest task from the waiting queue and assign the CPU to it and after that, it selects the next shortest task for execution by excluding the already executed task. This algorithm assumes that all the tasks arrive at the same time in the ready queue.

Ishwari et al [7] developed the Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems. It reduces the problem of starvation as the tasks with less remaining CPU burst time are assigned with the higher priorities and are executed first in the second round of algorithm. This algorithm assumes that all tasks arrive at the same time in the ready queue.

Manish et al [8] developed the Improved Round Robin scheduling algorithm which picks the first task from the ready queue and allocates the CPU to it for a time interval of up to 1 time quantum. After completion of task's time quantum, it checks the remaining CPU burst time of the currently running task. If the remaining CPU burst time of the currently running task is less than 1 time quantum, the CPU is again allocated to the currently running task for remaining CPU burst time.

Behera et al [9] developed an algorithm that reduces the number of context switching, average waiting time and average turnaround time. This algorithm arranges the tasks in ascending order of their burst times present in the ready queue. Then, the time quantum is calculated. For finding an optimal time quantum, median method is followed. Then, the time quantum is assigned to the tasks.

Sunita et al [1] proposed a novel grid-scheduling heuristic that adaptively and dynamically schedules tasks without requiring any prior information on the workload of incoming tasks. The approach models the grid system in the form of a state-transition diagram, employing a prioritized round-robin algorithm with task replication to optimally schedule tasks, using prediction information on processor utilization of individual nodes.

Pallab Banerjee et al [10] have presented a new algorithm called Generic scheduling algorithm.

In this scheduling algorithm the main idea is to adjust the time Quantum dynamically so that the generic algorithm performs better than the simple Round Robin scheduling algorithm.

Since the Round Robin algorithm is simple and efficient, it can be applied to Grid Computing model to improve the performance. Normally fixed time quantum is followed for round robin algorithm which may increase the waiting time of tasks. In this paper a novel round robin technique is proposed which uses variant time quantum for every cycle which reduces the waiting time of tasks.

3. Problem Definition

Round Robin scheduling is designed for time-sharing systems [11]. It is similar to FCFS scheduling, but preemption is added to the switch between tasks. A small time unit called the time quantum or time slice is defined. The ready queue is maintained as a circular queue. The CPU scheduler goes round the ready queue, allocating the CPU to each task for a time interval of up to 1 time quantum. The advantages of Round Robin are:

1. Effective in a general-purpose, time-sharing system or transaction-tasking system.

2. Fair treatment for all the tasks.

3. Low overhead on tasks.

4. Good response time for short tasks.

3.1 Scheduling criteria

To select an algorithm for a particular situation, many criteria have been suggested. These criteria are used for comparison and to make a substantial difference in deciding the best algorithm. The criteria include the following [6]:

Context Switch: A context switch is a task of storing and restoring context (state) of a preempted task, so that execution can be resumed from same point at a later time. Context switching is usually computationally intensive, that leads to wastage of time and memory, which in turn increases the overhead of scheduler, so the design of scheduling algorithms is to optimize these switches.

Throughput: Throughput is defined as number of tasks completed per unit time. Context switching and throughput are inversely proportional to each other.

Resource Utilization: This is a measure of how much busy the resource is. Usually, the goal is to maximize the resource utilization.

Turnaround Time: The time interval from the time of submission of a task to the time of completion is the turnaround time. Total turnaround time is the sum of the periods spent waiting to get into memory, waiting time in the ready queue, execution time on the resource and doing I/O.

Waiting Time: Waiting time is the total time a task has been waiting in ready queue. The scheduling algorithm does not affect the amount of time during which a task executes or does input-output; it affects only the amount of time that a task spends waiting in ready queue.

Response Time: Often, a task can produce some output fairly early and can continue computing new results while previous results are being produced to the user. Thus, response time is the time from the submission of a request until the first response is produced. So the response time should be low for the best scheduling.

It can be concluded that a good scheduling algorithm for real time and time sharing system must possess the following characteristics [6]: Minimum context switches, Maximum CPU utilization, Maximum throughput, Minimum turnaround time, Minimum waiting time, and Minimum response time.

4. Enhanced Round Robin Technique with Variant Time Quantum

In the Enhanced Round Robin Technique with Variant Time Quantum (ERRT), whenever a task is scheduled in a particular Grid resource by the Grid scheduler, it is immediately placed in the SCHEDULE queue. Initially ERRT calculates the average burst time of the tasks in the SCHEDULE queue at time 0. Then it takes the floor value of the average burst time as the time quantum for the round robin method. The tasks which have completed their execution in the first cycle are removed from the SCHEDULE queue and placed in the FINISHED queue. After the first cycle ERRT checks whether there are some more tasks waiting for the resource in the SCHEDULE queue. The remaining burst times of the tasks which are executed in the first cycle and the full burst times of the newly arrived tasks are added and its average is taken as the next time quantum for the next cycle. Then the round robin method is continued with the new time quantum. This step is repeated until there are no more tasks in the SCHEDULE queue of the resource.

4.1 Pseudo-code of ERRT

Step 1: Place all the tasks that have arrived at time 0 in the SCHEDULE queue

Step 2: Find the average burst time (ABT) of the tasks in the SCHEDULE queue

Step 3: Fix the Time Quantum tq=floor (ABT)

Step 4: Execute the Round Robin method for all the tasks in the SCHEDULE queue with tq as time quantum.

Step 5: Move the tasks with remaining burst time = 0 to the FINISHED queue.

Step 6: Place the newly arrived tasks in the SCHEDULE queue.

Step 7: Add the remaining burst times of the previously scheduled tasks and the full burst times of the newly arrived tasks.

Step 8: Find the average burst time (ABT) of the tasks and fix tq.

Step 9: Repeat Steps 4 to 8 until the SCHEDULE queue becomes empty.

5. Experiments and Results

A job consists of 10 tasks. Their arrival times and burst times are given in Table 1. As T1 arrives the system, it will be moved to the SCHEDULE queue and the resource will be allocated to it. This is because it is the first task that arrives to the system. So, its burst time b_t which equals 10ms will be used as the time quantum.

After it finishes its execution; T2, T3, T4, T5, T6, T7 and T8 must have arrived the system and will be placed in the SCHEDULE queue. At this point, these 7 tasks will be considered for execution. So the average of their burst times is computed (which is 5.4ms) and the floor value (i.e. 5ms) will be taken as the new time quantum. The same procedure is followed for every cycle until the SCHEDULE queue becomes empty.

Table 1 Tasks with their Arrival Times and Burst Times (in ms)

TASK_ID	AT	BT
T1	0	10
T2	2	5
T3	4	3
T4	5	2
T5	6	4
T6	8	6
T7	9	10
T8	10	8
T9	12	12
T10	14	11

Four performance criteria namely, average waiting time (AWT), average turnaround time (ATAT), average response time (ART) and numbers of context switches were studied. Round Robin (RR) and the proposed ERRT were simulated to observe these criteria. All tasks are independent and CPU bound, no task was I/O bound. The formulae used to calculate AWT, ART and ATAT are given in equations 1, 2 and 3.

Waiting time =Time first scheduled – Arrival Time

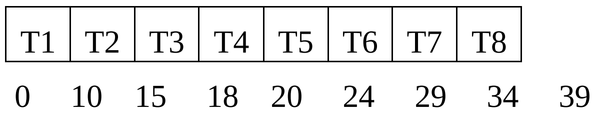
Average waiting time $=\sum_{i=1}^{n}\frac{WTi}{n}$ Eq 1

Turnaround time = Time of task completion- Arrival Time

Average Turnaround time $=\sum_{i=1}^{n}\frac{TATi}{n}$ Eq 2

Response Time = Time of task's first response – Arrival Time

Average Response Time $=\sum_{i=1}^{n}\frac{RTi}{n}$ Eq 3



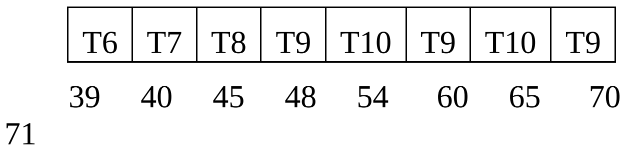


Fig. 1 Gantt Chart for Enhanced Round Robin Technique

Average waiting time, average turnaround time, average response time and number of context switches for the set of tasks given in table 1 are calculated using the equations 1,2 and 3 and the results are given in Table 2.

Waiting Time for ERRT

T1=0; T2=10; T3=15; T4=18; T5=20; T6=24+10=34; T7=29+6=35; T8=34+6=40; T9=48+6+5=59; T10=54+5=59;

Average Waiting Time = (0+10+15+18+20+34+35+40 +59+59) / 10 = 29

Turnaround Time for ERRT

T1=10-0=10; T2=15-2=13; T3=18-4=14; T4=20-5=15; T5=24-6=18; T6=40-8=32; T7=45-9=36; T8=48-10=38; T9=71-12=59; T10=70-14=56;

Average Turnaround Time = (10+13+14+15+18+32+36+ 38+59+56) / 10 = 29.1

Response Time for ERRT

T1=0-0=0; T2=10-2=8; T3=15-4=11; T4=18-5=13; T5=20-6=14; T6=24-8=16; T7=29-9=20; T8=34-10=24; T9=48-12=36; T10=54-14=40;

Average Response Time = (0+8+11+13+14+16+20+ 24+36+40) / 10 = 18.2

Table 2 Parameters used for Scheduling Algorithm

Scheduling Criteria	Round Robin	ERRT
Average Waiting Time	32	29
Average Turnaround Time	32.1	29.1
Average Response Time	18.2	18.2
Number of Context Switches	18	16

From Table 2 it is observed that ERRT produces Minimum context switches, Minimum turnaround time, Minimum waiting time and Minimum response time. Through these minimized values ERRT achieves Maximum CPU utilization and Maximum throughput. Table 3 shows the Total Turnaround Times produced by Round Robin scheduling and Round Robin Algorithm with Different Time Quantum Scheduling algorithms for different set of tasks. Graphical representation of the results are shown in figure 2.From Table 3 and Figure 2 it is observed that the proposed ERRT produces less completion time than Round Robin algorithm in all the cases.

Table 3 Comparison of RR and ERRT

Illustration No	No Of Tasks	Milli Seconds	
		RR	ERRT
1	5	122	102
2	8	198	164
3	10	251	232
4	13	482	457

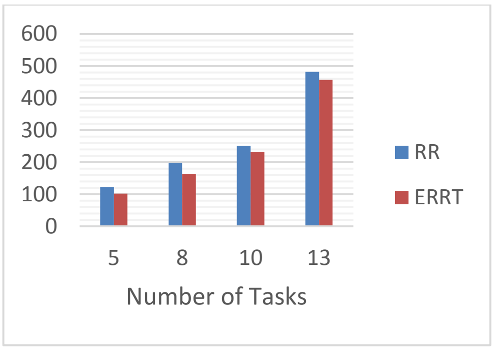


Fig. 2 Turnaround Time for Different Set of Tasks

6. Conclusion

Simple Round Robin algorithm is efficient, because it shares the available time among all the tasks which reduces the waiting time of tasks. But RR uses a fixed time quantum for all cycles which may decrease the system performance. In this paper a novel method called Enhanced Round Robin Technique is proposed which can be applied to Grid Computing model to improve the performance. The Results shows that the proposed technique is more efficient because it has less average waiting time, average turnaround time and number of context switches as compared to simple round robin. Performance of time sharing systems can be improved with the proposed technique and can also be modified to enhance the performance of real time system.

References

1. Sunita Bansal , Bhavik Kothari , ChittaranjanHota, "Dynamic Task-Scheduling in Grid Computing using Prioritized Round Robin Algorithm", International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011, pp. 472 - 477.
2. A. R. Dash. S. K. Sahu and S. K. Samantra, "An Optimized Round Robin CPU Scheduling Algorithm with Dynamic Time Quantum", International Journal of Computer Science, Engineering and Information Technology (lJCSEIT), Vol. 5, No.I, February 2015.
3. V.Vasudevan,R.Vijayalakshmi,"Heuristic Algorithm for Balancing Load in Grid Task Scheduling", International Journal of Computer Applications (0975 – 8887) Volume 67– No.15, April 2013, pp 38 -41.

4. M. K. Mishra and F. Rashid, "An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum", International Journal of Computer Science, Engineering and Applications (T.TCSEA), Vol. 4, No. 4, August 2014.
5. Marish Kr. Singla, “Task Scheduling Algorithms for Grid Computing with Static Jobs: A Review”, International Journal of Computer Science Engineering (IJCSE), Vol. 2 No.05 Sep 2013, pp. 218 – 221.
6. Ajit, S, Priyanka, G and Sahil, B, “An Optimized Round Robin Scheduling Algorithm for CPU Scheduling”, International Journal on Computer Science and Engineering (IJCSE), Vol. 02, No. 07, 2010, pp 2382-2385.
7. Ishwari, S. R and Deepa, G, “A Priority based Round Robin CPU Scheduling Algorithm for Real Time Systems”, International Journal of Innovations in Engineering and Technology (IJIET), Vol. 1 Issue 3, 2012, pp 1-11.
8. Manish K. M. and Abdul Kadir K, “An Improved Round Robin CPU Scheduling Algorithm”, Journal of Global Research in Computer Science, ISSN: 2229- 371X, Volume 3, No. 6, 2012, pp 64-69.
9. Behera, H.S, Mohanty, R and Debashree, N, “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, pp 10-15.
10. Pallab Banerjee, Riya Shree, RichaKumariVerma, “Generic Round Robin Scheduling for Real Time Systems”, International Journal of Advanced Research in Computer Science and Software Engineering Research, Volume 7, Issue 5, May 2017, pp. 148 – 155.
11. Mohammad Saber Iraji, "Time Sharing Algorithm with Dynamic Weighted Harmonic Round Robin", Journal of Asian Scientific Research,Vol. 5, No. 3, 2015, pp. 131-142