Task Scheduling algorithm in cloud computing: A Dynamic Variable Quantum Round robin

Ban hamed abdul ridah
ban8kut3@gmail.com
akdasalkinani@yahoo.com
rowad83@yahoo.com

Educational of wasit ,

Abstract: In recent years a Cloud computing became an attentional topic in IT. In cloud computing a Task allocation is a well-known problem. In the heterogeneous environments like CC task scheduling could be either static or dynamic manner to make the cloud computing more efficient and thus it rallies the users satisfaction. Round-robin fashion is most task scheduling algorithm suitable because it is fair but it had a static time slice problem, so in this paper we proposed dynamic quantum to allocate tasks to appropriate resource which will reduce turnaround time, waiting time, responds time and increase throughput attending to achieve high system performance.

Keywords: Round Robin, Dynamic Time Quantum, Cloud Environment.
1. Introduction

At the few current years, data traffic over the cloud network has been rising exponentially and therefore different innovation resolutions have been emerged to be able to control on data flow. However, there are many numbers of paths appearing in the cloud environment that connect users and their requested task to the cloud server. Therefore, an optimal path selection method needed for the access required cloud server[1].

Cloud computing allow users to get an accessing applications and data from anywhere at any time and. A simple example of a Cloud Computing provider is drobox and google drive which offered as a virtual storage service allow users access stored files and applications via the internet from any device. So major task of cloud is resource allocation to process or process scheduling [2].

Different algorithms are offered for cloud computing scheduling the most important them RR since it ideal in time shared environment. Efficiency of RR completely based on good selecting of suitable time quantum [3].

A time quantum is a small unit of time used in round robin and also named as a time slice. Each task in the Ready Queue (RQ) allocating to the resource for a time interval equal to 1 time slice [4]. When the burst time to the task greater than 1 time slice, that task is blocked and is placed back in to the RQ. in case of a new task arriving, that task will be inserting to the end of the circular queue [5].

To solve the static quantum problem, in this article we suggested a new different scheduling method of RR. In our suggested algorithm, value of time slice dynamically calculated. The sections of the article is organize as. Section 2 describe literature review. In Section 3 we describe our proposed technique and its illustration. In section 4, we experimentally analyze the performance of three scheduling algorithms, including our proposed algorithm, with three test cases. In last part 5 we supplied the conclusion and future work.
2. Related Work

A most of scheduling policy is used in cloud computing. Commonly two type of the scheduling strategies; Static and Dynamic strategy. In the first one, processes are arrives concurrently and available resource schedule updated after each process is schedule. While in the second one, task and machine set location and allocation is not going to fix. Dynamic algorithm worked in two manners: on line and batch modes heuristic scheduling [6].

At last few years many researcher studied RR to improve its performance in different techniques. This is done by CPU utilization, throughput, turnaround time, waiting time and number of context switching. In MMRR time quantum is taken as the range of the CPU burst time of all the tasks. The range of the tasks is the difference between smallest (minimum) and the largest (maximum) values. The authors [] calculate time quantum of the processes depending on their Priority and these processes are organized depending on Priority execute in the main Processor with their individual time quantum [7].

The main objective of using distributed algorithms is to arrange job's time and job's resource restrictions were presented in [8].their methodology needs each node to have full knowledge of the system rest, which really limitations its scalability. In addition their suggestion using deadlines as a finess of service collateral, to achieve additional requirements, such as production manufacture and research deadlines. Nimrod/G is a scheduling platform on the Grid for parameter sweep applications.

others In [9] suggested a feedback dynamic algorithm for pre-emptible task scheduling technique, Which conceder to increases resource utilization in clouds and feedback series of steps in above algorithms do well in the situation where resource contentions are fierce. They suggested a scheduling strategy called Dynamic Round-Robin (DRR) that efficiently reduces power consumption for VM scheduling and consolidation. The strategy goes to deploy the VM to servers and migrate VMs among servers.

In [10] a priority based dynamic resource allocation in cloud computing is suggests. Which considers the multiple SLA constraint and resource locatied by pre-emption mechanism for high priority job execution can make the resource utilization in cloud better. The prime Objective of this research provides dynamic resource provisioning and
achieve multiple SLA objectives through priority based scheduling. Since cost is the important aspect in cloud computing

3. The Suggested Algorithm

To solve the static time slice problem, we suggest an adaptive scheduling Algorithm of RR. In our Algorithm, dynamically determined the time slice value for iterating over tasks in non-static method.

Our algorithm clarified by sorting all tasks in acceding order according on burst time in main ready line. Calculate the mean of the burst time of tasks to use it as initial value to time quantum which dynamically recalculated in every round to each task based on eq1.

\[ TQ_{ij} = TQ^- + \frac{TQ^-}{(BT_{ij} + TQ_{i(j-1)})^2} \]  \\

\text{eq1}

| Proposed algorithm Pseudo code |
|--------------------------------|
| **Variables and functions**   |
| TQ: Time Quantum              |
| RQ: Ready Queue               |
| n: number of process          |
| Ti: Process at i\text{th} index |
| i, j: used as index of ready queue |
| BT: burst time of process     |
| TQ^-: mean of burst time of tasks, |
| TQ_{ij}: time slice of the process, \( T_i \) , in the round \( j \) |
| BT_{terminated}: burst time of finished process. |
| \( T_{new} \): burst time of new arrived process |
| Execute(): a function execute a process, \( T_i \) |
| **Input**:                    |
| Process, \( T_i \)           |
| **Output**:                   |
| Rescheduling all processes , \( T_i \) |
| **BEGIN**                    |
| Sort the processes in ascending order. |
| TQ^- = the mean of burst time of all processes |
| WHILE RQ are not empty        |
| BEGIN                        |
| Execute(\( T_i \))           |
| //Determine the new quantum time |
| \[ TQ_{ij} = TQ^- + \frac{TQ^-}{(BT_{ij} + TQ_{i(j-1)})^2} \]  \\
| eq1                          |
IF BT(T) = 0 THEN
\[ TQ^\sim = TQ^\sim - \frac{TQ^\sim}{BT_{\text{terminated}}} \]

IF new process \( T_{\text{new}} \) is arrived THEN
BEGIN
Insert \( T_{\text{new}} \) into RQ
\[ TQ^\sim = TQ^\sim + \frac{TQ^\sim}{BT_{\text{new}}} \]
END
\( j++ \)
END WHILE

FIGURE 1 ADAPTIVE RR ALGORITHM

4. Result and Discussion

To evaluate the performance our technique we make a simple comparison with the traditional RR and the modified RR technique presented in [11]. We use three data sets, clarified its details in table 1. Response time (RT), waiting time (WT), turnaround time (TRT) and throughput are the performance criteria applied to compare the algorithms.

| Task | Set1 | Set2 | Set3 |
|------|------|------|------|
|      | Burst Time | Arrival Time | Burst Time | Arrival Time | Burst Time | Arrival Time |
| t1   | 50   | 0    | 250  | 0    | 30   | 0    |
| t2   | 100  | 1    | 200  | 1    | 300  | 1    |
| t3   | 150  | 2    | 150  | 2    | 200  | 2    |
| t4   | 200  | 3    | 100  | 3    | 20   | 3    |
| t5   | 250  | 4    | 50   | 4    | 10   | 4    |
### Table 2: Experimental Results Applying on the First Data Set

| Techniques                   | TRT | WT  | RT  |
|------------------------------|-----|-----|-----|
| modified RR                  | 380 | 230 | 190 |
| Traditional RR (static time slice=8 unit) | 219.2 | 111.6 | 210 |
| Proposed algorithm           | 261.6 | 69.6 | 17.6 |

### Figure 2: Experimental Results Applying on the First Data Set

Figure 2 clarified that the proposed algorithm presented comparative turnaround time with traditional RR, considering to waiting and response time proposed algorithm presented best result over the other algorithms.

### Table 3: Experimental Results Applying on the Second Data Set

| Techniques                   | TRT | WT  | RT  |
|------------------------------|-----|-----|-----|
| modified RR                  | 610 | 460 | 290 |
| Traditional RR (static time slice=8 unit) | 281.4 | 102.6 | 250 |
| Proposed algorithm           | 252.6 | 102.6 | 17.6 |
FIGURE 3 EXPERIMENTAL RESULTS APPLYING ON THE SECOND DATA SET

Figure 3 show that traditional RR and proposed algorithm have equal waiting time, in turnaround and response time our proposed algorithm presented a good performance over the rest.

TABLE 4 EXPERIMENTAL RESULTS APPLYING ON THE THIRD DATA SET

| Techniques                              | TRT | WT  | RT  |
|-----------------------------------------|-----|-----|-----|
| modified RR                             | 258 | 146 | 58  |
| Traditional RR (static time slice=8 unit)| 137.8 | 38.8 | 80 |
| Proposed algorithm                      | 149.8 | 37.8 | 17.6 |

FIGURE 4 EXPERIMENTAL RESULTS APPLYING ON THE THIRD DATA SET
figure 4 clarified that turnaround time was not satisfied in proposed algorithm comparing with static RR but better than modify RR, while in response and waiting time the result was in possible ranges.

Summary of throughput presented in figure 5 show that all comparative technique offered an equal throughput using second dataset, in first and third dataset our algorithm shows a superior performance.

5. Conclusion and Future scope.

In this paper we make a little change traditional round robin in a way which will resolved the fixed unit of time quantum. Evaluating the performance of our adaptive algorithm by comparing it with traditional RR and modified RRs algorithms. After analyzing the results and evaluating the performance, the adaptive algorithm proved to be more suitable for scheduling tasks in heterogeneous environments such as the cloud computing environment, especially since the task expansion of resources was dynamically demonstrated.

As a future works we will try to applying our adaptive technique in Big Data platforms, we also intend works on finding more intelligence way to calculate the initial value to time quantum in the proposed scheduling algorithm.
Reference

1- Kosta, S., Aucinas, A., Hui, P., Mortier, R., & Zhang, X. (2012, March). Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In 2012 Proceedings IEEE Infocom (pp. 945-953). IEEE.

2- Wu, D., Rosen, D. W., Wang, L., & Schaefer, D. (2015). Cloud-based design and manufacturing: A new paradigm in digital manufacturing and design innovation. Computer-Aided Design, 59, 1-14.

3- Latiff, M. S. A., Madni, S. H. H., & Abdullahi, M. (2018). Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Computing and Applications, 29(1), 279-293.

4- Singh, M., & Agrawal, R. (2017, September). Modified round robin algorithm (mrr). In 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI) (pp. 2832-2839). IEEE.

5- Bhoi, S. K., Panda, S. K., & Tarai, D. (2014). Enhancing CPU Performance using Subcontrary Mean Dynamic Round Robin (SMDRR) Scheduling Algorithm. arXiv preprint arXiv:1404.6087.

6- Vasile, M. A., Pop, F., Tutueanu, R. I., Cristea, V., & Kołodziej, J. (2015). Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. Future Generation Computer Systems, 51, 61-71.

7- Dawood, A. J. (2012). Improving Efficiency of Round Robin Scheduling Using Ascending Quantum And Minumim-Maximum Burst Time. Journal of University of anbar for pure science, 6(2), 85-89.

8- Buyya, R., Murshed, M., Abramson, D., & Venugopal, S. (2005). Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost–time optimization algorithm. Software: Practice and Experience, 35(5), 491-512.

9- Lin, C. C., Liu, P., & Wu, J. J. (2011, December). Energy-efficient virtual machine provision algorithms for cloud systems. In 2011 Fourth IEEE International Conference on Utility and Cloud Computing (pp. 81-88). IEEE.

10- Agarwal, D., & Jain, S. (2014). Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint arXiv:1404.2076.

11- Mohapatra, S., Mohanty, S., & Rekha, K. S. (2013). Analysis of different variants in round robin algorithms for load balancing in cloud computing. International Journal of Computer Applications, 69(22).