ISSN:0975-9646

Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing

Pooja Samal, Pranati Mishra

Department of CSE,
CET

Abstract **— Cloud computing is the emerging internet based technology which emphasizes commercial computing. Cloud is a platform providing dynamic pool resources and virtualization. Based on a pay-as-you-go model, it enables hosting of pervasive applications from consumer, scientific, and business domains. To properly manage the resources of the service provider, load balancing is required for the jobs that are submitted to the service provider. Load balancing also helps in improving the performance of the centralized server. In the present work, various policies in relation to the algorithms developed are analyzed using an analysis tool, namely, cloud analyst. Comparison is also made for variants of Round Robin (RR) algorithms.**

Keywords—Cloud Computing; Virtual machines; Cloud service provider; Cloud Analyst; CloudSim; Cloud Service broker.

I. INTRODUCTION

Cloud computing has recently emerged as a new paradigm for hosting and delivering services over the Internet. It is attractive to business owners as it eliminates the requirement for users to plan ahead for provisioning, and also allows enterprises to start from the small and increase resources only when there is a rise in service demand [1]. Cloud computing is supported by state-of-the-art data centers that usually employ Virtual Machine (VM) technologies for consolidation and environment isolation purposes [2]. Cloud computing delivers an infrastructure, platform, and software (applications) as services that are made available to consumers in a pay-as-you-go model. In industry these services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) respectively [3]. Many computing service providers including Google, Microsoft, Yahoo, and IBM are rapidly deploying data centers in various locations around the world to deliver cloud computing services.

II. LOAD BALANCING

The goal of load balancing is to improve the performance by balancing the load among various resources (network links, central processing units, disk drives etc.) to achieve optimal resource utilization, maximum throughput, maximum response time, and avoiding overload [4]. Load balancing is a relatively new technique that facilitates networks and resources by providing a maximum throughput with minimum response time [5]. Dividing the traffic between servers, data can be sent and received without major delay. Different types of algorithms are available that helps distribution of traffic loaded, between available servers. A basic example of load balancing in our daily life can be related to websites. Without load balancing, users could experience delays, timeouts and possible long system responses. Load balancing solutions usually apply to redundant servers which help a better distribution of the communication traffic so that the website availability is conclusively settled [5]. The load balancing algorithms can be categorized mainly into two groups as discussed in the following section.

A. Static Algorithms

Static algorithms divide the traffic equivalently between servers. By this approach the traffic on the servers will be disdained easily and consequently it will make the situation more imperfect. This algorithm, which divides the traffic equally, is announced as round robin algorithm. However, there were lots of problems associated with this algorithm. Therefore, weighted round robin was developed to improve the critical issues of round robin. In weighted round robin algorithm each servers is assigned a weight and according to the highest weight they receive more connections. In a situation, when all the weights become equal, servers will receive balanced traffic [6].

B. Dynamic Algorithms

Dynamic algorithms designate proper weights on servers dynamically by searching the whole network. The lightest server is loaded to balance the traffic. However, selecting an appropriate server needs real time communication with the networks, which leads to extra traffic added to the system. Dynamic algorithm predicates on query that are made frequently on servers. However, sometimes prevailed traffic prevents these queries to be answered, and correspondingly more added overhead can be distinguished on network.

Although round robin algorithms are based on simple rule, more load is conceived on servers and thus unbalancing the traffic [6].

The remainder of this paper is organized as follows. Section 2 lists the related work. Section 3 describes the algorithm and defines the environment parameters. Section 4 experiments. Section 5 analyses of the experiment results. Section 6 gives the conclusions.

III. RELATED WORK

RR, MRR and TSPBRR had been used in the task scheduling in cloud computing. The research on them received good results and their efficiency had been proved. There are also many related improved work under study. Randles, M. et. al., [7] & Alakeel [8] have proposed a comparison of static and dynamic load balancing algorithms for cloud computing.

Padhy, R.P., Goutam, P., and Rao, P., [9] discussed some basic concepts of cloud computing and load balancing by studying some of the existing load balancing algorithms applied to clouds. Jiyin et.al, [10] have proposed a resource allocation mechanism with preemptable task execution which increases the utilization of clouds. They proposed an adaptive resource allocation algorithm for cloud system with preemptable tasks. However, their approach does not pertain to cost optimization and time optimization.

Stewart, D.B., and Khosla, P.K., [11] proposed the maximum-urgency-first algorithm, which can be used to predictably schedule dynamically changing systems. The scheduling mechanism of the maximum-urgency-first causes a critical task to fail. Thus, a modified maximum urgency first scheduling algorithm was proposed by I. Ahmad, Y.-K. Kwok, M.-Y. Wu, and K. Li which resolved the above mentioned problem.

Singh et. al [13] proposed a modified round robin (MRR), which is superior than RR and has less waiting response time, usually less pre-emption and context switching thereby reducing the overhead and saving of memory space. Yashuwaanth [14] proposed another MRR algorithm which overcomes the limitations of simple RR. Mohanty et. al. [15] have considered dynamic time quantum which changes with every round of execution. Their results show that PBDRR performs better than MRR algorithm in terms of reducing the number of context switches, average waiting time and average turnaround time.

H. Casanova et al. [16] and Baraglia et al. [17] proposed the heuristic algorithms to solve the scheduling problem based on various static data such as the execution time and system load. Unfortunately, all information such as execution time and workload cannot be determined in advance of dynamic grid environments.

Katevenis et al. [18] have proposed weighted round-robin cell multiplexing in a general-purpose ATM switch chip. Shreedhar and Varghese [19] have discussed the efficient fair queuing using deficit round robin. Wickrema et al. [20] present how Cloud Analyst can be used to model and evaluate a real world problem through a case study of a social networking application deployed on the cloud. The issues such as how the simulator can be used to effectively identify overall usage patterns and how such usage patterns affect data centres hosting the application are illustrated in the present work.

IV. SCHEDULING CRITERIA

For the task scheduling based on RR, MRR and TSPBRR, the criteria include the following:

Context Switch: A context switch is computing process of storing and restoring state of a CPU so that execution can be resumed from same point at a later time. Context switch are usually computationally intensive, lead to wastage of time, memory, scheduler overhead so much of the design of operating system is to optimize these switches.

Throughput: Throughput is defined as number of process completed per unit time. Throughput will be slow in round robin scheduling implementation. Context switch and throughput are proportional to each other.

CPU Utilization: We want to keep the CPU as busy as possible.

Turnaround Time: Turnaround time is sum of periods spent waiting to get into memory, waiting in ready queue, executing on CPU and doing input output. It should be less.

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

Response Time: Response time is the time it takes to start responding, not the time it takes to output the response. Large response time is a drawback in round robin architecture as it leads to degradation of system performance.

Response Time = Fint - Arrt + TDelay

Arrt is the arrival time of user request.
Fint is the finish time of user request.

TDelay = T + T(2) latency transfer

TDelay is the transmission delay
Tlatency is the network latency
Ttransfer is the time taken to transfer the size of data of a single request (D) from source location to destination.
Ttransfer=D/Bwperuser(3)Bwperuuser=Bwtotal/Nr(4)
Bwtotal is the total available of bandwidth
Nr is the number of user request in transmission

A good scheduling algorithm must possess following Characteristics:

1. Minimum context switches.
2. Maximum CPU utilization.
3. Maximum throughput.
4. Minimum turnaround time.
5. Minimum waiting time.
6. Minimum response time.

V. HEURISTICS

A. Scheduling based on RR

The scheduling process based on RR experimentation is represented in Fig.1.

1. The scheduler maintains a queue of ready Processes and a list of blocked and swapped out processes.
2. The PCB of newly created process is added to end of ready queue. The PCB of terminating process is removed from the scheduling data structures.
3. The scheduler always selects the PCB at head of the ready queue.
4. When a running process finishes its slice, it is moved to end of ready queue.
5. The event handler perform the following action,
   - *a) When a process makes an input -output request or swapped out, its PCB is removed from ready queue to blocked/swapped out list.*
   - *b) When input-output operation awaited by a process finishes or process is swapped in its process control block is removed from blocked/swapped list to end of ready queue.*

Fig.1. Pseudo code for task scheduling process based on RR

C. Scheduling based on MRR

In MRR algorithm, Time Slice (ITS) is calculated which allocates based on (range× total no of process (N)) divided by (priority (pr)× total no of process(p)). Range is calculated by (maximum burst time + minimum burst time) divided by the scheduling process based on MRR experimented in this paper is represented in Fig.2.

```
1) While (ready queue!=NULL)
{
For i to n
{
        Range=maxbt+minbt/2
        ts = ( range×N) / (pr×p)
} end of for
} end of while
```

Fig.2. Pseudo code for task scheduling process based on MRR

TABLE I. Calculation of time slice for MRR

Pr No	Burst Time (BT)	Priority (Pr)	Range (R)	N	P	Time Slice (TS) R×N/ Pr ×P
P1	25	2	15	5	5	8
P2	5	3	15	5	5	5
P3	15	1	15	5	5	15
P6	8	5	15	5	5	3
P5	10	4	15	5	5	4

Where,

N= Total no of process
P= Total no of priority
Range = Maximum BT+Minimum BT /2
Range = 5+25/2 =15
Time slice=Range ×N/ Pr×P
TS = 15 × 5/ 2 ×5 =8

D. Scheduling based on Time Slice Priority Based RR.(TSPBRR)

Let 'TQ$_i$' is the time quantum in round *i*. The number of rounds *i* varies from *1* to *n*, where value of *i* increments by *1* after every round till ready queue is not equal to NULL.

The scheduling process based on TSPBRR experimentation is represented in Fig.3.

```
1. Calculate TS for all the processes present in the ready queue.
2. While (ready queue!= NULL)
    {
    For i=1 to n do
    {
    if ( i ==1)
    {
    TQi = ½ TSi
        }
    Else
    {
    TQi = TQ i-1 + ½ TQ i-1
    }
    If (remaining burst time -TQ i) <=2
    TQ i = remaining burst time
    } End of For
    } End of while
3. Average waiting time, average turnaround time
and no. of context switches are calculated
    End
```

Fig.3. Pseudo code for Task scheduling process based on TSPBRR

VI. EXPERIMENT

Scheduling algorithms described in the previous section were implemented and tested in CloudSim. There were 5 numbers of users and 3 resources in the mod. For users in the model, each of them requests execution of 100 tasks with different length (in MI) between 1 and 50. The initial information of the resource in the model is described in Table II.

TABLE II. Calculation of time slice for MRR

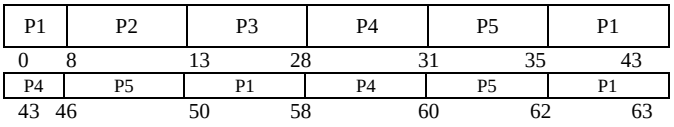
Pr No	Burst Time (BT)	Priority (Pr)	Range (R)	N	P	Time Slice (TS) R×N/ Pr ×P
P1	25	2	15	5	5	8
P2	5	3	15	5	5	5
P3	15	1	15	5	5	15
P6	8	5	15	5	5	3
P5	10	4	15	5	5	4

TABLE III. Calculation of time slice for proposed algorithm

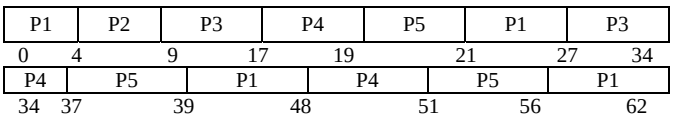
Process No	TS	ROUNDS				
		1st	2nd	3rd	4th	5th
P1	8	4	6	9	6	0
P2	5	5	0	0	0	0
P3	15	8	7	0	0	0
P4	3	2	3	3	3	0
P5	4	2	2	5	5	0

TABLE IV. Calculation of AVG TAT, AVGWT of MRR & TSPBRR

Algorithm	Average TAT	Average WT	CS
MRR	45.2	37.4	12
TSPBRR	42.4	32.4	13



Gantt chart for MRR



Gantt chart for TSPBRR

VI. ANALYSIS

The experiments as described in the previous section is carried out by considering 25 and 50 VMs which are present in different regions with 1 DC and 2 DC. The simulated results the different variants of RR are presented in Fig. 4 through Fig. 7. It is observed that the response time is better in case of TSPBRR as compared to others variants. It is also observed that the Avg TAT and Avg WT in TSPBRR superior than that in MRR algorithm.

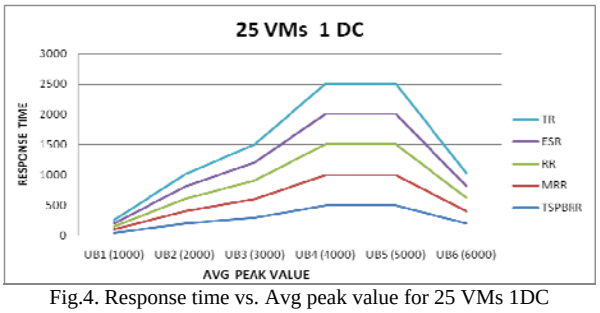


Fig.4. Response time vs. Avg peak value for 25 VMs 1DC

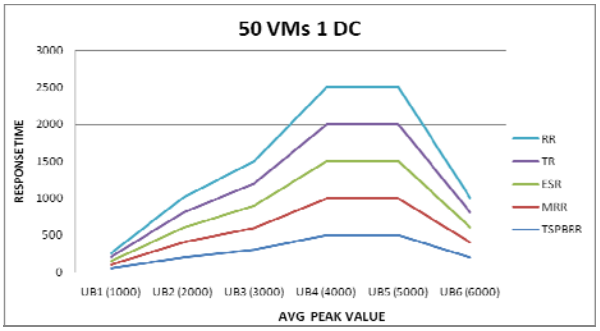


Fig.5. Response time vs. Avg peak value for 50 VMs 1DC

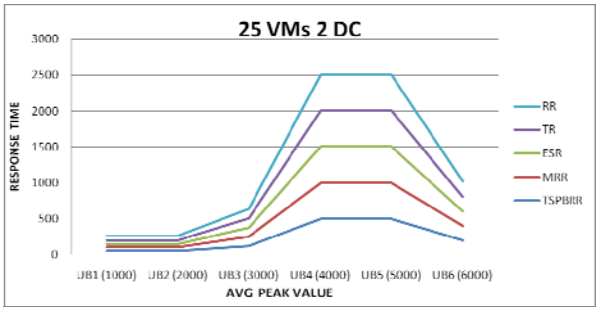


Fig.6. Response time vs. Avg peak value for 25 VMs 2DC

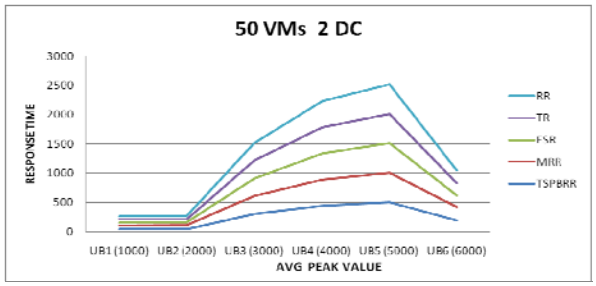


Fig.7. Response time vs. Avg peak value for 50 VMs 2DC

III. Conclusion

The load distribution problem on various nodes of a distributed system is solved in the present work to improve both resource utilization and job response time by analyzing the variants of RR algorithm. The overloading and under loading situations are avoided. Thus, load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. The proposed algorithm shows better response time as compared to the other algorithms.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, Cloud Computing: state of – the-art and research challenges, Springer, 2010, pp. 7-18.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, Xen and the art of virtualization, in: Proceedings of the 19th ACM Symposium on Operating Systems Principles, SOSP 2003, Bolton Landing, NY, USA, 2003, pp. 177.

[3] B.P. Rimal, E. Choi, and I. Lumb, A Taxonomy, Survey, and Issues of Cloud Computing Ecosystems, Cloud Computing: Principles, Systems and Applications, Computer Communications and Networks, Springer – Verlag, Chapter 2, 2010, pp. 21-46.

[4] A. Khiyaita, H. El Bakkali, M. Zbakh, D. El Kettani, Load Balancing Cloud Computing: State of Art", IEEE, 2010.

[5] R. Shimonski, Windows 2000 & Windows Server 2003 Clustering and Load Balancing. Emeryville. McGraw-Hill Professional Publishing, CA, 2003, pp. 2.

[6] R. X. T. and X. F. Z., A Load Balancing Strategy Based on the Combination of Static and Dynamic, in Database Technology and Applications (DBTA), 2010 2nd International Workshop, pp. 1-4, 2010.

[7] M. Randles, D. Lamb, and A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, 2010, pp. 551–556.

[8] A.M. Alakeel, A Guide to dynamic Load balancing in Distributed Computer Systems, International Journal of Computer Science and Network Security, Vol. 10, No. 6, 2010, pp. 153-160.

[9] R.P. Padhy and P.G.P. Rao, Load balancing in cloud computing system, Department of Computer Science and Engineering National Institute of Technology, Rourkela, Odisha, India, 2011.

[10] L. Jiyin, Q. Meikang, J-W. Niu, Y. Chen, and Ming, Adaptive Resource Allocation for Preeemptable Jobs in Cloud Systems, IEEE International Conference on Intelligent Systems Design and Applications, 2010, pp. 31-36.

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

[12] I. Ahmad, Y.-K. Kwok, M.-Y. Wu, and K. Li, "Experimental Performance Evaluation of Job Scheduling and Processor Allocation Algorithms for Grid Computing on metacomputers," Proc. IEEE 18th Int'l Parallel and Distributed Processing Symp. (IPDPS '04),pp. 170-177, 2004.

[13] A. Singh, P. Goyal, S. Batra, An Optimized Round Robin Scheduling Algorithm for CPU Scheduling, International Journal of Computer and Electrical Engineering, Vol. 2, No. 7, 2010, pp. 2383-2385.

[14] C. Yaashuwanth, and R. Ramesh, Design of Real Time Scheduler Simulator and Development of Modified Round Robin Architecture for Real Time System, International Journal of Computer and Electrical Engineering, Vol. 10, No. 3, 2010, pp. 43-47.

[15] R. Mohanty, H. S. Behera., Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems, International Journal of Computer and Electrical Engineering, Vol. 2, No. 2, 2011, pp. 46-50.

[16] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman, Heuristics for scheduling parameter, Journal of Theoretical and Applied Information Technology © 2005 - 2009 JATIT. All rights reserved. www.jatit.org 115 sweep applications in Grid environments", in Heterogeneous Computing Workshop", 2000, IEEE Computer Society Press, 2000, pp. 349–363.

[17] R. Baraglia, R. Ferrini, and P. Ritrovato, Astatic mapping heuristics to map parallel applications to heterogeneous computing systems, Concurrency and Computation: Practice and Experience, Vol. 17, No. 13, 2005, pp. 1579–1605.

[18] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, Weighted round-robin cell multiplexing in a general-purpose ATM switch chip, IEEE J. Sel. Areas Commun., Vol. 9, No. 8, 1991, pp. 1265–1279.

[19] M. Shreedhar, and G. Varghese, Efficient fair queuing using deficit round robin, IEEE Trans. Netw., Vol., 4, No. 3, 1996, pp. 375–385.

[20] B. Wickremasinghe, N. Rodrigo Calheiros, and R. Buyya, Cloud Analyst: A CloudSim-based Visual Modeller for Analyzing Cloud Computing Environments and Applications, IEEE, 2010.