Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 125 (2018) 717–724

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India

Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing

Mohit Kumar[a*], Kalka Dubey[b], S.C.Sharma[a,b]

[a]*Research Scholar, IIT Roorkee, india*
[a,b]*Professor, IIT Roorkee, India*

Abstract

There are many load balancing algorithm has been proposed for cloud computing in last decade but none of algorithm provides the elasticity with load balancing. We proposed a cloud architecture that is capable of handling the maximum user request before meet to deadline and provides an elasticity mechanism with the help of threshold based trigger strategy. Computational results (Table 1 & Figs. 2-5) shows that develop algorithm decrease the makespan time and enhance task acceptance ratio more than 10% compare to min-min algorithm, 30% compare to First come first serve (FCFS) and shortest job first (SJF) in all condition.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications.

Keywords: scalability, makespan time, virtual machine, elasticity, task scheduling;

1. Introduction

Cloud computing is an internet based computing technology in the field of computer science that provides the services (either in the form of software or hardware) to the users on the basis of pay per use. It provides the services like as software as a service, storage as a service, infrastructure as a service etc.[1] and three deployment models of cloud are public, private and hybrid. Public cloud services are available for general public over the internet. Amazon elastic compute cloud, Google appEngine, Window azure service platform etc are examples of public cloud those offered the services either pay per use basis or free. Private cloud is used for personal use or provides the service to

* Corresponding author: Tel. +91-9759950380
E-mail address: mohit05cs33@gmail.com

1877-0509
Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications
10.1016/j.procs.2017.12.092

single organization. Eucalyptus, OpenNebula, Openstack etc are example of private cloud those offered the similar advantages to public cloud. A hybrid cloud is combination of two or more than two public and private cloud which are bounded by service level agreement (SLA). User can send the request at any time from any geographical location for services, SLA selects the best resource within user defined deadline and budget. Elastic resource provisioning with quality of service (QoS) parameter (deadline, high availability, priority etc.) is one of the most challenging problem in the field of cloud computing. Therefore cloud service provider needs an efficient load balancing algorithm that reduces the makespan time as well as task rejection ratio within user defined deadline.

In the last few years, maximum company are trying to achieve the scalability in terms of platform, application and infrastructure level. Scalability is an important feature in cloud computing and can be divided into two part one is scale up other is scale out [2]. Scale up is also called vertical scalability and scale out is also called horizontal scalability. In this paper, we are using the horizontal scaling approach based upon the workload prediction with the help of user defined threshold at the time of service level agreement (SLA).

The reminder of the article is organized as follows: Section 2 describes the related work which is related to our research work such as existing scheduling & load balancing technique with virtual machine provisioning and deprovisioning, Section 3 we will discuss the proposed architecture and its components, section 4 problem formulation and proposed scheduling algorithm with elasticity, further section 5 is for analyze and comparison of experimental results and Section 6 conclusion.

2. Related Work

There are lot of algorithm have been proposed for load balancing and scalability of cloud resource in last decade. E.Coninck et al. proposed a dynamic auto scaling algorithm that reduces the execution time and makespan time of upcoming requests (application/task) considering the deadline as a constraint using the Openstack and cloudsim as a simulator [3]. M. Kumar and S.C Sharma [4] proposed an algorithm that not only reduce the makespan time of tasks but also increase the utilization ratio of the task considering the priority of tasks as quality of services parameter. F.Juarez proposed a dynamic energy aware scheduling algorithm that reduces the makespan time and energy using the private cloud as a tool for implementation [5]. In this paper, find out the makespan and energy consumption for results but integration of both parameters doesn't give optimal results i.e. one parameter at a time because trade-off occur between time and energy. Ye Feng et al., proposed a dynamic load balancing algorithm that reduce the task completion time and load balancing degree [6]. S. Abrishami, M. Naghibzadeh develop an algorithm for SaaS and IaaS that reduce the parameter cost and execution time where deadline as a constraint and java based simulator is used to implement the algorithm[7][8]. Proposed algorithm is implemented on Java based simulator that does not give the guarantee of cloud environment. The most important problem in the real environment is the inaccuracy of the estimated execution and transmission times. R.Naha, M.Othman and T.Somasundaram proposed broker based architecture for task scheduling and elasticity in cloud environment using cloud analyst and eucalyptus as a tool for implementation [9][10].

Li Xiaofang et al. [11] proposed an improved max-min algorithm for elastic cloud that monitor the load at virtual machine continuously. Proposed algorithm allocate the task to running virtual machine in such a way that it can improve the response time of tasks and resource utilization ratio. Coutinho et al. Analyze the elasticity behaviour in cloud computing [12] using the parameter response time and utilization of cpu. Further author's define the over provisioning and under provisioning condition in elasticity. Al-Dhuraibi, Yahya, et al. Review all the existing classical and recent elasticity solutions [13]. Author's compared the horizontal vs. vertical scalability pros and cons. Further they discuss the classification of elasticity mechanism, performance evolutions tools and elasticity solution in details. Galante et al. [14] present a comprehensive study of elasticity including the classification mechanism based upon the commercial and academic solutions in cloud environment. Some challenges and open issues which are associated with the use of elasticity concept are also discussed by the author's in this paper. Hu, Yazhou, et al. [15] discussed the elasticity concept and proposed a linear regression model to predict the upcoming workload in

cloud environment. Cloud resources are scaling based upon the predicted workload in this paper.

3. Proposed Architecture

The proposed cloud architecture for resource provisioning and deprovisioning with load balancing is shown in Figure1.

Job Request Handler It is work at software as a service level cloud platform. User submit the task request T_1, $T_2 \ldots T_n$ either graphical user interface or command line for service in the form of hardware, software etc. considering quality of service (QoS) parameter like deadline, priority, elasticity, availability etc. all the request (tasks) are checked by job request handler (gatekeeper) using Turing test and determine upcoming request is coming from legitimate user or not.

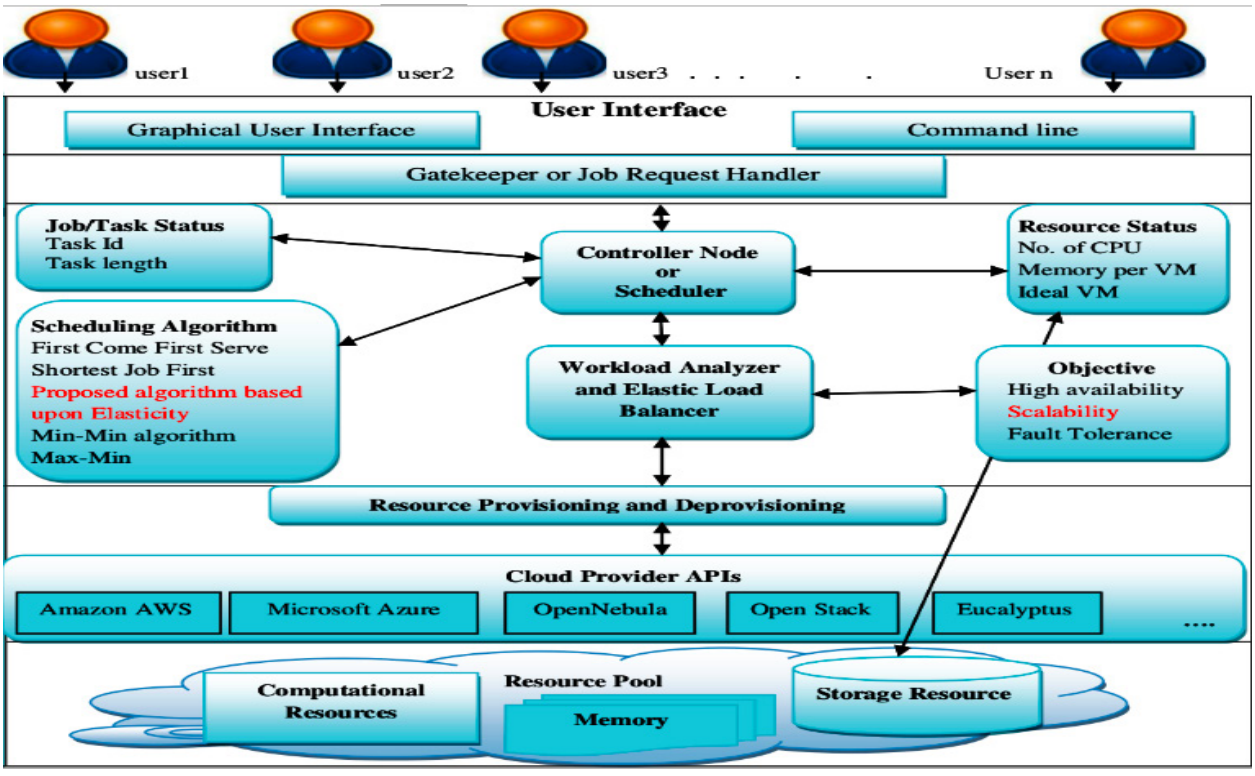


Figure 1 Cloud architecture for resource provisioning and deprovisioning

Controller Node or Scheduler This node is used to provide the interaction between SaaS and IaaS. Scheduler checks the quality of service parameter of request (task is deadline or priority based) and user required parameter like makespan time, execution time, cost, energy efficiency before assigned the tasks to cloud resource. After that scheduler schedule the task based upon the scheduling algorithm.

Workload analyzer and Elastic Load Balancer (ELB) The main aim of workload analyzer is to look at different characteristics of cloud workload like high computational workload and description of workload. ELB starts to monitor the virtual machine to find out the status of virtual machine (overloaded or underloaded conditions).

Cloud Resource Provisioning and deprovisioning (CRP and CRDP) The main objective of CRP is creation and deletion of virtual machine as per application (job) requirement that provides the better resource utilization and potential cost saving. Amazon cloud watch is available for scale up or scale down the resources in AWS cloud.

4. Problem Formulation and Proposed Algorithm

4.1 Problem formulation

To schedule all the upcoming workload in such a ways that cloud user can execute their task in minimum time considering deadline as constraint. Job request handler received N number of independent task request with task length (LT_{Ti}) and deadline of task is DT_{Ti}. Every task need processor speed p_s, number of cpu q_n, RAM R_r and bandwidth B in mbps. There are m heterogeneous running virtual machine R_1, $R_2 \ldots R_m$ in cloud environment. Scheduler try to map each task T_i to virtual machineR_j, if a resource R_j is match with task Ti then value of decision variable DV_{ij} is 1 otherwise its value is 0.

Execution time of tasks Ti at a particular virtual machine is $ET_{Ti} = LT_{Ti}/p_s * q_n$ (1)

When user submits the task to cloud resource with deadline, its completion time is depend that how much workload is available on that resource. The time required to complete the task on available cloud resource is expressed in equation 2. $RT_{Ti} = DT_{Ti} - AW_{Rj}$ (2)

Where AW_{Rj} represent the available workload at the resource R_j before allocated the task T_i and RT_{Ti} is the available remaining time to complete the allocated task T_i at resourceR_j. This remaining time (RT_{Ti}) should be greater than or equal to task execution time. $ET_{Ti} \leq R_{Ti}$ (3)

Execution time of virtual machine where tasks are executing in minimum time

$$ET_{Tj} = \min \sum_i^n ET_{Ti} * DV_{ij} \quad (4)$$

$$\text{makespan time } MST_{(Ti)} = \max \sum_j^m E_{Tj} \quad (5)$$

4.2 Proposed algorithm:

We developed a load balancing algorithm whose objective is to reduce the makespan time and task rejection ratio in cloud environment. We have created N number of task with different length (20000 to 400000 MI) and m number of virtual machine. Sort the task based on deadline (algorithm step 1 to 4). Some tasks have execution time more than their deadline. These types of tasks are discarded in cloud environment, if more tasks are discarded then performance start to degrade. Therefore developed algorithm balanced the load at all VM and increase the ratio of task meet with deadline using the elasticity concept (provisioning and deprovisioning of resources) as shown in algorithm 1. Firstly calculate the number of task unable to meet deadline in each interval after that average of

Algorithm 1 for Load Balancing with resource provisioning and deprovisioning

DL[i] deadline array and T[i] task array

1. ∀ i€ DL[i]
2. ∀ j€ i+1 to DL[i], if (DL[j]<DL[i]) # if condition is true go to step 3&4
3. Swap DL[j] with DL[i]
4. Swap T[j] with T[i] # otherwise j=j+1

UVM =under loaded virtual machine, OVM =overloaded virtual machine , TotalTask=N,
T_{rej} = number of rejected task in a interval, AT_{rej}= avg. number of task unable to meet deadline

5. Start the loop for Task 0 to N-1
6. Start loop for virtual machine 0 to m-1 # start to allocate the tasks
7. EPT[j]=$LT_{Ti}/p_s * q_n$ #EPT is expected processing time, End the for loop of virtual machine
8. Calculate the makespan time based on deadline
   For j=1 to m EPT[j]=EPT[j]+MET[j];
   # EPT[j] array represent the makespan time of task End for loop of VM
9. Find the minimum value of EPT[] ,
10. Compare the deadline of task with calculated execution time
11. If(DT_{Ti}>min EPT) then Assigned the task to VM. End for loop of VM;
    Otherwise T_{rej} ++; End for loop of Tasks
12. Start Task and VM for loop
13. Load increase at resource R_j after task assignment
    $W_{Rj} = W_{Rj} + ET_{Ti}$
14. Calculate the AT_{rej} in last z iteration

15. If (AT_{rej} >=N *.30), Increase VM by 20%.
    Else If (AT_{rej} > N * .1) Increase VM by 10%.
    Else ($AT_{rej} \leq$ N * .1) No need to increase the VM, End of for loop; End of for loop
16. Calculate UVM and OVM and sort the VM increasing and decreasing order
17. Calculate the UVM_{avg} in last z iterations
18. If (UVM_{avg}>.3*R_j), Reduce 20% VM of total running VM.
    Else If (UVM_{avg}> .1*R_j), Reduce 10% VM of total running VM
    Else ($UVM_{avg} \leq$ .1*R_j) No need to reduce the running VM,
19. End for loop; End for loop

rejected task (AT_{rej}) in last z interval is calculated and apply the user defined threshold conditions as per SLA. If value of AT_{rej} is more than or equal to the value of 30% of total tasks then add new VM 20%. If value of AT_{rej} is more than 10% of total tasks then add 10% new VM. If AT_{rej} is less than or equal to10% then there is no need to add the new VM. Therefore calculate the parameter AT_{rej} that is based upon recent (last) z interval. Further calculate the overloaded and underloaded VM after assigning the task to virtual machine based upon the defined threshold value and capacity of VMs. A VM is in overloaded mode if they utilize their capacity more than or equal to 90% and underloaded if utilize their capacity less than 20%. If any virtual machine is overloaded or under loaded condition then sorts VM in decreasing and increasing order and transfer the task from OVM to UVM. If average of under loaded virtual machine is greater than the 30 % of all available virtual machine then decrease the virtual machine 20 % for next interval. If it is more than 10% then decrease the virtual machine 10% for next interval shown in algorithm step 17to 20.

5. Analysis and Comparison of Experimental Results

We proposed and implement the load balancing algorithm for minimize the makespan time and increase the ratio of task to meet deadline using the cloudsim platform.

5.1 Makespan Time Calculations

We run the simulation more than one hour (approximate 100 times) on different number of task with random length cloudlet (tasks) and calculate the result using the space shared policy in cloudsim. Consider 10 virtual machine with bandwidth 1000 mbps, number of cpu for each virtual machine is 1. Consider the range of task is 10 to 30 at 10 virtual machine, 20 to 50 at 20 virtual machine and 100 tasks at 30 virtual machine, length of task is varying from 20000MI to 400000 MI. Computational results shows that proposed algorithm reduce the makespan time compared to FCFS, SJF and Min-Min algorithm as shown in Table 1.

Table 1 makespan time comparison

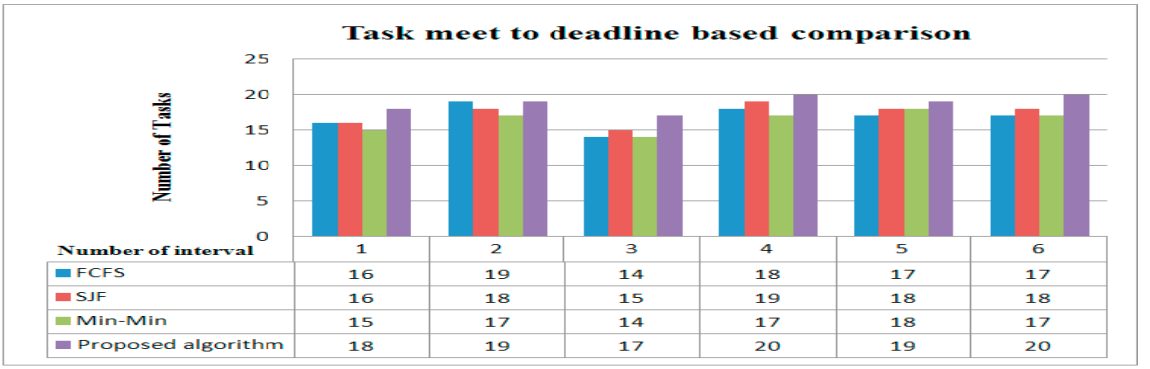
Task	Virtual machine	FCFS	SJF	Min-Min	Proposed algorithm
10	10	736	779	755	580
15	10	1081	830	784	603
20	10	1249	850	836	795
30	10	1486	1244	1147	1013
20	20	544	521	497	451
30	20	844	712	656	597
40	20	1061	898	824	752
50	20	1259	1084	1046	916
100	30	1489	1138	1058	970

5.2 Number of task meets to deadline

We computed the performance metric for analyzing the completion of tasks on or before the deadline specified by the user. Created a window of task in which 20 tasks with random length is sending continuously after 5 second interval. Created 10 VM to process the upcoming task and deadline of task is generated randomly. Developed load balancing algorithm considers the deadline as an important factor and find out the best virtual machine for the task so that task can be executed before the deadline expired. We run the simulation and calculated results shows that proposed algorithm complete more tasks before deadline rather than existing FCFS, SJF and Min-Min scheduling algorithm shown in Figure 2. The percentage of task meeting with deadline is approximate average 95% in proposed algorithm, 85% in FCFS, 86 in SJF and 81% in Min-Min algorithm. Further find out the results after increasing the task from 20 to 50 and VM from 10 to 30. Calculated results shows that proposed algorithm perform better than existing algorithm shown in Fig. 3.

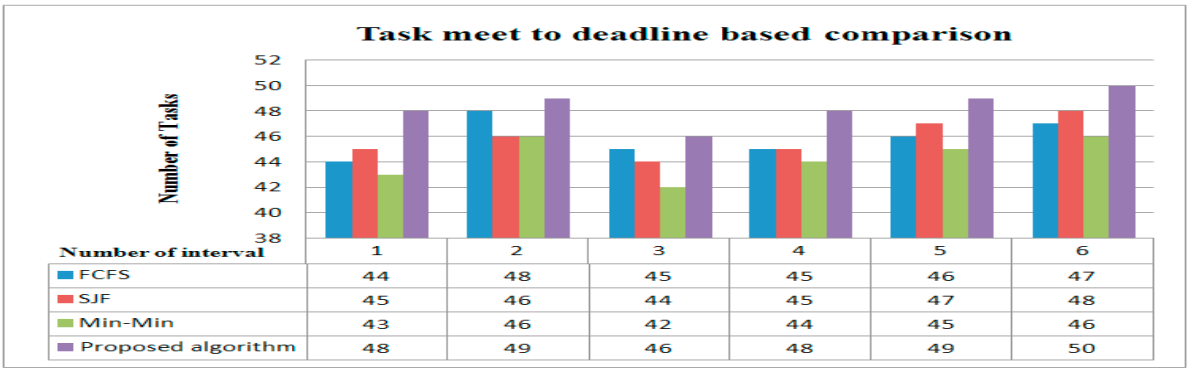
5.3 Provisioning and deprovisioning of resource

Elasticity is the degree to which a system is able to adapt to workload changes by provisioning and de-provisioning resources in an autonomic manner. If load is increased or decreased at host is above or below the threshold limit



Number of interval	1	2	3	4	5	6
FCFS	16	19	14	18	17	17
SJF	16	18	15	19	18	18
Min-Min	15	17	14	17	18	17
Proposed algorithm	18	19	17	20	19	20

Fig. 2 task meet to deadline comparison between proposed algorithm vs FCFS,SJF, Min-Min



Number of interval	1	2	3	4	5	6
FCFS	44	48	45	45	46	47
SJF	45	46	44	45	47	48
Min-Min	43	46	42	44	45	46
Proposed algorithm	48	49	46	48	49	50

Fig. 3 task meet to deadline comparison between proposed algorithm vs FCFS,SJF,Min-Min

then controller node call the ELB component to add or remove the virtual machine in cloud environment. In case I, send the 20 task with deadline at 10 VM. Run the simulation and plot the graph is shown in Figure 4 represent that only 17 tasks are accepted out of 20 tasks in first iteration. Controller node found that task rejection ratio is more than 10% therefore controller node passes the control to CRP for increasing the 10% VM. Now 11VM process the 20 upcoming tasks. Task rejection ratio is 10% or less than 10% for next iteration so there is no need to increase the virtual machine. This process is continuing until all the upcoming task has not finished.

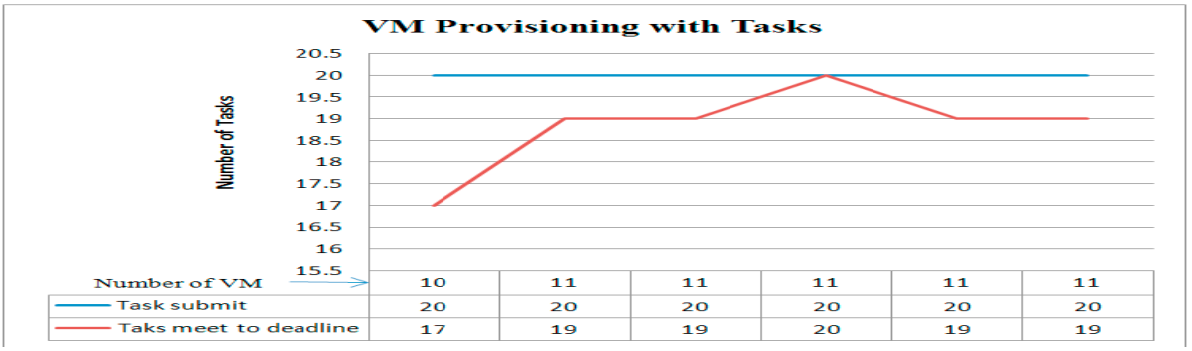


Figure 4 scale out by proposed algorithm in cloud environment

Consider II case in which increase the task 20 to 30 and start to process at 10 VM. More than 10% tasks (22 meet to deadline out of 30) are rejected. CRP increases the VM instance as per proposed algorithm. In next iteration 23 tasks meet with deadline and 7 tasks are rejected, average task rejection ratio is more than 10%, again 10% VM is increase. This process is continuing until AT_{rej} become 10 or less than 10%. Elasticity is achieved by algorithm shown in Figure 5.

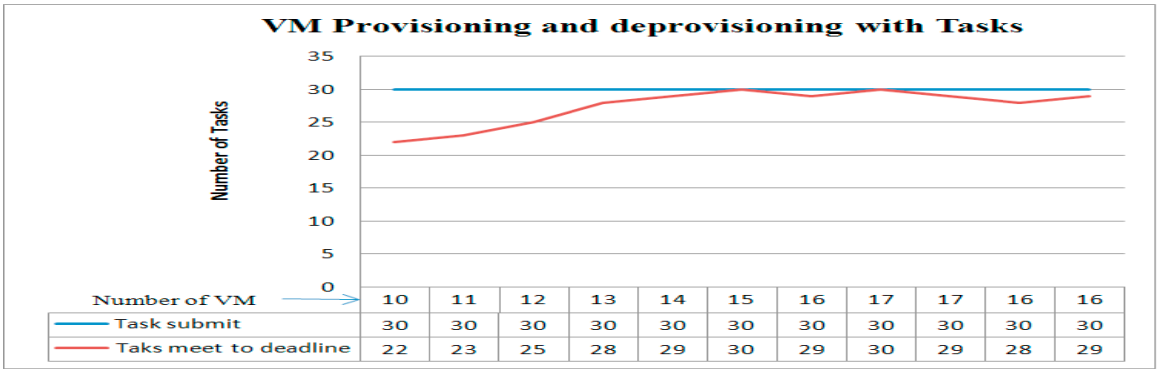


Figure 5 Elasticity in cloud environment by proposed algorithm

6. Conclusion

To achieve minimum makespan time and increase the ratio of tasks meet to the deadline in cloud environment, we have developed a load balancing algorithm in this paper considering deadline as QoS parameter. Develop algorithm is making the autonomic decision about the scale out and scale in based upon the upcoming demands and user defined threshold value. Proposed architecture contains the component like scheduler, workload analyzer, ELB, cloud resource provisioning and deprovisioning. Experimental results shows that under all possible condition proposed algorithm reduce the makespan time as well as task rejection ratio. The results have proven that proposed algorithm perform better scale out and scale in and increase the acceptance ratio of task than other existing conventional algorithm like FCFS, SJF and Min-Min shown in Table 1& Figure 2-5.

References

1. P.Sareen, "Cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud," International Journal of Advanced Research in Computer Science and Software Engineering 3.3 (2013).
2. Hwang, Kai, Yue Shi, and Xiaoying Bai, "Scale-Out vs. Scale-Up Techniques for Cloud Performance and Productivity," Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on IEEE, 2014.
3. D.Coninck, Elias, et al. "Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds," Journal of Systems and Software 118 (2016): 101-114.
4. M.Kumar, S.C. Sharma, Priority Aware Longest Job First (PA-LJF) Algorithm for Utilization of the Resource in Cloud Environment, INDIACom, 2016. pp. 415–420.
5. Juarez, Fredy, Jorge Ejarque, and Rosa M. Badia. "Dynamic energy-aware scheduling for parallel task-based application in cloud computing." Future Generation Computer Systems (2016).

6. Feng, Ye, et al. "A novel cloud load balancing mechanism in premise of ensuring QoS." Intelligent automation & soft computing 19.2 (2013): 151-163.
7. Abrishami, Saeid, and Mahmoud Naghibzadeh. "Deadline-constrained workflow scheduling in software as a service cloud." Scientia Iranica 19.3 (2012): 680-689.
8. Abrishami, Saeid, Mahmoud Naghibzadeh, and Dick HJ Epema. "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds." Future Generation Computer Systems 9.1 (2013): 158-169.
9. Naha, Ranesh Kumar, and Mohamed Othman. "Brokering and load-balancing mechanism in the cloud–Revisited." IETE Technical Review 31.4 (2014): 271-276.
10. Somasundaram, Thamarai Selvi, et al. "A broker based architecture for adaptive load balancing and elastic resource provisioning and deprovisioning in multi-tenant based cloud environments." Proceedings of International Conference on Advances in Computing. Springer India, 2013.
11. Li, Xiaofang, et al. "An improved max-min task-scheduling algorithm for elastic cloud." *Computer, Consumer and Control (IS3C), 2014 International Symposium on*. IEEE, 2014.
12. Coutinho, Emanuel Ferreira, Danielo Gonçalves Gomes, and José Neuman de Souza. "An analysis of elasticity in cloud computing environments based on allocation time and resources." *Cloud Computing and Communications (LatinCloud), 2nd IEEE Latin American Conference on*. IEEE, 2013.
13. Al-Dhuraibi, Yahya, et al. "Elasticity in Cloud Computing: State of the Art and Research Challenges." *IEEE Transactions on Services Computing* (2017).
14. Galante, Guilherme, and Luis Carlos E. de Bona. "A survey on cloud computing elasticity." *Utility and Cloud Computing (UCC), 2012 IEEE Fifth International Conference on*. IEEE, 2012.
15. Hu, Yazhou, et al. "Workload prediction for cloud computing elasticity mechanism." *Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on*. IEEE, 2016.