ISSN 2319 - 6629
Volume 4, No.3, April - May 2015
International Journal of Wireless Communications and Networking Technologies
Available Online at http://warse.org/pdfs/2015/ijwcnt02432015.pdf

A Weighted Mean Time Selective Scheduling Strategy for Load Balancing of Independent Tasks in Cloud Environment

Narendrababu Reddy.G[1], Dr. S. Phani Kumar[2],

[1]Asst.Prof.,GNITS, Research Scholar, GITAM,Hyderabad,gnbreddy25@gmail.com
[2]Prof. & HOD, CSE Department, GITAM University, Hyderabad, phanikumar.s@gitam.edu

ABSTRACT

Cloud computing nowadays becomes quite popular among a community of cloud users by offering a variety of resources. In cloud computing environments, resources and infrastructure are provided as a service over internet on demand. These resources need to be provisioned to the end users in most efficient manner to satisfy the SLAs. Load balancing of independent tasks in cloud computing environment is a crucial issue of resources allocation to the user requirements. In this paper we are presenting a new heuristic scheduling strategy for allocation of cloud resources to end users tasks on demand basis, which aims to achieve well balanced load across virtual machines for maximizing the throughput. This algorithm uses certain heuristics to select between two algorithms so that overall make-span of tasks on the machines is minimized. We evaluate our provisioning heuristics by comparing with existing load balancing and scheduling algorithms. Our approach illustrates that overall make-span of tasks on given set of VMs minimizes significantly in different scenarios.

Key Words— Cloud computing, Load balancing, Make-span, Min-Min, Max-Min.

1. INTRODUCTION

With the rapid development of processing and storage technologies and evolution of internet, computing resources have become cheaper, more powerful and more ubiquitously available than ever before. This technological trend has enabled the realization of a new computing model called Cloud Computing, in which resources are provided as general utilities that can be leased and released by users through the internet in an on-demand fashion.

According to Rajkumar Buyya et al.[1] "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers." Computing is being transformed to a model consisting of services that are commoditized and delivered in a manner similar to traditional utilities such as water, electricity, gas and telephone. In such a model, users access services based on their requirements without regard to where the services are hosted or how they are delivered.

Cloud computing [2] is an entirely internet based approach where all the applications and files are hosted on a cloud which consists of thousands of computers interlinked together in a complex manner. Cloud computing incorporate concepts of parallel and distributed computing to provide shared resources: hard ware, software and information to computers are other devices on demand. The end users can use these resources over a network on-demand basis as a "pay- per- use" model. The customer is interested in reducing the overall execution time of tasks on the machines. This leads to problems in scheduling of customer tasks within available resources.

In cloud platforms, load balancing (or resource allocation) takes place at two levels. First, when an application is uploaded to the cloud, the load balancer assigns the required instances to physical computers. Second, when an application receives multiple incoming requests, these requests should be each assigned to a specific application to balance the computational load across a set of instances of the same application. Load balancing must take into account the consideration of all kinds of cloud users. It must fulfill the requirements of all its users. The cloud users expect their jobs to be completed in fastest possible manner with high availability of resources. On the other hand the cloud provider invests such a huge capital with the aim of maximum utilization of all the installed resources in efficient and effective manner. Both cloud users and cloud providers except maximum throughput and improved performance for the money they invest. The main objective of the load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way[13].

Motivated by this problem, we propose a new load balancing algorithm for cloud systems to allocate resources to tasks using min-min or min-max algorithm and then scheduling them on either space shared or time shared basis. The rest of this paper is organized as follows. The second section describes the related works on existing load balancing techniques. Section three describes the proposed load balancing algorithm. Section four presents experimental results along with performance evaluation of the algorithm in comparison with existing algorithms followed by concluding remarks in section five.

2. RELATED WORKS

Load balancing [3] is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded and some others are under loaded. The goal of load balancing is improving the performance by balancing the load among the various resources to achieve optimal resource utilization, maximum throughput, shortest response time and avoiding overload. With proper load balancing, resource consumption can be kept to a minimum which will further reduce energy consumption.

To distribute load on different systems we use generally traditional algorithms like those used in web servers, but these algorithms do not always give the expected performance with large scale and distinct structure of service oriented data centers [4]. To overcome the shortcomings of these algorithms, load balancing has been widely studied by researchers and implemented by computer vendors in distributed systems.

In general load balancing algorithms follow two major classifications [5]:

- Depending on how the charge is distributed and how processes are allocated to nodes(system load);
- Depending on the information status of the nodes (system topology).

In the first case it designed as central approach, distributed approach or hybrid approach, in the second case as static approach, dynamic or adaptive approach [6].

A. Classification according to the System Load

a) Centralized approach: In this approach, a single node is responsible for managing the distribution within the whole system.

b) Distributed approach: In this approach, each node independently builds its own load vector by collecting the load information of other nodes. Decisions are made locally using local load vectors. This approach is more suitable for widely distributed systems such as cloud computing.

c) Mixed approach: A combination of above two approaches to take advantage of each approach.

B. Classification according to the System Topology

a) Static approach: This approach is generally defined in the design or implementation of the system.

b) Dynamic approach: This approach takes into account the current state of the system during load balancing decisions. This approach is more suitable for widely distributed systems such as cloud computing.

c) Adaptive approach: This approach adapts the load distribution to system status changes, by changing their parameters dynamically and even their algorithms. This approach is able to offer better performance when the system state changes frequently [6], [7].

To achieve the load balancing in cloud computing environment, so many resource provisioning algorithm have been proposed based on various factors like spatial distribution of cloud nodes, storage/replication, point of failure, algorithm complexity etc.In resource sharing environments, such as grids and clouds, a privileged resource management system is designed to manage how these resources are used.

Houle et al.[8] consider algorithms for static load balancing on trees treating that the total load is a fixed one. In [9], a New Time Optimizing Probabilistic Load Balancing Algorithm in Grid Computing is presented. This algorithm chooses the resources based on better past status and least completion time. The main purpose of this algorithm is to establish load balancing and reduce response time. In [10], a Task Load Balancing Strategy for Grid Computing is presented. In this, a hierarchical load balancing strategy and associated algorithms based on neighborhood property is discussed. This strategy privileges local balancing first i.e load balance within sites without communication between sites, then upper hierarchical balancing will take place and so on. Yagoubi proposed in [11], a hierarchical load balancing model as a new frame work to balance computing load in a GRID. This model suffers from communication bottlenecks.

3. PROPOSED ALGORITHM

To achieve our goal of minimizing the overall make-span of tasks on virtual machines and provide better quality of services, we design an algorithm that assigns tasks to best machines in such a way that it provides satisfactory performance to both, cloud users and providers. The algorithm is designed to make choice between Min-Min and Max-Min scheduling algorithm based on certain criteria.

Min-Min algorithm follows the following procedure: Phase1: It first computes the completion time of cloudlets on each VM and then for each cloudlet it chooses the VM which process the cloudlet in minimum possible time. Phase2: Then among all the cloudlets in MT the cloud let with minimum overall completion time is chosen and is allocated to VM on which minimum execution delay is excepted. The cloudlet is removed from the list of MT and the process continues until MT list is empty.

Max-Min algorithm follows the following procedure: Max-Min algorithm works similarly like Min-Min in phase 1. In phase 2, Max-Min allocates the cloudlet with maximum expected completion time to VM, i.e longer task is first allocated a VM. Max-Min algorithm allocates considerable long task (cloudlet) to one VM and smaller tasks to another VM based on their completion time. Min-Min on the other hand is useful when all the tasks are almost of same size. When all the tasks are of similar sizes then Min-Min allocates the cloudlet to a VM on which it executes in minimum possible time. This not only ensures that overall make-span of tasks on VM is reduced but also provide minimum delays in processing of tasks.

Using Cloud Sim[12] tool a simulation set-up was prepared to better understanding the working of resource allocation and task scheduling algorithms with the aim to

achieve load balancing in cloud environment. In the simulation tasks are modelled as cloudlets and machines are modelled as VMs on which cloudlets are executed and cloud itself is modelled as a data centre.

Let 'V' denotes the number of VMs and 'T' denotes the number of tasks. Execution time of 'i^{th}' task on 'j^{th}' VM is calculated using:

$$Exe_{ij} = \frac{filesize[i]}{MIPS[j]} \quad \text{---------- (1)}$$

All the tasks will be sorted according to minimum execution time.

Expected completion time of task on a VM is calculated using :

$$Comp_{ij} = Exe_{ij} + Wait_j \quad \text{---------- (2)}$$

'$Wait_j$' is the time for which 'i^{th}' task has to wait for 'j^{th}' VM to get ready.

Average completion time:

$$Average\_Comp_{ij} = \frac{\sum_{i=1}^{T}(comp[ij])}{T} \quad \text{---------- (3)}$$

Now to select between Max-Min and Min-Min, standard deviation (SD) is used.

$$SD = \frac{\sqrt{\Sigma(Comp[ij]-Average\_Comp[ij])}^2}{\sqrt{T}} \quad \text{--------- (4)}$$

Then, the sorted task list is searched for a location where the difference between two consecutive values of '$Comp_{ij}$' of tasks is sufficiently large as compared with other tasks, using the relation: $Comp_{ij}$ > SD. Based upon location following methods are invoked:

Case1: If the location is present in the upper half of the list then it implies that the list has greater number of sufficiently long tasks as compared to short tasks(i.e tasks of similar sizes are greater). In this case Min-Min outperforms Max-Min and hence Min-Min is invoked.

Case 2:if the location lies in the lower half of the list then it implies that list contains large number of short tasks but fewer long tasks. In this case Max-Min outperforms Min-Min. Hence Max-min is invoked.

Case 3: If no such location exist, that implies that all tasks are of almost same sizes with no major difference between any two tasks sizes (i.e difference < SD). In this case following conditions are used:

- If(SD< Average_$Comp_{ij}$), then the list contains all small size tasks. In this Min-Min is executed to allocate a VM to task.
- If(SD>= Average_$Comp_{ij}$), then Max-Min is executed to allocate a VM to task.

After allocation of a VM to a task that task is removed from the list of meta tasks and the process continues until all tasks are allocated to a VM.

Algorithm:

1.Inputs: Tasks sizes and MIPS of VM.
2. **for** all Tasks T in meta-tasks [MT]
   for all VMs
   $Exe_{ij} = \frac{filesize[i]}{MIPS[j]}$
3. Sort all tasks in MT in ascending order of their execution time.
4. **While** MT is not empty
   for all Tasks T in meta-tasks [MT]
   for all VMs
   $Comp_{ij} = Exe_{ij} + Wait_j$
5. **for** all Tasks T in meta-tasks [MT]
   find the task with minimum completion time and the resource that process it in minimum time.
6. **If** more than one resource obtains this minimum
   Then
   Select the VM which has been first initiated on FCFS basis.
7. Calculate SD using equation (4)
8. Find out location 'L' in MT where the difference between two consecutive values of $Comp_{ij}$ > SD
9. **IF** 'L' is present in the upper half or SD < Average_$Comp_{ij}$
   Then
   Allocate VM to task using Min-Min.
   Else
   Allocate VM to task using Max-Min.
10. Remove allocated task from MT

End While.

4. EXPERIMENTAL RESULTS

Performance Metrics: Make span of given cloudlets on given set of VMs is used as the performance metrics in this resource provisioning technique in cloud computing environment. Throughput of the heterogeneous system is the measure of the make span. Lesser the value of the make span of allocation algorithm better is the resource utilization.

To evaluate and compare the proposed strategy, a simulation environment has been setup using CloudSim[12]. The proposed policy is simulated in the following environment:

a) Cloudlets (tasks) are attributed by their file size.

b) VM capabilities are defined in terms of MIPS

c) The environment is static i.e list of tasks , resources and attribute values given before simulation.

The completion time of each task is calculated in the simulated environment in milliseconds.

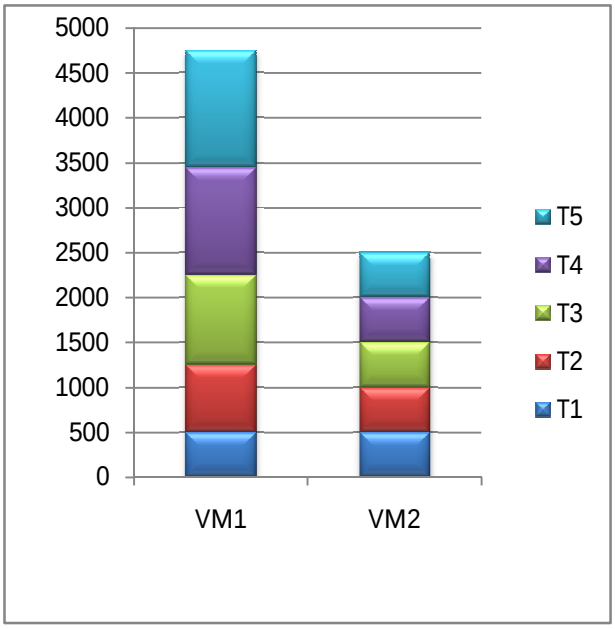

Figure 1: Make span of tasks without proposed algorithm

Figure 1 shows the overall make span of five cloudlets (Tasks T1,T2,T3,T4 & T5) on two VMs of different capacities without using proposed algorithm.

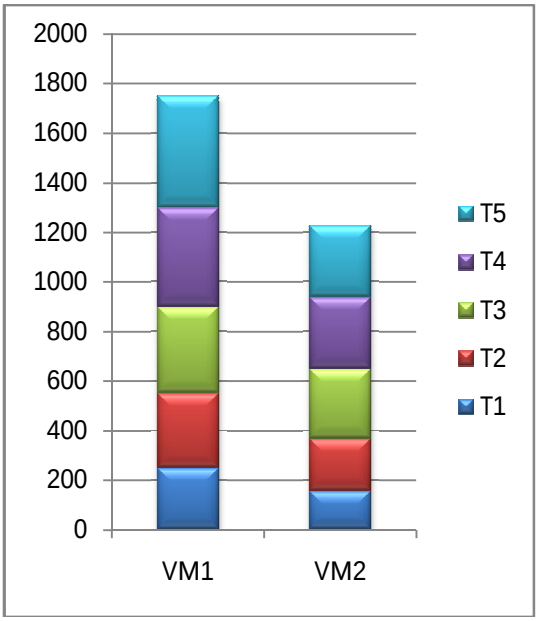

Figure 2: Make span of tasks with proposed algorithm

Figure 2 shows the overall make span of five cloud lets with proposed algorithm.

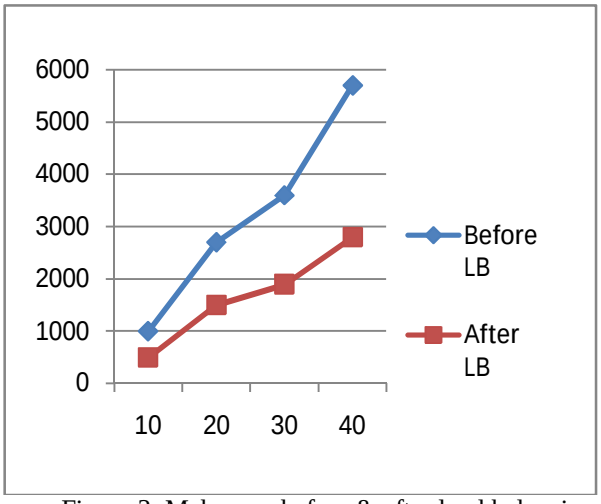

Figure 3: Makespan before & after load balancing.

Figure 3 illustrates the comparison of makespan before and after load balancing using the proposed weighted mean time selective algorithm. The X-axis represents the number of tasks and the Y-axis represents the makespan in milliseconds. With the proposed algorithm the makespan is reduced considerably. With more number of tasks, the difference in make span time is quite high and the algorithm provides the best results.

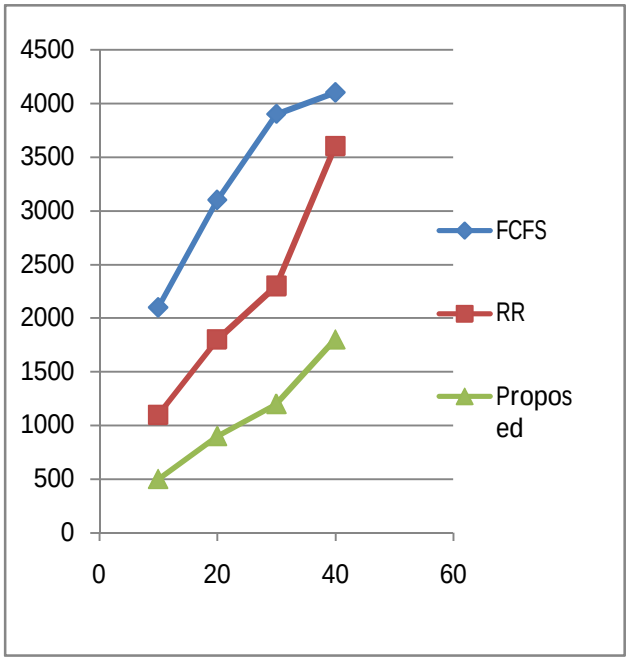

Figure4: Comparison with different algorithms.

Figure 4 shows the comparison of make span for FCFS, RR and proposed algorithms. The X-axis shows the number of tasks and the Y-axis shows make span in milliseconds. It is clearly evident from the graph that the weighted mean time algorithm is more efficient when compared with other regular algorithms.

5. CONCLUSION

Resource allocation on the cloud aims at avoiding under utilization of resources by balancing the nodes with appropriate tasks. The main objective of the load balancing algorithms is to utilize the resources effectively to reduce the make span i.e the overall completion time of the tasks. The algorithm proposed in this paper ensures that all resources are efficiently utilized and tasks are executed with minimum make span when compared with other existing algorithms. In future, we plan to extend this algorithm to jobs with dependent tasks i.e for work flow applications.

REFERENCES

[1] Rajkumar Buyya, C.S.Yeo,S.Venugopal,J.Broberg, I.Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility" in : Future Generation Computer Systems vol. 25, 2009, pp.599-616

[2] Qi Zhang, Lu Cheng, Raouf Boutaba, "Cloud computing: state of the art and research challenges" in: J Internet Serv Appl 2010, pp.7-18

[3] K.Ramana, A.Subrayanam and A.Ananda Rao, "Comparative Analysis of Distributed Web Server System Load Balancing Algorithms Using Qualitative Parameters"' VSRD-IJCSIT, vol.1(8),2011, pp.592-600.

[4] Yi Lu, Qiaomin Xie, A.Geller, J R.Larus,A.Greenberg: "A Novel Load Balancing Algorithm for Dynamically Scalable Web Services, IFIP PERFORMANCE 2011, 29th International Symposium on Computer Performance, Modelling, Measurements and Evaluation 2011, 18-20 October 2011, Amsterdam, Netherlands.

[5] Elarbi Badidi, "Architecture of service oriented distribution of objects". Doctoral Thesis 20 July 2000.

[6] N.Shivaratri, P.Krueger and M.Singhal, "Load distributing for locally distributed systems", IEEE Computer 25(2),pp.33-44, December 1992.

[7] T.L Casavant and J.G Kuhl, "A Taxonomy of Scheduling in General Purpose Disrtributed Computing Systems". IEEE Transcations on Software Engineering,14(2),pp.141-154, February 1988.

[8] M.Houle, A.Symnovis, D.Wood, Dimension-exchange algorithms for load balancing on trees, in proc. Of 9th Int. Colloquium on Structural Information and Communication Complexity, Andros, Greece, June 2002, pp 181-196.

[9] M.Moradi, M.A. Dezfuli,, M.H.Safavi, Department of Computer and IT Engineering, Amirkabir University of Technology ,Tehran, Iran, A New Time Optimizing probabilistic Load Balancing Algorithm in Grid Computing IEEE 978-1-4244-6349-7/10. 2010.

[10] B. Yagoubi, Y.Slimani, Taski load balancing strategy for Grid Computing, Journal of Computer Science 3(3) (2007) pp186-194

[11] B. Yagoubi, Distributed load balancing model for Grid Computing, in : African Conference on Research in Computer Science and Applied Mathematics, October, 2008,pp631-638.

[12] R.N. Calheiros, R.Ranjan, C.A.F.D.Rose, R.Buyya, CloudSim: A Novel frame work for modeling and simulation of cloud computing environments and evaluation of resources provisioning algorithms, Software:Practice and Experience 41 (2011) 23-50.

[13] D.L Eager, E.D. Lazowska, J.Zahorjan, Adaptive load sharing in Homogeneous distributed systems, The IEEE Transactions on Software Engineering 12(5),1986, 662-675.

[14] G.Narendrababu Reddy, S.Phani Kumar, Review of Load Balancing Techniques in Cloud Computing Environment: Challenges and Algorithms, International Journal of Advanced Research in Computer Science, Vol 5, No.4, (2014) pp 157-162.