IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 4, Ver. III (Jul – Aug. 2014), PP 59-69
www.iosrjournals.org

Dynamic Load Balancing For Cloud Computing Using Heuristic Data and Load on Server

Vinay Darji, Jayna Shah, Rutvik Mehta

Abstract: *Cloud computing is emerging trend in Information Technology community. Cloud resources are delivered to cloud users based on the requirements. Because of the services provided by cloud, it is becoming more popular among internet users. Hence, the number of cloud users is increasing day by day. Because of this, load on the cloud server needs to be managed for optimum resource utilization. This research proposes new load balancing algorithm which considers parameter like weight of each task, execution time of each task, current load and future load on the server. Current load balancing algorithms don't consider these parameters together for load balancing. Proposed scheme selects the best node based on these parameters to achieve optimum use of resources.*

Index terms: *Cloud computing, load balancing, min-min algorithm, resource utilization*

I. Introduction

Cloud computing is a concept used to describe a variety of computing concepts that involve a large number of computers connected through a real-time communication network such as the Internet. In science, cloud computing is a synonym for distributed computing over a network, and means the ability to run a program or application on many connected computers at the same time. The phrase also more commonly refers to network-based services, which appear to be provided by real server hardware, and are in fact served up by virtual hardware, simulated by software running on one or more real machines. Such virtual servers do not physically exist and can therefore be moved around and scaled up (or down) on the fly without affecting the end user.[1] In generalized way it describes how organization take their existing IT infrastructure and hand over to skilled person who can precisely build or manage it so that organization can achieved their goal by focusing on new ways and techniques that help the business at hand rather than becoming expert in building servers, managing storage or protecting data. Some of the most prominent examples of cloud provider are Amazon EC2, Google App Engine, Google apps etc. Developers, on a whole, can obtain the advantages of a managed computing platform, without having to commit resources to design, build and maintain the network. On profound studies, some of the important problems of cloud were examined, like data security, data loss, load balancing etc.

In cloud platforms, resource allocation (or load balancing) takes place Marjory at two levels.

At First Level: When an application is uploaded to the cloud, the load balancer assigns the requested instances to physical computers, attempting to balance the computational load of multiple applications across physical computers.

At Second Level: When an application receives multiple incoming requests, each of these requests must be assigned to a specific application instance, to balance the computational load across a set of instances of the same application [2]. One of the major issue is load balancing. Load balancing [3] is a computer networking method for distributing workloads across multiple computing resources, such as computers, a computer cluster, network links, central processing units or disk drives. Load balancing aims to optimize resource use, maximize throughput, minimize response time, and avoid overload of any one of the resources. Using multiple components with load balancing instead of a single component may increase reliability through redundancy. In Cloud Computing, if more load will be consigned on nodes of cloud, on gradually increasing the number of users and if the cloud provider is not configured with any good mechanism for balancing then there would be improper load balancing and capacity of cloud servers would not be utilized properly. This will confiscate or seize the performance of heavy loaded node. If some good load balancing technique is implemented, it will equally divide the load (here term equally defines low load on heavy loaded node and more load on node with less load now) and thereby we can maximize resource utilization. [4][5]

II. Related work

In current situation, every organization is propagating towards the use of cloud computing. Number of users for cloud computing are increasing day by day. During that period of time cloud provider will need to maintain more effective load balancing and do the efficient resource management than the current scenario. And it is costly to buy new hardware just to balance load instead of fully utilizing current resources. Thus a better load balancing system is needed to handle much load and to maximize utilization of current resources. There are

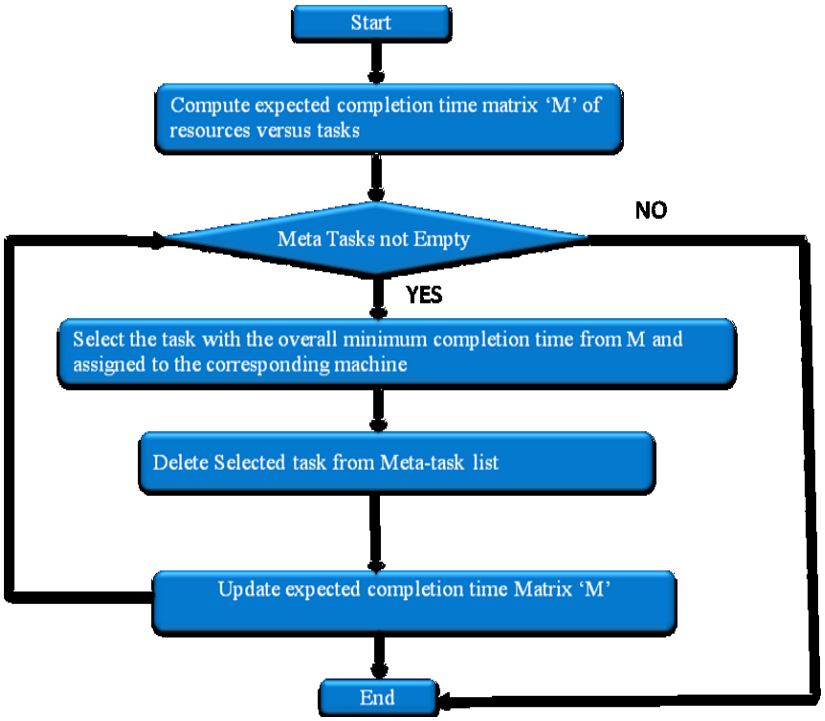
several static and dynamic type of Load Balancing Algorithms on which various researches have been made. According to historical data and current state of the system and through genetic algorithm, this strategy computes ahead the influence it will have on the system after the deployment of the needed VM resources and then chooses the least-affective solution, through which it achieves the best load balancing and reduces or avoids dynamic migration. This strategy solves the problem of load imbalance and high migration cost by traditional algorithms after scheduling. Experimental results prove that this method is able to realize load balancing and reasonable resources utilization both when system load is stable and variant.[6][9] In the paper[10], the datacenter controller assigns the requests to a list of VMs on a rotating basis. The first request is allocated to a VM- picked randomly from the group and then the DataCenter controller assigns the subsequent requests in a circular order. Once the VM is assigned the request, the VM is moved to the end of the list. In this algorithm, there is a better allocation concept known as Weighted Round Robin Allocation in which one can assign a weight to each VM so that if one VM is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the DataCenter Controller will assign two requests to the powerful VM for each request assigned to a weaker one. The major issue in this allocation is this that it does not consider the advanced load balancing requirements such as processing times for each individual requests. In Throttled algorithm[10] the client first requests the load balancer to find a suitable Virtual Machine to perform the required operation. The process first starts by maintaining a list of all the VMs each row is individually indexed to speed up the lookup process. If a match is found on the basis of size and availability of the machine, then the load balancer accepts the request of the client and allocates that VM to the client. If, however there is no VM available that matches the criteria then the load balancer returns -1 and the request is queued. Another technique is spread spectrum technique in which the load balancer spread the load of the job in hand into multiple virtual machines. The load balancer maintains a queue of the jobs that need to use and are currently using the services of the virtual machine. The balancer then continuously scans this queue and the list of virtual machines. If there is a VM available that can handle request of the node/client, the VM is allocated to that request. If however there is a VM that is free and there is another VM that needs to be freed of the load, then the balancer distributes some of the tasks of that VM to the free one so as to reduce the overhead of the former VM. Figure2. better explains the working of the ESCE algorithm. The jobs are submitted to the VM manager, the load also maintains a list of the jobs, their size and the resources requested. The balancer selects the job that matches the criteria for execution at the present time. Though there algorithm offers better results as shown in further section, it however requires a lot of computational overhead[7][10]. CARTON[8][11] is used for cloud control that unifies the use of LB and DRL. LB (Load Balancing) is used to equally distribute the jobs to different servers so that the associated costs can be minimized and DRL (Distributed Rate Limiting) is used to make sure that the resources are distributed in a way to keep a fair resource allocation. DRL also adapts to server capacities for the dynamic workloads so that performance levels at all servers are equal. With very low computation and communication overhead, this algorithm is simple and easy to implement. Central Load Balancing Policy[12][13] is used for Virtual Machines (CLBVM) that balances the load evenly in a distributed virtual machine/cloud computing environment. This policy improves the overall performance of the system but does not consider the systems that are fault-tolerant. Task Scheduling based on LB[10][14] is a two-level task scheduling mechanism based on load balancing to meet dynamic requirements of users and obtain high resource utilization. It achieves load balancing by first map-ping tasks to virtual machines and then virtual machines to host resources thereby improving the task response time, resource utilization and overall performance of the cloud computing environment. Another algorithm is **Min-min algorithm**

The Min-min heuristic begins with the set U of all unmapped tasks. Then, the set of minimum completion times, M, for each ti U, is found. Next, the task with the overall minimum completion time from M is selected and assigned to the corresponding machine (hence the name Min-min). Last, the newly mapped task is removed from U, and the process repeats until all tasks are mapped (i.e., U is empty). Min-min is based on the minimum completion time, as is MCT. However, Min-min considers all unmapped tasks during each mapping decision and MCT only considers one task at a time. Min-min maps the tasks in the order that changes the machine availability status by the least amount that any assignment could. Let ti be the first task mapped by Min-min onto an empty system. The machine that finishes ti the earliest, say mj, is also the machine that executes ti the fastest. For every task that Min-min maps after ti, the Min-min heuristic changes the availability status of mj by the least possible amount for every assignment. Therefore, the percentage of tasks assigned to their first choice (on the basis of execution time) is likely to be higher for Min-min than for Max-min. The expectation is that a smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest. The algorithm is as follows:

Step 1: [calculate the makespan of each node based on execution time]
Expected execution time:

$$E_{ij} = IV_i / PS_j$$

Where i= task
j= resource
IV= Instruction Volume
PS= Processing Speed
Apply min-min task scheduling algorithm to calculate makespan.



Theoretical Analysis

Task	IV(MI)	DV(Mb)
T1	100	50
T2	200	60
T3	300	70
T4	200	80
T5	100	50
T6	400	40

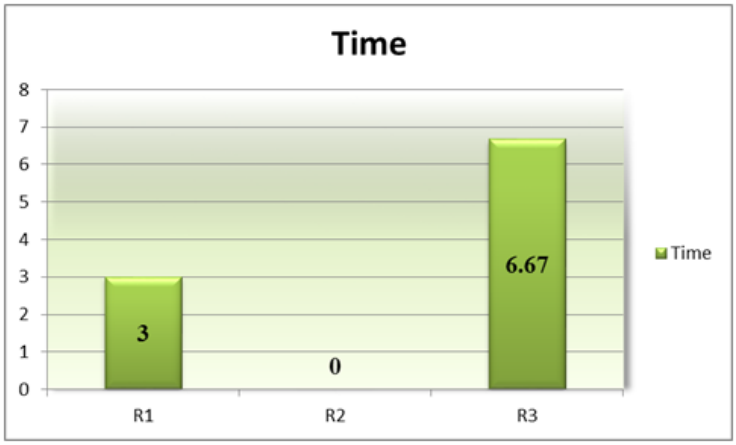
Task specification

Resource	PS(MIPS)	BW(MbPS)
R1	100	100
R2	50	150
R3	150	150

Resource specification

	R1	R2	R3
T1	1	2	0.67
T2	2	4	1.33
T3	3	6	2
T4	2	4	1.33
T5	1	2	0.67
T6	4	8	2.67

Expected execution time matrix



Makespan

Result

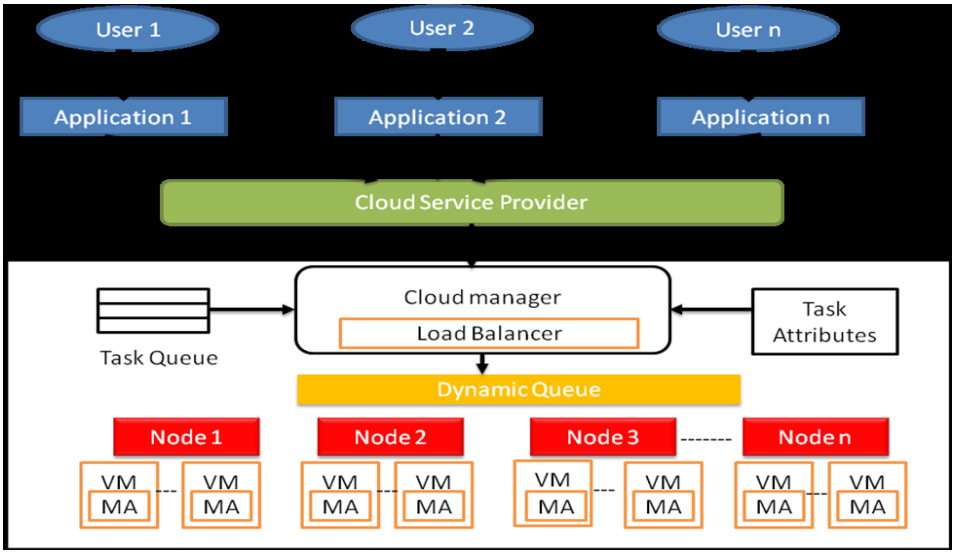
```
Output - CloudSim_S (run)   Minmin.java
run:
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 4 to VM #0
0.1: Broker: Sending cloudlet 0 to VM #2
0.1: Broker: Sending cloudlet 3 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #2
0.1: Broker: Sending cloudlet 2 to VM #2
0.1: Broker: Sending cloudlet 5 to VM #2
0.7666666666666666: Broker: Cloudlet 0 received
1.0966666666666667: Broker: Cloudlet 4 received
2.0966666666666667: Broker: Cloudlet 1 received
3.0966666666666667: Broker: Cloudlet 3 received
4.096666666666667: Broker: Cloudlet 2 received
6.763333333333334: Broker: Cloudlet 5 received
6.763333333333334: Broker: All Cloudlets executed. Finishing...
6.763333333333334: Broker: Destroying VM #0
6.763333333333334: Broker: Destroying VM #1
6.763333333333334: Broker: Destroying VM #2
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
```

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    0        SUCCESS        2             2        0.67        0.1        0.77
    4        SUCCESS        2             0        1        0.1        1.1
    1        SUCCESS        2             2        1.33        0.77        2.1
    3        SUCCESS        2             0        2        1.1        3.1
    2        SUCCESS        2             2        2        2.1        4.1
    5        SUCCESS        2             2        2.67        4.1        6.76
*****Datacenter: Datacenter_0*****
User id         Debt
3               106.8
***********************************
BUILD SUCCESSFUL (total time: 0 seconds)
```

III. Architecture

Below is the working model on which the research is to be carried. This model provides us the platform to implement the research on load balancing within cloud computing.

There can be number of users in the system. Each user is having an application and there will be a request from each application. All requests are sent to cloud service provider. Cloud service provider forwards these requests to cloud manager. Cloud manager contains load balancer which services each request which is stored in task queue based on task attribute associated with that request. Task attributes are task ID, Instruction Volume in MI, Data Volume in Mb and type of task which may be either Information processing task or entertainment related task or computational task. Based on these things, load balancer will assign tasks to particular virtual machines(nodes). There will be monitoring agent in node which will compute load whenever required.



System architecture

Task attribute

ID	Instruction volume(MI)	Data volume(Mb)	Type of task

Resource specification

ID	Processing speed(MIPS)	Bandwidth(MbPS)

IV. Algorithm

Below is the scenario of how the algorithm is implemented. Depending upon the entire research the concept can be implemented through the following algorithm.

Pseudo code:

Step 1: [calculate the makespan of each node based on execution time]

Expected execution time:

$$E_{ij} = \max\left(IV_i / PS_j, DV_i / BV_j\right)$$

Where i= task

j= resource

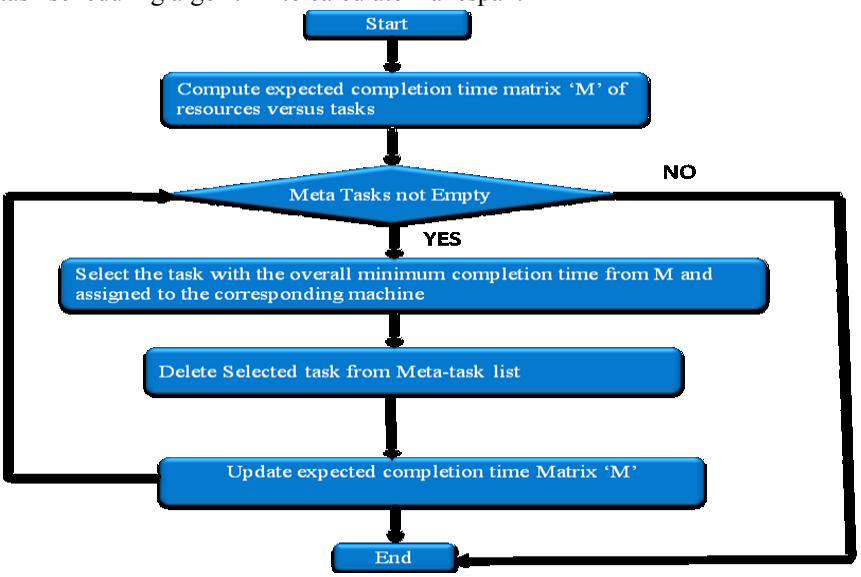
IV= Instruction Volume

PS= Processing Speed

DV= Data Volume

BW= Bandwidth

Apply min-min task scheduling algorithm to calculate makespan.



Min-min load balancing algorithm

Step 2: [Find average makespan]

$$Avg_{makespan} = \sum_{i=1}^{n} makespan_i / n$$

Where n= number of node

Step 3: [Calculate available capacity of each node]

$$AR = Avg_{makespan} - makespan$$

Step 4: [Calculate current load factor of each node]

$$\text{CLF} = \sum_{i=1}^{k} (\text{Weight of task based on task attribute})$$

Where k=number of tasks on each node

If (Task attribute = Information processing Task)

{

Weight = 1

}

Else if (Task attribute = Entertainment related Task)

{

Weight = 2

}

Else (Task attribute = Computational Task)

{
Weight = 3
}

Step 5: [Calculate future load factor FLF based on heuristic data]

$$FLF(Ti) = \sum_{i=1}^{l-1} [CLF(Ti)/l-1] \text{ where l= time and l>2}$$

FLF = CLF if l=1 or l=2

Step 6: [Calculate assignment factor (AF) for each node]

AF= AR + CLF + FLF

[Consider only positive values of AF]

Step 7: [Find min_AF and divide all AF by that factor]

Min _AF = min (All AF)

Step 8: [Calculate Queue factor]

Queue factor = AF / Min_AF

Step 9: [Generate Dynamic Queue based on Queue factor]

Theoretical Analysis

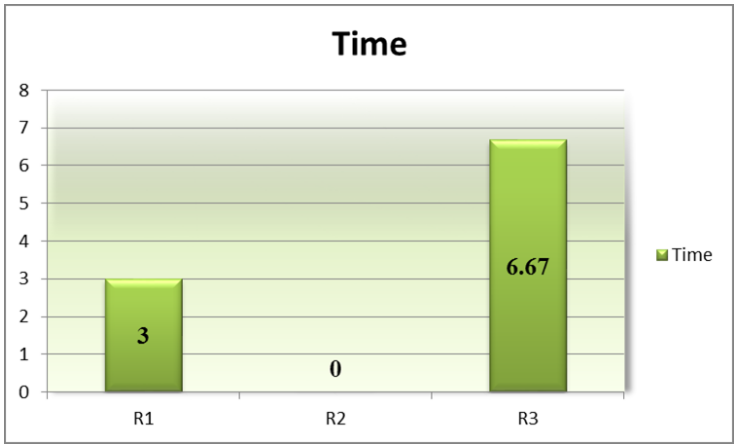
Task	IV(MI)	DV(Mb)
T1	100	50
T2	200	60
T3	300	70
T4	200	80
T5	100	50
T6	400	40

Task specification

Resource	PS(MIPS)	BW(MbPS)
R1	100	100
R2	50	150
R3	150	150

Resource specification

Step-1



Expected execution time matrix

	R1	R2	R3
T1	1	2	0.67
T2	2	4	1.33
T3	3	6	2
T4	2	4	1.33
T5	1	2	0.67
T6	4	8	2.67

Makespan

Step-2

$$Avg_{makespan} = \sum_{i=1}^{n} makespan_i / n$$

Where n= number of node

$Avg_{makespan}$= 3+0+6.67/3

= 3.22

Step-3

$$AR = Avg_{makespan} - makespan$$

AR_1= 3.22 – 3 = 0.22

AR_2= 3.22 – 0 = 3.22

AR_3= 3.22 – 6.67 = -3.45

Step-4

$$\text{CLF z} \sum_{i=1}^{k} (\text{Weight of task based on task attribute})$$

Where k=number of tasks on each node

Here,

CLF_1= 2 +1 = 3

CLF_2=0

CLF_3=3+3+2= 8

Step-5

Initially, FLF = CLF

So,

FLF_1= 3

FLF_2= 0

FLF_3= 8

Step-6

AF_1= AR_1 + CLF_1 + FLF_1 = 0.22 + 3 + 3 = 6.22

AF_2= AR_2 + CLF_2 + FLF_2 = 3.22 + 0 + 0 = 3.22

AF_3= AR_3 + CLF_3 + FLF_3 = -3.45 + 8 + 8 = 12.55

Step-7

Min _AF = min (All AF) = min (6.22 , 3.22 , 12.55) = 3.22

Step-8

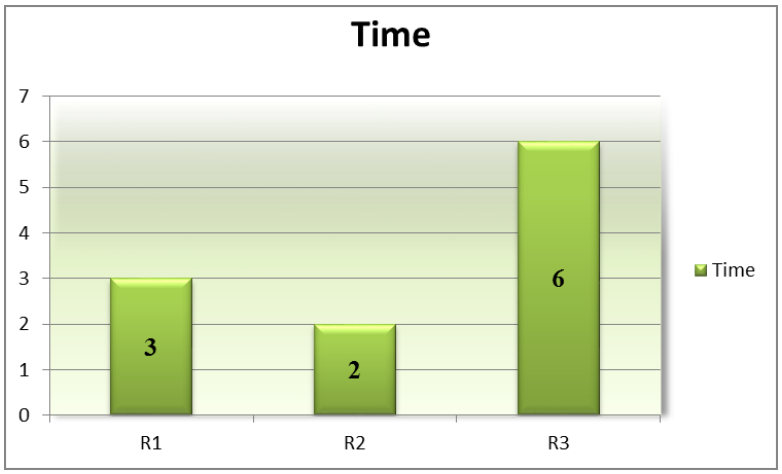
Queue factor = AF / Min_AF

Q_1 = 6.22 / 3.22 = 1.93 = 2

Q_2 = 3.22 / 3.22 = 1

Q_3 = 12.55 / 3.22 = 3.89 = 4

Step-9



New Makespan

V. Results

Task	IV(MI)	DV(Mb)
T1	100	50
T2	200	60
T3	300	70
T4	200	80
T5	100	50
T6	400	40

Task specification

```
Output - CloudSim_S (run)
run:
Starting Scheduling....
Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2, Host #0
0.1: Broker: VM #1 has been created in Datacenter #2, Host #0
0.1: Broker: VM #2 has been created in Datacenter #2, Host #0
0.1: Broker: Sending cloudlet 4 to VM #0
0.1: Broker: Sending cloudlet 1 to VM #2
0.1: Broker: Sending cloudlet 0 to VM #1
0.1: Broker: Sending cloudlet 3 to VM #0
0.1: Broker: Sending cloudlet 2 to VM #2
0.1: Broker: Sending cloudlet 5 to VM #2
```

```
1.1: Broker: Cloudlet 4 received
1.4333333333333333: Broker: Cloudlet 1 received
2.0933333333333333: Broker: Cloudlet 0 received
3.0933333333333333: Broker: Cloudlet 3 received
3.433333333333333: Broker: Cloudlet 2 received
6.1: Broker: Cloudlet 5 received
6.1: Broker: All Cloudlets executed. Finishing...
6.1: Broker: Destroying VM #0
6.1: Broker: Destroying VM #1
6.1: Broker: Destroying VM #2
Broker is shutting down...
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
Datacenter_0 is shutting down...
Broker is shutting down...
Simulation completed.
Simulation completed.
```

```
========== OUTPUT ==========
Cloudlet ID    STATUS    Data center ID    VM ID    Time    Start Time    Finish Time
    4        SUCCESS        2            0        1        0.1        1.1
    1        SUCCESS        2            2        1.33        0.1        1.43
    0        SUCCESS        2            1        1.99        0.1        2.09
    3        SUCCESS        2            0        1.99        1.1        3.09
    2        SUCCESS        2            2        2        1.43        3.43
    5        SUCCESS        2            2        2.67        3.43        6.1
*****Datacenter: Datacenter_0*****
User id         Debt
3               106.8
**********************************
Scheduling is finished!
BUILD SUCCESSFUL (total time: 0 seconds)
```

VI. Comparison

Resource\Algorithm	Min-min	Proposed
R1	3	3
R2	0	2
R3	6.67	6

Comparison of completion time of various algorithm

Resource\Algorithm	Round-robin	Min-min	Proposed
R1	2	2	2
R2	2	0	1
R3	2	4	3

Comparison of task allocation of various algorithm

VII. Conclusion And Future Work

As cloud computing is a vast and emerging area, there are many issues related to it. Load balancing is one of them and it needs to be solved to provide better customer service. This research focuses on 4 parameters , namely , weight according to the type of task, current load factor, future load factor, makespan. Considering these 4 parameters, the proposed scheme achieves optimum resource utilization.

In the future work, performance of the proposed work needs to be evaluated on the real world i.e. private cloud and public cloud. Network delay and various other parameters need to be considered.

References

[1] Martin Randles, David Lamb, 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, pp551-556, 2010

[2] Jianzhe Tai, Juemin Zhang, JunLi, WaleedMeleis and NingfangMi "A R A: Adaptive Resource Allocation for Cloud Computing Environments under Bursty Workloads" 978-1-46730012-4/11 ©2011 IEEE.

[3] J. F. Yang and Z. B. Chen, "Cloud Computing Research and Security Issues," International Conference on Computational Intelligence and Software Engineering (CiSE), Wuhan, 10-12 December 2010, pp. 1-3.

[4] K. Ramana, A. Subramanyam and A. Ananda Rao, Comparative Analysis of Distributed Web Server System Load Balancing Algorithms Using Qualitative Parameters, VSRD-IJCSIT, Vol. 1 (8), 2011, 592-600

[5] Chaczko, Z., Mahadevan, V., Aslanzadeh, S., & Mcdermid, C., "Availability of Load Balancing in Cloud Computing", International Conference on Computer and Software Modeling, 2011.

[6] A.Khiyaita, M.Zbakh, H. El Bakkali, Dafir El Kettani, "Load Balancing Cloud Computing : State of Art", IEEE, 2012.

[7] Qi Zhang, Lu Cheng , Raouf Boutaba, " Cloud computing: state-of-the-art and research challenges", Springer, 20 April 2010

[8] Rashmi. K. S., Suma. V, Vaidehi. M, "Enhanced Load Balancing Approach to Avoid Deadlocks in Cloud", International Journal of Computer Applications, June 2012.

[9] Nidhi Jain Kansal, Inderveer Chana, " Cloud Load Balancing Techniques : A Step Towards Green Computing", IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012

[10] Nidhi Jain Kansal, Inderveer Chana, "Existing load balancing techniques in cloud computing: a systematic review", Journal of Information Systems and Communication, February 2012

[11] Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, "A Scheduling Strategy on Load Balancing of Virtual Machine sourcein Cloud Computing Environment", 3rd International symposium on Parallel Architectures, Algorithms and Programming, IEEE 2010.

[12] Tanveer Ahmed, Yogendra Singh, "Analytic Study Of Load Balancing Techniques Using Tool Cloud Analyst", International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue 2, Mar-Apr 2012.

[13] Stanojevic R. and Shorten R. (2009) IEEE ICC, 1-6.

[14] Bhadani A. and Chaudhary S. (2010) 3rd Annual ACM Banga-lore Conference.

[15] Fang Y., Wang F. and Ge J. (2010) Lecture Notes in Computer Science, 6318, 271-277.

[16] http://cloudbus.org/cloudsim

[17] Rajkumar Buyya, Christian Vecchiola, and Thamarai Selvi, Mastering Cloud Computing, International Edition: Morgan Kaufmann, ISBN: 978-0-12-411454-8, Burlington, Massachusetts, USA, May 2013; and Indian Edition: Tata McGraw Hill, ISBN-13: 978-1-25-902995-0, New Delhi, India, Feb 2013.

[18] Rajkumar Buyya, James Broberg, and Andrzej Goscinski (eds), Cloud Computing: Principles and Paradigms, 644 pages, ISBN-13: 978-0470887998, Wiley Press, New York, USA, February 2011.