Volume 4, Issue 2, February 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Analysis of Public Cloud Load Balancing using Partitioning Method and Game Theory

Mangal Nath Tiwari *
Dept. of CS, APS University
Rewa, MP, India

Kamalendra Kumar Gautam
Dept. of CS, APS University
Rewa, MP, India

Dr Rakesh Kumar Katare
Dept. of CS, APS University
Rewa, MP, India

Abstract—Cloud computing is a provision of providing networked, on-line, on-demand services pay per use basis. Several issues as scalability, security, performance etc are discussed so far by many researchers for the cloud computing. Cloud partitioning is an optimal approach for public cloud. In public cloud environment various nodes are used with required computing resources situated in different geographic locations, so this strategy simplifies the load distribution across the multiple nodes, but fault tolerance and load balancing are most important problems obtaining high performance in the system. Load balancing is the process of distribution of workload among different nodes or processor. The purpose of load balancing is to improve the performance of a cloud environment through an appropriate distribution strategy. Game theory is the formal study of conflict and cooperation. Game theoretic concepts apply whenever the actions of several agents are interdependent. The game theoretic algorithms help to obtain a user optimal load balancing which ultimately improves overall performance of cloud computing. This paper introduces a better approach for public cloud load distribution using partitioning and game theory concept to increase the performance of the system.

Keywords— Cloud computing, Dynamic Load Balancing (DLB), Game Theory, Public Cloud, Cloud Partitioning.

I. INTRODUCTION

Cloud computing is the use of computing resources that are delivered as a service over a. Load balancing and provisioning in cloud computing systems is really a challenge job. For solving such problem always a distributed and dynamic solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfils the required demands Jobs cannot be assigned to appropriate servers and clients individually for efficient load balancing as cloud is a very complex structure and components are present throughout a wide spread area. Here some uncertainty is attached while jobs are assigned. The aim is to provide an evaluation and comparative study of these approaches [13]. Workload distribution problem in cloud computing environment is very crucial and complex task till today, because prediction of user request arrivals on the server is not possible. In cloud environment, each virtual machine has different capabilities, so it becomes more complex to schedule job and balance the work-load among nodes. In this paper we introduce a dynamic strategy to balance workload among nodes. This scheme provides more flexibility and performance in the system. Virtualization is very useful concept in context of cloud systems. Virtualization means “something which isn’t real”, but gives all the facilities of a real. It is the software implementation of a computer which will execute different programs like a real machine. Virtualization is related to cloud, because using virtualization an end user can use different services of cloud Ashish et al. [1]. The remote datacenter will provide different services in a full or partial virtualized manner Public cloud applications, storage, and other resources are made available to the general public by a service provider. These services are free or offered on a pay-per-use model. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure Ashish et al. [1]. Public cloud services may be free or offered on a pay-per-usage model. The term "public cloud" arose to differentiate between the standard model and the private cloud, which is a proprietary network or data canter that uses cloud computing technologies, such as virtualization. Examples of public clouds include Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, Google Drive, Google Apps, and Windows Azure Services Platform [6]. In this paper we reviewed a dynamic strategy to balance workload among nodes with the help of cloud partitioning and game theory concept. In this work various nodes are used with required computing resources situated in different geographic location.

II. LOAD BALANCING

“The load balancing technique used to make sure that none of the node is in idle state while other nodes are being utilized”. In order to balance the lode among multiple nodes you can distribute the load to another node which has lightly loaded. Thus distributing the load during runtime is known as Dynamic Load Balancing technique. Load balancing algorithm can be divided into two categories as 1) Static and 2) Dynamic. In static load balancing algorithm, all the information about the system is known in advance, and the load balancing strategy has been made by load balancing algorithm at compile time. This load balancing strategy will be kept constantly during runtime of the system.

In contrast, dynamic algorithm is implemented at running time, and the load balancing strategies change according to the real statement of the system. Though, the dynamic algorithm has better adaptability, it is sensitive to the accuracy of the load information or statement of system. Many researchers have proposed several algorithms for load balancing. In cloud computing when a computation is requested by any system it is distributed to all the slaves existing in that cloud. So the way in which the distribution is being done must get the response from all the slaves at the same time so that there should not be any waiting for any particular computing device to reply before further processing could happen. But in the real time clouds heterogeneous computing devices exists and any process's execution time on the slave is required to be estimated. So the main feature that is must in any load balancer is the asymmetric load distribution.

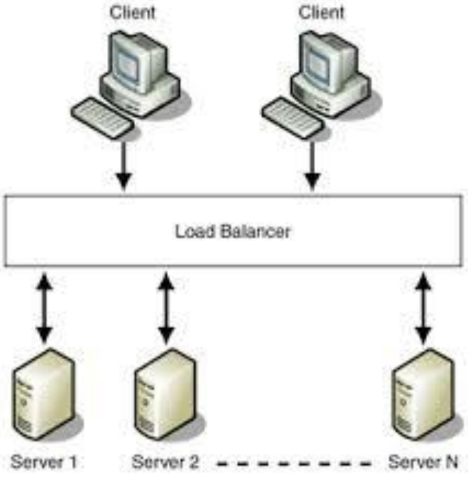

Figure: 1 – Simple view of Load Balancer

A greater ratio of workload is required to be given to those with higher computation capabilities. But sometimes just higher computation power only cannot help is deciding how much share of the task is required should be assigned to that system. This assignation of proper task to proper system in heterogeneous computing infrastructure is done by load balancer. Load balancer is also responsible for 'Priority Activation' which means that when the number of slave computing device devices drops below a certain point the load balancer must wake some of the sleeping devices to maintain the computing performance for the client. Load balancing in cloud computing was described in by Adler [3]. There are many load balancing algorithms, such as Round Robin (RR), Equally Spread Current Execution Algorithm (ESCEA), and Ant Colony algorithm (ACA). Nishant et al. [4] used the ant colony optimization method in nodes load balancing. Randles et al. [5] gave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. They concluded that the ESCEA algorithm and throttled algorithm are better than the Round Robin algorithm. Some of the classical loads balancing methods are similar to the allocation method in the operating system, for example, the Round Robin algorithm and the First Come First Served (FCFS) rules. The Round Robin algorithm is used here because it is fairly simple.

III. CLOUD PARTITIONING MODEL

A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet [6]. Public cloud is made up of several nodes situated in deferent geographic location. Cloud partitioning is a method to make partitions of huge public cloud is some segment of cloud.

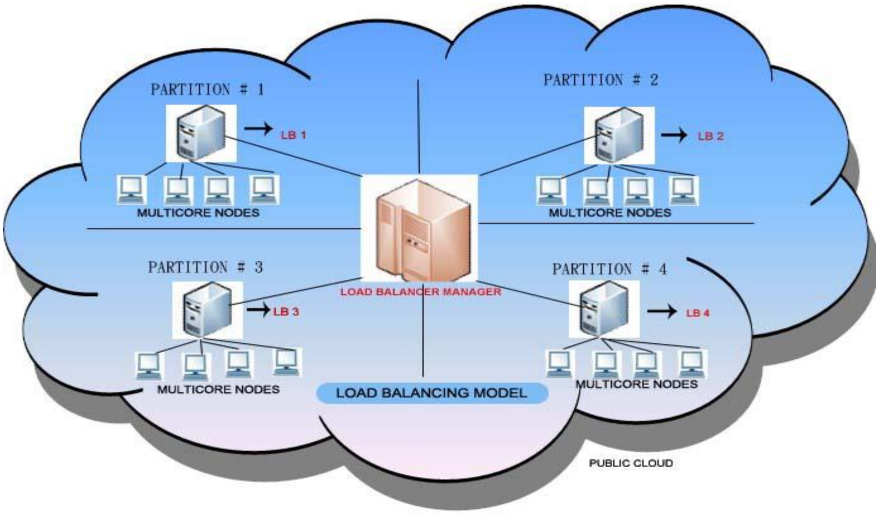

Figure – 2: Cloud partitioning model

A cloud partition has several nodes belongs to a particular area, these subarea of the public cloud based on the geographic locations. These subareas are considered to be as cluster of nodes with a load manages. Model of the cloud partitioning is depicted on the following figure. In this model load balancing is implemented in two steps, in first steps public cloud is partitioned into four subarea named Partition#1, Partition#2, Partition#3 and Partition#4. Each partition has a Load balancer (LB) associated with multiple multi-core nodes. There is a main controller system which manages the load balancer called Load Balancer Manager (LBM). After partitioning the public cloud into different partitions, load balancing then starts.

A. Working of Load Balancer Manager (LBM):

In this model, Load Balancer Manager (LBM) is responsible for the following task-

- Receives the jobs from different end users.
- Choose a specific partition for the received jobs.
- Check the status of the cloud partition (Status may be in one of these: IDLE, NORMAL, and HEAVY).
- If the partition Status=HEAVY the no allocation will be done, it means all nodes are overloaded already.

If partition Status=IDLE or NORMAL then forward the Jobs to the respective Load balancer (LB). Now the load balancer activated and starts its work.

B. Possible status of Cloud Partition:

Cloud partition can be in one of the following three statuses –

- IDLE – In this, most of the nodes are in idle state.
- NORMAL – In this status, some of the nodes are in idle status while some others are overloaded.
- HEAVY – In this status of the cloud partition, most of the nodes are overloaded.

Any node is available to get job for execution only if it returns to normal status. Load Balancer Manager checks the status of the partition and dispatches the user requests to the partition which is in IDLE status.

C. Calculation of Load Degree (LD) for a node:

The Load Degree (LD) of a node in any cloud partition is calculated from following equation-

$$LD\ (N) = \sum_{i=1}^{m} Xi*Fi$$

Here, N=Current Node, Fi are the parameter either static or dynamic where Fi(1<=i<=m), m represents the total number of parameter . Xi are weights that may differ for different kinds of job for all (1<=i<=n).

Average Load Degree (LD) of the cloud partition will be calculated as-

$$Avg\_LD= \sum_{i=1}^{n} LD\ (Ni)/n$$

D. Possible load status of node:

According to the calculation of load degree for the node three load status of the node are defined as follows-

IDLE: When LD (N)=0
NORMAL: 0<LD (N) <=High_LD
OVERLOADED: High_LD<=LD (N)

Any cloud partition having the status=HEAVY is not selected by the Load Balancer Manager and likewise any node having the Load Degree (LD) =OVERLAODED is not eligible for the processing. Only cloud partition having IDLE or NORMAL load status and Node having IDLE or NORMAL load degree are selected for scheduling and load balancing.

IV. LOAD BALANCING TECHNIQUE

Here we are going to discuss some load balancing technique for both the partition having either load status=idle or load status=normal. In this section mainly we will discuss about the load balancing technique for the cloud partition having load status=normal using game theory.

A. For cloud partition having idle status:

In this situation, this cloud partition has the ability to process jobs as quickly as possible so a simple load balancing method can be used [7]. There are lots of works has been done for load balance algorithm such as the Random algorithm, the Weight Round Robin, and the Dynamic Round Robin[12]. The Round Robin (RR) is used here because it is very simple method for load balancing. The Round Robin algorithm does not record the status of each connection so it has no status information. In a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called “Round Robin based on the load degree evaluation”. Before the Round Robin step, the nodes in the load balancing table are ordered based on the load degree from the lowest to the highest. The system builds a circular queue and walks through the queue again and again. Jobs will then be assigned to nodes with low load degrees. The node order will be changed when the balancer refreshes the Load Status Table. However, there may be read and write inconsistency at the refresh period T. When the balance table is refreshed, at this moment, if a job arrives at the cloud partition, it will bring the inconsistent problem. The system status will have changed but the information will still be old. This may lead to an erroneous load strategy choice and an erroneous nodes order.

```
Step: 1   Start
Step: 2   Store Nodes in the LBT based on Load_Degree(LD)
Step: 3   If load_degree(N)=low then
                  Load_degree(N)=job
Else
                  LoadBalancer refresh the LoadStatusTable at the refresh period T
Step: 4   Set the value of flag
Step: 5   Read the value of flag (before period T and after T+1)
          If flag=read then
                  Output Load_Degree for RoundRobin
          Else
                  Input Load_Degree for RoundRobin
Step: 6   Set the SystemStatus and Set the SystemInfo
Step: 7   End
```

Algorithm: 1- RR Algorithm with Load Degree Evaluation

To resolve this problem, two Load Status Tables should be created. A flag is also assigned to each table to indicate Read or Write. When the flag = “Read”, then the Round Robin based on the load degree evaluation algorithm is using this table. When the flag = “Write”, the table is being refreshed, new information is written into this table. Thus, at each moment, one table gives the correct node locations in the queue for the improved Round Robin algorithm, while the other is being prepared with the updated information. Once the data is refreshed, the table flag is changed to “Read” and the other table’s flag is changed to “Write”.

B. *For cloud partition having Normal status:*

This situation is more complex than the idle status situation, because in these situations jobs are dispatched faster by the cloud Load Balancer Manager (LBM) and each user wants to execute his job at shortest response time so the public cloud needs an optimal approach to complete the job execution at minimum response time. To solve such problem Penmatsa and Chronopoulos [9] has proposed “static load balancing strategy based on game theory for distributed systems”. This paper is the base of our review work and we consider that the implementation of distributed system, the public cloud load balancing can be viewed as a game. The purpose of load balancing is to improve the performance of a system through an appropriate distribution of the application load. [10] A general formulation of this problem is as follows: given a large number of jobs, find the allocation of jobs to computers optimizing a given objective.

“Game theory is the formal study of decision-making where several players must make choices that potentially affect the interests of the other players”. Game theory is the formal study of conflict and cooperation. Game theoretic concepts apply whenever the actions of several agents are interdependent. These agents may be individuals, groups, firms, or any combination of these. The concepts of game theory provide a language to formulate structure, analyze, and understand strategic scenarios [12].

A game is a description of strategic interaction that includes the constraints on the actions that the players can take and the players' interests, but does not specify the actions that the players do take. A solution is a systematic description of the outcomes that may emerge in a family of games. Game theory suggests reasonable solutions for classes of games and examines their properties. There are three types of game theocratic load balancing techniques for public cloud.

Global Method – In this case there is only one decision maker that optimizes the response time of the entire system over all jobs and the operating point is called social (overall) optimum [10].

Cooperative Method – In this case there are several decision makers (e.g. jobs, computers) that cooperate in making the decisions such that each of them will operate at its optimum. Decision makers have complete freedom of pre-play communication to make joint agreements about their operating points [10].

Non cooperative Method – In this case there are several decision makers (e.g. users, jobs) that are not allowed to cooperate in making decisions. Each decision maker optimizes its own response time independently of the others and they all eventually reach equilibrium. This situation can be viewed as a non cooperative game among decision makers.

The equilibrium is called Nash equilibrium [11] and it can be obtained by a distributed non cooperative policy. At the Nash equilibrium a decision maker cannot receive any further benefit by changing its own decision [10].

Our study of the so far literature has shown that the load balancing strategy for public cloud is implemented as Non-cooperative game theory, which is describes as here under –

C. Mathematical Model:

Study of this mathematical model is based on [10]. In the game of load balancing for the public cloud the players would be nodes in each cloud partition and the user jobs dispatched by the Load Balancer Manager (LBM).

We assume that there are n nodes in each partition and p jobs dispatched by the LBM. Now the Load Balancer (LB) has to decide on how to distribute user jobs to available nodes such that they will operate optimally. In the following, we present the notations we use and then define the non-cooperative load balancing game.

μ_i −*Average Processing Time (APT), where i=1, 2, 3…… n*

$\O j$−*Job's Average Throughput where j=1, 2, 3………….. n*

$\Phi - \sum_{J=1}^{m} \O_j$, *is the Total Job Arrival Time (TJAT)*

Thus user j (j=1, 2, 3,……m) must find the fraction Sji of all its jobs that are assigned to the node i such that expected execution time of this job is minimized. Let us assume that Sji is the fraction of job j is assigned to node i. The vector Sj=(Sj1, Sj2, …….. Sjn) is called the load balancing strategy of user job j. And the vector S= (S1, S2, …….. Sn) is called the strategy profile of load balancing game.

In order to determine a solution for our load balancing game we consider an alternative definition of the Nash equilibrium. Nash equilibrium can be defined as a strategy profile for which every user's load balancing strategy is a best reply to the other users' strategies. This best reply for a user will provide a minimum expected response time for that user's jobs given the other users' strategies. This definition gives us a method to determine the structure of the Nash equilibrium for our load balancing game. The computation of Nash equilibrium may require some coordination between the users. Here this is necessary in the sense that users need to coordinate in order to obtain the load information from each computer. From the practical point of view we need decentralization and this can be obtained by using greedy best reply algorithms [11]. In these algorithms each user updates from time to time its load balancing strategy by computing the best reply against the existing load balancing strategies of the other users.

V. CONCLUSIONS AND FUTURE WORKS

A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet [6]. Public cloud is made up of several nodes situated in deferent geographic location. Cloud partitioning is a method to make partitions of huge public cloud is some segment of cloud. A public cloud is one based on the standard cloud computing model, in which a service provider makes resources, such as applications and storage, available to the general public over the Internet. Public cloud is made up of several nodes situated in deferent geographic location. Cloud partitioning is a method to make partitions of huge public cloud is some segment of cloud. The object of study in game theory is the game, which is a formal model of an interactive situation. It typically involves several players; a game with only one player is usually called a decision problem.

In future study we will try to find other load balance strategy because other load balance strategies may provide better results, so tests are needed to compare different strategies. Many tests are needed to guarantee system availability and efficiency. Also we will address the development of game theoretic models for load balancing in the context of uncertainty as well as game theoretic models for dynamic load balancing in future. We also plan to develop dynamic load balancing schemes based on dynamic game theory that provide fairness by taking the current system load into account and also consider other aspects of heterogeneity.

ACKNOWLEDGMENT

I wish to thank all the people who gave me an unending Support right from stage the idea was conceived. I would like to thank my supervisor Dr. R.K. Katare for their helpful, valuable comments and suggestions.

I express my sincere and profound thanks to Er. R.B. Singh and Er. A.K. Singh, which always stood as the helping and guiding support for me.

REFERENCES

[1] Ashish Kumar Singh, Sandeep Sahu, Mangal Nath Tiwari, R. K. Katare, *Scheduling Algorithm with Load Balancing in Cloud Computing*, IJCA, pp-48-43, 2011.

[2] Shaoyi Song, Tingjie Lv, and Xia Chen, *Research Article Load Balancing for Future Internet: An Approach Based on Game Theory*, Hindawi Publishing Corporation Journal of Applied MathematicsVolume 2014, Article ID 959782.

[3] B. Adler, *Load balancing in the cloud: Tools, tips and techniques*, http://www.rightscale.com/info center/whitepapers/Load-Balancing-in-the-Cloud.pdf, 2012.

[4] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, N. Nitin, and R. Rastogi, *Load balancing of nodes in cloud using ant colony optimization*, in Proc. 14th International Conference on Computer Modelling and Simulation (UKSim), Cambridgeshire, United Kingdom, Mar. 2012, pp. 28-30.

[5] M. Randles, D. Lamb, and A. Taleb-Bendiab, *A comparative study into distributed load balancing algorithms for cloud computing*, in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications, Perth, Australia, 2010, pp. 551-556.

[6] M. Rouse, *Public cloud*, http://searchcloudcomputing, techtarget.com/definition/public-cloud, 2012.

[7] Gaochao Xu, Junjie Pang, and Xiaodong Fu, *A Load Balancing Model Based on Cloud Partitioning for the Public Cloud*, IEEE Transactions on Cloud Computing Year-2013.

[8] D. MacVittie, *Intro to load balancing for developers —the algorithms*, https://devcentral.f5.com/blogs/us/ Introtoload-balancing-for-developers-ndash-the-algorithms, 2012.

[9] S. Penmatsa and A. T. Chronopoulos, *Game-theoretic static load balancing for distributed systems*, *Journal of Parallel and Distributed Computing*, vol. 71, no. 4, pp. 537-555, Apr. 2011.

[10] D. Grosu, A. T. Chronopoulos, and M. Y. Leung, *Load balancing in distributed systems: An approach using cooperative games*, in Proc. 16th IEEE Intl. Parallel and Distributed Processing Symp., Florida, USA, Apr. 2002, pp. 52-61.

[11] T. Basar and G. J. Olsder., *Dynamic Noncooperative Game Theory*, SIAM, 1998.

[12] Theodore L. Turocy, Bernhard von Stengel, *Game Theory*, CDAM Research Report LSE-CDAM-2001-09 October 8, 2001.

[13] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, *Availability and load balancing in cloud computing*, presented at the 2011 International Conference on Computer and Software Modeling", Singapore, 2011.

[14] Ashish Kumar Singh, Sandeep Sahu, Kamalendra Kumar Gautam, Mangal Nath Tiwari, *Private Cloud Scheduling with SJF, Bound Waiting, Priority and Load Balancing*, IJARCSSE, Volume 4, Issue 1, January 2014 ISSN: 2277 128X.

[15] Vinay Kumar Kaushik, Hemant Kumar Sharma, Dinesh Gopalani, L*oad Balancing In Cloud Computing Using High Level Fragmentation Of Dataset*, International Conference on Cloud, Big Data and Trust 2013, Nov 13-15, RGPV.

[16] Divya Thazhathethil et al., *A Model for load balancing by Partitioning the Public Cloud*, IJIRCCE, Vol. 2, Issue 1, January 2014.

AUTHORS PROFILES

Mangal Nath Tiwari completed the M. Sc. (IT) and M. Phil (CS) Degrees from APS University, Rewa in 2004 and 2010, respectively. Currently he is a student of Ph.D. in APS University, Rewa, Madhya Pradesh, India. E-Mail: mangal.tiwari81@gmail.com

Dr. Rakesh Kumar Katare received the M.Tech. And Ph.D. in Computer Science from Devi Ahilya Vai University Indore, India. Currently he is Professor in Deptt. Of Computer Science, APS University, Rewa, Madhya Pradesh, India. He is also supervisor of Mangal Nath Tiwari, E-Mail: katare_rakesh@yahoo.com

Kamalendra Kumar Gautam completed MSc(CS) from Makhanlal Chaturvedi National University for Journalism and Communication,Bhopal and M.Phil(CS) from APSU, Rewa, MP. Currently he is research scholar in APS University, Rewa, Madhya Pradesh, India.
E-Mail: gautamtata@yahoo.com