International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 2, February 2015, PP 1-6
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Load Balancing Model in Cloud Computing

Akshada Bhujbal, Prajakta Jakate, Manasi Wagh, Madhura Pise, Prof.M.V.Marathe

Department of Computer Engineering, Pune Vidyarthi Griha's College of Engineering and Technology, Pune, India

ABSTRACT

Cloud computing is efficient and scalable but to maintain the stability of processing so many jobs in the Cloud computing environment is a very complex problem. Cloud Computing possesses sharing of resources. Cloud may involve resources like networks, servers, storage and services. The article includes a better load balance model for the cloud based on the Cloud partitioning concept. A switch mechanism is chosen for different strategies and for different situations. The round robin and game theory algorithm improves the efficiency and performance in the Cloud environment

Keywords: *Cloud, Load balancing, Cloud partition, Round Robin algorithm, Game theory algorithm*

INTRODUCTION

Cloud computing is an attracting technology in the field of computer science. NIST gave a definition of cloud computing for enabling ubiquitous, convenient, on demand network access to a shared pool of configurable computing resources (e. g., networks, servers, storage, applications, and services). This can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. NIST defined the five important characteristics of cloud computing that includes On-demand self-service, Global network access, Distributed resource pooling, Scalable, Measured service [2].

Cloud computing architectures are inherently parallel and distributed. They also serve the needs of multiple clients in different scenarios. The distributed architecture deploys resources distributive to deliver services efficiently to users in different geographical locations [3]. Clients in a distributed environment generate a request randomly, in any processor. So the major drawback of this randomness is associated with task assignment. The unequal task assignment to the processor creates imbalance i. e., some of the processors are overloaded and some of them are under loaded [4].

The objective of Load balancing is to achieve a high user satisfaction and resource utilization ratio. It is also used to avoid the situation where nodes are either heavily loaded or under loaded in the network, hence improving the overall performance of the system. Proper Load balancing helps in utilizing the available resources optimally, thereby minimizing the resource consumption [5].

Load Balancing Concepts

Load balancing is a process of distributing load. The load is distributed on individual nodes to maximize throughput, and to minimize the response time. It also removes a condition in which some of the nodes are heavily loaded while some others are light. [6] Load balancer is a software program which receives connection request from clients and forwards it to one of the backend server. Then the server replies accordingly. Hence the client will be unaware of the data is stored. This separation provides security benefit and prevent attacks. In cloud computing environment, the job arrive randomly with random CPU utilization. Because of random arrival of job specific resources heavily loaded and, the other resources are less loaded.

RELATED WORK

There are many studies being conducted in this area. Load balancing concept in cloud computing was described in a white paper written by Adler[7] who introduced the tools and techniques commonly used for load balancing in the cloud. Chaczko et al.[8] state that the load balancing plays important

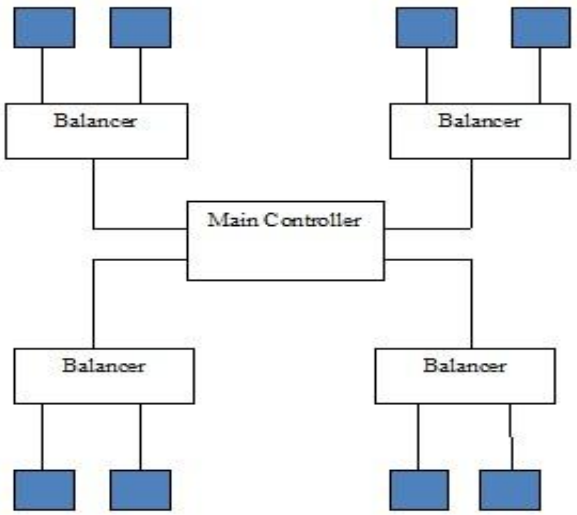
**Address for correspondence*

role for improving the performance and maintaining stability. There are many load balancing algorithms like Ant Colony algorithm, Round Robin, and Equally Spread Current Execution Algorithm. A. Subramanyam and A. AnandaRao have [9] have put forth that Load balancing algorithm tries to balance the total system load and also presented the performance analysis of various load balancing algorithms In the existing system the partitioning of cloud is based on area which might not always be well suited in all situations hence we need a new partitioning method from which we get better performance.

ARCHITECTURE

Main Controller and Balancers

The load balancing is done by the main controller and the balancers. The main controller first assigns jobs to the suitable cloud partition. The main controller deals with information for each partition. The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs.



ALGORITHM

Assigning Jobs to the Cloud Partition

When a job arrives at the cloud, first of all the main thing to do is to choose the right partition. There are three types of cloud partition status:

- Idle: When the percentage of idle nodes exceeds alpha (α), change to idle status. If it is in idle status, this job should be transferred to another partition by using Round Robin algorithm.
- Normal: When the percentage of the normal nodes exceeds beta (β), change to normal load status.
- Overload: When the percentage of the overloaded nodes exceeds gamma γ, change to overloaded status. If it is overload, this job should be transferred to another partition. And exactly to which partition it should be transferred is selected using above two algorithms.

The parameters alpha (α), beta (β) and gamma (γ) are to be set by the cloud partition balancers. When the load status of a cloud partition is idle or normal, then this partitioning can be done locally. If the cloud partition load status is not normal or idle, then this job should be get transferred to another cloud partition. How to assign the jobs to the nodes is decided by the partition load balancer. If one cloud server is overloaded and it again getting new client request while other servers are in Idle or Normal state then following algorithms are used.

Calculation of Load Parameter

Step 1: Define a load parameter set:

F = {F1, F2, ..Fm} with each Fi (1 <= i<= m, Fi∈ [0,1]) parameter being either static or dynamic. m

represents the total number of the parameters.[10]

Step 2: Compute the load degree as:

$$Load\_degree(N) = \sum_{i=0}^{m} \alpha_i F_i \quad (1)$$

$\alpha_i \left(\sum_{i=0}^{n} \alpha_i = 1\right)$ are weights that may differ for different kinds of jobs. N represents the current node.[10]

Step 3: Define evaluation bench marks.

Then calculate the average cloud partition degree from the node load degree statistics as:

$$Load\_degree_{avg} = \frac{\sum_{i=0}^{n} \text{Load\_degree}(N_i)}{n} \quad (2)$$

The bench mark Load_degreehigh is then set for different situations based on the Load_degreeavg.[10]

Step 4: Then three load status levels of nodes defined as [10]:

- Idle

  When Load_degree(N) = 0 (3)

  There is no job being processed by this node so the status is changed to Idle.

- Normal

  For 0 < Load degree(N) <= Load_degreehigh (4)

  The node is normal and it can process other jobs.

- Overloaded

  When Load degreehigh <= Load_degree(N) (5)

The node is not available and cannot receive jobs until it returns to the normal.[10].The cloud partition balancers create the Load Status Tables and the load degree results are input into the Load Status Tables. Each balancer has a Load Status Table and refreshes it each fixed period T. The balancers use table to calculate the partition status. Every partition status has a different load balancing solution. When a job arrives at a cloud partition, the job is get assigned by balancer the job to the nodes according to its current load strategy. As the cloud partition status changes this strategy is changed by the balancers.

Round Robin Algorithm

Step 1: Job arrives at the main controller.

Step 2: Job is assigned to balancer according to balancer status, request location.

Step 3: In particular partition P.

Set i=0

Set s[n] as no. of servers arranged in increasing order of jobs in P for all n=1,..,n

Set s[c] as no. of idle servers in P from s[n] for all c=1,..,n

Step 4: When job arrives

If s[c] != NULL

Then, send the connection to s[i]

i= i+1;

If i == c

Then i=1

Else Go to game theory

End if

Step 5: go to step 1

Game Theory

The players in the game are the nodes and they contend for jobs. Consider in the current cloud partition there are n nodes with N jobs arriving, then define the following parameters:

μi : Processing ability of each node, i = 1,…, n.

*Φ*j : Time spent for each job.

$$\Phi = \sum_{j=0}^{N} \varphi_j$$

Φj: Time spent by the entire cloud partition,

$$\Phi < \sum_{j=1}^{N} \mu_i$$

S_{ij}: Fraction of job j that assigned to node i ($\sum_{j=0}^{N} S_{ij} = 1$ and 0 <= S_{ij} <= 1).

In this model, the most important step is finding appropriate value of Sij. Here "the best reply" method proposed by Grosuet al.[11] can be used to calculate Sij of each node. The Nash Equilibrium here is to minimize the response time of each job.

MATHEMATICAL MODEL

Compute Load Degree

We have to check the status of system to know system is overloaded or not because on the basis of status we are transferring the request to balance load. Finding Status of system we require Load degree so,

Input: Set of Parameters.

Output: 1. Load Degree

2. Load Degree Average

Define a load Parameter Set: F={f1,f2……fm}

m= represent the total no. of parameters

Compute the load degree as: Load_Degree(N)=$\sum^{m}_{i=1}\alpha iFi$

Where i=1…m

Fi=(1<=i<=m; Fi€[0,1])

Average Cloud partitioning load Degree:

Load_Degree_Avg=($\sum^{m}_{i=1}$Load_Degree(Ni))/n

Classify Load

Input:

1. Load Degree.

2. Load Degree Average

Output:

Status Information

- Idle
  Load_Degree(n)=0;
- Normal
  $0<Load\_Degree(n)<=Load\_Degree_{high}$
- Overload
  $Load\_Degree_{high}<=Load\_Degree(n)$.

Non Cooperative load balancing game

Input:

Sji be the fraction of jobs that user j send to computer i .

The vector sji=(Sj1,Sj2……Sjn) is called the load balancing strategy of user j.

The vector Sj = (S1,S2, ….Sm) is called the strategy profile of the load balancing game.

Process:

- The expected response time at computer I is
  Fi(S) = 1/ μi-Σ j=1 to m sjiϕk
- The overall expected response time of user j is given by
  Dj(S)=Σi= 1 to nsji Fi(S) = Σi=1 to nsji/μi-Σk=1 To m skiϕk
- The goal of user j is to find a feasible load balancing strategy S ji such that Dj(S) is minimized

Output:

The decision of user j depends on the load balancing decisions of other users since Dj(S) is a function of s.

CONCLUSION

The overall goal of this project is the balancing of load on clouds. This will improve the performance of cloud services. And prevent the overloading of server, which would otherwise degrade the performance. The response time will also improve. Thus overall performance of cloud services will remains unchanged. In case of the system fails even also partially, it having backup plan. It will maintain the stability of the system. There are provisions to accommodate future modifications in the system.

This algorithm will ensure the optimum utilization of cloud resources. This algorithm will cut the economic cost for an organization because less resources will be required than static algorithms to handle the user requests.

REFERENCES

[1] P. Mell and T. Grance, The NIST definition of cloud computing, http://csrc. nist. gov/ publications/nistpubs/800-145/SP800-145. pdf, 2012.

[2] P. Mell and T. Grance "The NIST definition of Cloud Computing" version 15. National Institute of Standards and Technology (NIST), Information Technology Laboratory (October 7, 2009).

[3] M. D. Dikaiakos, G. Pallis, D. Katsa, P. Mehra, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research", IEEE Journal of Internet Computing, Vol. 13, No. 5, Septe9mber/October 2009, pages 10-13.

[4] A. Khiyaita, H. EI Bakkli, M. Zbakh, Dafir EI Kettani, " Load Balancing Cloud Computing: State Of Art", 2010, IEEE.

[5] Ram PrassdPandhy (107CS046), P Goutam Prasad rao (107CS039). "Load balancing in cloud computing system" Department of computer science and engineering National Institute of Technology Rourkela, Rourkela-769008, Orissa, India May-2011.

[6] K. Ramana, A. Subramanyam and A. AnandaRao, "Comparative analysis of distributed web server system load balancing algorithms using qualitative parameters," VSRD-IJCSIT, Vol. 1 (8), 2011, 592-600 [4]. Network Working Group J.

[7] 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.

[8] 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.

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

[10] GaochaoXu, Junjie Pang, and Xiaodong Fu, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud," IEEE transactions on cloud computing year 2013.

[11] Daniel Grosu and Anthony T. Chronopoulos, "Noncooperative load balancing in distributed systems," Journel of parallel and distributed computing ELSEVIER Comput. 65 (2005) 1022 – 103

AUTHOR'S BIOGRAPHY

Prof. Ms. M. V. Marathe is an assistant professor under department of Computer Engineering. She is having 12+ years' experience in the field of teaching.

Ms. Akshada Bhujbal is a student pursuing her B.E. Degree under Department of Computer Engineering from University of Pune. She is presently working on mathematical module of the load balancing model and cloud environment.

Ms. Prajakta Jakate is a student pursuing her B.E. Degree under Department of Computer Engineering from University of Pune. She is currently involved in system developing activities for developing algorithm for load balancing model.

Ms. Manasi Wagh is a student pursuing her B.E. Degree under Department of Computer Engineering from University of Pune. She is currently involved in system developing activities for developing algorithm and designing GUI.

Ms. Madhura Pise is a student pursuing her B.E. Degree under Department of Computer Engineering from University of Pune. She is currently involved in system developing activities for developing algorithm for load balancing model.