ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

A Load Balancing Model Based on Cloud Partitioning for Public Cloud Using Game Theory

Prakash Krishna Shinde
M.Tech Student,
Department of Computer Science Engineering,
Marri Laxman Reddy Institute of Technology & Management.

Mr G.Prabhakara Reddy
Associate Professor,
Department of Computer Science Engineering,
Marri Laxman Reddy Institute of Technology & Management.

Abstract:

Load balancing in the cloud computing environment has an important impact on the performance. Good load balancing makes cloud computing more efficient and improves user satisfaction. This article introduces a better load balance model for the public cloud based on the cloud partitioning concept with a switch mechanism to choose different strategies for different situations. To provide fairness to all the jobs in the system, we use a cooperative game to model the load balancing problem. Our solution is based on the Nash Bargaining Solution (NBS) which provides a Pareto optimal solution for the distributed system and is also a fair solution. An algorithm for computing the NBS is derived for the proposed cooperative load balancing game. To provide fairness to all the users in the system, the load balancing problem is formulated as a non-cooperative game among the users who try to minimize the expected response time of their own jobs. We use the concept of Nash equilibrium as the solution of our non-cooperative game and derive a distributed algorithm for computing it.

Key words:

Game theory; public cloud; switching mechanism; Nash bargaining solution.

1.Introduction:

Cloud computing is an attracting technology in the field of computer science. In Gartner's report[1], it says that the cloud will bring changes to the IT industry. The cloud is changing our life by providing users with new types of services . A static load balancing problem for both single-class jobs and multi-user jobs in a distributed computer system that consists of heterogeneous host computers (nodes) interconnected by a communication network. Jobs arrive at each computer according to a time-invariant exponential process.

Load balancing is achieved by transferring some jobs from nodes that are heavily loaded to those that are idle or lightly loaded[2].

1.1. Load balancing for single-class jobs:

This load balancing problem is formulated as a cooperative game among the computers and the communication subsystem. The several decision makers (e.g., computers and the communication subsystem) cooperate in making decisions such that each of them will operate at its optimum. The decision makers have complete freedom of pre-play communication to make joint agreements about their operating points. Based on the Nash Bargaining Solution (NBS) which provides a Pareto optimal and fair solution, we provide an algorithm (CCOOP) for computing the NBS for our cooperative load balancing game. The objective of this cooperative load balancing scheme is to provide fairness to all the jobs, i.e. all the jobs (of approximately the same size) should experience approximately the same expected response time independent of the computers allocated for their execution.

1.2. Load balancing for multi-user jobs:

This problem is formulated, taking into account the users' mean node delays and the mean communication delays, as a non-cooperative game among the users. Each user minimizes her/his own response time independently of the others and they all eventually reach an equilibrium. We use the concept of Nash equilibrium as the solution of our non-cooperative game and derive a distributed algorithm (NCOOPC) for computing it. The objective of this non-cooperative load balancing scheme is to provide fairness to all the users i.e. all the users should have approximately the same expected response time independent of the computers allocated for the execution of their jobs (of approximately the same size).

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

1.3.Game Theory:

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.Lets 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 . 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.

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.

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 [2] 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. 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.

C.Mathematical Model:

Study of this mathematical model is based on . 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
-Average Processing Time (APT), where i=1, 2, 3…… n
Øj-Job's Average Throughput where j=1, 2, 3……..
n

$\Phi - \sum_{i=1}^{m} \emptyset_i$

, 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 assi gned 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.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

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 . In these algorithms each user updates from time to time its load balancing strategy by computingthe best reply against the existing load balancing strategies of the other users.

1.4. Switching Mechanism:

End users from different locations submit their jobs to the Cloud Environment. All these jobs are received by a Main Controller which is a single node to manage all the partitions. Nodes under each partition are managed by a Load Balancer. Main Controller distributes the jobs to the Load Balancer by checking its partition status.

The partition may be in 3 states as Idle, Normal and Overload states. The partition status is set by the Load Balancer based on the parameters as Number of CPUs, the CPU processing speeds, the available memory size, the memory utilization ratio, the CPU utilization ratio and network bandwidth etc.,

The jobs are received by the Load Balancers and the Load Balancing algorithms are applied to the partitions. Here the Switching Mechanism is applied. Switching Mechanism is the process of switching over to the 2 different algorithms according to the 2 different situations. Switching Mechanism contains two different algorithms,

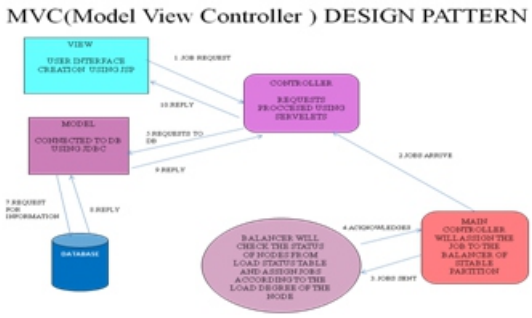
one simple algorithm for Idle state partitions and another one effective algorithm for Normal state partitions. Round Robin is a simple and cheap algorithm that can be used for Idle state partitions. An effective algorithm for Normal state partition should prevent the partitions becoming overloaded state.

2.Related Work:

Several studies have been made on load balancing stratergy for single class and multi-class job strategy the Load balancing in cloud computing was described in a white paper written by Adler[3] who introduced the tools and techniques commonly used for load balancing for public cloud. There are many load balancing algorithms, such as Round Robin, Equally Spread Current Execution Algorithm, and Ant Colony algorithm.Randles et al.[4] gave a compared analysis of some algorithms in cloud computing by checking the performance time and cost. Some of the classical load 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. Game theory approach algorithm is used here because it is fair[2].

3.System Model:

Several cloud computing strategies are focused with public cloud. A public cloud is based on the standard cloud computing model, with service provided by a service provider . There are 3 parts in the design pattern.ViewThe user interface is created using java scripts which is used for a effective front end and application. When user click on the submit button. A job request is sent to controller. ControllerThe business logic is implemented in this module using servelets.ModelThe whole application is connected to the database in this part using Java Database Connection driverFig 1:



3.1 Main controller:

The main controller first assigns jobs to the suitable cloud partition and then communicates with the balancers in each partition to refresh this status information. Since the main controller deals with information for each partition, smaller 1data sets will lead to the higher processing rates.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

3.1.1 Psuedo Code:

STEP 1: START

STEP 2: JOBS ARRIVE AT MAIN CONTROLLER

STEP 3: ASSIGN JOB TO BALANCER (JOB,BALANCER i)

STEP 4: IF (STATUS== OVERLOADED)

STEP 5:ASSIGN JOB TO BALANCER i+1

STEP 6: ELSE

STEP 7:ASSIGN JOB TO BALANCER I

STEP 8:ELSE

STEP 9:RETURN

3.2 Balancer :

The balancers in each partition gather the status information from every node and then choose the right strategy to distribute the jobs. The relationship between the balancers and the main controller is shown in Fig.2..()

3.2.1. Psuedo Code:

STEP 1:CHECK STATUS OF EVERY PARTITION

STEP 2:GET STATUS FROM LOAD STATUS TABLE

STEP 3:IF (STATUS CODE==IDLE||STATUS CODE==NORMAL)

STEP 4:ASSIGN JOB TO PARTITION

STEP 5:ELSE IF(STATUS CODE==OVERLOADED)

STEP 6:GOTO OTHER PARTITION

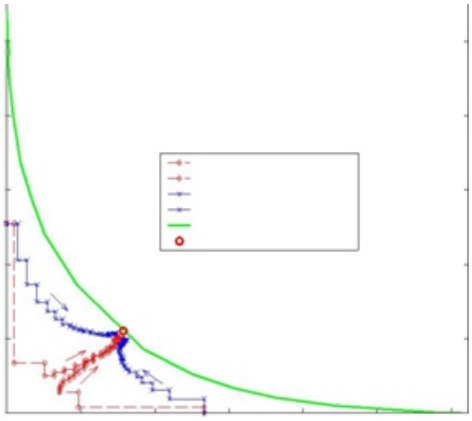
STEP 7:ELSE

STEP 8:RETURN TO MAIN CONTROLLER

STEP 9:STOP

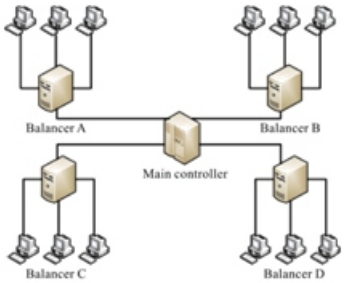
4.Nash Bargaining Solution:

(NBS) for cooperative games. The Nash Bargaining Solution is different from the Nash Equilibrium for noncooperative games. In a cooperative game the performance of each player may be made better than the performance achieved in a noncooperative game at the Nash Equilibrium.Assume that there are M players. Player 1….M has fi(x)_ as objective function. Each fi is a function from X _ to R where X__ (_ a positive integer) is a nonempty, closed and convex set, and _ is bounded above. We want to maximize simultaneously all fi(x)_ Let U_ be the minimal performance required by the players without cooperation to enter the game.In other words,Ui__ represents a minimum performance guarantee that the system must provide to the player i.is called the initial agreement point.The main goal of our work is to provide fairness to the users and the users' jobs i.e., all the users and their jobs should experience approximately equal expected response time (expected queuing delay + processing time + any communication time) or be charged approximately equal price for their execution independent of the computers allocated, and we will show that game theory provides a suitable framework for characterizing such schemes. Most of the previous work on load balancing did not take the fairness of allocation into account or considered fairness in a system without any communication costs.For distributed systems in which all the jobs belong to a single user (single-class), we use a cooperative game to model the load balancing problem which takes the average system information into account (static load balancing). Our solution is based on the Nash Bargaining Solution which provides a Pareto optimal solution for the distributed system and is also a fair solution. We then extend the system model Solution space and solution trajectories for NBS-based and symmetric decentralized algorithms. Arrows indicate direction of convergence of algorithm.



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

to include jobs from various users (multi- user/multi-class job distributed system) and include pricing to model a Grid system. For a grid system, we propose three static price-based job allocation schemes whose objective is to minimize the expected price for the grid users. One scheme provides a system optimal solution and is formulated as a constraint minimization problem and the other two schemes provide a fair solution and are formulated as non-cooperative games among the users. We use the concept of Nash equilibrium as the solution of our non-cooperative games and derive distributed algorithms for computing it. We also extend the proposed static load balancing schemes for multi-user jobs and formulate two schemes that take the current state of the system into account (dynamic load balancing). One dynamic scheme tries to minimize the expected response time of the entire system and the other tries to minimize the expected response time of the individual users to provide a fair solution.



Axiomatic Foundation. Based on 4 axioms first defined by John Nash in 1950 [Nash, 1950], a unique optimal bargaining solution be-tween two agents can be found if the set of feasible solutions is com-pact and convex. Let us define such a two-agent bargaining problem by B = (V1(x), V2(x), d, S), where x F = [x1, x2], is as above with p = 2, Vi : Rni R are the agents' Von Neumann-Morgenstern utility functions [von Neumann and Morgenstern, 1944], d = (d1, d2) R2 is the disagreement point which defines the cost incurred by each agent if no agreement is reached and S R2 is the compact, convex set of all feasible utility pairs that improve on d. We define xB F to be the optimal bargaining solution with optimal utility sB S.Nash showed that a unique optimal solution exists which maximizes the product of the utility functions of both players if the following four axioms are satisfied. It was Nash who first chose to use the product of utilities to determine the Nash Bargaining Solution, and although there is no clear interpretation of this construct in relation to the bargaining problem, its simplicity has allowed for its wide adoption and varied uses (see [Osborne and Rubinstein, 1994] for an alternative formulation).

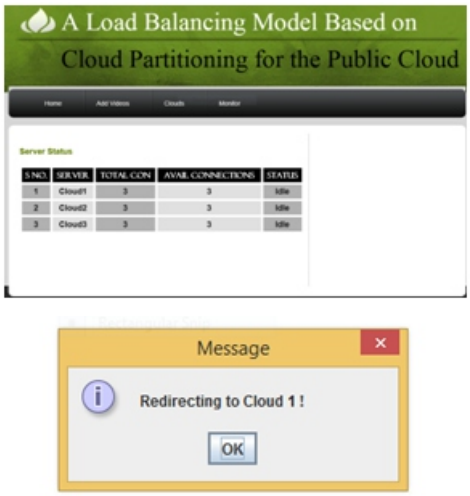
Axiom 4.1 Axiom of Rationality: Each agent prefers the locally optimal solution.

Axiom 4.2 Axiom of Symmetry: If S is symmetric about the line V1 = V2, then the optimal bargaining utility lies on that line.

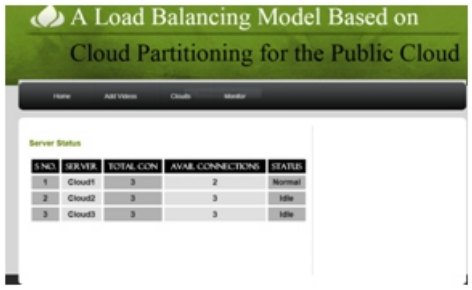
Axiom 4.3 Axiom of Linear Invariance: Neither scaling nor o set of either utility function a ects the resulting bargaining solution.

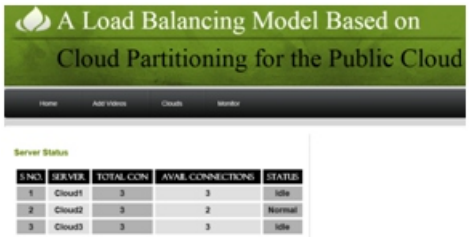
Axiom 4.4 Axiom of Independence of Irrelevant Alternatives:

Proof Outline (After Nash, [Nash, 1950]). To show existence and uniqueness of an optimal bargaining solution, we invoke properties of compactness and uniqueness of S, respectively. To show that the optimal solution maximizes the product of the utilities of both agents, the following elegant set of arguments was developed based on the fouraxioms. If both agents are rational they will try to maximize their local utility, Vi. If both utility functions are linearly invariant, then both can be scaled and o set such that d = (0, 0) and sB = (1, 1). Let B = (V1(x), V2(x), d, S), where S is augmented to include all points such that the sum of the two utilities is less than 2 (ie. let S be the triangle formed by the points {(0, 0), (2, 0), (0, 2)}). Since S is symmetric, by Axiom 4.2, sB _ must be on the line V1 = V2, and thus sB_ = (1, 1). By Axiom 4.4, we see that sB_ S, and so is also the optimal solution to the original problem. The final step is to see that sB is the point of maximum product of utility improvements (V1(x)−d1), (V2(x)−d2), and hence that maximizing the product of utility improvements determines the unique optimal bargaining solution.



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal





CONCLUSION:

It is important to evaluate solutions for cloud balancing implementations with an eye toward support for the needs of an actual IT department. The global and local application delivery solution chosen to drive a cloud balancing implementation should be extensible, automated, and flexible, and the vendors involved need to look favorably upon standards. There are challenges associated with the implementation of such a strategy, some of which might take years to address. But the core capabilities of global and local application delivery solutions today make it possible to build a strong, flexible foundation that will enable organizations to meet current technical and business goals and to extend that foundation to include a more comprehensive cloud balancing strategy in the future.

Acknowledgment:

I would like to thank Prof. ……., for accepting me to work under his valuable guidance. He closely supervises the work over the past few months and advised many innovative ideas, helpful suggestion, valuable advice and support.

REFERENCES:

[1]Xu, Gaochao, Junjie Pang, and Xiaodong Fu. "A load balancing model based on cloud partitioning for the public cloud." IEEE Tsinghua Science and Technology, Vol. 18, no. 1, pp. 34-39, 2013.

[2]P. Mell and T. Grance, —The NIST definition of cloud Computing, online available at: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2012.

[3]N. G. Shivaratri, P. Krueger, and M. Singhal, —Load distributing .

[4]Zhu, Yan, Huaixi Wang, Zexing Hu, Gail-Joon Ahn, Hongxin Hu, and Stephen S. Yau. "Efficient provable data possession for hybrid clouds." In Proceedings of the 17th ACM conference on Computer and communications security, pp. 756-758. ACM, 2010.

[5]A. Rouse, —Public cloud, available at: http://searchcloudcomputing.techtarget.com/definition/public-cloud.

[6]Lori MacVittie, —Cloud Balancing: The Evolution of Global Server Load Balancing, F5 white paper, online available at: http://www.f5.com/pdf/white-papers/cloud-balancing-white-paper.pdf, 2013.

[7]D. MacVittie, "Intro to load balancing for developers — The algorithms, https://devcentral.f5.com/blogs/us/introto-load-balancing-for-developers-ndash-the-algorithms, 2012.

[8]Doddini Probhuling L. —Load Balancing Algorithms In Cloud Computing, International Journal of Advanced Computer and Mathematical Sciences, ISSN 2230-9624. Vol. 4, Issue 3, pp. 229-233, 2013.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

[9]Naimesh D. Naik and Ashilkumar R. Patel —Load Balancing Under Bursty Environment for Cloud Computing, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181, Vol. 2, Issue 6, pp. 17 – 26, June – 2013.

[10]Kaviani, Nima, Eric Wohlstadter, and Rodger Lea. "MANTICORE: A framework for partitioning software services for hybrid cloud." In Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on, pp. 333-340, 2012.

[11]Patel, Parveen, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A. Maltz, Randy Kern et al. "Ananta: cloud scale load balancing." In Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM, pp. 207-218. ACM, 2013.

[12]P. Jamuna and R.Anand Kumar — Optimized Cloud Partitioning Technique to Simplify Load Balancing, International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Volume 3, Issue 11, pp. 820 – 822, November 2013.

[13]S. Chong, J. Liu, A. Myers, X. Qi, K. Vikram, L. Zheng, and X. Zheng, —Building secure web applications with automatic partitioning, in Proceedings of the Symposium on Operating Systems Principles (SOSP), 2009.

[14]G. Hunt and M. Scott, —The Coign automatic distributed partitioning system, in Proc. of the Symposium on Operating Systems Design and Implementation (OSDI), 1999.

[15]R. Newton, S. Toledo, L. Girod, H. Balakrishnan, and S. Madden, —Wishbone: Profile-based Partitioning for Sensornet Applications, in Proc. of the NSDI, 2009.

[16]S. Agarwal, J. Dunagan, N. Jain, S. Saroiu, and A. Wolman, —Volley: Automated data placement for geo-distributed cloud services. in NSDI, 2010.

[17]A. Wieder, P. Bhatotia, A. Post, and R. Rodrigues, —Orchestrating the deployment of computations in the cloud conductor, in NSDI, 2012.

[18]S. Y. Ko, K. Jeon, and R. Morales, —The HybrEx model for confidentiality and privacy in cloud computing, in Proc. of HotCloud, 2011.

[19]A. Li, X. Yang, S. Kandula, and M. Zhang, —CloudCmp: Shopping for a Cloud Made Easy, HotCloud, 2010.

[20]P. Teregowda, B. Urgaonkar, and C. Giles, —CiteSeerx: a Cloud Perspective, in Proc. of the HotCloud Workshop, 2010.

[21]H. L. Truong and S. Dustdar, —Composable cost estimation and monitoring for computational applications in cloud computing environments, Procedia CS, vol. 1, no. 1, pp. 2175–2184, 2010.

[22]B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, —Clonecloud: elastic execution between mobile device and cloud. in Proc. of the European Symposium on Operating Sytems (EuroSys), 2011.

[23]M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, —Cloudward bound: planning for beneficial migration of enterprise applications to the cloud, in SIGCOMM, 2010.

[24]C. Liu, B. T. Loo, and Y. Mao, —Declarative automated cloud resource orchestration, in Proc. of the Symposium on Cloud
Computing, 2011.