Load Balancing in Cloud Computing Systems using Divisible Load Scheduling

Haridas Kataria[1], Vipul Pant [2]

[1] Lecturer CSE, Govt. Polytechnic for Women, Sirsa,
[2]Lecturer CSE, Govt. Polytechnic for Women, Sirsa,

Abstract - Load balancing in cloud computing systems is a big challenge now. As it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfill the required demands, jobs cannot be assigned to appropriate servers and clients individually for efficient load balancing. Here some uncertainty is attached while jobs are assigned.

Our aim is to provide an evaluation study of some of the methods of load balancing in large scale Cloud systems, demonstrating different distributed algorithms for load balancing and to improve the different performance parameters for the clouds of different sizes.

Keywords- Cloud Computing, Load Balancing, throughput, latency

I. INTRODUCTION

"Cloud computing" is a term, which involves virtualization, distributed computing, networking, software and web services. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc. Central to these issues lies the establishment of an effective load balancing algorithm. The load can be CPU load, memory capacity, delay or network load. Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work.

Cloud computing is an emerging computing paradigm which is rapidly gaining consideration in the IT industry. Since cloud computing still is in its infancy, there are many open research challenges. Cloud computing is an on demand service in which shared resources, information, software and other devices are provided according to the clients requirement at specific time. It's a term which is generally used in case of Internet. The whole Internet can be viewed as a cloud. Capital and operational costs can be cut using cloud computing. Always a distributed solution is required. Because it is not always practically feasible or cost efficient to maintain one or more idle services just as to fulfill 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.

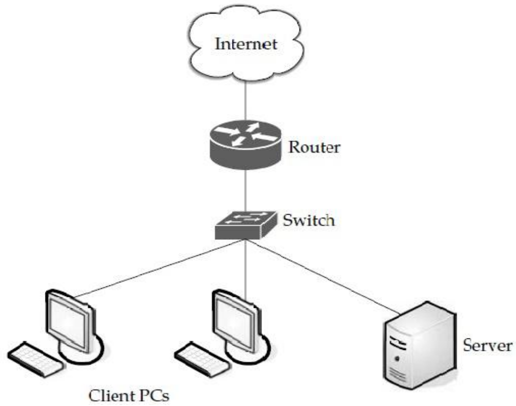

Figure 1: A cloud is used in network diagrams to depict the Internet.

Load balancing is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system.

The important things to consider while developing such algorithm are :

- estimation of load,
- comparison of load,
- stability of different system,
- performance of system,
- interaction between the nodes,
- nature of work to be transferred,
- selecting of nodes and
- many other ones.

This load considered can be in terms of CPU load, amount of memory used, delay or Network load.

Goals of Load balancing:

i. To improve the performance substantially.
ii. To have a backup plan in case the system fails even partially.
iii. To maintain the system stability.
iv. To accommodate future modifications in the system.

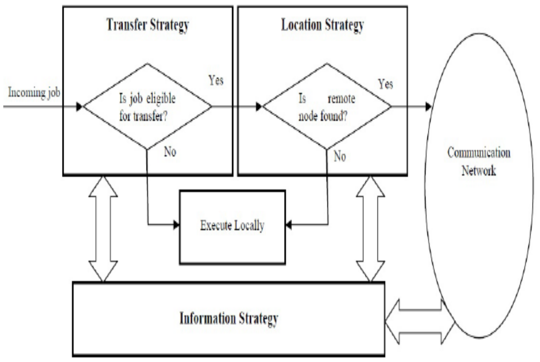

Figure 2: Interaction among components of a load balancing algorithm

This paper is divided into four sections. The Section I give the introduction, Section II represents the architecture of Cloud, Section III represents the proposed work & result evaluation, and finally Section IV concludes the work done.

II. ARCHITECTURE

The architecture of a cloud computing system is usually structured as a set of layers.

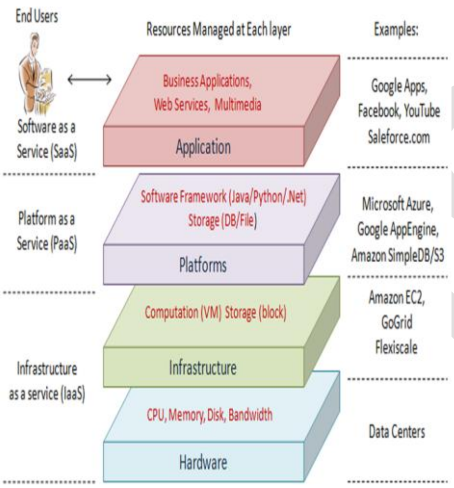

Figure 3: The architecture of a cloud system.

A typical architecture of a cloud system is shown in figure. At the lowest level of the hierarchy there is the hardware layer, which is responsible for managing the physical resources of the cloud system, such as servers, storage, network devices, power and cooling systems. On the top of the hardware layer, resides the infrastructure layer, which provides a pool of computing and storage resources by partitioning the physical resources of the hardware layer by means of virtualization technologies. Built on top of the infrastructure layer, the platform layer consists of operating systems and application frameworks.

The purpose of this layer is to minimize the burden of deploying applications directly onto infrastructure resources by providing support for implementing storage, database and business logic of cloud applications. Finally, at the highest level of the hierarchy there is the application layer, which consists of cloud applications.

III. PROPOSED WORK

Here it is assumed that a cloud consists of many networks and each of those networks has different topology, so, to understand the concept of load balancing, the star topology is taken.

Implementation of a Distributed Load in a Star Network

Step 1: Method for Processing a Task:

Assume that there is a three station that is working on a star network.
Station 1 denoted as a process of a receiving a tasks.
Station 2 denoted as a computing process.
Station 3 denoted as a transmission process.
All of the three station connected with a computer worker and denoted as-

r € {1, 2, ……….., R}

In the process of receiving a task at station 1, the task flow arriving at the receiver is a poison process and the entire process of receiving task from master worker is exponential distribution with a mean value of $M_{1,r}$ task per second.
The processing at station 2 is an exponential distribution with a mean value of $M_{2,r}$ task per second.
The Transmission of the result back to the master worker at the station 3 is performed in a manner that the results are packetized into data packets with exponential distribution. Hence, the transmission of task at the station 3 is exponentially distributed with a mean value of $M_{3,r}$ task per second.
A computing worker r is operating based on the following procedures:
• The computing worker r is a tandem connected sequential processing chain.
• The master worker does not assign a new task to the computing worker r if an application task is in process at Station 1, even if Station 2 and/or Station 3 are empty.
• An application task is blocked when it completes the process at any Station and finds that the next Station is busy.

Step 2: Steady State Diagram with Computing Tasks.

$\mathbf{q_0}$ state defines System is empty.
$\mathbf{q_4}$ state defines Application task is in process at Station 1 only.
$\mathbf{q_6}$ state defines Application tasks are in process at Station 1 and 2 only.
$\mathbf{q_7}$ state defines Application tasks are in process at Station 1, 2 and 3.
$\mathbf{q_5}$ state defines Application tasks are in process at Station 1 and 3 only.
$\mathbf{q_1}$ state defines Application task is in process at Station 3 only.
$\mathbf{q_3}$ state defines Application tasks are in process at Station 2 and 3 only.

q_2 state defines Application task is in process at Station 2 only.

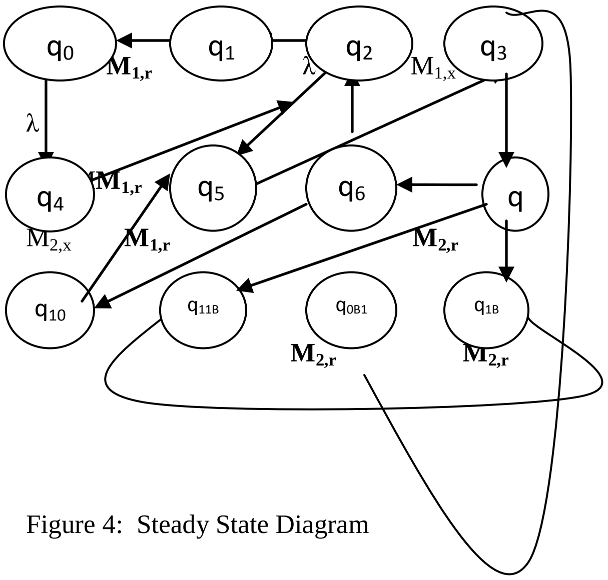

Figure 4: Steady State Diagram

q_{10B} state defines Application task is blocked at the output of Station 1 because Station 2 is occupied.
q_{11B} state defines Application task is blocked at the output of Station 1 because both Station 2 and 3 are occupied.
q_{0B1} state defines Application task is blocked at the output of Station 2 because Station 3 is occupied.
q_{1B1} state defines Application task is blocked at the output of Station 2 because both Station 1 and 3 are occupied.
Figure shows a Markov model for the operating process in the computing worker, where all possible operating states and transitions between all states are presented. When the computing worker is operating in steady-state, its steady state equations (Equations (1)–(12)) can be obtained by the following procedures: Considering the state "q_0", which is directly related to the two states "q_4" and "q_1". When the computing worker in state "q_0" is starting to receive a new task from the master worker, it transits to state "q_4" at the rate of λ, which is an outbound flow from the state "q_0". On the other hand, when the computing worker is in state "q_1" completes the process of transmitting data back to the master worker, it transits to state "q_0" at the rate of $M_{3,r}$, which is an inbound flow into the state "q_0". When the operation is stable, the outbound flows from the state "q_0" is equal to the inbound flow to the state "q_0". Consequently, we obtain Equation (1) as:
$\alpha_r\lambda pq_0 = M_{3r}pq_1$
in which the left side represents the outbound flow and right side represents the inbound flow. Here α signifies the action being taken and symbol p indicates the probability.
Similarly, applying the same strategy to the rest of the states, we can obtain the steady-state Equations (2)–(12) for all the corresponding states. The steady-state equations for this multidimensional Markov processing chain are then as follows:

$\alpha_r\lambda pq_0 = M_{3,r}pq_1$ …………… (1)
$M_{1,r}pq_4 = \alpha_r\lambda pq_0 + M_{3,r}pq_5$ … (2)
$(\alpha_r\lambda + M_{2,r})pq_2 = M_{1,r}pq_4 + M_{3,r}(pq_3 + pq_{0B1})$ …… (3)
$(\alpha_r\lambda + M_{3,r})pq_1 = M_{2,r}pq_2$ …… (4)
$(M_{1,r} + M_{3,r})pq_5 = \alpha_r\lambda pq_1 + M_{2,r}(pq_6 + pq_{10B})$ …………… (5)
$(\alpha_r\lambda + M_{2,r} + M_{3,r})pq_3 = M_{1,r}pq_5$ …………… (6)
$(M_{1,r} + M_{2,r} + M_{3,r})pq_7 = \alpha_r\lambda(pq_3 + pq_{0B1})$ …………… (7)
$(M_{1,r} + M_{2,r})pq_0 = \alpha_r\lambda pq_2 + M_{3,r}(p_7 + pq_{11B} + pq_{1B1})$ ………… (8)
$(\alpha_r\lambda + M_{3,r})pq_{0B1} = M_{2,r}pq_3$ …………… (9)
$(M_{1,r} + M_{3,r})pq_{1B1} = M_{2,r}(pq_7 + pq_{11B})$ …………… (10)
$(M_{2,r} + M_{3,r})pq_{11B} = M_{1,r}(pq_7 + pq_{1B1})$ …………… (11)
$M_{2,r}pq_{10B} = M_{1,r}pq_7$ …………… (12)

Step 3: Processing Time at Computing worker

The calculation of the processing time at each computing worker allows a cloud provider to predict the usage pay on each computing worker. The processing time for a task to be successfully completed in computing worker r is the sum of the processing times taken for that task to be successfully completed in Stations 1, 2 and 3. The average processing time on a computing worker is equivalent to the ratio of the time spent for all the tasks to be successfully completed by the computing worker r over the average of the sizes of all the tasks assigned to the computing worker r.

$$T_r = \frac{Ur}{\alpha r\lambda} = \frac{\sum\sum\sum (n1+n2+n3)pn1n2n3}{\alpha r\lambda}$$

The distributed network may follow different topologies. The tasks are distributed over the whole network. One topological network connects with the other through a gateway. One of the physical topologies forming a cloud is shown in the diagram. This distributed network is a cloud because some of the nodes are Mobile clients, some of them are Thin and some are Thick clients. Some of them are treated as masters and some are treated as slaves. There are one or more data centers distributed among the various nodes which keeps track of various computational details. Our aim is to apply the Divisible Load Scheduling Theory(DLT) proposed for the clouds of different sizes and analyze different performance parameters for different algorithms under DLT and compare them..

IV. CONCLUSION

This paper describes that how divisible load scheduling theory can be applied in case of clouds. It also explains the proposed system model, the various notations used and analysis of measurement and reporting time. Here the inverse link speed b is taken as 1 and the inverse measurement speed a is 0.5 for both the cases. Number of master computers is taken to be constant equal to 50. The time is smaller in case of simultaneous reporting as compared to sequential reporting. It is because in case of sequential reporting, some of the slaves receive almost zero load from its master. Number of effective slaves in this case is less as compared to the simultaneous reporting case. Hence with increase in no. of slaves with respect to a master, the finishing time remains almost same in case of sequential reporting whereas in case of simultaneous reporting, the finishing time decreases for the increase in no. of slaves corresponding to a single master. The finishing time can be improved by increasing the number of slaves under a master computer in a cloud only to some extent before saturation in case of sequential measurement and sequential reporting strategy. But finishing time can be decreased significantly in case of simultaneous measurement start and simultaneous reporting termination by increasing the no. of slaves under a single master computer.

REFERENCES

[1]. Anthony T.Velte, Toby J.Velte, Robert Elsenpeter, Cloud Computing A Practical Approach, TATA McGRAW-HILL Edition 2010.

[2]. Martin Randles, David Lamb, A. Taleb-Bendiab, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, 2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops.

[3]. Mladen A. Vouk, Cloud Computing Issues, Research and Implementations, Proceedings of the ITI 2008 30th Int. Conf. on Information Technology Interfaces, 2008, June 23-26.

[4]. Ali M. Alakeel, A Guide to Dynamic Load Balancing in Distributed Computer Systems, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[5]. Martin Randles, Enas Odat, David Lamb, Osama Abu- Rahmeh and A. Taleb-Bendiab, “A Comparative Experiment in Distributed Load Balancing”, 2009 Second International Conference on Developments in eSystems Engineering.

[6]. Peter S. Pacheco, ”Parallel Programming with MPI”, Morgan Kaufmann Publishers Edition 2008

[7]. Mequanint Moges, Thomas G.Robertazzi, ”Wireless Sensor Networks: Scheduling for Measurement and Data Reporting”, August 31, 2005

[8]. Boost C++ Libraries. Available: http:www.boost.org.

[9]. General algebraic modeling system (GAMS). Available: http://www.gams.com.

[10]. Google AppEngine: Run your web apps on Google’s infrastructure. Available: http://code.google.com/appengine/.

[11]. IBM Smart Cloud. Available: http://www.ibm.com/cloud-computing.

[12]. JTC1/SC22/WG21 - The C++ Standards Committee. http://www.open-std.org/jtc1/sc22/wg21/.

[13]. Linear algebra package (lapack). Available: http://www.netlib.org/lapack/.

[14]. Microsoft connected service framework (CSF).Available:http://www.microsoft.com/serviceproviders/solutions/connectedservicesframework.mspx.

[15]. Kiam Heong Ang, Gregory Chong, and Yun Li. PID control system analysis, design, and technology. IEEE Transactions on Control Systems Technology,13(4):559–576, 2005.

[16]. Panos J. Antsaklis and Anthony N. Michel. Linear Systems. Birhaüser, Boston, 2006.

[17]. Danilo Ardagna, Barbara Panicucci, Marco Trubian, and Li Zhang. Energyaware autonomic resource allocation in multi-tier virtualized environments. IEEE Transactions on Services Computing, 99(PrePrints), 2010.

[18]. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel

[19]. Jr. Arthur E. Bryson and Yu-Chi Ho. Applied Optimal Control: Optimization, Estimation, and Control. Taylor & Francis, revised edition, 1975.

[20]. Jerry Banks, John S. Carson, II, Barry L. Nelson, and David M. Nicol. Discrete-Event System Simulation. Prentice Hall, 5th edition, 2010.

[21]. Jacques F. Benders. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik, 4(1):238–252, 1962.

[22]. John R. Birge and François Louveaux. Introduction to Stochastic Programming. Springer Science+Business Media, LLC, 2nd edition, 2011.

[23]. S. Bittanti, P. Bolzern, and M. Campi. Exponential convergence of a modified directional forgetting identification algorithm. System Control Letter, 14:131–137, 1990.

[24]. Peter Bloomfield. Fourier analysis of time series: An introduction. Wiley- Interscience, 2nd edition, 2000.

[25]. Peter Bodík, Rean Griffith, Charles Sutton Armando Fox, Michael Jordan, and David Patterson. Statistical machine learning makes automatic control practical for Internet datacenters. In Proc. of the 2009 USENIX Conf. on Hot Topics in Cloud Computing (HotCloud’09), 2009.

[26]. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris, and David Orchard. Web Services Architecture. Working Group Note NOTE-ws-arch-20040211, W3C Web Services Activity, Feb 2004.