Available online at www.elixirpublishers.com (Elixir International Journal)

Information Technology

Elixir Inform. Tech. 87 (2015) 35538-35541

ISSN: 2229-712X

An Effective Selection Policy in Load Balancer to Enhance the Cloud Performance

Shoney Sebastian[1] and Ganeshan N[2]

[1]Department of Computer Science, Christ University, Bangalore.

[2]Director (MCA), RICM, Bangalore.

ARTICLE INFO

Article history:

Received: 26 August 2015;

Received in revised form:

26 September 2015;

Accepted: 01 October 2015;

Keywords

Cloud computing,
Selection policy,
Load Balancer,
Dynamic algorithms,
Static algorithms.

ABSTRACT

Due to the wide acceptability of the industry for cloud computing, variety of applications are designed targeting to cloud platform which makes efficient load balancing a major bottleneck for the above. To improve the efficiency of the cloud platform, a judicial distribution of workload among the available resources needs to be ensured. To share the workload in an effective way, a proper load balancing strategy is important. This strategy comprises of information policy, triggering policy, resource type policy and selection policy. Since cloud applications are running in a distributed scenario, through this paper, researcher suggests a selection policy for load balancer in a distributed environment.

Introduction

Software Applications have taken a lead position in the field of Information Technology to reduce the human workload. In the case of distributed applications, the scalability of the application is a matter of concern in the present dynamic scenario. The fast developments in computing resources have reduced the cost of hardware and increased the processing capability of the system remarkably. Still hosting a distributed application in a high end system is not recommended due to many reasons. When there is a huge hit in the application which is beyond the limit of the system, there is no way to scale it. Moreover when the usage of the application is minimum, the entire infrastructure is underutilised.

Cloud computing is an online service model by which hardware and software services are delivered to customers depending upon their requirements and pay as an operating expense without incurring high cost [1]. Basically cloud computing is a set of services that provide Infrastructure resources using Internet media and data storage on a third party server. It has three dimensions known as Software level service, Platform level service and Infrastructure level service [2].

One type of cloud computing environment is a cluster of physical machines maintained in-house by an organization. These physical machines all run as virtual machines on which the organization's own applications will run. By using virtual machines on a number of physical machines, the organization can improve reliability and performance as well as the dynamic provision of the appropriate resources to various applications depending on demand. An alternative scenario is to get the service from a third-party cloud provider in which the organization's requirements will be taken care by the providers. New machine instances can be obtained from the cloud providers using a metered payment plan. This approach allows organizations to gain computing power on demand without the need to maintain hardware [3].

Tele:

E-mail addresses: shoney.sebastian@christuniversity.in

With the rapid expansion of the industry [4], service providers are building new data centers, augmenting the existing infrastructure to meet the increasing demand. Many of the existing applications today really don't require a dedicated infrastructure for the entire application to run. Some of the applications like, Online voting, Online auctioning, flash crowds [5] etc. needs the infrastructure for a specific time period. After the time period, the allotted resources can be rescheduled for other applications in need. So instead of owing a complete infrastructure for hosting such applications, it is better to move such applications into a cloud environment. The entire security of the application will be taken care by the cloud providers. When we move the application to the cloud environment, the main concern is security and scalability. A good designed load balancer can effectively scale it up or down the resources as per the requirements. Identifying the task which is to be migrated during a load balancing process is extremely important to decide the performance of the load balancer. Since the task of rescheduling can bring a lot more internal overheads, it is important to have a minimum overheads in an efficient load balancer during the task migration. The algorithm suggested in this paper for selection policy is designed by keeping Software as a Service platform of cloud computing in mind.

The rest of the paper is organized as follows. In Section II, the authors describe the load balancing problem Section III briefs out the proposed algorithm for triggering a load balancing algorithm. In Section IV, we present the experiment results and section V deals with the conclusion of the work.

Load Balancing Problem

Resource Management and workload distribution are the two main functions of any distributed systems. Workload needs to be evenly shared among the resources to improve the global throughput of these systems. Load balancing problem has been discussed in traditional distributed systems literature for more than two decades. Various algorithms, strategies and policies have been proposed, implemented and classified [6]. There are

two types of Load Balancing Algorithms: static and dynamic. In static algorithms, load balancing decisions are taken at the time of compilation so that the resource requirements are estimated in advance. Where as in Dynamic load balancing algorithms, decisions related to load balancing are taken at the run time after analysing a number of load factors. Load balancing algorithms can be defined by their implementation of the following policies [7]:

a. Information policy: Decide what workload information to be collected, from where it is to be collected and When should collect.

b. Triggering policy: determine the appropriate period to start a load balancing operation.

c. Resource type policy: classifies a resource as server or receiver of tasks according to its availability status.

d. Selection policy: defines the tasks that should be migrated from overloaded resources (source) to most idle resources(receiver).

The paper focuses on to propose a strategy to identify the task to be rescheduled in case of a load unbalance. An algorithm is suggested to identify the over loaded application which in turn needs to be migrated to a new node for stabilising the load of the current node

Load Calculation

No standardized methods are discussed in the literature to calculate the load of a system. T.F. Adbelzaher thinks that load is mainly related to the number of requests and bandwidth [8].

Various load parameters have been discussed in the literature such as CPU queue length, CPU utilization, Load on memory, Load on physical disc reading, load on physical disc writing, number of active connections, network bandwidth etc.

Let n is the number of parameters considered for finding the load occupied by an application i, p1,p2,p3...pn are the load on parameter 1,2,3..n.Then Loadi (of application i) can be calculated as

$$\text{Loadi}=\left|r1r2r3r4.....,rn\right|\begin{vmatrix}Loadp1\\Loadp2\\Loadp3\\.....\\Loadpn\end{vmatrix} \quad \text{eqn (1)}$$

Where s=[r1,r2,r3,....,rn] is a set of constants.[9]

The value of these variables varies depends up on the type of load parameter.

Proposed Algorithm

A typical cloud system consists of a large number of worker nodes located in one place or distributed in different geographical locations which can be interconnected and virtualised to get a massive capacity in terms of CPU, memory, bandwidth, and storage. Each worker node possesses an initial load, which represents an amount of work to be performed, and may have a different processing capacity. To minimize the time needed to perform all tasks, the workload has to be evenly distributed over all nodes which are based on

their processing capabilities. This is where the need of load balancing is arising. The load balancing problem is closely related to scheduling and resource allocation. A proper load balancer evenly distributes workload among the available resources in a system to optimize the average response time of the applications [10].

Even though an efficient load balancer optimise the average response time, the strategy for identifying the task which is to be migrated to the new node in case there is a load imbalance is also equally important to decide the performance. A load balancer can act in a centralised environment and in a distributed environment. In centralised environment, the resource request first comes to load balancer and the load balancer will schedule the load after seeing the load of connected systems. In a distributed environment, the resource request will be randomly assigned to any node and when the node exceeds its limit will invoke the load balancer to redistribute the load to the other nodes. The algorithm suggested in this paper invoke the load balancer in a distributed environment.

As per equation (1), the load of a particular node is the sum of the load of all the applications running on the specific node. In normal cases, the sum of the loads of all the applications will be within the peak threshold value, PT, of the node. But sometimes the application's requirements will change drastically which will lead to a load imbalance. In such cases, the application needs to be migrated to a new node in order to fulfil its requirements as well as to stabilise the load. The proposed algorithm for selecting the resource is given below.

The following notations are used in the algorithm

MTL- Minimum Threshold Load, PTL – Peak Threshold Load, N- total no. of load parameters such as cpu load memory load ect.., TNA- Total No. of Applications running in the node, I – varies from 1 to TNA, LoadI – Load of application I, HL- Application with Highest Load.

1. Calculate the values of MTL and PTL of each node using equation (1)

2. Find the load of the application I on each parameter from 1 to N

3. Calculate LoadI using equation(1) by substituting proper values for r1,r2,r3…rn.

4. Repeat steps 3 and 4 for finding the LoadI of each application running in the specified node where I varies from 1 to TNA with suitable values for r1,r2,r3...rn

5. Find the Mean M1 of the loads from 1 to TNA applications and the mean M2 of the loads, by exempting the load of the highest loaded application from 1 to TNA

6. If (MTL+M2) < PTL and (MTL+M)>PT

{

Migrate the application with load HL to a new node

}

Experiments and Results

Experiment is conducted on a system having the following configuration Pentium® Dual- Core CPU T4200 @ 2.00Ghz 2.00GHz,Memory 2GB,Operating System 32 bit. Five applications having different complexity being developed and each application is tested individually to find out the

load variation in different parameters. Each application is designed to run in a different thread and in every 1 Mille Second the system load consists of different parameters such as CPU load, Memory Load, Physical disc read and Physical Disk Write Load is calculated. Data collected for 100000 Mille Seconds was written to an Excel Sheet and the average value of the load factors calculated for each application.

Experimental Result Analysis

The experiment is conducted for different no of parameters and the observation result is summarized below.

For Two Parameters

Two parameters such as load on CPU and load on Disk Read operation is tested individually on 5 different applications and Table1 shows the calculated Load of different applications when the values of r1=r2=.5. Figure 1 shows the corresponding graph.

Table 1

Sl No	App No.	CPU load	disk read	r1	r2	loadi
1	App1	42.74378	13131	0.5	0.5	6586
2	App2	67.23621	2310.4	0.5	0.5	1186
3	App3	72.50488	3919.4	0.5	0.5	1995
4	App4	64.34736	21970	0.5	0.5	11017
5	App5	62.11647	8E+06	0.5	0.5	4E+06

Table 2

App Nos	CPU load	Mem load	diskread(a)	r1	r2	r3	Loadi
App1	42.74378	1028	13130.73	0.25	0.25	0.5	6833.05
App2	67.23621	1036	2310.417	0.25	0.25	0.5	1431.017
App3	72.50488	1017	3919.391	0.25	0.25	0.5	2232.072
App4	64.34736	1256	21970.44	0.25	0.25	0.5	11315.31
App5	62.11647	954	20120	0.25	0.25	0.5	10314.03

Table 3

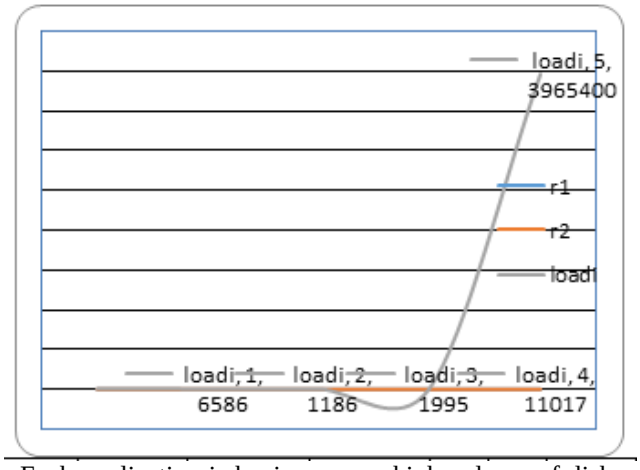
App Nos	CPU load	Mem load	disk read	disk write	r1	r2	r3	r4	Loadi
App1	4.27E+01	1028	1.31E+04	1.74E+03	0.25	0.25	0.25	0.25	3.99E+03
App2	6.72E+01	1036	2.31E+03	2.43E+03	0.25	0.25	0.25	0.25	1.46E+03
App3	7.25E+01	1017	3.92E+03	1.20E+03	0.25	0.25	0.25	0.25	1.55E+03
App4	6.43E+01	1256	2.20E+04	7.42E+03	0.25	0.25	0.25	0.25	7.68E+03
App5	6.21E+01	954	1.75E+04	9.87E+03	0.25	0.25	0.25	0.25	7.10E+03

From the graph, the load of application5 gone to extreme value which will result in to a high value in the mean M. if the MTL+M value exceeds the PTL, then application 5 needs to be migrated to a new node.

For 3 parameters

In this experiment, three load parameters, Load on CPU, Load on disk read operation and Load on disk write operation, tested in 5 different applications.

Figure 1



Each application is having a very high volume of disk read operation which is considered during the load calculation by giving a higher value to the constant r3. Table 2 shows the average values of load on these parameters and the overall system load. From the graph shown in Figure2, the variation of load on different applications is minimum. In this case the mean M1 and M2 will be almost same which indicate that the node will be maintaining the load balancing among the running applications. In case a load imbalance then application 4 will be taken out.

For 4 parameters

The parameters such as CPU load, Memory Load, Disk read operation and Disk write operation are tested across all five applications with an interval of 1 millisecond and sample data is collected for 100000 millisecond. From the sample data, the average values of CPU load, memory load, Load on disk read and Load on disk write is calculated for each application which is summarised in Table 3 along with the overall load of the system. From graph shown in Figure 3, it is clear that the load requirements of different applications are almost same which means not much variation in the values of M1 and M2. In case of a load unbalance, application 4 needs to be transferred to a new node to make the current node stable.

Figure 2

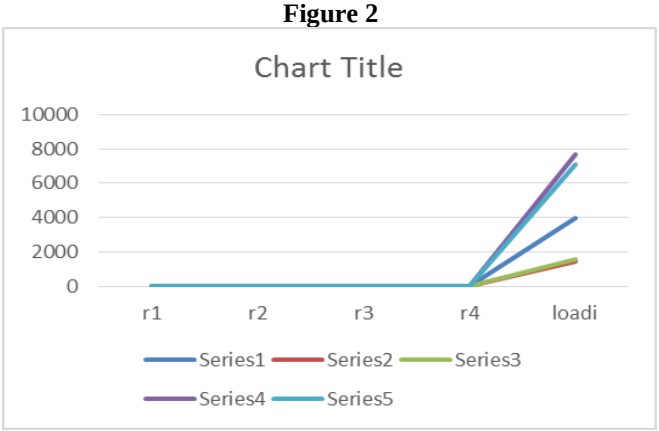
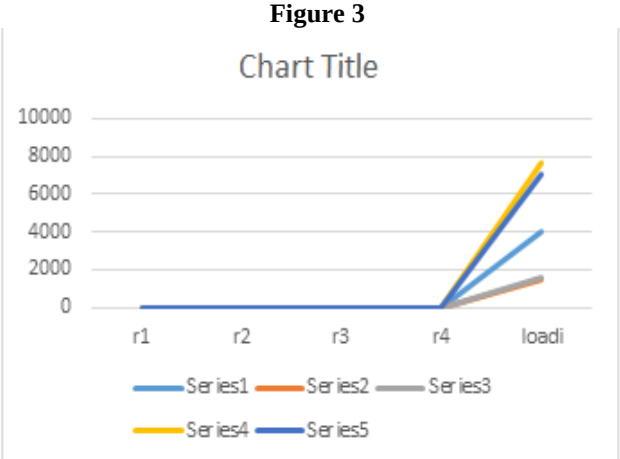


Figure 3



Conclusion

The paper addressed the problem of load balancing in cloud environments and the importance of having an effective selection policy to improve the performance of the load balancer. The algorithm is suggested by keeping in mind the distributed nature of cloud platform. Work load balancing is a major research challenge in the Cloud computing domain, we hope that the approach discussed in this paper for selecting a task for migration,in case of a load imbalance will definitely improve the performance of Cloud Computing.

References

[1] S. Bandyopadhyay, S. R. Marston, Z. Li, A.Ghalsasi,"Cloud Computing: The Business Perspective", November 2009.

[2] R. Fox,"Digital Libraries: The systems analysis perspective", Library in the Clouds, vol. 25, no. 3, pp. 156-161, 2009.

[3] Emil Ong, "Resin 4.0 Technical White Paper"

[4]Facebook Statistics. https://www.facebook.com/ press/info.php?statistics.

[5] Shibu Daniel, Minseok Kwon. "Prediction-based Virtual Instance Migration for Balanced Workload in the Cloud Datacenters",

[6]Javier Bustos Jimenez, Robin Hood: An Active Objects Load Balancing Mechanism for Intranet.

[7] H.D. Karatza. Job scheduling in heterogeneous distributed systems, Journal. Of Systems and Software, 56:203-212,1994

[8] Adbelzaher T F, Bhatti N. Web Server Queue by Adaptive Content Delivery,International Workshop on Quality of Service, London, UK,1999.

[9] Zhang Ying, "An Improved load balancing strategy based on LVS", Science & Technology,2008

10] K.Y. KABALAN, W.W. SMAR AND J.Y. HAKIMIAN. "Adaptive load sharing in heterogeneous systems: policies, modifications and simulation", *Int. Journ. of SIMULATION*, vol. 3, num. 12, pages 89-100, 2002.