International Journal of Engineering Research and General Science Volume 3, Issue 4, July-August, 2015
ISSN 2091-2730

Dynamic Approach for Load Balancing in CMS

Karishma Bhagwan Badgujar[1], Prof. P. R. Patil[2]

Department of Computer Engineering,

Pune Institute of Computer Technology,

Pune, India.

karishmabadgujar13@gmail.com[1], pictianpravin@gmail.com[2]

Abstract — A centralized hierarchical cloud-based multimedia system (CMS) consist of a resource manager, server clusters and according to the task characteristics cluster heads in which for multimedia service tasks the resource manager assigns clients' requests to server clusters, and then to the servers within its server cluster and then each cluster head distributes the assigned task. For such a complicated CMS, however, to design an effective load balancing algorithm, it is a research challenge that spreads the multimedia service task load on servers with the minimal cost, for transmitting multimedia data between server clusters and clients, while the maximal load limit of each server cluster is not violated. So, in this paper, each server cluster only handles a specific type of multimedia task in a more practical dynamic multiservice scenario, and each client requests a different type of multimedia service at a different time. By such scenario an integer linear programming problem can be modelled, in general which is computationally intractable. Also an efficient genetic algorithm also solved by this system with an immigrant scheme, for dynamic problems which has been shown to be suitable. In CMS, with dynamic multiservice load balancing the proposed genetic algorithm can efficiently cope which is demonstrate by simulation results.

Keywords — Cloud computing, genetic algorithm, load balancing, multimedia system.

INTRODUCTION

For various multimedia computing and storage services there a huge number of users' demands through the Internet at the same time because of that cloud-based multimedia system (CMS) [3] emerge. In a distributed system, through the Internet there are storing and processing their multimedia application data and meet different multimedia QoS requirements, a huge number of clients simultaneously which is supported by infrastructure, platforms, and software. For most multimedia applications Considerable Computation are required, which are often performed on mobile devices with constrained power. So that there strongly required the assistance of cloud computing. In general, the utilities based on cloud facilities offer the cloud service providers to client. So that to request multimedia services and process clients does not need to take much cost. By doing so, for the utilized resources by the time, the clients only need to pay, and on powerful cloud servers multimedia applications are processed.

A number of server clusters and a resource manager composed by a centralized hierarchical CMS (as shown in Fig. 1) and each of which is coordinated by a cluster head, and we assume the servers in different server clusters to provide different services. Such a CMS is operated as follows. Each time for different server clusters the CMS receives clients' requests for multimedia service tasks; the resource manager of the CMS assigns those task requests according to the characteristics of the requested tasks. Subsequently, for some server within the server cluster the cluster head of each server cluster distributes the assigned task. The performance of the whole CMS is affected by the load of each server cluster significantly which is not hard to observe. In general, the resource manager of the CMS is distributing the task load across server clusters, and so that, it can be able to cope with load balancing in the CMS.

The remainder of this paper is organized as follows: Section 2 introduces the literature survey that describes existing load balancing techniques with their pros and cons. Section 3 gives our problem description. Section 4 describes the proposed system. Section 5 discusses the mathematical model for proposed system. Section 6 describes about the results. Conclusion is drawn in Section 7.

RELEATED WORK

Following are some existing systems which are previously stated by researchers:

A. Round Robin Algorithm:

In round robin framework [4], no of methods are divided in the middle of all processors. Allotment of courses of action is carried out in round robin manner. Here remote processor is in charge of designation and keeping up the request of courses of action in round robin manner. Here dispersion of burden between courses of action is equivalent however execution time require for each one employment is diverse. Because of that a few techniques burden might intensely stacked and others may stay perfect. Here all http appeal having comparative nature and are utilized as a part of web servers and disseminated just as.

B. Connection Mechanism:

In connection mechanism [5], load balancing algorithm is depend on part of dynamic scheduling algorithm known as least connection mechanism. Initially it counts number of active connection and depending upon that it distributes load among them. Each connection is assign by index and that index is maintained at load balancer. Index number increases when a new connection is added and decreases if any connection finishes.

C. Throttled Load Balancing Algorithm:

Throttled algorithm [6], uses virtual machine technique. Here load balancer checks the client request load and perform that operation on particular server which can able to handle load. Here selection of load server is done by load balancer depending upon client request.

D. Equally Spread Current Execution Algorithm:

This algorithm [7], utilizes need of methods to be executed in processor. It ascertain the heap of each one procedure and exchange it to virtual machine which conveys load among light weight processors disseminate the heap arbitrarily by checking the size and take less time, so give amplify throughput. Here burden balancer disseminates the heap into numerous virtual machines so this strategy is known as burden spreading system.

E. Task Scheduling Algorithm:

Y. Fang et al. [7], proposed a system of two-level undertaking planning which is focused around burden adjusting. It used to meet necessities of element clients which will help to get use of high asset. It firstly maps burden adjusting errands to virtual machines and host assets which enhance reaction time of specific assignment alongside asset use and execution of the cloud.

F. Biased Random Sampling:

M. Rundles et al. [8], gives new strategy for adaptable and dispersed burden adjusting methodology for framework area irregular inspecting which adjusts the heap in the middle of hubs of the framework. This produces a virtual chart of every hub which speaks to server load. Hub is meant as an image in chart and steered as level of assets. It uses decentralize burden adjusting plan which ready to handle huge system framework load.

G. Min-Min Algorithm:

In Min-Min algorithm [9], user request are considered as unassigned tasks. Among them minimum completion time task is figure out. Again from that minimum task minimum time is separated which require less time to complete. Depending upon time requirement of tasks then it is assigned to respective machine. When task is completed then it is removed from list of task assigned to machine and its execution time is assigned to newer upcoming task. This procedure is repeated until the user requests are being assigned to server. But it can lead to starvation its maindrawback of this system.

H. Max-Min Algorithm:

Min-min algorithm and max-min algorithm [9], both are same but they differ only after finding out minimum assigned times. Among all tasks it selects the maximum task which requires maximum time to execute. Depending upon that time task is scheduled on particular machine. After that maximum time of tasks registered on server machine and the completed task time is removed from that machine.

I. Token Routing Algorithm:

In this paper [10], the task is divided into token and this token is distributed among all available machines. Main problem arises when bottleneck problem come in picture. Also load distribution criteria are not fixed here. Heuristic approach is use to remove the drawback of token ring algorithm based on load balancing. Decision taken by this algorithm is fast and effective. Here agent does not need to have knowledge about their own load and neighbors load. They build their own base of knowledge to pass the token and to make decision. And it is actually derived from previously generated tokens. So it avoids creation of communication overhead.

In some previous works on other issues of cloud computing or distributed computing have also existed, Also note that GA has been applied to dynamic load balancing in, but their GA was designed for distributed systems, not specific to the CMS. In addition, they did not have any multiservice concern.

PROPOSED SYSTEM

A. System Overview:

Centralized and decentralized are two categories of CMS. This paper considers a centralized CMS as illustrated in Figure 1, which consists of a number of server clusters as well as a resource manager each of which is coordinated by a cluster head. Different from the decentralized CMS, for multimedia service tasks each time it receives clients' requests, the global service task load information collected from server clusters adored by the resource manager of the centralized CMS, and decides the amount of client's requests assigned to each server cluster so that in terms of the cost of transmitting multimedia data among client and server clusters, the load of each server cluster is distributed as balanced as possible. The decision of assignment is based upon the characteristics of different service requests and the information collected from server clusters. As fewer overheads are imposed on the system the centralized framework is scalable in comparison to a decentralized framework, and hence, there are a lot of applications have implemented centralized framework.

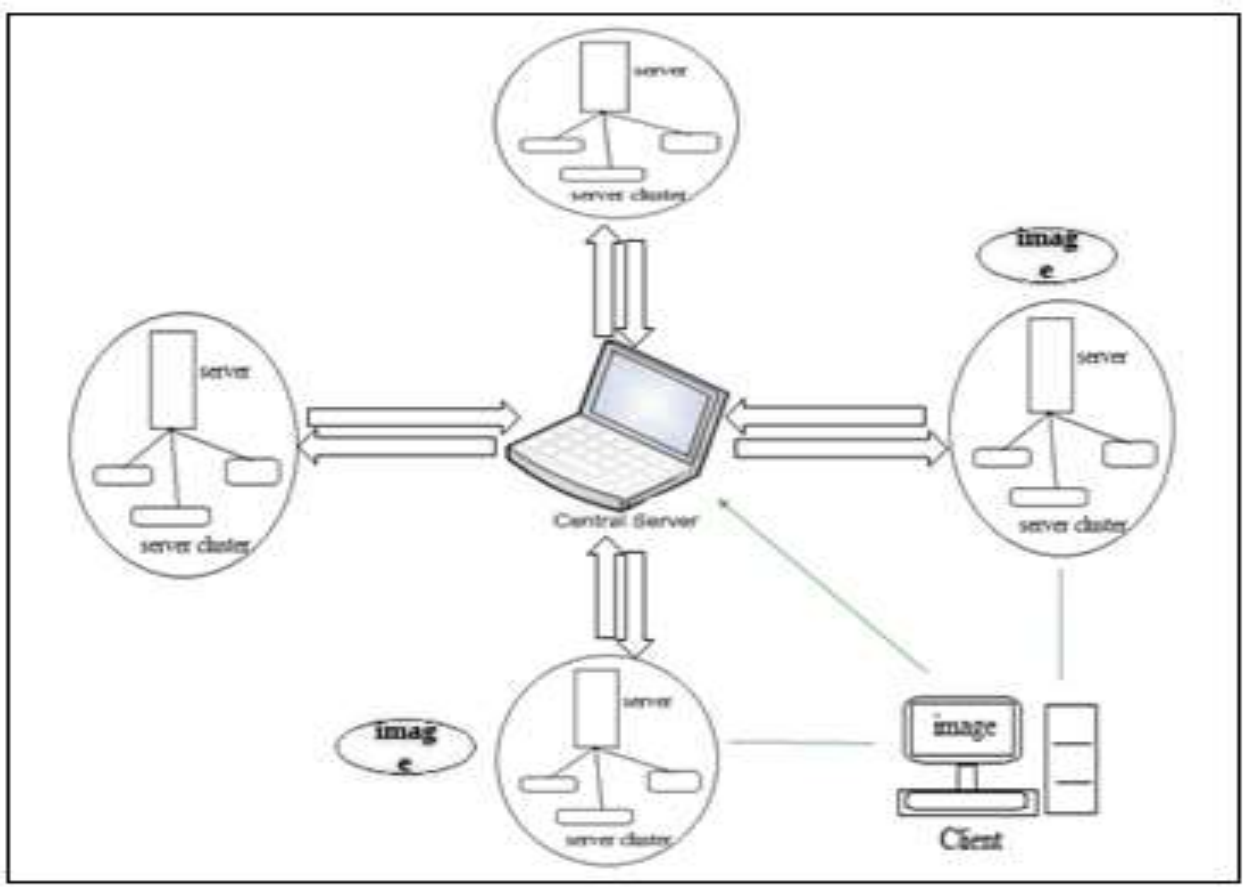

Fig. 1: System Architecture [1]

Here we are going to achieve load balancing based on the type of user request. That means if user request for video data upload and download then it will be handled only by video server clusters. Here we are going to use four main modules:

1. Service Module:

When a client requests for multimedia upload and download, then the request of client is being handle by central server. Depending upon the request, data is categorized to cluster head. Initially clients request is send to that server which is nearest for distance to client machine. At that server, client requests will be handling. If nearest server is busy or overloaded at that instance then request is transfer to next nearby server.

2. Proximity Module:

In this module, server capacity is maintained and not to be exceeded. This module checks consistency of data provided by server to client by limiting client request load. It also calculates network proximity using Genetic algorithm (GA) for values of client request and server load.

3. Weight Calculation Module:

This module measures CPU usage of server in each category. Then it compute landmark order and server utilization ratio. It is also responsible for minimization and calculation of weight and link assessment.

4. Load Balancing Module:

Initially this module consider weighted bipartite graph of client requested cluster head. Then it removes links which do not provide reliable data. After that it calculates network proximity. Then it balances load by computing latency order produced by weight calculation module.

B. Algorithm Used:

Step 1: Initially server status will be 0 as all the servers are available. Resource Manager maintains a data structure comprising of the Job ID, Server ID and Server Status.
Step 2: When there is a queue of requests, the cloud manager parses the data structure for allocation to identify the least utilized server. If availability of servers is more then, the server with least hop time is considered.
Step 3: The Resource Manager updates the data structure automatically after allocation.
Step 4: The Resource Manager periodically monitors the status of the servers for the distribution of the load, if an overloaded server is found, and then the c\Resource Manager migrates the load of the overloaded server to the underutilized server.
Step 5: The decision of selecting the underutilized server will be based on the hop time. The server with least hop time is considered.
Step 6: The Resource Manager updates the data structure by modifying the entries accordingly on a time to time basis
Step 7: The cycle repeats from Step2.

RESULT AND DISCUSSION

To process the different jobs, cloud resource manager redirects the job to specific server. Cloud resource manager identifies the request, and then selects the Cluster specific to multimedia type & request the best available server to process the job. Servers are configured & are responsible to process the job effectively and minimize the response time. The following table shows the Jobs & available servers to handle the request. Proximity & Weight calculation module identifies the best available node to process the request.

Table 1: Comparison

Job Id	Server Id
Job_1, Job_2	S1
Job_1, Job_2, Job_n	S2
Job_2 , Job_6	S3
⋮	⋮
Job_4, Job_n	S_n

The comparison of methods is shown here for existing approach & dynamic approach for load balancing:

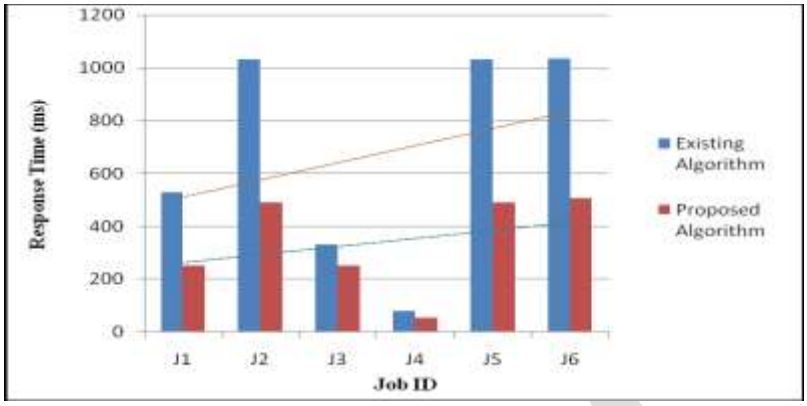

Fig. 2: Graph of Job Processing Comparison

CONCLUSION

The load balancing is implemented in the cloud computing environment to provide on demand resources with high availability. But the existing load balancing approaches suffers from various overhead. The enhanced load balancing approach using the efficient cloud management system is proposed to overcome the aforementioned limitations.

Henceforth, propose technique can produce better and efficient mechanism for load balancing than existing systems. It also refers servers by distance that means request will be redirected to nearest server first for achieving efficient balancing. The evaluation of the proposed approach will be done in terms response time and also by considering the hop time.

REFERENCES:

[1] Karishma B. Badgujar et al, Dynamic load balancing Mechanism in multiservice cloud storage, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014.

[2] Chun-Cheng Lin, Hui-Hsin Chin, Student and Der-Jiunn Deng, Dynamic multiservice load balancing in cloud-based multimedia system, IEEE Systems Journal, vol. 8, no. 1, march 2014.

[3] W. Hui, H. Zhao, C. Lin, and Y. Yang, "Effective load balancing for cloud-based multimedia system," in Proc. Int. Conf. Electron. Mech. Eng. Inform. Technol., 2011, pp. 165–168.

[4] Zhong Xu, Rong Huang (2009), "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems", CS213 Parallel and Distributed Processing Project Report.

[5] P. Warstein, H.Situ and Z.Huang (2010), "Load balancing in a cluster computer", in proceeding of the seventh International Conference on Parallel and Distributed Computing, Applications and Technologies, IEEE.

[6] Ms. Nitika, Ms. Shaveta, Mr. Gauravraj; "Comparative Analysis of Load Balancing Algorithms in Cloud Computing", International Journal of Advanced Research in Computer Engineering & Technology Volume 1, Issue 3, May 2012.

[7] Y. Fang, F. Wang, and J. Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing", Web Information Systems and Mining, Lecture Notes in Computer Science, Vol. 6318, 2010, pages 271-277.

[8] T. R. V. Anandharajan, Dr. M. A. Bhagyaveni, "Co-operative Scheduled Energy Aware Load-Balancing technique for an Efficient Computational Cloud", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011.

[9] T. Kokilavani, J.J. College of Engineering & Technology and Research Scholar, Bharathiar University, Tamil Nadu, India "Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing", International Journal of Computer Applications (0975 – 8887) Volume 20– No.2, April 2011.

[10] ZenonChaczko, VenkateshMahadevan, ShahrzadAslanzadeh, Christopher Mcdermid (2011), "Availabity and Load Balancing in Cloud Computing", International Conference on Computer and Software Modeling IPCSIT vol.14 IACSIT Press, Singapore 2011.