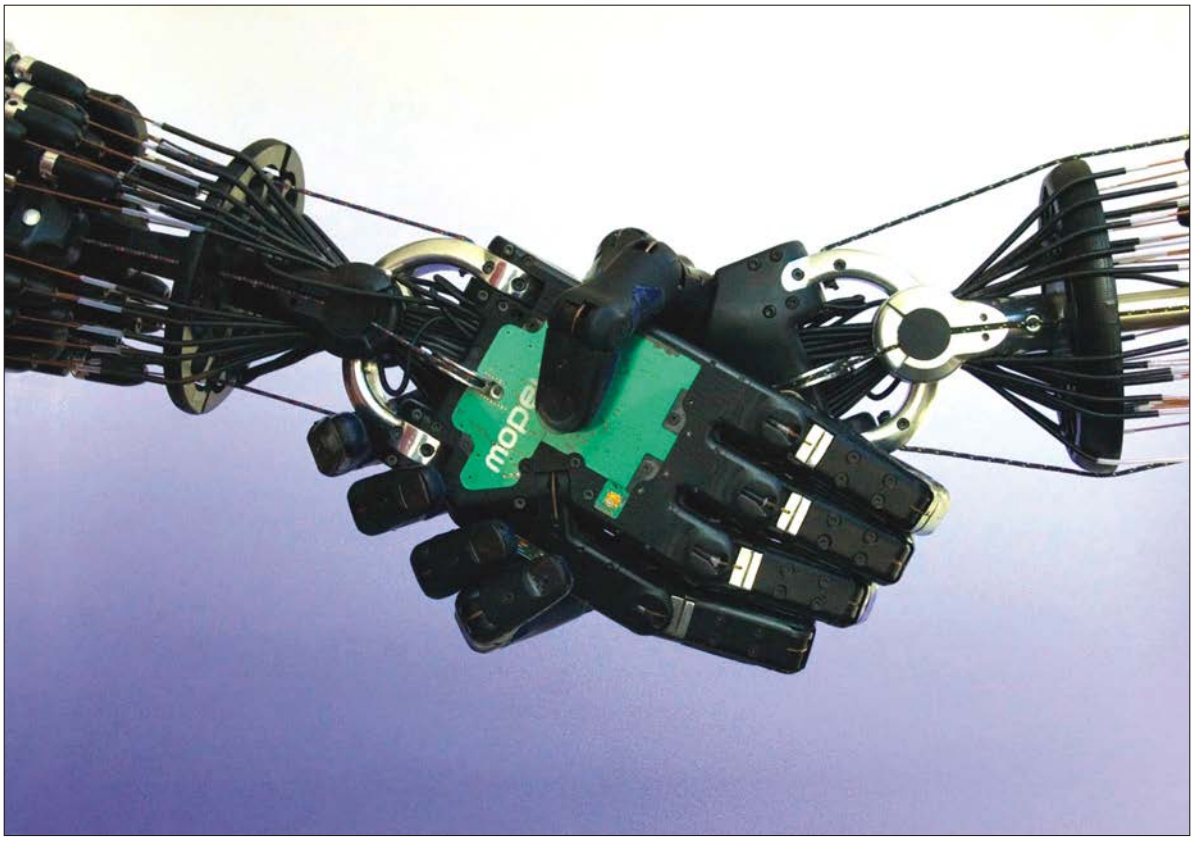

Journal of Artificial Intelligence

ISSN 1994-5450

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Artificial Intelligence 5 (4): 200-206, 2012
ISSN 1994-5450 / DOI: 10.3923/jai.2012.200.206

Priority Based Load Balancing in Cloud for Data Intensive Applications

[1]E. Iniya Nehru, [2]S. Sujatha, [3]P. Seethalakshmi and [4]N. Sridharan
[1]National Informatics Centre, Chennai, Tamil Nadu, India
[2]Department of Computer Applications, [3]Department of Computer Science, Anna University of Technology, Tiruchirapalli, Tamil Nadu, India
[4]Department of Information Science and Technology, Anna University, Chennai, Tamil Nadu, India

Corresponding Author: E. Iniya Nehru, National Informatics Centre, Chennai, Tamil Nadu, India

ABSTRACT

The number of Internet users is growing at an exponential rate every day. Large number of users tries to retrieve the data in many applications like examination results, which leads to very high load on a server. This also results in reduction of throughput and there is strong need for developing a system with an efficient load balancing algorithm to retrieve the intensive data within a reasonable response time. By the use of cloud technology, this can be achieved. Cloud Computing encompasses virtualization, distributed environment and provides on-demand services. The objective of this study was to balance the requests by identifying the IP address and use a predefined policy to retrieve the data from a distributed database environment using virtualization techniques.

Key words: Cloud computing, virtualization, distributed database, load balancing, IP addressing

INTRODUCTION

Cloud Computing has become an enabling technology after Distributed Computing, Parallel Processing and Grid Computing. It exhibits characteristics like scalability, reliability, multi-tenancy, empowerment of end-users. It supports various deployment models and uses Service Oriented Architecture (SOA), reduces information technology overhead for the end-user, gives greater flexibility, location independence, provides on-demand services and elasticity etc. (http://en.wikipedia.org/wiki/Cloud_computing).

Now-a-days many platforms are available to deploy an application in the cloud. Eucalyptus an open-source platform is used to provide Infrastructure as a Service (IaaS). Private cloud computing environment can be created by using this platform. VMware, Xen and KVM hypervisor are some of the virtualization technologies which are also supported by *Eucalyptus* (Baun and Kunze, 2009) to implement the cloud abstractions. The proposed system has been designed to support load balancing in cloud environment.

In order to achieve load balancing, information is stored in clusters located in the same physical location or even spread over a network of inter-connected computers by means of distributed database technology. MySQL Cluster is a prominent distributed database management system which is designed to provide high availability of data across data nodes and high performance. It supports Replication, Horizontal data partitioning and Hybrid Storage. In the proposed system, both cloud computing and distributed database technologies are used to provide a better load balancing environment for data intensive applications.

MySQL Cluster Database provides services with full capability to cover the peak demands. In Cloud environment based on demand, requests are assigned to the different VM's (Nurmi *et al.*, 2009). Requests are sent across a farm of MySQL cluster servers in which the data are replicated to avoid single point failure and also does load balancing (Kaitsa *et al.*, 2009), there-by a better response time is achieved. Advanced eager scheduling (Neary and Cappello, 2003) achieves fault tolerance and load balancing by dynamically breaking down the tasks and by performing parallel computing.

The performance of distributed server is improved by FIRM (Serpanos and Antoniadis, 2000) in which the requests are distributed by First Come First Serve and with Round Robin basis. FIRM achieves saturation throughput and a guaranteed service with minimum response time.

Non-preemptive scheduling leads to poor performance like a processor intensive job getting assigned to a slow machine and also due to excessive idle times. Speed of computations can be improved (McLaughlin *et al.*, 1998) using preemptive scheduling. A new genetic algorithm based task scheduling has been proposed and its performance has been proved on many applications on the grid (Akhtar, 2007). A method for implementing distributed database applications on the Web using Java (Duan, 1996) has been proposed. When Java based approaches are used on the web then high degree of object mobility is also achieved. An interference aware scheduler has been designed to handle high data intensive jobs (Chiang and Huang, 2011). Iterative application of map reduce paradigm will produce good response time for most data intensive applications (Fox, 2011).

The data and information typically are stored in a centralized storage that makes easy for administrator to manage and manipulate those data. However this method is limited by the capacity of database server and the way it is processed. Distributed storage techniques has been used for developing personalized e learning systems where intelligent agents are used to enhance modularity, reusability and reliability (Al-Sakran, 2006). Information is stored in many nodes (Pukdesree *et al.*, 2006) by the concept of distributed database. Two approaches one on simple data intensive applications and the other on distributed file systems are applied and their performance are compared (Miceli *et al.*, 2010).

A single shared cluster can support multiple applications using database replication policies (Soundararajan *et al.*, 2006). Peak load conditions and failure conditions are easily handled to maintain application level performance using these replication policies.

Google App Engine (GAE) Datastore API and distributed database technologies (Bunch *et al.*, 2010) like AppScale can be used. It analyzes how each database differs in implementing the API. Also it describes the implementation and use the platform to empirically evaluate each of the databases. In addition, it says that AppScale can be integrated with each database.

MySQL Cluster Network Database (Hutchings *et al.*, 2010) improves the performance of database utilization in terms of distributed data and distributed processing. The data is stored in each storage node in the cluster by using hashing function. Virtualization technology in data centers has been implemented to improve the efficiency of the data centers (Uddin *et al.*, 2012). Here there will be more than one database server in the system, but the data itself is stored in a centralized storage which is shared. In this case, each database server will handle user's requests in parallel.

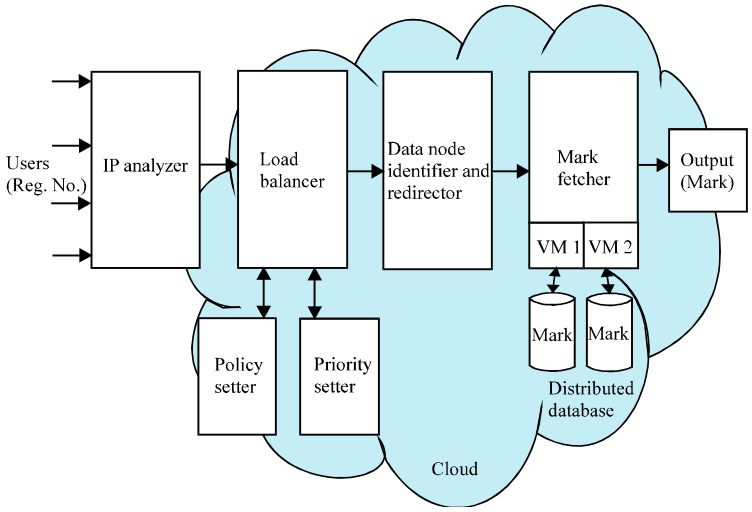

Fig. 1: System architecture of priority based load balancing in cloud for student mark retrieval system

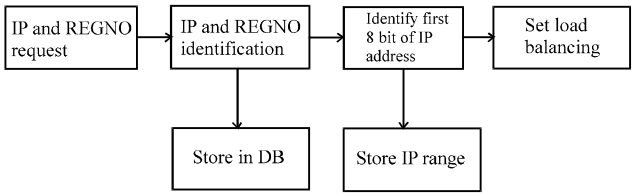

Fig. 2: Block diagram of IP analyzer

SYSTEM DESIGN

Student mark retrieval system has been considered as a case study for a data intensive application. The architecture of the system is described in terms of its components and their functionalities. Figure 1 shows the overall architecture of the Student Mark Retrieval System in which IP based Priority and Round Robin Load Balancing in cloud is implemented.

The overall system architecture shows two major processes namely, student marks database creation in a distributed environment and retrieval of mark by load balancing. Student marks database creation involves creating many MySQL instances with replication of data in all Virtual Machines. Retrieval of marks involves priority based on the incoming IP address, the defined policy settings and then assigning the request to a suitable virtual server.

IP analyzer: In IP Analyzer, IP address of the incoming request is analyzed. The Fig. 2 shows the block diagram of IP Analyzer. The IP address and register number are stored in a database with its number count. From the incoming requests, IP address range will be identified by analyzing its first 8-bit block. Based on that, Load Balancing and Policy Setter are done.

Load balancer and policy setter: In Load Balancer and Policy Setter, the identified first 8-bit of IP address will be checked for the frequency of visits from the same 8-bit IP address block and

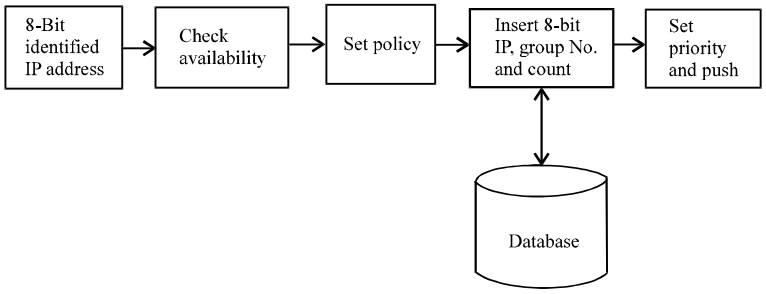

Fig. 3: Block diagram of load balancer and priority setter

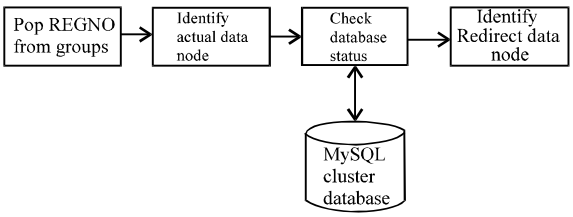

Fig. 4: Data node identifier and redirector

based on number of visits and the policy, the request is pushed into a suitable virtual server. Figure 3 shows the block diagram of Load Balancer and Policy Setter. Also after pushing the request, a counter is used to count the number of visits from the same 8-bit address block in a database. As and when the request is completed, the count value for this 8 bit IP block is decremented. When the entire database servers get filled by the maximum requests possible to be processed, the new requests waits till any one of the database server have a space to process the requests.

Data node identifier and redirector: Data Node Identifier and Redirector checks the counter values for each IP address block in each VM, the VM which has the lowest value in the counter is selected and the requests will be redirected to this VM. Figure 4 shows the block diagram of data node identifier and redirector. MySQL connections are established with the MySQL server and student marks are retrieved from the distributed database. Thus by having data in many VM's/nodes the marks will be retrieved from any of the data nodes.

IMPLEMENTATION

Distributed database creation: MySQL Cluster provides shared-nothing clustering capabilities for the MySQL database management system. MySQL Cluster is implemented through an additional storage engine available within MySQL called Network Database (NDB) or NDBCLUSTER. MySQL Cluster uses three different types of nodes (processes). Figure 5 shows the block diagram of the Distributed Database creation in two hosts.

Data storage and retrieval in data nodes is done using a MySQL server (mysqld). Data nodes can be queried directly using the NDB API. Student marks are stored in a distributed database by replicating the data in all data nodes.

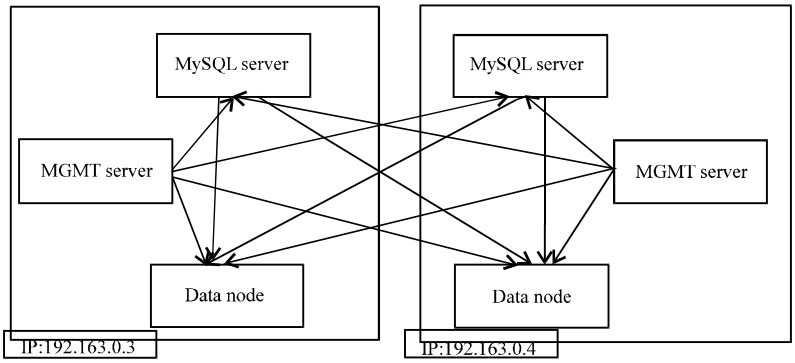

Fig. 5: Block diagram of distributed database creation in (a) 1st and (b) 2nd host

Table 1: Virtual machine types

Priority	Type	Virtual machines (VM's) categories
P1-VM1 type	Large	4 CPU , 8 GB RAM
P2-VM2 type	Medium	2 CPU, 4 GB RAM
P3-VM3 type	Small	2 CPU, 2 GB RAM

Load balancer and policy setter: Initially three VM's are configured with the priority as shown in Table 1.

Steps Involved in VM allocation:

Step 1: Identify the incoming request with IP address and Register number
Step 2: Analysis of IP address to recognize the starting 8-bit IP address block
Step 3: Save the starting 8-bit IP address block in database
Step 4: Initially the identified First 8-bit IP address block counter value is checked in each VM. If there is more than one VM which has the lowest count for this IP address block, the VM with the highest priority (VM1-P1 highest priority, VM3-P3 lowest priority already defined) is selected. Then the request will be pushed to that VM and also the counter is increased
Step 5: Once the request has been processed, then the count should be decremented in the database for the corresponding 8 bit IP address block in the corresponding VM

Based on the above priority based and round robin algorithm the requests for marks are scheduled and the marks retrieved from the data base. The performance of this algorithm is compared against a single server and also 3 VM's in a simple round robin method and it is proved that IP based priority and round robin method is efficient among the three methods.

RESULTS AND DISCUSSION

The implemented priority based round robin load balancing algorithm using cloud computing is now compared in Fig. 6 against a single server (without VM) and also requests processed in a round robin method with any priority. It is seen the average response time under normal conditions varies between 3.75 to 11.25 msec for a request. Assuming that three Internet Service Providers

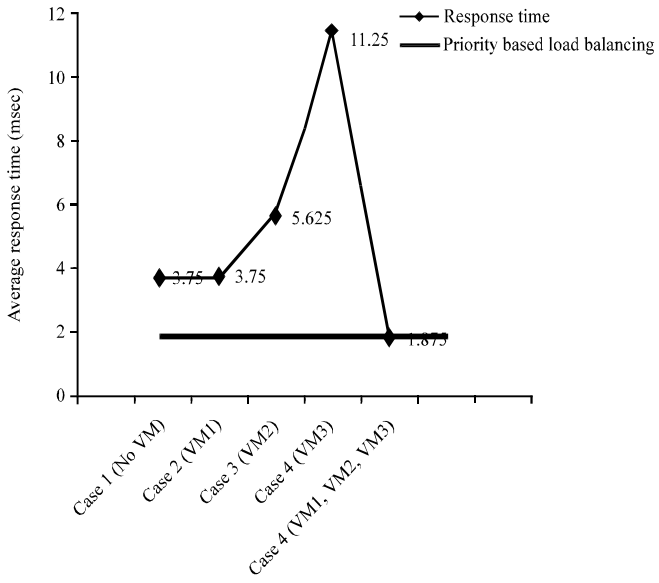

Fig. 6: Comparison of average response time-single server vs. 3 VM's (cloud)

are available, the average response time fluctuates between 3.75, 5.625 and 11.25 msec. It is seen that using the IP based priority based round robin method the average response time for a request is 1.875 msec.

CONCLUSION

From the results the priority based round robin load balancing algorithm has shown a minimum of fifty percent improvement when the number of queries increases and is seen more suitable for data intensive applications. Therefore it has been concluded that a priority based Round robin load balancing algorithm is more suitable for data intensive applications.

REFERENCES

Akhtar, Z., 2007. Genetic load and time prediction technique for dynamic load balancing in grid computing. Inform. Technol. J., 6: 978-986.

Al-Sakran, H., 2006. An agent-based architecture for developing e-learning system. Inform. Technol. J., 5: 121-127.

Baun, C. and M. Kunze, 2009. Building a private cloud with eucalyptus. Proceedings of 5th IEEE International Conference on E-Science Workshops, December 9-11, 2009, Oxford, UK., pp: 31-32.

Bunch, C., N. Chohan, C. Krintz, J. Chohan and Y. Nomura *et al.*, 2010. An evaluation of distributed datastores using the appscale cloud platform. Proceedings of 3rd International Conference on Cloud Computing, July 5-10, 2010, Miami, Florida, USA., pp: 305-312.

Chiang, R.C. and H.H. Huang, 2011. TRACON: Interference-aware scheduling for Data-intensive applications in virtualized environments. Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, November 12-18, 2011, IEEE Computer Society, pp: 1-12.

Duan, N.N., 1996. Distributed database access in a corporate environment using Java. Proceedings of 5th International World Wide Web Conference, May 6-10, 1996, Paris, France, pp: 1149-1156.

Fox, G.C., 2011. Data intensive applications on clouds. Proceedings of the 2nd International Workshop on Data Intensive Computing in the Clouds, November 14, 2011, Seattle Washington, pp: 1-2.

Hutchings, A., A. Morgan and G. Vanderkelenm, 2010. MySQL cluster tutorial. Proceedings of the O'Reilly MySQL Conference and Expo, April 12, 2010, Santa Clara, CA.

Kaitsa, M., I. Stavrakas, T. Kontogiannis, I. Daradimos, M. Panaousis and D. Triantis, 2009. Load balancing incoming IP requests across a farm of clustered MySQL servers. Proceedings of the International Conference on Computer as a Tool, September 9-12, 2009, Warsaw, pp: 546-550.

McLaughlin, D., S. Sardesai and P. Dasgupta, 1998. Preemptive scheduling for distributed systems. DARPA/AFRL-Rome, Intel Corporation and NSF.

Miceli, C., M. Miceli, B. Rodriguez-Millai and S. Jha, 2010. Understanding performance of distributed data-intensive applications. Phil. Trans. R. Soc. A, 368: 4089-4102.

Neary, M.O. and P. Cappello, 2003. Advanced eager scheduling for Java-based adaptive parallel computing. Concurrency Computat.: Pract. Exp., 1: 1-2.

Nurmi, D., R. Wolski, C. Grzegorczyk, C. Obertelli, S. Soman, L. Youseff and D. Zagorodnov, 2009. The eucalyptus open-source cloud-computing system. Proceedings of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, May 18-21, 2009, Shanghai, China, pp: 124-131.

Pukdesree, S., A. Sukstrienwong and V. Lacharoj, 2006. Evaluating of distributed database on PC cluster computers. Proceedings of 10th WSEAS International conference on Computers, July 13-15, 2006, Vouliagmeni, Athens, Greece, pp: 1322-1326.

Serpanos, D.N. and P.I. Antoniadis, 2000. FIRM: A class of distributed scheduling algorithms for high-speed atm switches with multiple input queues. Proc. IEEE INFOCOM Annu. Joint Conf. IEEE Comput. Communi. Soc., 2: 548-555.

Soundararajan, G., C. Amza and A. Goel, 2006. Database replication policies for dynamic content applications. Proceedings of the 1st ACM SIGOPS/EuroSys European Conference on Computer Systems, April 18-21, 2006, ACM New York, pp: 89-102.

Uddin, M., A.A. Rahman, A. Shah and J. Memon, 2012. Virtualization implementation approach for Data centres to maximize performance. Asian J. Sci. Res., 5: 45-57.