Adaptive Load Balancing Policy for Web Server Custer

Ajay Tiwari
School of Computer Science
Devi Ahilya University (DAVV)
Indore, India
e-mail: tiwariajay8@yahoo.com

Abstract—The paper highlights the significance of open source software for distributed computing environment and proposes an adaptive load balancing model using open source software for distributed computing environment. The load balancing strategies used by the model are based on load of the system. The proposed algorithm uses current load, response ratio and processor utilization of the nodes of the web server cluster to evaluate the performance.

Keywords*- Open Source Software; Distributed Computing; Dynamic Load Balancing;Response Ratio; Scheduling*

*****

I. INTRODUCTION

Research work in the field of software technology shows that the preferred software development method always been the one that seemed to work best within the contemporary technological and economic constraints, particularly the costs of computer ownership, programming personnel and data communications [9]. The idea of Open Source Software came with the same philosophy in the early days of computing. In 1950s and for many years later, computer manufacturers released their software free along with the hardware. The supplied software was in both source code and object code form. The accompanied software was used as a marketing tool for the hardware. To run specific applications, computer users wrote their own or hired computer programmers to write application software.

The software world observed a transformation in 1964, with the launch of IBM System 360. It was a standard computer platform which expanded computer population especially medium size business organizations. Most of the new computer owners did not have the resources to hire computer programmers and therefore an application software vacuum was created which was filled by software companies. These companies wrote special purpose software viz. insurance, railway reservation system etc. and generic software viz. payroll, inventory management etc. These software were quite expensive as the software development was done by dedicated and expensive team of programmers. Initially companies supplied their software in both source and object code form which was used for customization at the time of need but later, due to competition, companies stopped disclosing the source code.

Another boom was observed in the software world in late 1970s, with the advent of personal computers owing low cost of computers, the computer population soared and the number of companies providing software solution for these computers increased exponentially. For the new PC environment, software companies invested huge amount and was no longer feasible for them to disclose their source code in this competitive era. In the early 1990s, popularity of Internet changed the work culture of the computer professionals. Now it was possible for them to work in collaborative manner. This gave the birth to today's open source community. Linux was the most popular open source product of the community. Open source products were soon available in almost all the established software categories [9].

The work done in the paper is the extension work of [2]. The rest of the paper is organized as follows. Section II describes the revolution in the field of Open Source Software. The proposed architecture and frame work is presented in Section III. The simulation results are discussed in Section IV.

II. OPEN SOURCE SOFTWARE REVOLUTION

Open Source Software (OSS)/Free Software refer to the software that are not copyrighted. Software are free and can be used without any restriction. In OSS/free software, users can run, copy, modify and distribute (copy of the original or modified version) the software.

Open source software, together with clusters and grids, offered a cost effective environment for web-enabled applications. The cost effective hardware and software solutions for implementing web technology increases the web solutions which results processing load on the intranet and Internet. Load balancing can be effectively used to balance this workload on cluster, server farm, grids etc. [10][3].

Load balancing is a technique to distribute workload over two or more resources in order to achieve increased performance and is achieved by using load balancers. Load balancer can be a hardware or software and follow some policies to assign the requests to the resources. Typically, these can be random, round robin, resource load based, resource computing power based and least connection based. Security is one of the additional features of load balancers as they hide the network and resources behind the scene that are performing the real task [7][14].

Linux Virtual Server (LVS) is an open source Linux load scheduling and balancing software which is used to build a highly available cluster of nodes with high-performance. And provides services like web service, mail service, ftp service, VoIP service etc. [6]. Apart from LVS, Red Hat Cluster Suite is an easy to use cluster software implementation from Linux leader Red Hat. MySQL has introduced dynamic load balancing capability to its database platform and enterprise subscription since 2008. It easily handles multi-core processors, large memory systems and proxy servers that serve as a middleware layer between the client and database server. It also prioritizes queries and workloads, based on user input. Xiao et al have suggested a variety of open source software to support HTTP/S, Mail (POP3, IMAP, SMTP), JMS, TCP, UDP, VFS, SMS, XMPP, FIX, database, proxy and operating systems etc [5].

The SimGrid and SimGrid v2 are among the most popular open source grid simulation tools available to the researchers [4]. SimGrid provides set of abstractions and functionalities to build simulator for the custom applications. Its network topology is fully configurable and the users have to define topology to suit their needs. The resources in SimGrid are modeled using two performance characteristics latency and service rate. It is a discrete event simulator developed on top of the "SimJava" API and supports the economy based scheduling algorithms. It supports simulation of computational Grid as well as the data Grid [15]. Various extensions to GridSim have been released over the period of time. These extensions include concept of advance resource reservation, failure detection, improved network structure & topology and buffer management across the network [13][11].

III. Proposed Load Balanceer Model

This section explains the structure of proposed dynamic load balancer model. The Web Server Cluster (WSC) in consideration consists of a load balancer and number of replicated servers. Load balancer receives the client requests and distributes them to most suitable server of the cluster. The proposed load balancer comprises of Internal Proxy, Load Database (LDB) module, Load Controller (LC) module and Decision Maker (DM) module as shown in Figure 1.1

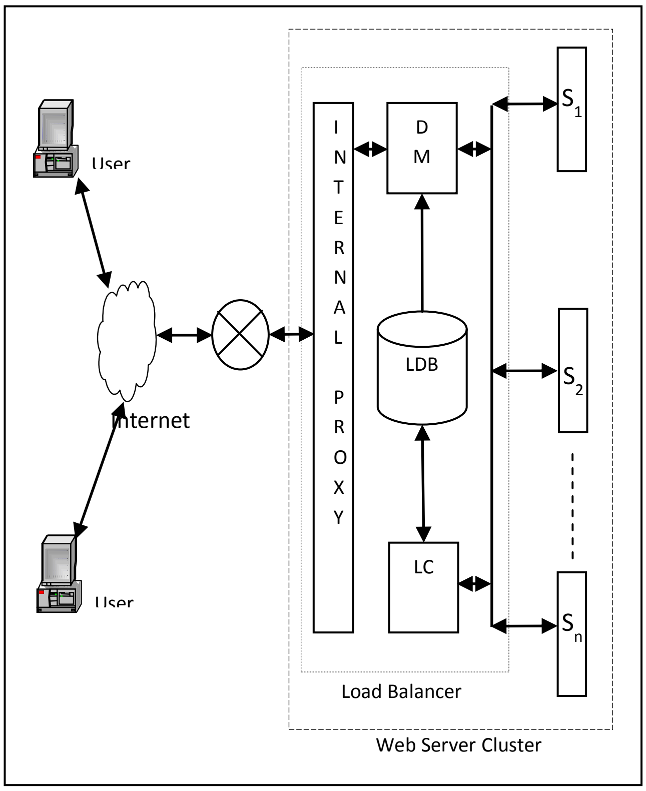

Figure 1.1 Proposed Load Balancer Model

All the client requests are received and responded by Internal Proxy of load balancer where it performs compression and caching to guarantee fast response time for most frequent requests. As Internal Proxy is only entry point to the web cluster, it can also be used as firewall wherever required. LDB stores the current load of the nodes and also stores the value of response ratio and processor utilization of the nodes to evaluate the performance of the WSC. Load of the nodes are determined with the help of heterogeneity factor and queue length of the nodes. As heterogeneity factor of nodes is constant, LDB keeps their value until the nodes are alive. Therefore, only queue length of the nodes is updated in the LDB for load calculation of the nodes of the WSC. LDB is updated by LC as well as nodes of the WSC. LC collects the queue length parameter of the node periodically and updates LDB whereas nodes of the WSC uses state change driven policy. According to the policy, a node updates its queue length value at LDB whenever it changes from one level to another. The levels are lightly loaded, moderate loaded and heavily loaded. DM only refers LDB for optimal forwarding of the client requests. Nodes of the WSC also refer LDB to migrate processes using sender initiated or receiver initiated policies.

Two types of communication between load balancer and nodes of the WSC are possible. One is request-response data and the other is control information. Request-response is done between DM and nodes whereas control information communication takes place between LC and nodes. Algorithm does not consider communication overheads as these are negligible.

Addition and removal of nodes (scaling) in the WSC is handled by LC. If the overall load of WSC increases up to some predefined level, LC selects a node from available nodes and makes it available to WSC by updating LDB. Similarly, it removes some resources during off time when WSC load is less than a predefined limit by removing entries from LDB. Health checking of the nodes of the WSC is done during routine collection of values of their parameters. If LC does not listen a node for a predetermined time interval, it sends a message to the node to check whether it is alive or went down and updates its table accordingly.

A. *Infomal Discription of the Proposed Model*

The assumptions of the algorithm are as follows:

- The scheduler has perfect information while making scheduling decision.
- The scheduling as well as communication overheads are negligible.
- The incoming requests are independent and can be executed at any time and in any order.

WSC comprises of n replicated nodes, each serving its queue and interconnected by high-speed network with negligible communication delay. The nodes are of heterogeneous nature in term of processor speed, RAM, cache memory and front side bus. The system is simulated for the setup of n=10 nodes of varying hardware profile.

Almost all the load balancing algorithms use some load indices to measure the load of the nodes. Most of the researchers have considered queue length of the node as load indicator [1][8]. The proposed algorithm considers heterogeneity factor along with queue length of the nodes for lode calculation. For lightly loaded, moderately loaded and heavily loaded WSC, the scheduler uses different node selection policies which are as follows:

a) For lightly loaded WSC, scheduling algorithm uses random policy to distribute requests in which a node is selected randomly with each node having equal probability.
b) In moderately and heavy loaded WSC system, scheduling algorithm uses least loaded policy in which a node is selected with minimum load.
c) A lightly loaded node uses receiver initiated (pull) policy to pull the load from the relatively heavily loaded nodes of WSC.
d) A heavily loaded node uses sender initiated (push) policy to push the load to the relatively less loaded nodes.

Load Li of ith node in WSC can be computed as:

$$L_i = HF_i * \sum_{j=1}^{np_i} t_j$$

where, npi is the number of processes on node i, tj is the remaining service time of process j and HFi is the heterogeneity factor of node i and can be computed as:

$$HF_i = \frac{PC_i}{\sum_{j=1}^{n} PC_j}$$

where, PCi is the processing capability of ith node and calculated on the basis of the nodes processing speed, cache and RAM.

To test the performance of WSC, the response ratio and processor utilization indices are being used by the algorithm.

Response ratio R of a process is calculated as follows:

$$R = t / (t+w) \qquad 0 < R \le 1$$

where, t is service time of process and w is the missed time.

Mean response Rmean time of WSC having n nodes is calculated as:

$$R_{mean} = \frac{\sum_{i=1}^{n} R_i}{n} \qquad 0 < Ri \le 1$$

where, Ri is the response ratio of node i and calculated as:

$$R_i = \frac{\sum_{j=1}^{np_i} \left({t_j}/{(t_j + w_j)} \right)}{np_i}$$

where, tj and wj are the service time and missed time of process j on node i and npi is the number of processes at node i.

Similarly, mean processor utilization Umean of WSC having n nodes is calculated as:

$$U_{mean} = \frac{\sum_{i=1}^{n} U_i}{n}$$

where, Ui is the processor utilization of ith node of WSC.

B. Foraml Discription of the Proposed Model

Algorithm shown in Table 1.1 starts with initialization of parameters of LDB and nodes of the WSC and describes the functioning of load balancer. Similarly the algorithm shown in Table 1.2, describes how load balancer handles the client requests.

TABLE 1.1 LOAD UPDATING AND BALANCING ALGORITHM

```
1) Parameters and LDB are initialized.
   Thread I  // Updation of Load Data Base(LDB) by Load Controller (LC)
2) Step ( i) and (ii) are repeated infinitely
    i) LC collects the value of parameters periodically.
    ii) LC calculates the load of the nodes and updates the LDB.
   Thread II  // Updation of LDB by Nodes of WSC
3) LDB is updated by nodes of the WSC whenever their parameters value changes from one level to other.
Thread III  // Nodes Perform Process Migration
4) Step ( i) and (ii) are repeated infinitely
    i) A lightly loaded node selects most heavily loaded node from the LDB and sends a request (pull policy) for process migration.
    ii) A heavily loaded node selects least lightly node from the LDB and sends a request (push policy) for process migration.
   Thread IV  // Health Monitoring by LC
5) Step ( i) and (ii) are repeated infinitely
    i) LC calls removeResource function periodically to remove access nodes entry from LDB or the nodes which goes down abruptly.
    ii) LC calls addResource function whenever additional resources are needed.
```

TABLE 1.2 REQUEST AND RESPONSE FORWARDING ALGORITHM

```
Thread //Request forwarding Through DM Module
      Step ( i) to step (vi) will be repeated infinitely
    a) Load balancer waits for the client requests
    b) After arrival of requests, if the requested object is in the internal proxy cache, response is sent back.
    c) Else request is forwarded to DM.
    d) DM refers the LDB and request is redirected to the least loaded node.
    e) Response is sent back to the internal proxy via DM.
    f) Internal proxy caches the result for TTL time and responds to the client.
```

IV. SIMULATION AND RESULT ANALYSIS

This section discusses the simulation results and result analysis.

A. Simulation

To simulate the proposed model real network of six nodes have been used [12]. Out of six nodes, one node is dedicated as load balancer with Red Hat Cluster Suite and rest of the machines are used as the nodes of the cluster. As the nodes of the cluster are replicated, Apache tomcat and My-SQL are installed in rest of the five nodes of the cluster. The client programs also run on the load balancer machine. Internal Proxy, Decision Maker (DM), Load Data Base (LDB) and Load Controller (LC) modules are written using multithreaded approach of Java programming and perform their respective tasks as discussed in Section III, Table 1.1 and Table 1.2 using the formula of Section III (A).

A Java function generates artificial requests (workload) where arrival follows Poisson distribution and service follows exponential distribution. Artificial workloads have a greater flexibility as compared to real workloads and are easier to reproduce. The nodes of the cluster are of different processing capability and their heterogeneity factor is calculated by using formula of Section III (A) and stored for load calculation of the nodes. The proposed model is compared with random, round robin and weighted round robin algorithms. The same set of input request is simulated for random, round robin, weighted round robin and proposed algorithm and mean response time and mean processor utilization parameters are calculated for all five nodes using the formula of Section III (A).

TABLE 1.3 MEAN RESPONSE TIME OF THE SERVERS FOR DLB POLICIES

Node ID	Random	RR	WRR	Proposed
Node 1	377.24	313.62	287.37	265.26
Node 2	372.12	298.89	319.22	294.18
Node 3	289.72	365.47	279.98	289.74
Node 4	380.23	335.56	299.31	269.83
Node 5	263.46	301.19	331.23	309.67

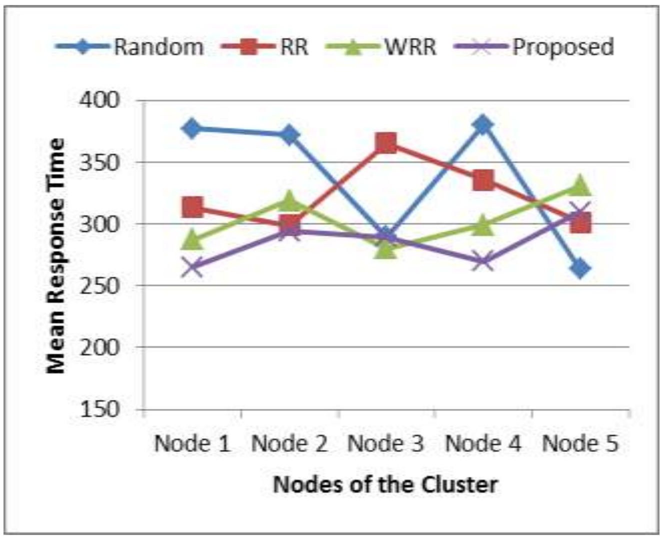

Figure1.2 Comparison of Mean Response Time of the Servers

B. Simulation Result and Result Analysis

Table 1.3 and Table 1.4 shows the mean response time and mean server utilization for nodes of the cluster using random, round robin, weighted round robin and proposed algorithm and the comparison is depicted in Figure 1.2 and Figure 1.3. For each algorithm, mean response time and mean server utilization is computed for a predetermined set of inputs. Table 1.3 and Figure 1.2 shows that mean response time ranges from 263.46 to 458.24, 298.89 to 365.47, 279.98 to 331.23 and 265.26 to 309.67 for random, round robin, weighted round robin and proposed algorithm respectively and is dispersed around 79.67, 27.86, 21.49 and 18.25 about the mean. Similarly, as shown in Table 1.4 and Figure 1.3, the mean server utilization improves as we move from random to proposed algorithm. Although upper bound is approximately same for all the algorithms, lower bound varies from 42% to 75% as we move from random to proposed algorithm which shows a significant difference in the mean server utilization. Figure 1.2 and Figure 1.3 show the smoothness of mean response time and mean server utilization and one can easily observe that the smoothness improves as we move from random to proposed algorithm. Whereas the performance of random is highly zigzag, the results obtained by proposed algorithm are more consistent.

TABLE 1.4 MEAN SERVER UTILIZATION FOR VARIOUS DLB POLICIES

Node ID	Random	RR	WRR	Proposed
Node 1	76	91	68	81
Node 2	49	65	93	93
Node 3	93	84	70	79
Node 4	56	49	81	75
Node 5	42	84	58	83

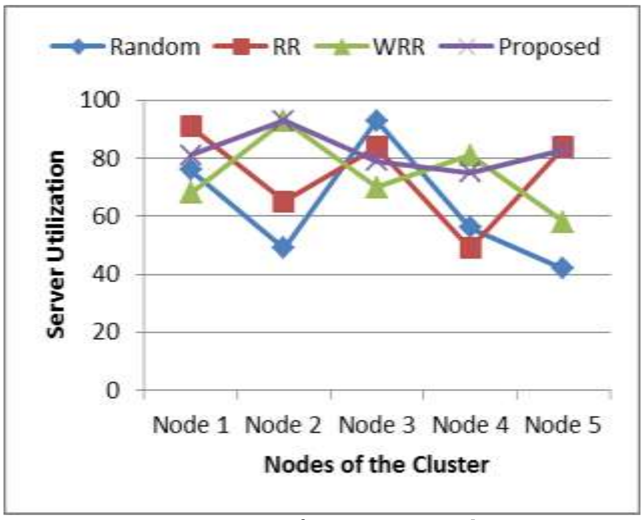

Figure1.3 Comparison of Mean Server Utilization

V. SUMMARY

Open source software are necessary for the rapid development of IT based applications particular in the developing countries to draw the benefits of the technology for the common men. The popularity of distributed computing over past decades has posed a challenge to the researchers and software developers. The use of dynamic load balancing should be the essential feature of the distributed system software to have better performance and improved response time. Dynamic load balancing is one of the critical scheduling problems in DCE on a cluster of replicated servers which faces a constant pressure of increased network traffic and diverse load levels. The problem is aggravated with the growing complexity of web based applications and services.

The chapter investigated various open source software used in the area of distributed computing and presents a load balancing model using these software. The proposed model is compared with random, round robin and weighted round robin. Mean response time and mean server utilization parameters are calculated for performance comparison. Simulation results show that the performance of the proposed model is better than random, round robin and weighted round robin scheduling algorithms. The proposed load balancing model can be useful for incorporating load balancing features in existing and new open source software.

Acknowledgment

I would like to give my heartfelt thanks to my Ph D supervisor Dr. Priyesh Kanungo for taking on multitude of roles that provided guidance and direction for this research. I would like to express deep sense of gratitude for his truly insightful thoughts that envisioned the topic of the paper.

References

[1] A Karimi, Z Faraneh , A Jantan and A R Ramli, "A New Fuzzy App.roach for Dynamic Load Balancing Algorithm," International Journal of Computer Science and Information Security (IJCSIS), Vol. 6, No 1, 2009, pp. 1-5.

[2] A. Tiwari, P. Kanungo, "A Model for Dynamic Load Balancing in Open Source Software for Distributed Computing Environment," Proceeding of CONSEG2012: 6th International Conference on Software Engineering IEEE, Indore, Sept 2012, pp. 1-5.

[3] *Chin* Lu and S Lau, *An* adaptive load balancing algorithm *for* heterogeneous *distributed systems with multiple task classes*, 16th IEEE International Conference on Distributed Computing Systems (ICDCS '96), 1996.

[4] H. Mehta, P. Kanungo and M. Chandwani, Performance Enhancement of Scheduling Algorithms in Clusters and Grids using Improved Dynamic Load Balancing Techniques, WWW 2011, March, 2011, Hyderabad, India. ACM 978-1-4503-0637-9/11/03

[5] HL. Xiao, Y. Zhu, L.M. Ni, Z. Xu, "Incentive-based scheduling for market-like computational grids," IEEE Transactions on Parallel and Distributed Systems, Vol. 19, No. 7, Jul 2008, pp. 903-913

[6] J Aweya, M Ouellette, D, Montuno, B. Doray and K Felske, An adaptive load balancing scheme for web servers, International Journal of Network Management, pp 3-39, 2002.

[7] L Cherkasova, M DeSouza and Shankar Ponnekanti, Performance Analysis of Content-Aware Load Balancing Strategy FLEX: Two Case Studies, Annual Hawaii International Conference on System Sciences, 2001.

[8] L Wenzheng, S Hongyan and N Algorithm, "Load Balancing in Cluster Systems," Proceedings of the 2010 14th International Conference on Computer Supp.orted Cooperative Work in Design, IEEE, 2010 .

[9] M K Campbell, "Historical Reflections: Will the Future of Software be Open Source?" Communications of the ACM, Vol. 51, No. 10, Oct 2008, pp. 21-23.

[10] M Colajanni, P S Yu and V Cardellini, Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers, 18th IEEE International Conference on Distributed Computing Systems (ICDCS'98), pp. 295-302, 1998.

[11] P Werstein, H Situ and Z Huang, Load Balancing in a Cluster Computer, Seventh International Conference on Parallel and Distributed Computing, Applications and Technology, (PDCAT'06) IEEE, 2006.

[12] R Sharma, "Dynamic Load Balancing in Network of Workstations of Different Computing Powers," Ph D Thesis, Department of Computer Engineering, IET, Devi Ahilya University, Indore, MP, India, Jul 2013.

[13] V Cardellini, M Colajanni and P. Yu, Dynamic Load Balancing on Web-Server Systems, Internet Computing Volume 3, Issue 3, IEEE, ISSN: 1089-7801, pp 28 – 39, May 1999.

[14] W. Yang, S. Li and D. Cheng, A Load Balancing Strategy in Web Cluster System, Third International Conference on Natural Computation, IEEE, 2007.

[15] Z Lin, L Ping and S Yuan, A Content-based Dynamic Load-Balancing Algorithm for Heterogeneous Web Server Cluster, Journal of Intelligent Computing and Applications (JICA), ISSN: 0974–410X, pp 21-29, Jan– June 2009.