VOL. 10, NO. 10, JUNE 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

EFFICIENT SERVER LOAD BALANCING THROUGH IMPROVED SERVER HEALTH REPORT

Saifullah M. A. and M. A. Maluk Mohammed
Department of Computer Science Engineering, M A M College of Engineering, Tiruchirappalli, India
E-Mail: ssg_saif@mamce.org

ABSTRACT

To achieve scalability and high availability of the service offered by web server clusters, an efficient server load balancing policy is required. A critical part of the load balancing policy is to find the best available server to assign the load. For that, server load needs to be calculated. In this paper, the parameters required to assess the load of the server are explored. An important load parameter, 'number of open file descriptors' is identified to find the load on a server along with existing load parameters, CPU cycles and free memory. The server load reporting is improved by extending SNMP agent to report server resources including 'number of open file descriptors'. Performance metrics used in test scenarios are: Throughput, HTTP Response Time and Error rate and Normalized Throughput. Tests were done in two different scenarios: normal condition scenario and the other scenario with high load on web servers. The load balancing results of the server cluster by comparing our implementation with known load balancing algorithm used on web-clusters, Round Robin (RR) and state full algorithm Least Connections (LC) are described. Our experimental results show that the previously mentioned algorithms can be outperformed by our proposed adaptive mechanism, Scalable Load Balancing (SLBL) algorithm. Our experimental results show that the performance of the cluster of web servers is significantly improved by the proposed adaptive algorithm SLBL over the existing algorithms, RR and LC. The average service request rate that can be serviced by the SLBL algorithm is around 1.27 times more than that of LC and around 1.93 times more than that of RR.

Keywords: dynamic load balancing, internet computing, web cluster.

1. INTRODUCTION

Web-based services are continuously growing due to their demand and Web servers are getting lot of service requests than ever as the Web has become the standard, simple and default interface to access the services of data centres, information systems, e-commerce sites and application service providers. The performance challenges of Web-based architecture have increased due to the rapid growth of diversified client devices, the increased complexity of middleware application software, and the need of client authentication and system security, and the need to ensure high availability of services of web based systems in a cost-effective way. Despite the fact that both the server and network capacity have raised a lot, and better architectural solutions are being applied, the challenge of short & acceptable response time continues to question the research of web systems and clusters [1], [2], [13]. Because when a server is overloaded, the response times acquired by the customer grow and this can result in losing the revenue of sale operation if we refer to a commerce site [13]. Thus, still there are some challenges to be resolved from the customer's viewpoint in terms of acceptable response time.

Server load balancing can solve these challenges as it makes multiple servers take part in the same service and share the same work, since the capacity of a single server is limited. It gives crucial benefits such as availability, scalability, security and manageability of Web systems. One of the popular types of load balancing is cluster-based servers also known as server farms. Service or content is replicated on multiple servers in this option to achieve load balancing benefits, but this requires robust load balancing strategy. A server load balancing strategy consists of distributing or assigning the processes or tasks of a parallel application across the available servers. An efficient load balancing strategy avoids the condition where some servers are busy with multiple jobs queued up while others are idle [11], [12], [15], [16], [17], [57]. Hence, the process of task distribution is more complex as it requires selecting the right or best server among available servers. One of the difficult problems of target server selection is when to decide that a server is overloaded.

Usual server load measurements or parameters such as CPU load, free memory are indicative of server load to some extent only as these parameters do not indicate the resources of a server such as available file descriptors (fd). Even if the system has enough free memory and enough free CPU cycles but without enough free file descriptors, service cannot be offered. This is one of the likely situations for the increased response time when the server is congested or overloaded. So 'number of open file descriptors' is an important load parameter to be monitored on the server. The existing load balancing algorithms have not considered this parameter as explained in the next section on Related Work. Our proposed algorithm includes 'number of open file descriptors' parameter along with CPU load and free memory parameters.

The organization of rest of the paper is as follows: Section II gives the details of related work. Section III gives the proposed architecture of cluster based load balancing system. Section IV gives load balancing algorithms for cluster based web servers. Section V describes proposed SLBL algorithm. Section VI describes the extensions made to Net-SNMP agent. Section VII explains enhancements done to HAProxy load balancer. Section VIII discusses our experimental test bed setup, experiments carried out and performance evaluation of load balancing algorithms. Section IX gives conclusions and future work.

2. LITERATURE REVIEW

The categorization purposed by Cardellini *et al*. in [12] is based on the place where the distribution logic is applied when forwarding a request to the chosen server of a web server system which is locally distributed: at the network and at the Web system [29], at the client, at the Domain Name System (DNS) [28], [38], [39], [42], [43], [44]. In this paper, we consider the first option as it is the most popular, dispatcher-based clusters or cluster based web server system. In cluster based load balancing system, the incoming requests are received by a centralized distributing entity of the Web system and it forwards them to the servers of the cluster. The infrastructure of Web system is the sole component that is controlled by the content provider directly in the cluster based web server system. Other devices that are part of the network such as backbones, DNS systems, and routers cannot be controlled by a individual organization. Web server-based cluster architecture [1], [2], [13], [25], [26], [40], [41], [57] is made up of a collection of web servers that are interconnected by a high speed network and locally distributed.

We focus on finding out the load on servers and selection logic of the load balancing algorithm. The existing load balancing algorithms have not considered the proposed important parameter [9], [10], [19], [20], [21], [22], [23], [24]. We describe in detail the mechanisms and the parameters used for the estimation of load on web servers. We discuss the performance evaluation of our algorithm towards the efficiency of cluster based load balancing system.

3. PROPOSED ARCHITECTURE OF CLUSTER LOAD BALANCING SYSTEM

Figure-1 shows the architectural structure of the proposed cluster based web system. The proposed architecture of cluster offers only one interface to the clients; therefore, it is seen as a single device. The customers are not cognizant of the IP addresses and names of the web servers that makes the architecture of cluster.

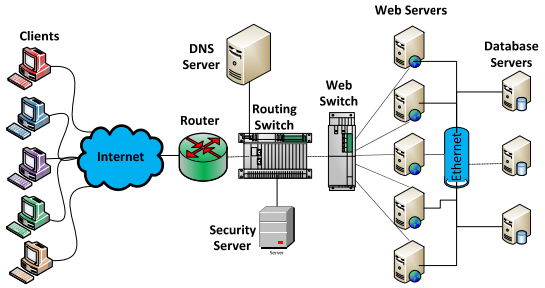

Figure-1. Proposed Architecture of cluster load balancing system.

The clients access the applications hosted in the cluster system by sending their requests to the virtual IP address that corresponds to the centralized node that acts as the user interface or front-end of the cluster architecture. The front end node of such a system is known as the web switch. The web switch forwards all the in-bound packets received from the clients to the web server devices according to the selected load balancing algorithm. The critical component of the web switch or distribution entity is the load balancing algorithm, that selects the best suited target web servers to respond to client requests [2], [18], [27].

4. EXISTING ALGORITHMS OF LOAD BALANCING FOR CLUSTER BASED WEB SYSTEMS

The routing process of cluster based web system is coordinated by the load balancing algorithms of the distribution entity. We categorize load balancing algorithms employed at a non content aware distribution node into four different classes: 1. Non-adaptive and stateless, 2. Non-adaptive and stateful, 3. Adaptive and stateless and 4. Adaptive and stateful. We call the algorithms which consider or not of client connection requests accordingly as stateful or stateless algorithms. We call the algorithms which consider or not the feedback of web server status measurements and adjust their behaviour depending on the status measurement transitions respectively as adaptive or non-adaptive algorithms.

Non-adaptive and stateless algorithms do not keep track of any type of state information of web server system. Examples of these kinds of algorithms are Round Robin and Random algorithms. In spite of the fact that the Round-Robin DNS scheduler [45] is based on a similar principle, per host DNS caching usually leads the cluster system to the definite state of imbalanced load. In opposition, Round-Robin implementation into virtual web server [46] is much better to DNS Round Robin because of its each connection scheduling. Round-Robin and Random algorithms can be extended easily to satisfy web systems of diversified sizes [2], [47]. For example if Si is an indication of the server capacity, a relative server capacity can be defined as:

$$R_i = \frac{S_i}{\max\{S_1,..S_2,..S_k\}}, 0 \le R_i \le 1$$

As far as Round-Robin policy is concerned, heterogeneous capacities can get different probabilities. As for Random algorithm, the relative server capacity can be compared with a random generated number p, where $0 \le p \le 1$ in order to circulate to another server.

Non-adaptive and stateful algorithms keep an eye on client connections at the centralized distribution entity. Examples for this category are Weighted Least Connections (WLC, Least Connections (LC), algorithms [48], [49]. Furthermore, Never Queue Scheduling algorithms and Shortest Expected Delay (SED) use identical approach to Least Connections algorithm and the web server with the shortest expected delay gets the client connections [50].

Adaptive and stateless algorithms consider server state conditions but do not observe connection state information. These algorithms get information from monitor agent processes running on either the centralized distribution entity or the web servers. The server status information received from monitor agent is processed in order to decide metric values and therefore weights used for balancing decisions. Some regularly used parameters are: "memory usage", "CPU load", "ICMP request-reply" time and "disk usage". In some instances, web servers status parameter values are generally saved by network monitor services like SNMP that run on the web switch. An example of this policy known as Central Load Balancing for Virtual Machines (CLBVM) is introduced in [51]. In other instances, SNMP manager run on web switch queries status parameter values of web servers that are maintained by SNMP agents on the web servers [52], [57]. There are different prediction algorithms used for load balancing, examples of them are [38], [39], [53], [54].

Adaptive and stateful algorithms take into account server state conditions and also keep track of client requests [49], [55]. Stateful adaptive implementations include MALD (Mobile Agent based Load balancing); it implements scalable load balancing on distributed web servers using mobile agent's technology [56].

The predictive probabilistic load balancing policy (PPLB) that uses queuing model to predict the usage of each server and uses adaptive weights depending on a utility function that follows the deviation and difference in forecasted average and measured response time of web server [39]. Simulated Annealing Load Spreading Algorithm (SALSA), employs an energy function that places each web server based on the following parameters: processing capability of web server, arrival request rate, request processing rate, and the average waiting time of each request in the queue of web server [58], [59].

Though Adaptive Load Balancing Mechanism (ALBM) [60] obtains performance information by monitoring the applications running on the nodes, monitoring of few critical applications and critical resources is missing, which are proposed in this work.

5. SLBL ALGORITHM

In this work, a stateful adaptive load balancing algorithm, called Scalable Load BaLancing (SLBL) algorithm is designed and implemented. SLBL estimates web server's load in order to make balancing decisions. The metrics used to dynamically adjust web server load are: CPU load average, free memory and number of open file descriptors. Metric calculation is performed by agents that run at the web server. Based on these metric values collected from agents running on web servers, server load estimation occurs at the web switch. This whole process is repeated periodically. Initial web server load values along with server configuration in terms of metrics that is CPU speed, total RAM and total number of file descriptors and also polling period with weight age to each metric are given by the decision maker, as an initial estimation of the balancing point of system. The possible weight age of each parameter is given in Table-1.

Table-1. Load parameters.

CPU load	Free memory	No. of open file descriptors
0	1	0
1	0	0
1/3	1/3	1/3
1/2	0	1/2
W1	W2	W3

First row of the above table is for the administrator who is interested only in CPU load. Second row is for the administrator who is interested only in free memory. Third row is for the administrator who is interested only in number of open file descriptors. Fourth row shows equal weight age for each parameter. And the last row shows any weight w1 for first parameter and w2 for second parameter and w3 for third parameter can be given with the condition that w1 + w2 + w3 = 1. So the administrator has the choice to give his own weight to each metric. Web server health probe (process) periodically runs on the web switch to collect the values of metrics from agents at each web server and the values are updated at web switch to calculate the adaptive server load of each web server for that probing period.

Along with the monitoring of important server resources, it is also necessary to monitor the status of critical services of the servers. SLBL algorithm monitors the status of the critical applications: web service, file service and DNS service including TCP and UDP services.

6. RESOURCE MONITORING ENHANCEMENTS TO SNMP AGENT

SNMP (Simple Network Management Protocol) is one of the most popular protocols for monitoring and controlling the health and welfare of computer equipment, network equipment, and servers. We used SNMP mechanism for its simplicity and wide deployment.

Net-SNMP is an open source suite of applications used to implement all the major versions of SNMP, SNMP v1 [30], SNMP v2c [31] and SNMP v3 [32]. It supports both the versions of IP protocol, IPv4 and IPv6. Net-SNMP has implementation for lot of MIB (Management Information Base) [3] information modules. Furthermore, Net-SNMP is highly extensible for proprietary or experimental uses too. Net-SNMP's implementation is available for Linux and Unix-like operating systems and also for Microsoft Windows. The extension of Net-SNMP agent requires both MIB extension and code extension.

A new function percentage_of_open_fds() is written to find out the number of open file descriptors on a system using /proc/sys/fs/file-nr . Net-SNMP agent needs to get the number of open file descriptors and return this number to HAProxy whenever it polls.

Pseudo code of the function percentage_of_ open_fds() is given below:

Function percentage_of_open_fds()

Step-1. *Open file /proc/sys/fs/file-nr for reading.*

Step-2. *On successful opening of the above file, read its contents into a string variable of length 1035.*

Step-3. *Tokenize the string obtained from /proc/sys/fs/file-nr into words[0], words[1] and words[2].*

Step-4. *Assign the values as following:*

Step-5. *numOpenFDs = atol(words[0]);*

Step-6. *unUsedFDs = atol(words[1]);*

Step-7. *maxFDs = atol(words[2]);*

Step-8. *Calculate FDRatio = numOpenFDs / maxFDs * 100;*

Step-9. *Round of FDratio as SNMP does not support float values.*

Step-10. *Return usedFDratio rounded integer obtained from above step as percentage.*

End of percentage_of_open_fds() function

7. HAPROXY LOAD BALANCER WITH ENHANCED SERVER RESOURCE POLLING

In this section the details about the HAProxy code that was modified for the load balancing implementation is described. A new option 'resourcechk' is added for resource monitoring of the server namely 'open file descriptors' on the servers under proxies section as shown below:

```
global
maxconn 4096 # Total Max Connections.
defaults
listen http_proxy 192.168.78.239:8088
balance roundrobin # Load Balancing algorithm
option resourcechk
## Define the servers to balance
server server1 192.168.78.240:80 weight 1 maxconn 512
```

Configuration parsing module and server health checking module are extended to poll the open file descriptors on web servers and set the status of the server accordingly. If the server has crossed soft limit of open file descriptors, an error will be set on its status. Otherwise, it will be available for load balancing. 'Number of open file descriptors' is collected by hr NumOf Open File Descr OID.

8. PERFORMANCE EVALUATION

We describe in this section, the test bed setup used for evaluating the performance of the proposed load balancing system. In our experimental scenarios, we used a web cluster with web servers of equivalent processing power connected to a web switch by LAN, as depicted in Figure-1. The web servers use Apache software, version 2.2 and Net-SNMP version 5.6.1.1 is used in our experiments. We performed our cluster tests with a set of Perl scripts and httperf tool [14]. The configuration of web servers and web-switch is Pentium 4 computer with a CPU that operates at 2GHz and 1GB memory. The configuration of web clients is Pentium 4 computer with a CPU that operates at 2GHz and 512MB memory. The operating system that runs on the web-switch is a Linux [5] operating system. Communication between web server and server health polling process of web-switch is done through SNMP.

The load balancer was tested with a set of five web servers. Initial test was to find out the suitable polling interval for our proposed SLBL algorithm to get the values of server health parameters. The HTTP client requests retrieve 10Kb object from the cluster of web servers. This test is performed at a rate of 50 HTTP requests per second. A test program to occupy the file descriptors was run on each web server to simulate the case of file descriptor exhaustion.

As described above initial tests were run to measure the performance of the load balancer based on the polling interval with which we update the status information of web server. And also in this experiment fds load on the servers was toggled to find out the right polling interval. The polling interval for updating server load information was increased from 100ms to 500ms, 1second, 5seconds and finally 10 seconds. Due to SNMP packet round trip time, lower intervals than 100ms are not preferred. Bigger values than 10 seconds are not selected as the statue of web server changes much faster. Number of connection failures due to file descriptors exhaustion was measured during different frequencies of updates for our algorithm SLBL and RR (Round Robin).

Figure-2 shows that 500 milliseconds update and lower intervals gave no connection failure but higher polling intervals showed connection failures. In case of 10 sec, as the status of server load is not available with load

balancer (LB), LB sends to the loaded server so error starts increasing right away. In case of 5 sec polling the

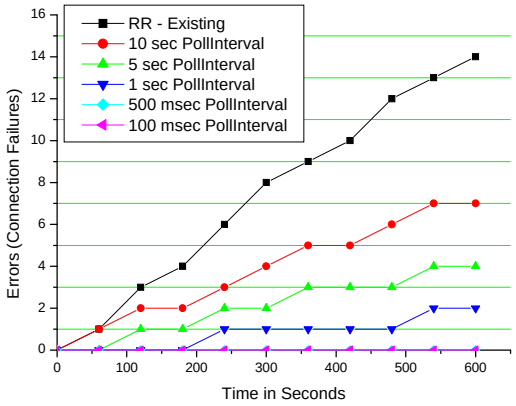

Figure-2. Errors (Connection Failures) over different polling intervals.

update of server load came earlier than 10 seconds so LB avoids loaded server to some extent so errors are less. Similarly in case of 1 sec polling, the status of loaded server comes much earlier than higher polling intervals so errors are less comparatively. There were no results for 500ms and 100ms because of toggling of heavy load on servers is well detected with these polling intervals so LB did not send the traffic to these loaded servers hence there were no errors. So 500ms line and 100ms lines are not shown in Figure-2. As the polling replies did not affect the RR algorithm, we can see that values are same for RR algorithm for all polling intervals.

Different scenarios were tested after finding the right polling interval. Each experimental scenario consists of clients that send HTTP requests to the web servers. The submission rate of requests varies for every experiment accordingly. For every scenario, a fixed set of requests is executed and for each rate performance metrics of the web cluster system are measured. The performance metrics used are the following:

Throughput, this is one of most important metric for the performance measurement of the cluster of web servers. This is measured in Kbps or KBpbs or Mbps units.

HTTP response time is defined as the time from the initial HTTP request being sent until the complete HTTP response is received (in ms). In addition, in our scenarios, HTTP response time is expressed as an average value over all client HTTP requests per request rate.

Error Rate, (or connection failures) is the percentage of the requests that were not serviced or delayed service more than 10 seconds. Error rate is an important performance metric of the balancing algorithm. More specifically, as errors increase HTTP requests that failed service increase. This is an indication of sub-optimal selections of the balancing algorithm that lead HTTP requests to timeout.

Scalability evaluation of the cluster of web servers is also an important measurement.

In our experimental scenarios, we investigate the following balancing algorithms: stateless and adaptive Round Robin (RR) and state full non adaptive least connections (LC). We compare those results with measurements from our SLBL implementation in the following scenarios:

I. The web servers have no initial load and the clients retrieve an object from the cluster. This scenario resembles the case of normal network conditions and lack of web server loads.

II. The first web server's CPU is loaded to test in what extent load balancing algorithms will follow server resource exhaustion.

III. Scalability evaluation is an important performance measurement of the cluster of web servers. The scalability of the cluster is measured with the increasing size of cluster.

A. Scenario I

In this scenario HTTP client requests retrieve 10Kb object from the cluster of web servers. This operation is initially performed at a rate of 100 HTTP requests per second until 1200 HTTP requests per second, with a step of 50 requests per second. The algorithms that are put to test are: RR, LC and our SLBL implementation. The purpose of this scenario is to compare the performance of the stateless RR and of the state full LC balancing algorithms with the performance of the SLBL load balancing algorithm.

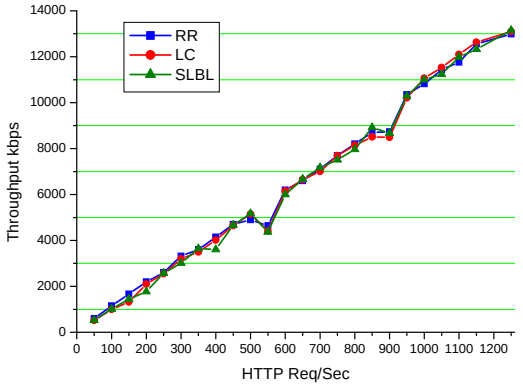

Figure-3. Scenario I. Throughput kbps over clients Req/sec.

Figure-3 shows throughput variation of three algorithms, RR, LC and SLBL for variation of HTTP request rate under normal conditions. We can observe that the throughput of three algorithms is almost same but for 550 HTTP requests/sec and around 950 HTTP requests/sec, throughput was less. This reduction of throughput is due to the network activity during that time.

The same can be verified in Figure-6 which shows error rate of three algorithms RR, LC and SLBL for variation of HTTP request rate under normal conditions. We can observe in Figure-5 that for 550 HTTP requests/sec and 950 HTTP requests/sec, Error rates are high for three algorithms, these are the same points when throughput was shown less in Figure-3.

Figure-4 shows the response time variation of three algorithms RR, LC and SLBL for variation of HTTP request rate under normal conditions. We can observe that initially the RR showed slightly better response time compared to other algorithms LC and SLBL but as the request rate increased above 800 req/sec gradually the difference in response times of LC, SLBL and RR reduced. The reason for little higher response.

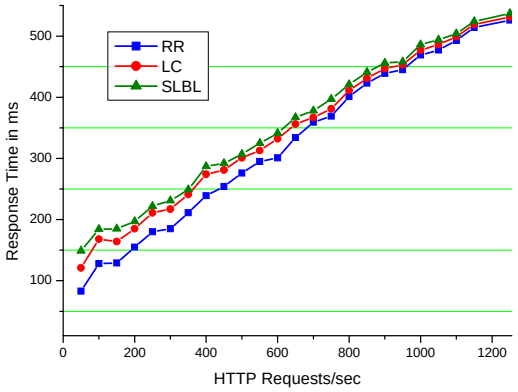

Figure-4. Scenario I. Response time in ms over clients Req/sec.

time of LC and SLBL compared to RR response time is because LC and SLBL uses more resources of web switch and network. And other observation is that in normal scenarios with higher request rates response time is almost same for all these three algorithms.

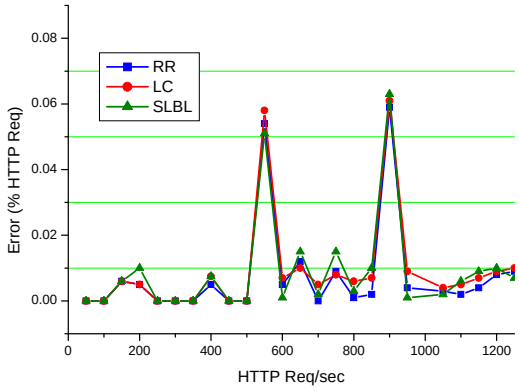

Figure-5. Scenario I. Error rate over clients Req/sec.

The results of this scenario in terms of throughput (Figure-3) show that, under normal conditions, all algorithms perform almost equivalently. This was something expected for this scenario since none of the balancing mechanisms of LC or SLBL is actually used. This is the case where a "blind mechanism" such as RR outperforms "intelligent mechanisms" (LC, SLBL), contribute only to consumption of web switch CPU and network resources.

The conclusion of scenario I is that simple algorithms like RR are sufficient as long as web servers are of the same processing power and have equivalent network resources at their disposal. Furthermore, HTTP requests of small HTTP response content in bytes can be efficiently balanced over a web cluster of uniformly distributed servers, with the use of simple balancing algorithm like RR.

B. Scenario II

In this scenario, the resources of server, file descriptors are exhausted or overloaded, while HTTP clients retrieve 10Kb object from the cluster of web servers. This operation is initially performed at a rate of 10 HTTP requests per second, with a step of 10 requests per second, until 150 HTTP requests per second. The purpose of this scenario is to compare the performance of the stateless RR and the state full LC balancing algorithms with the SLBL algorithm under severe unbalancing conditions due to resource overloading of two of the web servers. To achieve resource overloading of the first web server, resource intensive script is run on two web servers.

Figure-6 shows the throughput of the cluster as increasing the HTTP client requests per second for the three algorithms RR, LC and SLBL. Initially throughput is almost the same for all three algorithms. But as request rate increased RR algorithm's throughput decreased compared to other two algorithms. Above 90 requests per seconds SLBL algorithm showed better throughput compared to LC algorithm as shown the graph (Figure-6) as expected, LC and SLBL algorithms outperform

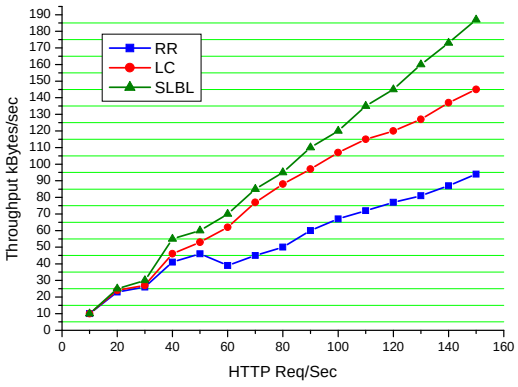

Figure-6. Scenario II. Throughput Kb/sec over clients Requests/sec.

RR in terms of throughput (Figure-6) and we can view more clearly the advantages of an adaptive mechanism such as SLBL.

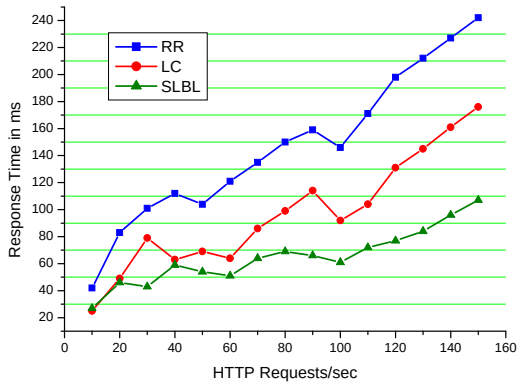

Figure-7. Scenario II. Average HTTP response time over clients Requests/sec.

The response time of the cluster as increasing the HTTP client requests per second for the three algorithms RR, LC and SLBL is shown in Figure-7. In this Figure, the RR algorithm showed comparable response time similar to other algorithms initially, but as connections increase, its response time increases (degrades) rapidly and also SLBL algorithm selection causes HTTP flows to maintain better average

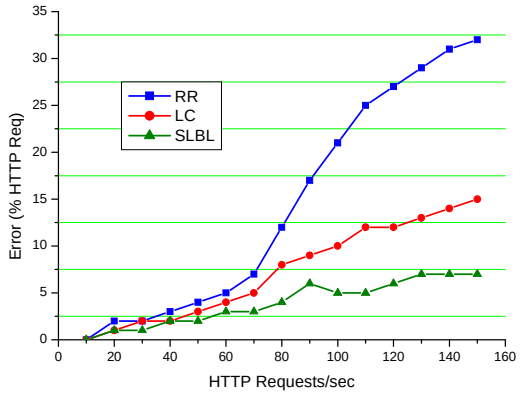

Figure-8. Scenario II. Error rate over clients requests/sec.

HTTP response time than LC algorithm there are fluctuations in the curves of all the three algorithms, i.e., up and downs in the curves are due to network activity during that time.

In Figure 8 shows the error rate of the cluster as increasing the HTTP client requests per second for the three algorithms RR, LC and SLBL. We can observe that initially the error rates of all the three algorithms were less this is due to fewer requests gone to the overloaded servers. As the rate of client requests increased error rate also increased due to many requests going to the overloaded servers. It can be seen that from the rate of 90 requests per second and above many requests timed out. This large number of errors causes HTT P response time to degrade rapidly.

The results of this scenario in terms of the throughput of the cluster (Figure-6) shows that initially the throughput is almost same for all the three algorithms this can be verified by the cause of low error rate observed initially in Figure-8. Above 90 requests per seconds SLBL algorithm showed better throughput compared to other two algorithms this is due to that fact that error rate increased for the other two algorithms around the same time which can be observed in Figure-9. The results of response time of the cluster (Figure-7), shows better response time for SLBL algorithm after 90 requests per second it is due to the lower error rate for SLBL algorithm compared to other algorithms that can be seen during the same time in Figure-8.

The conclusion of scenario II is that simple algorithms like RR are not sufficient for the scenario of overloaded servers. Furthermore, the SLBL algorithm successfully spots load conditions, due to the polling of server resources, number of open file descriptors and thus manages to perform better than the LC algorithm and RR algorithm.

C. Scalability evaluation

In addition to throughput and response time evaluation, performance evaluation of scalability of the cluster of web servers is the other important measurement. With the increase in the number of web servers, more service requests can be served by the cluster of web servers in the measured unit of time. The load balancing strategy plays a crucial role in rising

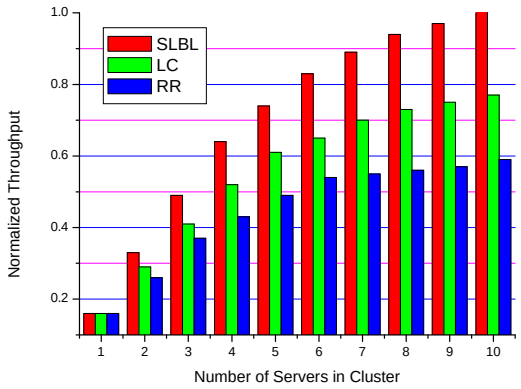

Figure-9. Scalability of Cluster Throughput.

the scalability of the cluster of web servers. Thus, we evaluated the scalability of the cluster in our setup using the value of maximum attainable throughput; it is

measured in requests/second unit. We increased the number of servers in the cluster from 1 to 10 in this set of experiments and generated enough clients in each experiment to measure the maximum throughput of the cluster by varying the numbers of servers in the cluster.

The scalability of the cluster with the increasing size of cluster (the number of web servers in the cluster) is shown in Figure-9. The proposed SLBL algorithm shows the highest scalability by virtue of the improved load balancing strategy and better usage of cluster resources. Contrary to LC, the SLBL benefits from extra status information regarding the health and resources each web server. However, LC has lesser amount of load balancing overhead compared to SLBL while the size of the cluster is expanded. Therefore, under the conditions of low load, the maximum throughput of LC still reaches that of SLBL, and SLBL and LC policies have linear rise for the throughput as the number of nodes increases. RR shows lowest performance compared to SLBL and LC, due to its poor load balancing strategy of simple request allocation technique which assigns requests blindly without caring the status of web servers. Thus, RR comes to a satiation point with much lesser throughput value compared to other schemes. This is the main cause for the very slow growth of scalability as the size of cluster increases.

These performance results clearly indicate that the proposed SLBL algorithm works better than the two other algorithms (RR and LC). Also, under overload conditions, the SLBL algorithm provides stable throughput due to its polling mechanism to get the status of resources on web servers, while the two other algorithms face poor conditions and the throughputs of LC and RR are reduced. In short, the average service request rate that can be serviced by the SLBL algorithm is around 1.27 times more than that of LC and around 1.93 times more than that of RR.

9. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a novel load balancing algorithm using improved health reporting of web servers with extra parameters for cluster-based web servers that is called SLBL. 'Number of open file descriptors', an important load parameter is identified to find the load on a server along with existing load parameters, CPU cycles and free memory. The server load reporting is improved by the use of extended Net-SNMP agent to report the status of server resources including 'number open file descriptors'. HAProxy is used as a load balancer in this setup and it considers the value of 'number of open file descriptors for selecting a server with more resources. Tests were done in two different scenarios: normal scenario and the other is with high load on servers. Performance metrics that are used: throughput, response time and error rate or connection failures. The load balancing results of the server cluster of our implementation are compared with the known load balancing algorithms used on web clusters, Round Robin (RR) and Least Connections (LC). We show that the previously mentioned algorithms can be outperformed by the proposed adaptive load balancing mechanism SLBL. Our experimental results show that the performance of the cluster of web servers is significantly improved by the proposed adaptive algorithm SLBL over the existing algorithms, RR and LC. The average service request rate that can be serviced by the SLBL algorithm is around 1.27 times more than that of LC and around 1.93 times more than that of RR. The core selection and control process of SLBL algorithm can be enhanced with the help of real time status information of transactions and service processes and also using Business Activity Monitoring for the improvement of load balancing efficiency.

REFERENCES

[1] Bourke Tony. 2001. Server load balancing, O'Reilly and Associates. Inc., Sebastopol, CA.

[2] Cardellini Valeria, Emiliano Casalicchio, Michele Colajanni and Philip S. Yu. 2002. The state of the art in locally distributed Web-server systems. ACM Computing Surveys (CSUR). 34(2): 263-311.

[3] K. McCloghrie, M.Rose. 1998. Management Information Base for Network Management of TCP/IP-based internets. IETF, RFC 1066. Available: http://tools.ietf.org/html/rfc1066.

[4] J. Case, M. Fedor, M. Schoffstall, J. Davin. 1998. A Simple Network Management Protocol. IETF, RFC 1067. Available: http://tools.ietf.org/html/rfc1067.

[5] Linux Operating System, http://fedoraproject.org/get-fedora.

[6] W Hardaker *et al*. Net-SNMP package. Available from: http://www.net- snmp.org.

[7] S. Waldbusser, P. Grillo. 2000. Host Resources MIB. IETF, RFC 2790, Available: http://tools.ietf.org/html/rfc2790.

[8] Willy Tarreau *et al*. HAProxy the Reliable, High Performance TCP/HTTP Load Balancer. Available: http://haproxy.1wt.eu/.

[9] Sandeep Sharma, Sarabjit Singh and Meenakshi Sharma. 2008. Performance Analysis of Load Balancing Algorithms. World Academy of Science, Engineering and Technology.

[10] Zhong Xu, Rong Huang. Performance Study of Load Balancing Algorithms in Distributed Web Server

Systems. CS213 Parallel and Dis- tributed Processing Project Report.

[11] Carter R. L and Crovella M. 1995. Dynamic Server selection in the Internet. Tech. Rep. TR-95-014, Computer science Department, Boston University, Boston, MA.

[12] G. R. Andrews, D. P. Dobkin and P. J. Downey. 1982. Distributed allocation with pools of servers. In ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing. pp. 73-83.

[13] Katja Gilly, Carlos Juiz, Ramon Puigjaner. 2011. An up-to-date survey in web load balancing, World Wide Web. 14(2): 105-131.

[14] D. Mosberger and T. Jin. 1998. httperf: A Tool for Measuring Web Server Performance. Performance Evaluation Review. 26(3): 31-37.

[15] Andreolini M., Colajanni M., Nuccio M. 2003. Kernel-based web switches providing content-aware routing. In: Proc. of the 2nd IEEE International Symposium on Network Computing and Applications (NCA'03).

[16] Aron M., Druschel P., Zwaenepoel W. 2000. Cluster reserves: a mechanism for resource management in cluster-based network servers. In: Proc. of ACM SIGMETRICS.

[17] Borzemski L., Zatwarnicki K.: 2003. Afuzzy adaptive request distribution algorithm for cluster-based web systems. In: Proc. of the 11th Euromicro Conference on Parallel, Distributed and Network- Based Processing (Euro PDP).

[18] Cardellini V., Colajanni M., Yu P.S. 1999. Dynamic load balancing on web- server systems. IEEE Int. Comp. 3(3): 28-39.

[19] Derek L. Eager, Edward D. Lazowska, John Zahorjan. 1986. Adaptive load sharing in homogeneous distributed systems. IEEE Transactions on Soft- ware engineering. 12(5): 662-675.

[20] Y.Wang and R. Morris. 1985. Load balancing in distributed systems. IEEE Trans. Computing. C-34, no. 3, pp. 204-217.

[21] M. Zaki, W. Li and S. Parthasarathy. 1997. Customized dynamic load balancing for a network of workstations. Journal of Parallel and Distributed Computing: Special Issue on Performance Evaluation, Scheduling, and Fault Tolerance.

[22] P. L. McEntire, J. G. O'Reilly and R. E. 1984. Larson Distributed Computing: Concepts and Implementations. New York: IEEE Press.

[23] L. Rudolph, M. Slivkin-Allalouf, E. Upfal. 1991. A Simple Load Balancing Scheme for Task Allocation in Parallel Machines. In Proceedings of the 3rd ACM Symposium on Parallel Algorithms and Architectures. pp. 237-245.

[24] William Leinberger, George Karypis, Vipin Kumar. 20002. Load Balancing Across Near-Homogeneous Multi-Resource Servers. 0-7695-0556-2/00, IEEE.

[25] S. Kontogiannis, S. Valsamidis, P. Efraimidis and A. Karakos. 2009. Probing based load balancing for web server farms. In Proc. of the 13th Panhellenic Conference on Informatics. pp. 175-180.

[26] S. Kontogiannis, S. Valsamidis and A. Karakos. 2011. ALBL, ALBL/HSC algorithms: Towards more scalable, more adaptive and fully utilized balancing systems. Journal of Computing. 3(2): 4-19.

[27] Brisco T.P. 1995. DNS support for Load Balancing. RFC 1794.

[28] Iyengar A., Challenger J., Dias D., Dantzig P. 2000. High-performance web site design techniques. IEEE Int. Comp. 4, 17-26.

[29] Colajanni M., Yu P.S. 2002. A performance study of robust load sharing strategies for distributed heterogeneous web server systems. IEEE Trans. Knowl. Data Eng. 14(2): 398-414.

[30] Case J., Fedor M., Schoffstall M. and J. Davin. 1990. The Simple Network Management Protocol. RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance Systems International, and the MIT Laboratory for Computer Science.

[31] Case J., McCloghrie K., Rose M. and Waldbusser S. 1993. Introduction to version 2 of the Internet-standard Network Management Framework. RFC 1441, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University.

[32] Case J., Mundy R., Partain D. and B. Stewart. 1999. Introduction to Version 3 of the Internet-standard Network Management Framework. RFC 2570.

[33] Harrington D., Presuhn R. and B. Wijnen. 2002. An Architecture for Describ- ing Simple Network Management Protocol (SNMP) Management Frame-works. STD 62, RFC 3411.

[34] Case J., Harrington D., Presuhn R. and B. Wijnen. 2002. Message Processing and Dispatching for the Simple Network Management Protocol (SNMP). STD 62, RFC 3412.

[35] Levi D., Meyer P. and B. Stewart. 2002. Simple Network Management Proto- col (SNMP) Applications. STD 62, RFC 3413.

[36] Blumenthal U. and B. Wijnen 2002. User-Based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3). STD 62, RFC 3414.

[37] Wijnen B., Presuhn R. and K. McCloghrie. 2002. View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP). STD 62, RFC 3415.

[38] Daniel W. Yoas, Greg Simco, Resource utilization prediction: a proposal for information technology research, Proceedings of the 1st Annual conference on Research in information technology, October 11-13, 2012, Calgary, Alberta, Canada.

[39] Saeed Sharifian, Seyed A. Motamedi, Mohammad K. Akbari. 2011. A predictive and probabilistic load-balancing algorithm for cluster-based web servers, Applied Soft Computing. 11(1): 970-981.

[40] R. Jayabal, R.Mohan Raj. 2014. Design and Implementation of Locally Dis- tributed Web Server Systems using Load Balancer. International Journal of Engineering Sciences and Research Technology. ISSN: 2277-9655.

[41] Yuanhao Zhou1, Li Ruan1, Limin Xiao1, Rui Liu1. 2014. A Method for Load Balancing based on Software-Defined Network, Advanced Science and Technology Letters. 45(CCA 2014): 43-48, http://dx.doi.org/10.14257/astl.2014.45.09 ISSN: 2287-1233 ASTL.

[42] Hugo H. Kramer, Vinicius Petrucci, Anand Subramanian, Eduardo Uchoa. 2012. A column generation approach for power-aware optimization of virtualized heterogeneous server clusters, Computers and Industrial Engineering. 63(3): 652-662.

[43] Sabato Manfredi, Francesco Oliviero, Simon Pietro Romano. 2013. A dis- tributed control law for load balancing in content delivery networks. IEEE/ACM Transactions on Networking (TON). 21(1): 55-68.

[44] Saeed Sharifian, Seyed A. Motamedi, Mohammad K. Akbari. 2010. An approximation-based load-balancing algorithm with admission control for cluster web servers with dynamic workloads, The Journal of Supercom- putting. 53(3): 440-463.

[45] Schemers R. J. 1995. ldnamed: A load balancing name server in perl. In Proc. of 9th USENIX conference on System administration. pp. 203-210.

[46] Zhang W. 2000. Linux server clusters for scalable network services. In Proc. of Ottawa Linux Symposium. pp. 437-456.

[47] Colajanni M., Yu P. S. and Dias M. D. 1998. Analysis of task assignment poli- cies in scalable distributed web-server systems. IEEE Trans. on Parallel Distributed Systems. 9-6: 585-597.

[48] CISCO. 2007. CISCO Services Modules - Understanding CSM Load Balancing Algorithms.http://www.cisco.com/-warp/public/117/csm/lb_algorithms.pdf.

[49] W. Zhang. 2003. Build highly-scalable and highly-available network services at low cost. Linux Magazine. 3: 23-31.

[50] A. Weinrib and S. Shenker. 1988. Greed is not enough: Adaptive load sharing in large heterogeneous systems. In: Proc. of IEEE INFOCOM. pp. 986-994.

[51] B. Abhay and C. Sanjay. 2010. Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. In Proc. of COMPUTE: The Third Annual ACM Bangalore Conference. New York, NY, USA: ACM. pp. 1-4.

[52] J. Batheja and M. Parashar. 2003. A framework for Adaptive Cluster Computing Using Javaspaces. Cluster Computing. 6-3(3): 201-213.

[53] M. Andreolini and S. Casolari. 2006. Load prediction models in web-based systems. In: Proc. of the 1st international conference on Performance evaluation methodologies and tools. ACM. p. 27.

[54] M. Andreolini, S. Casolari and M. Colajanni. 2008. Models and Framework for Supporting Runtime Decisions in Web-Based Systems. ACM Transactions on the Web. 2(3): 17-43.

[55] P. O'Rourke and M. Keefe. 2001. Performance Evaluation of Linux Virtual Server. In: Proc. of the 15th LISA System Ad-ministration Conference. pp. 79-92.

[56] J. Cao, Y. Sun, X. Wang and S. K. Das. 2003. Scalable load balancing on distributed web servers using mobile agents. Parallel and Distributed Computing. 63(10): 996-1005.

[57] M. A. Saifullah, Dr. M. A. Maluk Mohammad. 2014. Server Load Balancing Through Enhanced Server Health Report. International Journal of Applied Engineering Research. 9(24): 28497-28520.

[58] B. Boone, S. Van Hoecke, G. Van Seghbroeck, N. Joncheere, V. Jonckers, F. De Turck, C. Develder and B. Dhoedt. 2010. Salsa: QoS-aware load balancing for autonomous service brokering. Systems and Software. 83(3): 446-456.

[59] M. A. Saifullah and M. A. Maluk Mohamed. Scalable load balancing using virtualization based on approximation. IEEE International Conference on Computer Communication Technology, December 11 - 13, Hyderabad.

[60] Choi, E.: Performance test and analysis for an adaptive load balancing mechanism on distributed server cluster systems. Future Gener. Comput. Syst. 20, 237–247