Load Balancing as a Strategy Learning Task

K. Kungumaraj[1], T. Ravichandran[2]
Karpagam University
Coimbatore – 21, India
[2]Hindusthan Institute of Technology
Coimbatore – 32, India

ABSTRACT: *A distributed system consists of independent workstations connected usually by a local area network. Load Balancing system puts forward to a new proposal to balance the server load in the distributed system. The load balancing system is a set of substitute buffer to share the server load, when their load exceeds its limit. The proposed technique gives an effective way to overcome the load balancing problem. Serving to more number of client requests is the main aim of every web server, but due to some unexpected load, the server performance may degrade. To overcome these issues, network provides an efficient way to distribute their work with the sub servers which is also known as proxy servers. Allocating work to the sub server by their response time is the proposed technique. The secure socket layer with Load balancing scheme has been introduced to overcome those server load problems. Storing and serving effectively and securely is more important so that desired algorithm is going to implement for load distribution and security enhancement named as Secure Socket Layer with Load Balancing and RSA Security algorithm respectively. Calculating response time of each request from the clients has been done by sending an empty packet over the networking to all the sub servers and response time for each sub server is calculated using the Queuing theory. In this Load Balancing system, the SSL based load distribution schemes has been introduced for better performance.*

Keywords: Load Dispatcher, IP Spraying, Load Prediction, Latency time, Throughput, Load reduction

Received: 12 October 2011, Revised 8 November 2011, Accepted 19 November 2011

1. Introduction

The IT infrastructure is playing an increasingly important role in the success of a business. Market share, customer satisfaction and company image are all intertwined with the consistent availability of a company's web site. Network servers are now frequently used to host ERP, e-commerce and a myriad of other applications. The foundation of these sites, the e-business infrastructure is expected to provide high performance, high availability, secure and scalable solutions to support all applications at all times.

However, the availability of these applications is often threatened by network overloads as well as server and application failures. Resource utilization is often out of balance, resulting in the low-performance resources being overloaded with requests while the high-performance resources remain idle. Server load balancing is a widely adopted solution to performance and availability problems. Server load balancing is the process of distributing service requests across a group of servers.

The highest performance is achieved when the processing power of servers is used intelligently. Advanced server load-balancing products can direct end-user service requests to the servers that are least busy and therefore capable of providing the fastest response times. Necessarily, the load-balancing device should be capable of handling the aggregate traffic of multiple servers.

In order to achieve web server scalability, more servers need to be added to distribute the load among the group of servers, which is also known as a server cluster. When multiple web servers are present in a server group, the HTTP traffic needs to be evenly distributed among the servers. In the process, these servers must appear as one web server to the web client, for example an internet browser. The load balancing mechanism used for spreading HTTP requests is known as IP Spraying. The equipment used for IP spraying is also called the Load Dispatcher or Network Dispatcher or simply, the Load Balancer. In this case, the IP sprayer intercepts each HTTP request, and redirects them to a server in the server cluster. Depending on the type of sprayer involved, the architecture can provide scalability, load balancing and failover requirements. The main advantages of using load balancing algorithm are,

- Increased scalability
- High performance
- High availability and disaster recovery

2. Objectives

The main objective of this research are,

- Review the characteristic of the server and proxy servers.
- Review the different aspects of reducing the load of the server.
- Propose a new concept called SSL_LB Scheme.
- Analysis, design and find the new solution.
- Stimulate as a real time research project and obtain better results.

3. Load Balancing Technique

The main server S and the respective sub servers S_1, S_2, S_3….S_n. When the Client requests to the loaded server is represented as R. The multiple client requests have been assigned R_1, R_2…R_n. The encrypted files will be stored and shared on the network is F. Load of the server LS will be calculated by total number of client request. It will check with the server information to identify the server load. The empty packet P sends to the sub servers and the response RT will be calculated. The selected server with the response time is represented as S_i.

The server, which receives the request from another node, generates and encrypts the dynamic content using the forwarded session key. Finally, it returns the reply to the initial node, which sends the response back to the client. We assume that all the intra communications in a cluster are secure since these nodes are connected through the user-level communication and are located very closely.

```
If (RSA Encrypted) $CR_1 \rightarrow S$ then
    Check load of S.
IF LS(Main Server S) exceeds THEN
    Calculate Response Time R and Send $P \rightarrow S_1, S_2, S_3 \ldots S_n$
    $CR_1 \rightarrow S_i$.
```

The requests arriving at the Web switch of the network server are sent to either the Web server layer or the application server layer according to the requested service by the client. Since the SSL connection is served by a different type of HTTP server (Hypertext Transfer Protocol Secure (HTTPS)) and a different port number, the requests for the SSL connection are passed on to the distributor in the application server layer. To solely focus on the performance of the application server, it ignores the latency between the Web switch and the distributor and logically represents them as one unit. When a request arrives at the distributor, it searches its lookup table to determine whether there is a server that has the session information of the client and then forwards the request to the server. Otherwise, it picks up a new server to forward the request. The forwarded server establishes a new SSL connection with the client. If the request is forwarded to a highly loaded server, the server in turn sends the request with the session information to a lightly loaded server.

The server identifies the available server by sending an empty packet. If the sub server is free then it will responds immediately, through the response time it allocates the clients to the proxy servers. End-user requests are sent to a load-balancing system that determines which server is most capable of processing the request. It then forwards the request to that server. Server load balancing can also distribute workloads to firewalls and redirect requests to proxy servers and caching servers.

4. Load Balancing Algorithm

Step 1: Representing the server S_r and its proxy servers by defining the IP address and Host name $S_r(i_1, i_2, i_3, .. i_n)$. This helps to allocate and construct the network structure with main server and proxy server before proceeding.

Step 2: Save those Network construction with relevant details. This can be done with proper authentication.

Step 3: Select the file (F) to be shared with the proxy servers.

Step 4: Encrypt the files with the help of private and public keys. This can be done with the help of RSA algorithm.

Step 5: Save those encrypted files on the sub servers. These files will be automatically stored on the proxy's, the proxy servers could be identified by the network construction module, which stores the IP addresses.

Step 6: The uploaded files are sharable and the requested node R(n) can be download those files which they needed.

Step 7: The next step is evaluating the server load. When client requests the server, server calculates the load by the number of open connections N(S). The request of the clients will be categorized into 2 types such as dynamic and static requests.

Step 8: The server distributes an empty packet EP $(S_{i1}, S_{i2}, \ldots\ldots S_{in})$ to all sub servers and gathers the response. The response time will be calculated through queuing method.

Step 9: The response time will be calculated and compared by using the total number of requests and download time of the sub servers.

Step 10: The user request is redirected to the sub server and the user can download the files and decrypt using the private key.

5. Results and Discussion

This section introduces the experiments performed on the overall implementation in order to measure its performance. Two sets of experiments are used: determination of the performance strategy based on the task, and the efficient of the scheduling period on the performance. The experimental results were averaged over 3 sub servers, and computed at a 90% interval.

In order to ensure consistency in the experiment results, these were run at different node to ensure that the hosts used were free from jobs other than those generated by the main server. The default values used in the experiments are given in the following Table 7.1.

Parameters	**Value**
Number of sub servers	3
Scheduling period	1 second
Expected Response time (approximately)	1 second
Performance measurement	1 second

Table 1. Parameter for the Implementation

The experimental values are given in the above table. The usage of the servers in implementation process, the scheduling period, expected response time for each task and the performance measurement time were given in the above table.

The measurement based on the number of task and the performance has been measured by the percentage. The effective and the performance was high when the number of task was low. When the task level increases the performance was reduced slightly.

This is the main drawback of those algorithms and methods. To overcome those issues the proposed algorithm has been implemented. The parameter was same for the comparison. The implementation of the SSL_LB algorithm was tested with some limited load the figure implies that.

The following values are the outcome of the proposed system. The evaluation between the sub serservers based on the response time will be calculated and verified before data process.

Server Name	Starting Time	Ending Time	Response Time
Server1	16:52:14.0156250	16:52:14.0312500	0.015625
Server2	16:52:14.0156250	16:52:14.0313700	0.015745
Server3	16:52:14.0156250	16:52:14.0324600	0.016835

Table 2. Server Response Time

This experiment confirmed this fact, and indicated that this improvement implies an improvement in response time. Furthermore, these improvements follow the same pattern indicating a strong impact of load balancing on response time. On the other hand, when decreasing a larger number of movements of jobs is required to maintain all loads within and a larger number of wrong job transfers follow due to the outdated load information, and the asynchrony of schedulers. At the other extreme the band is so large, the priority to load balancing are already within the schedule. Hence the existing algorithm has not much improvement to perform, and so importance of the SSL-LB was increased.

The proposed sub servers are tested on 3 test beds. The response time was calculated from the 3 servers by using the above formula finally the best server will do the task which the server allocates. Through the response time the server could calculate the download speed and the task completion time. The comparison process will be considered before task navigation. The chart and the table represents the outcome of the proposed algorithm. The each sub server responded with a specified time period. Through that the best one will be selected.

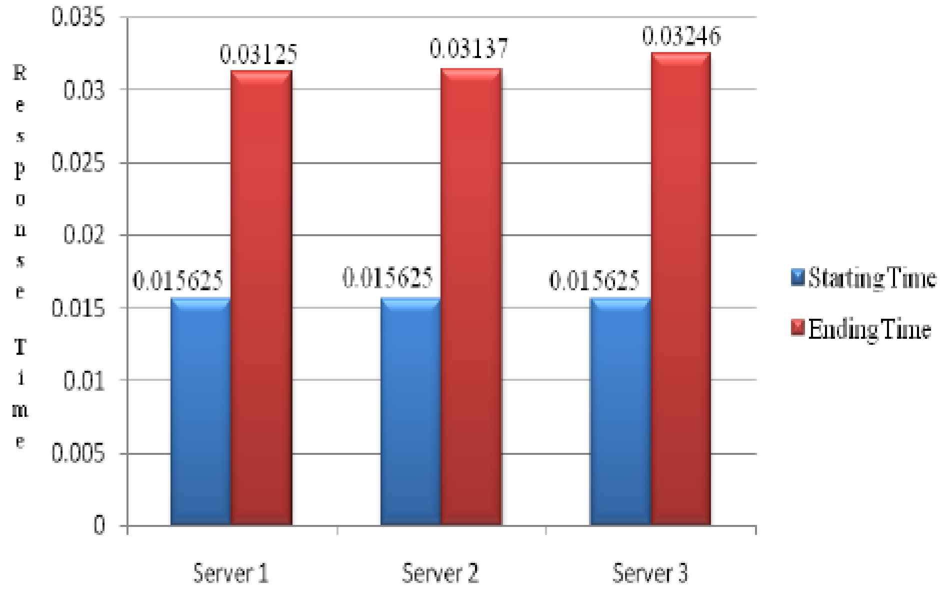

Figure 1. Response time comparison for three different servers

The above table represents the download time of the server after got the response. The various measurements and comparative study defines the response time of the sub server was increased in the SSL_LB algorithm. Our simulation results show the above response time and the server which have been allocated when the load was high. Finally the result proofs the above discussion regarding the load balancing and its performance issues.

Selected Server Name	Request Sent time	Got Response at	Response Time
Server1	16:52:14.0156250	16:52:14.0312500	0.015625

Table 3. Outcome of the proposed system

Load balancing system consists of two performance metrics:

- Latency
- Throughput.

Latency is the time between the arrival of a request at a server and the completion of the request. Throughput is the number of requests completed per second. The latency and throughput results of the three models are in a 16-node application server. The results are measured as the parameter of the pareto distribution for the incoming request interval. When the request distribution decreases, the request interval becomes shorter and, consequently, the load on the servers increases. When analyzing the performance of this system with the existing proposals this shows a major benefit of load distribution.

Metrics	**SSL_Session**	**SSL_LB**	**RR**
Latency	4.3	2.4	4.8
Throughput	2.5	4.4	2.0
Coverage	5.9	8.6	6.5
Security	6.7	8.7	5.0

Table 5. Comparative measurement of three different algorithms

Achieving these performance benefit in the domain of server load balancing concept is not a small task, even the load has increased the performance will be effectively analyzed.

The performance impact of Load Balancing can be measured in four key areas:

a) Latency

b) Throughput

c) Coverage

d) Security

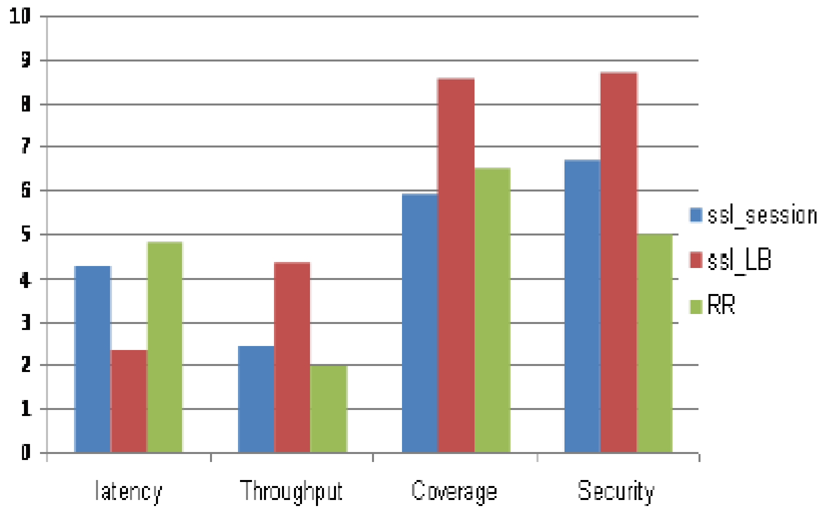

Figure 2. Performance comparison of three different algorithms

a) Latency:
In practice, hosts are added to a Network Load Balancing cluster in proportion to the request rate as the client load increases. When this is the case, the server may respond later. This will affect the client. This system propose to minimize the latency when the client requesting a file. This can be done by load balancing scheme which regulates user request and makes the prompt response.

b) Throughput:
Throughput is the average rate of successful message delivery over a communication channel. Network throughput is the sum of the data rates that are delivered to all terminals in a network. **Throughput** to clients, which increases with additional client traffic that the cluster can handle prior to saturating the cluster hosts (higher is better).

c) Coverage:
Dealing the client requests efficiently even the serer load capacity exceeds is more important for every load balancing scheme. But in the existing proposals RR and SSL_Session methods are considering only a limited set of client request. This makes the performance better than the other two schemes.

d) Security :
Sharing the files in the network makes every file available in the sub server. So that the sub server can respond to their clients more effectively. But the security issues may create problems by using sub servers. Preventing those files from the security threads is more important, in this system the files are shared and stored after the encryption, so that security is high than the existing schemes.

6. Conclusion

Server load balancing is a powerful technique for improving application availability and performance in service provider, web content provider and enterprise networks, but piecemeal implementation can also increase network cost and complexity. Server load balancing devices that don't operate at wire speed can also create their own performance bottlenecks.

References

[1] SSLeay Description and Source, http://www2.psy.uq.edu.au/ftp/Crypto/, 2007.

[2] Abbott, S. (1997). On the Performance of SSL and an Evolution to Cryptographic Coprocessors, *In:* Proc. RSA Conf., Jan.

[3] Allen, C., Dierks, T. (1997). The TLS Protocol Version 1.0, IETF Internet draft, work in progress, Nov.

[4] Amza, C., Chanda, A., Cox, A. L., Elnikety, S., Gil, R., Cecchet, E., Marguerite, J., Rajamani, K., Zwaenepoel, W. (2002). Specification and Implementation of Dynamic Web Site Benchmarks, *In:* Proc. IEEE Fifth Ann. Workshop Workload Characterization (WWC-5), Nov.

[5] Andreolini, M., Casalicchio, E., Colajanni, M., Mambelli, M. (2004). A Cluster-Based Web System Providing Differentiated and Guaranteed Services, Cluster Computing, 7 (1) 7-19.

[6] Apostolopoulos, G., Aubespin, D., Peris, V., Pradhan, P., Saha, D. (2000). Design, Implementation and Performance of a Content-Based Switch, *In:* Proc. INFOCOM.