ISSN:2229-6093

AN EFFICIENT LOAD BALANCING ALGORITHM FOR A DISTRIBUTED COMPUTER SYSTEM

K.Kungumaraj, M.Sc., B.L.I.S., M.Phil.,
Research Scholar,
Karpagam University,
Coimbatore - 641 021.

Dr. T.Ravichandran, B.E (ECE), M.E(CSE), Ph.D., MISTE.,
Principal,
Hindusthan Institute of Technology,
Coimbatore – 641 032.

ABSTRACT

This Load Balancing algorithm puts forward a new proposal to balance the server load. 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 SSL_LB and RSA 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. 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.

1. INTRODUCTION

Internet server applications must be able to run on multiple servers to accept an ever increasing number of users and networks need the ability to scale performance to handle large volumes of client requests without creating unwanted delays. So, load balancing algorithm must be implemented for better performance as well as to increase the ability to handle more number of users. For these reasons, clustering is of wide interest to the enterprise. Clustering enables a group of independent servers to be managed as a single system for higher availability, easier manageability, and greater scalability.

Today, very few enterprises can afford to host their company's web site on a single, monolithic server. Rather, sites are deployed on server clusters that improve performance and scalability. To provide fault tolerance and hide cluster detail from site visitors, a load balancing and application acceleration appliance sites between the Internet and a server cluster, acting as a virtual server.

Due to the growing popularity of the Internet, data centers/network servers are anticipated to be the bottleneck in hosting network-based services, even though the network bandwidth continues to increase faster than the server capacity. It has been observed that network servers contribute to approximately 40 percent of the overall delay, and this delay is likely to grow with the increasing use of dynamic Web contents. For Web-based applications, a poor response time has significant financial implications.

The Above figure-1 represents the overall system model. 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.

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. These servers must appear as one web server to the web client, for example an internet browser.

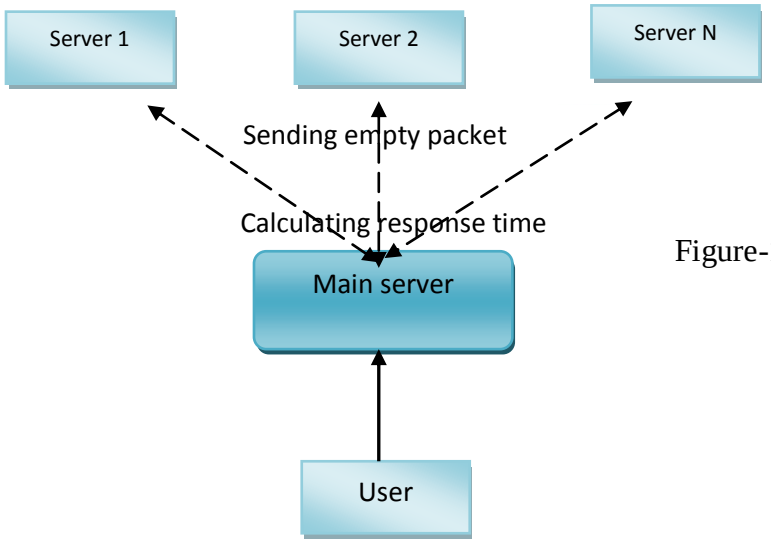

Figure-1 System Model

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.

OBJECTIVES :

- Review the characteristic of the server and proxy servers.
- Review the different aspects of reducing the server load.
- Propose a new concept called SSL_LB Scheme.
- Analysis, design and find the new solution.

NEED FOR THE SYSTEM :

Network Load Balancing is superior to existing system such as round robin DNS (RRDNS), which distributes workload among multiple servers but does not provide a mechanism for server availability. If a server within the host fails, RRDNS, unlike Network Load Balancing, will continue to send it work until a network administrator detects the failure and removes the server from the DNS address list. This results in service disruption for clients. This project also has advantages over other load balancing solutions—both hardware- and software-based—that introduce single points of failure or performance bottlenecks by using a centralized dispatcher. Because the project Load Balancing has no proprietary hardware requirements, any industry-standard compatible computer can be used. This provides significant cost savings when compared to proprietary hardware load balancing solutions.

ADVANTAGES :

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

2. PROBLEM FORMULATION :

Complex systems make increasing Demands on web servers. Multiple Objects can interfere, and high volumes can overwhelm Systems. Fixes need to be identified early in this research, and Clients have scalability concerns, and must warrantee some level of scalability with industry accepted metrics. The basic performance challenges for both the browser

and server sides of the equation and advises on an overall approach for identifying and attacking performance bottlenecks. Identifying load of the servers is more complicated process. The identification of load refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently. As such load identification is most relevant for multi-user systems; often one built using a client/server model, such as web servers. However, other types of software systems can also be used for load testing.

There are few simple symptoms shows network server load. If the server load exceeds its limit then the application will automatically gets slow down and response from the server will be very low.

Single physical Origin or Proxy Server may not be able to handle its load. For Web-based applications, a poor response time has significant financial implications due to the long response time resulting from the Secure Sockets Layer (SSL), which is commonly used for secure communication between clients and Web servers. Even though SSL is the de facto standard for transport layer security, its high overhead and poor scalability are two major problems in designing secure large-scale network servers. Deployment of SSL can decrease a server's capacity by up to two orders of magnitude. In addition, the overhead of SSL becomes even more severe in application servers. Application servers provide Dynamic contents and the contents require secure mechanisms for protection. Generating dynamic content takes about 100 to 1,000 times longer than simply reading static content. Moreover, since static content is seldom updated, it can be easily cached. Several efficient caching algorithms have been proposed to reduce latency and increase throughput of front-end Web services. However, because dynamic content is generated during the execution of a program, caching dynamic content is not an efficient option like caching static content. Server load may increase slightly when more number clients requesting at a particular time.

Prediction of change in load:

Many load balancing algorithms consider only load on a virtual server, ignoring the volume it is assigned in the ID space. Since volume is often closely correlated with rate of object arrival, it could reduce the chance that a node's load increases significantly between periodic load balances by avoiding the assignment of

virtual servers with light load but large volume to nodes with little unused capacity. This suggests a predictive scheme which balances load based on, for example, a probabilistic upper bound on the future loads of a virtual server.

Balance of multiple resources:

In the Load balancing system, a difficult task is considering both bandwidth and storage. This would be modeled by associating a load vector with each object, rather than a single scalar value.

Load balancing system is automatically distribute (load balance) client requests across a number of servers providing the same service to the client, such as Internet Information Services (IIS). With a production database that allows updates however, only one server can be active. In a high-availability environment, you create a Load balancing cluster to easily route client requests from the original primary server to a promoted secondary server without having to update each client directly. Although Load balancing requires you to manually reconfigure the Load balancing cluster to point clients to a secondary server, manually modifying the load balancing is much faster than manually changing the connection information for each client.

3. METHODS

Load Balancing Algorithm

If $CR_1 \longrightarrow S$ then

Check load of S.

If LS exceeds then

Calculate R and then Send $P \longrightarrow S_1, S_2, S_3 \ldots S_n$

$CR1 \longrightarrow S_i$.

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.

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.

The above figure-1 represents the architecture of the proposed mechanism. The requests from various clients are gathered in the server side, the server load will be calculated using the SSL_LB scheme if the server load exceeds the sub servers' details can be collected. After that the server sends as empty packet to all the sub servers, depends on the response time the client request will be navigated to the particular sub server. The client navigation will be considered

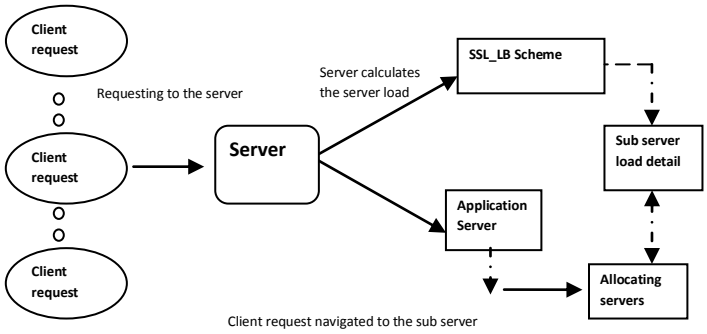

Figure - 1 : SSL_Load Balancing Architecture

SSL_Load Balancing (SSL-LB) Algorithm:

Step 1: Representing the server and its sub servers by defining the IP address and Host name. 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 files 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 client 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. 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 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.

The server load will be calculated by determining some elements such as number of connections made in the network, proxy server allocation. The approximate server weight consist with the number of open connections, here the concept cluster has been proposed to combine all requesting clients for a particular server at a time.

4. RESULTS AND DISCUSSION

Load balancing system consists of two performance metrics: Latency and 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 three models are in a 16-node application server. When the request distribution decreases, the request interval becomes shorter and, consequently, the load on the servers increases. 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:

- Latency
- Throughput
- Coverage
- Security

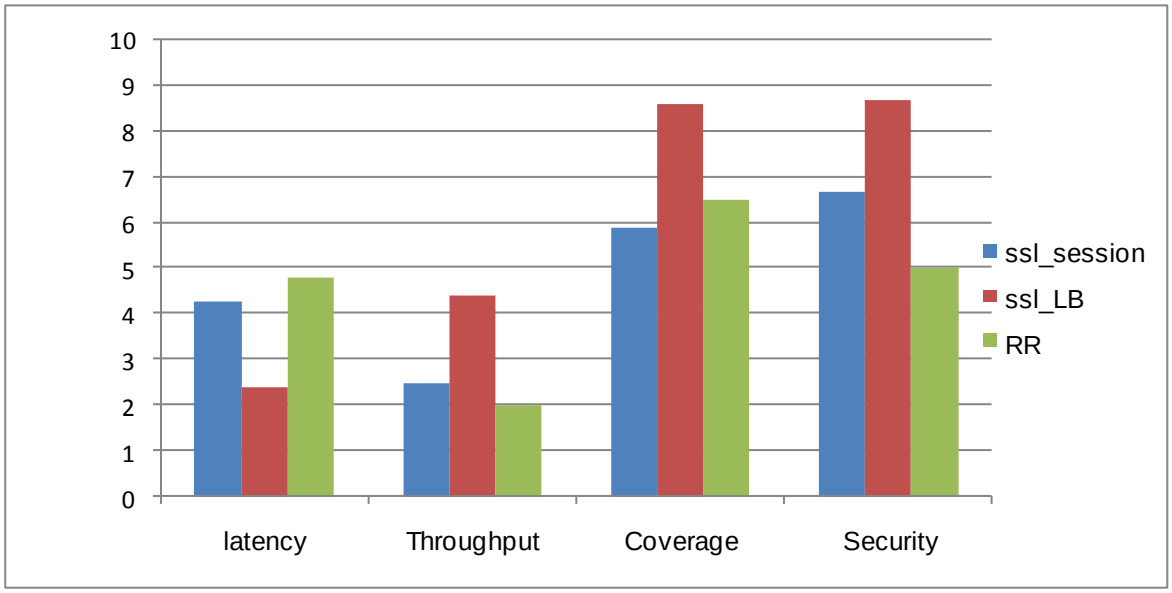

Figure-2 **Performance Comparison**

The above Figure-2 describes the performance comparison between the existing approaches such as RR and SSL_with_session with the proposed system. This result shows the effectiveness of the proposed system by using three parameters such as latency, throughput and security.

5. CONCLUSION

Through this Research paper, assume that 80 percent of the requests are dynamic and are automatically directed to the application servers. The remaining 20 percent are serviced by the Web servers. It will be more adaptable to the 99 percent dynamic requests in the future and effective load distribution will be made.

This Research is carry out for pay load reduction of the web server when the server being busy and investigated the performance implications of the SSL protocol for providing a secure service in a cluster-based web application server. An ultimate aim of this thesis is to stimulate this scenario as a real time project, analysis and evaluate results. Using three proxy servers, proposed a back-end forwarding scheme for distributing load to the lightly loaded server and improving server performance to obtain the better results. In this system examined different aspects of using IP any cast as a mechanism for load distribution and service location in the Internet.

Load balancing systems help optimize server workloads in virtual data center environments. Focal Point can enhance this function by offering virtual fabric memory partitions for storage and data as well as providing low latency storage access. Focal Point can also help distribute the processing load across multiple load balancing cards or systems, providing scalable solutions. Finally, Focal Point offers advanced frame forwarding and security features ideally suited to advanced load balancing systems.

REFERENCES :

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

[2] S. Abbott, "On the Performance of SSL and an Evolution to Cryptographic Coprocessors," Proc. RSA Conf., Jan. 1997.

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

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

[5] M. Andreolini, E. Casalicchio, M. Colajanni, and M. Mambelli, "A Cluster-Based Web System Providing Differentiated and Guaranteed Services," Cluster Computing, vol. 7, no. 1, pp. 7-19, 2004.

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

[7] G. Apostolopoulos, V. Peris, and D. Saha, "Transport Layer Security: How Much Does It Really Cost?" Proc. INFOCOM, 1999.

[8] M.F. Arlitt and C.L. Williamson, "Internet Web Servers: Workload Characterization and Performance Implications," IEEE/ACM Trans. Networking, vol. 5, Oct. 1997.