International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

Implementation of Secured Protocol in Clustered Web Servers

G. Sri Lakshmi[*1], Dr. K. Kungumaraj[2]

*[1]Department of Computer Science, Mother Teresa Women's University,Kodaikanal, Tamil Nadu, India
[2]Department of Computer Applications, ArulmiguPalaniandavar Arts and Science College for Women, Palani, TamilNadu, India

ABSTRACT

The application server handles dynamic and sensitive Web information that need security from listening stealthily, altering, and fabrication. Despite the fact that the Secure Sockets Layer (SSL) is the most famous convention to give a protected channel between a customer and a cluster based organize server, its high overhead corrupts the server execution impressively and, in this manner, influences the server adaptability. In this way, enhancing the execution of SSL-empowered system servers is basic for outlining adaptable and superior server farms. In this venture, the effect of SSL offering is analysed and ssl-session-mindful conveyance in group based system servers. In a back-end sending plan, called SSL with bf that utilizes a low-overhead client level correspondence instrument like Virtual Interface Engineering (VIA) which is utilized to accomplish a decent load adjust among server hubs. To look at the three circulation models for organize servers, Round Robin (RR), ssl_with_session, and ssl_with_bf, through reproduction. The trial result with I6-hub and 32-hub bunch designs demonstrates that, in spite of the fact that the session reuse of ssl_ with session is basic to enhance the execution of application servers. The proposed back-end sending plan can additionally upgrade the execution because of better load adjusting. The ssl_with_bf plan can limit the normal inactivity by around 40 percent and enhance throughput over an assortment of workloads.

Keywords : Web Services, Load balancing, SSL, Web servers, low overhead user level.

I. INTRODUCTION

As the network traffic inside a company's network or website increases, the load on server or server pool increases. Each ask for to get to applications or data from a server adds to the general preparing limit that it can deal with. This

expansion in client get to keeps on including until, eventually, the server can't deal with any more movement, and accidents. Associations can maintain a strategic distance from this additional server load and potential server farm fall with a responsive server load balancer. Server load balancing is a way for servers to effectively handle high-volume traffic and avoid decreased load times and accessibility problems. By properly and evenly distributing network and web traffic to more than one server, organizations can improve throughput and application response times. Data centres implementing a server load balancing solution utilize a hardware device known as a multi-layer switch to distribute network traffic, while maintaining optimal performance in application delivery[3].

Availability with Server Load Balancing

In many business IT infrastructures, multiple network paths exist to guide user access to internal and external networks. With server load balancing, users no longer experience network downtime, slow information retrieval, or failed connections. By maintaining alternate routes to destination pages and applications through distributed server requests, server load balancing provides users with guaranteed access to the information. This fail-over system provides a "backup" path in case one server loses functionality. By ensuring application availability over business networks, organizations can gain continued infrastructure support to maintain a high level of performance.

Web Server Load Balancing

When HTTP traffic increases on an organization's website, load balancing helps distribute the flow to the appropriate web servers. Web server load balancing evenly distributes the surplus of traffic flow to multiple servers with a multi-layer switch. The switch intercepts every HTTP request directed at an organization's website, redirecting them to the appropriate server within the data center. Through web server load balancing, web traffic increases are efficiently distributed, resulting in optimized page load times and information delivery.

II. DESCRIPTION OF THE PROBLEM

Existing Frame Work

In existing framework, they have used to build up the venture utilizing Round Robin [RR] model and SSL_with_Session display. Those models are not viable [1]. Those models are not ready to give the out invest energy and the exhaustive put likewise lesser than that their normal yield. These models had made the Latency issue and insignificant through put. For this issue they acquainted the SSL_with_bf show is with defeated the current issues. We are going to implement SSL_with_BackForwarding model in our proposed system to produce high through put and reduce latency. Despite of using SSL for message transfer not all the requests are secured. Because of this lack in security, it becomes a bottleneck in many E-Commerce Web sites[2].

Proposed System

In our Proposed System, We are going to implement the SSL_with_BackForwarding model (Algorithm) is to overcome the problem of existing system. This model will reduce the latency and increase the throughput than the existing system (Round Robin model and SSL_with_Session)[4]. The Secure Socket Layer_with_BF model is very helpful for load balancing of the server. This will reduce the load of the server while the server is being busy. These are the advantages of our proposed system. The ssl backforwarding scheme can minimize the average latency by about 40 percent and improve throughput across a variety of workloads.

Problem Statement

The current calculations are static, which isn't changed in light of various system topology. The dormancy diminishment rate is low, which cause colossal system movement. The SSL security is as it were in light of RSA cryptographic system, however the RSA encryption strategy utilizes 1024 piece keys which is effectively crackable. Complex frameworks make expanding Demands on web servers. Numerous Objects can meddle, and high volumes can overpower Systems. Fixes should be distinguished right on time in this examination, and Clients have versatility concerns, and should warrantee some level of adaptability with industry acknowledged measurements. The essential execution challenges for both the program and server sides of the condition and exhorts on a general approach for distinguishing and assaulting execution bottlenecks. Distinguishing heap of the servers is process that is more convoluted. The ID of load alludes to the act of demonstrating the normal use of a product program by mimicking different clients getting to the program simultaneously. In that capacity stack distinguishing proof is most pertinent for multi-client frameworks; frequently one manufactured utilizing a customer/server display, for example, web servers. Be that as it may, different sorts of programming frameworks can likewise be utilized for stack testing. There are couple of basic side effects demonstrates arrange server stack. On the off chance that the server stack surpasses its cut-off then the application will naturally gets back off and reaction from the server will be low. Single physical Origin or Proxy Server will most likely be unable to deal with its heap. For Web-based applications, a poor reaction time has huge money related ramifications because of the long reaction time coming about because of the Secure Sockets Layer (SSL), which is ordinarily utilized for secure correspondence amongst customers and Web servers. Despite the fact that SSL is the true standard for transport layer security, its high overhead and poor versatility are two noteworthy issues in planning secure extensive scale organize servers. Organization of SSL can diminish a server's ability by up to two requests of greatness. [5-7] What's more, the overhead of SSL turns out to be considerably more extreme in application servers. Application servers give Dynamic substance and the substance require secure components for assurance. Producing dynamic substance takes around 100 to 1,000 times longer than basically

perusing static substance. Also, since static substance is at times refreshed, it can be effortlessly stored. A few productive reserving calculations have been proposed to lessen inactivity and increment throughput of front-end Web administrations. Nevertheless, because dynamic substance is created amid the execution of a program, storing dynamic substance is not a proficient choice like reserving static substance. Server load may increment somewhat when more number customers asking for at a specific time.

III. OBJECTIVES

To provide a dynamic algorithm, which balances network traffic, based on different kinds of topologies. To provide an efficient latency reduction rate to the network traffic. To provide secured uncrackable key for each client request and efficient load balancing technique.

IV. IMPLEMENTATION

Load Balancing Algorithm

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 CR_1 S then
Check load of S.
If LS exceeds then
Calculate R and then Send P S_1, S_2, S_3…Sn
CR_1 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.

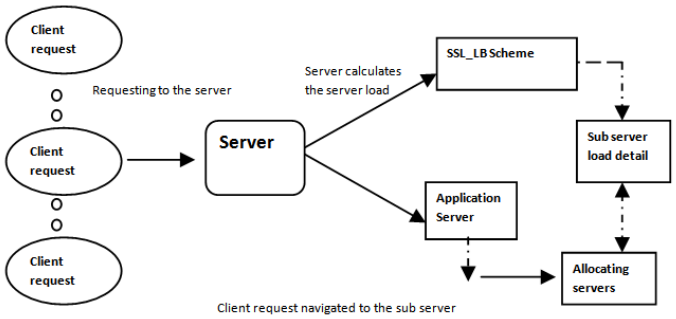

Figure 1: SSL_Load Balancing Architecture

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 server's 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. The Secured Socket Layer Protocol is depicted in the below figure -2.

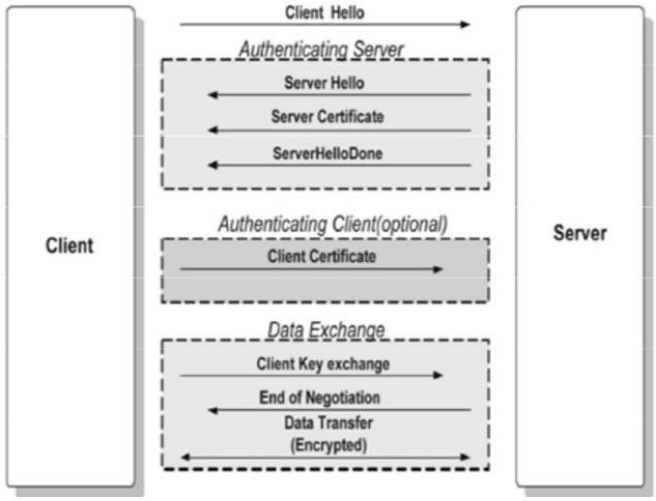

Figure 2: SSL Protocol

V. CONCLUSION

We explored the performance implications of the SSL convention for giving a safe administration in a load based application server and proposed a back-end sending plan for enhancing server execution through a superior load adjust. The proposed ssl_with_bf conspire abuses the hidden client level correspondence with a specific end goal to limit the intracluster correspondence overhead. We looked at three application server models, RR, ssl_with_session and ssl_with_bf, through reproduction. The recreation demonstrate catches the VIA correspondence attributes and the application server plan in adequate detail and uses sensible numbers for SSL encryption overheads acquired from estimations[8]. Reproduction with 16-hub and 32-hub bunch designs with an assortment of workloads gives the accompanying conclusions: First, plans with reusable sessions, conveyed in the ssl_with_session and ssl_with_bf models, are basic for limiting the SSL overhead. Second, the normal dormancy can be diminished by around 40 percent with the ssl_with_bf demonstrate contrasted with the ssl_ with_session show, bringing about enhanced throughput. Third, ssl_with_bf yields a superior execution with the blended customers, though the execution of the ssl_with_session display is corrupted because of the expanding skewness. At last, ssl_with_bf is heartier than ssl_with_session in taking care of variable document sizes.

VI. REFERENCES

[1]. Kumaravel, A. and K. Rangarajan, 2013. Routing algorithm over semi-regular Tessellations, 2013 IEEE Conference on Information and Communication Technologies, ICT 2013.

[2]. K. Kungumaraj& Dr. T. Ravichandran (2012) Secure Socket Layer based Load Balancing Algorithm for a Distributed Computer System .International Journal of Computer and Internet Security, 04, (1-7).

[3]. Khanaa, V., K. Mohanta and T. Saravanan, 2013.Comparative study of web communications over fiber using direct and external modulations, Indian Journal of Science and Technology, 6(suppl 6): 4845- 4847.

[4]. Kumar Giri, R. and M. Saikia, 2013. Multipath routing for admission control and load balancing in wireless mesh networks, International Review on Computers and Software, 8(3): 779-785.

[5]. Leay, S.S., 2007. Description and Source, http://www2.psy.uq.edu.au/ftp/Crypto.

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

[7]. Allen, C. and T. Dierks, Nov, 1997. The TLS Protocol Version 1.0, IETF Internet draft.

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