International Journal of Computing Science and Information Technology, **2013, Vol.3,Iss.1 13-18**
ISSN: 2278-9669, January 2015 (http://ijcsit.org)

DYNAMIC LOAD BALANCING STRATEGIES FOR A DISTRIBUTED COMPUTER SYSTEM

[1] J.VIJAYABHARATHI **[2] Dr.K.KUNGUMARAJ**

[1]Assistant Professor, Department of Computer Science, Mother Teresa Women's University Kodaikanal.

[2] Assistant Professor, Department of Computer Science, APA College For Women, Palani.

Abstract

A distributed system consists of independent workstations connected usually by a local area network. 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. Network server 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 based on their response time is the proposed Dynamic Load Balancing technique.

Keywords: Dynamic Load Balancing, Proxy Server, Client-Server, Distributed System.

1. Introduction

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. For example, E-Biz reported about $1.9 billion loss in revenue in 1998 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 the server's capacity up to two orders of magnitude.

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 improve performance and availability problems. Server load balancing is the process of distributing service requests across a group of servers.

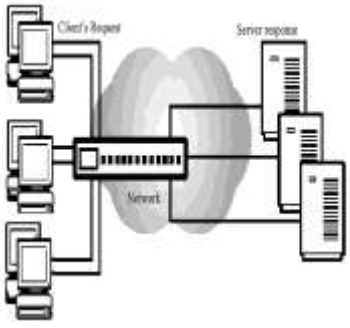

Figure-1 : Client – Server interaction

Figure-1 represents how clients interact with the server. End-user requests are sent to a load-balancing device that determines which server is most capable of processing the request. Then it forwards the request to that server. Server load balancing can also distribute workloads to firewalls and redirect requests to proxy servers and caching servers.

2. RELATED WORKS

In a single Web server environment, the cost of the SSL layer was studied by Apostolopoulos et al. [1] using the Netscape Enterprise Server and Apache Web server, and it was shown that the session reuse is critical for improving the performance of Web servers. This study was extended to a cluster system that was composed of three Web server nodes [2]. The paper has described the architecture of the L5 system and has presented two application experiments: Routing HTTP session based on Uniform Resource Locators (URLs) and Session-aware dispatching of SSL connections. The SSL-session reuse scheme is also investigated in [3], which presented a session-based adaptive overload control mechanism based on SSL connections differentiation and admission control.

Guitart et al. [4] proposed a possible extension of the Java Secure Socket Extension (JSSE) API to allow the differentiation of resumed SSL connections from new SSL connections. Recent studies on data centers have focused on cluster based Web servers [5], [6], [7] and the following works are related to the researcher research. Aron et al. [5] has proposed the backend request forwarding scheme in cluster-based Web servers for supporting HTTP1.1 persistent connections. The client requests are directed by a content-blind Web switch to a Web server in the cluster by a simple distribution scheme such as the RR Domain Name System (DNS). The first node that receives the request is called the initial node. The initial node parses the request and determines whether to service it locally or forward it to another node based on the cache and load balance information. The forwarded request is sent back to the initial node for responding to the client. However, this study does not consider the impact of user-level communication and SSL-enabled application servers. The first effort that has analyzed the impact of user-level communication on distributed Web servers is the PRESS model [6]. The clients in the PRESS model communicate with the cluster using TCP over a Fast Ethernet, whereas the intra-cluster communication uses VIA over connectionless local area network (cLAN) [6]. It is shown that the server throughput can improve up to 30 percent by deploying VIA.

J.H. Kim et al. [7] shows that, in addition to taking advantage of a user-level communication scheme, co-scheduling of the communicating processes reduces the average response time by an additional 25 percent. Due to the low cost of the intra-cluster communication, reading a file from a remote cache turns out to be faster than reading the file from the local disk. Implementation on an 8-node cluster shows that PRESS can improve the server throughput by about 29 percent compared to the TCP/IP model.

Zhou et al. [8] have deployed VIA between a database server and the storage subsystem. They implemented the interface, called Direct Storage Access (DSA), to support the Microsoft SQL Server to use VIA. However, none of these studies has investigated the application server performance with SSL offering. Amza et al. [9] have explored the characteristics in several Web sites, including auction, online bookstore, and bulletin board sites, using synthetic benchmarks. In their study, the online bookstore benchmark reveals that the CPU in the database server is the bottleneck, whereas the auction and bulletin board sites show that the CPU in the Web server is the bottleneck. Cecchet et al. [10] examines the performance and scalability issues in Enterprise Java Beans (EJB) applications. They have modeled an online auction site like eBay and have experimented on it by several EJB implementations. Their test shows that the CPU on the EJB application server is the performance obstacle. In addition, the network is also saturated at some services.

The design of InfiniBand data centers is studied in [11]. It compares the performance between Socket Direct Protocols (SDP) and native sockets implementation over InfiniBand (IPoIB). This paper only uses the user-level communication in a data center without any intelligent distribution algorithm or architectural support for secure transactions.

3. SYSTEM MODEL

Many content-intensive applications have scaled beyond the point where a single server can provide adequate processing power. Both enterprises and service providers need the flexibility to deploy additional servers quickly and transparently to end-users. Server load balancing makes multiple servers appear as a single server – a sin-

gle virtual service – by transparently distributing user requests among the servers. The load of a server should be dispersal when the weight exceeds its limit. Today more number of applications is depending on a particular server. User level has been increased more than the expected weight of the server so that the load of the server may increase slightly. This situation needs a solution to balance the load of the server. Number of sub-servers may allocate to overcome this problem. The benefit of load balancing concept is to reduce the load of the server effectively by allocating some works to their sub or proxy servers.

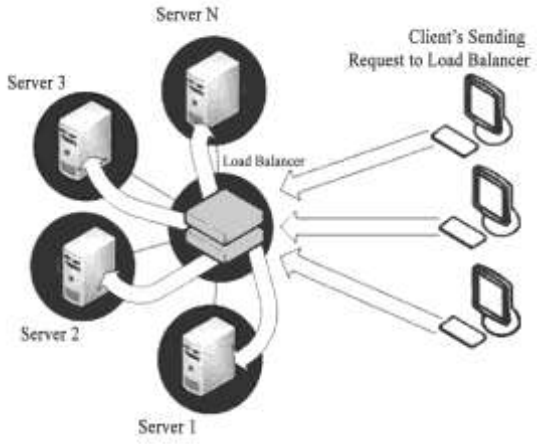

Figure-2 Outline of Load Balancing System

Figure-2 shows an outline of the Secure Socket Layer Server Load Balancing (SSL-LB) systems. The highest performance of a server is achieved when the processing power of servers are used intelligently. The other benefit of server load balancing is its ability to improve application availability. If an application or server fails, load balancing can automatically redistribute end-user service requests to other servers within a server farm or to servers in another location. Server load balancing also prevents planned outages for software or hardware maintenance from disrupting service to end-users. Distributed server load-balancing products can also provide disaster recovery services by redirecting service requests to a backup location when a catastrophic failure disables the primary site.

The proposed system has been created with the aim of increasing the performance and reducing the download time by allocating the sub-server concepts. This compares and measures the characteristics of different dynamic load balancing strategies for heavily and lightly loaded homogenous distributed systems. This system uses a SSL-LB model with non-pre-emptive task scheduling with the assumption that the task arrival is Poisson and service times are exponentially distributed. Furthermore, it considers a periodic broadcast type communication network for load information distribution. For each algorithm, the researcher analyzes the influence of different parameters on response time and on the system throughput. Later, the communication costs and the security parts are considered and their influence is discussed. For the security purpose, RSA algorithm has been implemented. The effective job allocation strategy based on the response time and performance of the sub-servers has been considered. The effectiveness of these algorithms is evaluated by simulations. Average response time and load imbalance are used as performance indices to measure the relative merits of the algorithms. In the end, the results obtained using these load balancing strategies are compared with those obtained in the absence of load balancing.

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 server provides an efficient way to distribute their work with the sub-servers which is also known as proxy servers. The SSL-LB load balancing algorithm introduces a new technique to balance the server load. The proposed technique gives an effective way to overcome the load balancing problem. The load balancing system is a set of substitute buffer to share the server load, when their load exceeds its limit. Allocating work to the sub-server based on their response time is the proposed technique. Storing and serving effectively and securely is more important so that desired algorithms going to implement for load distribution and security enhancement named as RSA and SSL-LB respectively. Calculating the response time of each request from the clients has been done by sending an empty packet over the networking to all the sub-servers. Response time for each sub-server is calculated using the Queuing theory. In this system, the SSL load distribution scheme has been introduced for effective performance.

Clusters are usually deployed to improve performance and availability provided by a single computer. Load

balancing is a technique used to spread out workload among many processes, computers, networks, disks or other resources, so that no single resource is overloaded. 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. 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.

4. RESULTS AND DISCUSSIONS

This section introduced the experiments performed on the overall implementation in order to measure its performance of SSL-LB. Two sets of experiments are used: determination of the performance strategy based on the task, and the efficiency of the scheduling period on the performance. The experimental results were averaged over 3 sub servers, and computed at a 90% interval. This experiment confirmed that the strong impact of SSL-LB load balancing on response time. The queuing model used in the SSL-LB algorithm is M/M/c. The exponential inter-arrival times with mean 1/λ, exponential service times with mean 1/μ and c parallel identical servers. Customers are served in order of arrival then the occupation rate per server,

$$\rho = \lambda / c\mu ,$$

is smaller than one.

In order to ensure consistency in the experiment results, the default values used in the experiments are given in the following Table-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-1. The usage of the servers in implementation process, the scheduling period, expected response time for each task and the expected performance measurement time are given in the above Table-1.

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

The proposed sub servers are tested on 3 test beds. The main server sends an empty pocket to the proxy servers Server1, Server2 and Server3. The Server1, Server2 and Server3 will receive this empty pocket and return back to the main server. The proxy server with lightly loaded will return the empty pocket quickly and the proxy server with heavily loaded will return the empty pocket slowly. The response time is calculated from the 3 servers and the best server will do the client request. Through the response time, the server could calculate the download speed and the task completion time. Table-2 shows the three proxy server's response times.

Server1, Server2 and Server3 will receive the empty pocket from the main server at the same time 16:52:14.0156250. The Server1 will return the empty pocket back to the main server 16:52:14.0312500. The

response time for server1 is 0.015625 ms. The Server2 will return the empty pocket back to the main server 16:52:14.0313700. The response time for server2 is 0.015745 ms. The Server3 will return the empty pocket back to the main server 16:52:14.0324600. The response time for server2 is 0.016835 ms.

Figure-3: Response time comparison for three different servers

The above Figure-3 shows Server1, Server2 and Server3 response times graphically. For each sub-server responded with a specified time period. Through that the best one will be selected. Server1 have the less response time compared to Server2 and Server3 and client request can be allocated to Server1 as shown in the Table-3.

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

The various measurements and comparative study defines the response time of the sub-server which was increased in the SSL-LB algorithm. The above result proves the SSL-LB load balancing and its performance issues.

5. CONCLUSION

Server load balancing is a powerful technique for improving application availability and performance in service provider, web content provider and enterprise networks. But implementation can also increase network cost and complexity. The key feature of server load balancing is its ability to direct service requests intelligently to the most appropriate server. The main aim of this research is to increase total throughput under an increased load when users are added. SSL-LB algorithm is used to distribute automatically (load balance) the client requests across a number of servers providing the same service to the client such as Internet Information Services (IIS).

REFERENCES

[1] Apostolopoulos G., V. Peris and D. Saha, 1999. Transport Layer Security: How Much Does It Really Cost?. Proc. INFOCOM, 2 : 717-725.

[2] Apostolopoulos G., D. Aubespin, V. Peris, P. Pradhan, and D. Saha, 2000. Design, Implementation and Performance of a Content-Based Switch. In Proceedings of the 2000 IEEE Computer and Communications Societies Conference on Computer Communications (INFOCOM-00), Los Alamitos, 3 : 1117–1126.

[3] Guitart J., D. Carrera, V. Beltran, J. Torres, and E. Ayuade, 2005. Session-Based Adaptive Overload Control for Secure Dynamic Web Applications. Proc. International Conference Parallel Processing (ICPP ’05), p. 341 – 349.

[4] Aron M., P. Druschel, and W. Zwaenepoel, 1999. Efficient Support for P-HTTP in Cluster-Based Web Servers. Proc. Usenix Ann. Technical Conf., p. 185-198.

[5] Carrera E.V., S. Rao, L. Iftode, and R. Bianchini, 2002. User-Level Communication in Cluster-Based Servers. Proc. Eighth International Symp. High-Performance Computer Architecture (HPCA ’02), p. 275-286.

[6] Kim J.H., G.S. Choi, D. Ersoz, and C.R. Das, 2004. Improving Response Time in Cluster-Based Web Servers through Co-scheduling. Proc. 18th International Parallel and Distributed Processing Symposium, p. 88-97.

[7] Zhou Y., A. Bilas, S. Jagannathan, C. Dubnicki, J.F. Philbin, and K.Li, 2002. Experiences with VIA Communication for Database Storage. Proc. 29th Ann. International Symp. Computer Architecture, p. 257-268.

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

[9] Cecchet E., J. Marguerite, and W. Zwaenepoel, 2002. Performance and Scalability of EJB Applications. Proc.

17th ACM SIGPLAN Conf. Object-Oriented Programming, Systems, Languages, and Applications. p. 246-261.

[10] Balaji P., S. Narravula, K. Vaidyanathan, S. Krishnamoorthy and D.K. Panda, 2004. Sockets Direct Procotol over InfiniBand in Clusters: Is It Beneficial?. Proc. IEEE International Symp. Performance Analysis of Systems and Software (ISPASS '04), p.28-35.

[11] Yingvu zhu, Yiming Hu, 2003. Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems. Peer-to-Peer Computing, Proceedings, Third International conference, 16(4) : 349-361.

[12] Byers J., J. Considine, and M. Mitzenmacher, 2003. Simple load balancing for Distributed Hash Tables. Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS), Berkeley, 2735: 80-87.

[13] Byers J., J. Considine, and M. Mitzenmacher, 2003. Simple load balancing for Distributed Hash Tables. Proceedings of International Workshop on Peer-To-Peer Systems (IPTPS), Berkeley, 2735: 80-87.

[14] Alonso R., 1986. The Design of Load Balancing Strategies for Distributed Systems. Future Directions in Computer Architecture and Software Workshop, Seabrook Island, SC, p. 1-6.

[15] Chi-Chung Hui, Samuel T.Chanson, 1999. Improved Strategies for Dynamic Load Balancing. IEEE Concurrency, 7(3) : 58-67.

[16] Yang C.S., M.Y. Luo, 2000. A content placement and management system for distributed Web-server systems. Proc. of IEEE 20th Int. Conf. on Distributed Computing Systems (ICDCS'2000), Taipei, Taiwan, p. 691-698.