International Research Journal of Interdisciplinary & Multidisciplinary Studies (IRJIMS)
A Peer-Reviewed Monthly Research Journal
ISSN: 2394-7969 (Online), ISSN: 2394-7950 (Print)
Volume-I, Issue-I, February 2015, Page No. 17-21
Published by: Scholar Publications, Karimganj, Assam, India, 788711
Website: http://www.irjims.com

Customized Automatic Load Sharing For a Network of Workstations

Dr.K.Kungumaraj
Asst. Professor, Dept. of Computer Science, APA College for Women, Palani
&
J.Vijayabharathi
Asst. Professor, Dept. of Computer Science, Mother Teresa Women's University Kodaikanal

Abstract

There are various Internet applications such as financial transactions, database access are always running on the Internet Server. Those applications must be able to run on multiple servers to accept an ever increasing number of users and networks need the ability to scale the performance to handle large volumes of client requests without creating unwanted delays. So, load sharing concept 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.

Keywords: Load Sharing, Domain Name Server, Distributed Systems.

I. Introduction: Over the years, as PCs have become more powerful and as larger computers have become capable of hosting web sites, the high-availability and continuous-operation features of server load balancers have become more prominent. Server load balancers have gained wide acceptance and popularity almost over the years, and their capabilities have evolved well beyond moniker's suggestion. It would be more correct to refer to today's robust load balancers as traffic managers, but even this description sells these products short. Today's leading server load balancers are hybrids of various products, incorporating a number of technologies into a streamlined network appliance. Robust load balancers are not only capable of balancing the load between multiple servers and ensuring availability, they offer some incorporate firewalls and proxy servers, and some have switching capabilities.

The commonly used technique for server location and load distribution is to use enhanced versions of Domain Name Service. A Domain Name Server (DNS) can resolve the same name to different IP addresses for the purpose of load distribution. Round Robin DNS and application layer any castings are examples of this scheme. The problem with these schemes is that they are not capable of determining the availability of a given server and continue to send client requests to failed servers. Since intermediate name servers cache the resolved name-to-IP-address mapping, changes in DNS information propagates slowly through the Internet. Even if a network administrator detects a failed server and removes its DNS records, the whole Internet world may not become aware of this fact for hours or possibly days. Consequently the failed server continues to receive requests, resulting in HTTP failures to end users. Caching of name-to-IP-address mapping by intermediate name servers also makes fine grain load distribution difficult. It is possible to time-out cached mappings quickly. However, that has several undesirable side effects. It increases the load on the DNS server and network traffic significantly. It also means that each (when caching is disabled) HTTP request to the web server has to be preceded by a DNS query to the name server.

The other major problem with DNS based schemes is that they failed to factor in network topology and load in making decisions. Consequently, a client may be directed to a far away server or to a server attached to congested network even when one close to it is available. For fine grain load

balancing in a server cluster, many popular Web sites use a connection routing front-end (also called connection router) that distributes connections to different server nodes. It maintains a view of the load on each server in order to forward new connections to the most appropriate server. One can combine a DNS based scheme with connection routing front-ends in order to get the best of both approaches. However, this does not solve the problem of locating the closest server for the requesting client. Additionally, since all connections to a server cluster have to pass through the connection router, it is a single point of failure and may become a bottleneck at high loads.

II. Related Works: Yuh-Ming Chin et al. [1] has proposed load balancing system which is distributing the resource effectively across the global networks. When the load distribution has implemented, the feasibility analysis must be considered. The server and the proxy servers must act quickly according to the client's requests. The bandwidth and the response time decide the efficiency of the load balancing system. The paper 'Minimizing File Download Time in stochastic peer-to peer networks' has involved in decrementing the download time. The peer-to-peer (P2P) file-sharing applications are becoming increasingly popular and account for more than 70% of the Internet's bandwidth usage. Measurement studies show that a typical download of a file can take from minutes up to several hours depending on the level of network congestion or the service capacity fluctuation. The author considers two major factors that have significant impact on average download time, namely, the spatial heterogeneity of service capacities in different source peers and the temporal fluctuation in service capacity of a single source peer. It points out that the common approach of analyzing the average download time based on average service capacity is fundamentally flawed. It rigorously proves that both spatial heterogeneity and temporal correlations in service capacity increase the average download time in P2P networks and then analyzes a simple, distributed algorithm to effectively remove these negative factors, thus minimizing the average download time. It shows through analysis and simulations that it outperforms most of other algorithms currently used in practice under various network configurations. The adoption of SSL (Secure Sockets Layer) to secure data across the Internet and World Wide Web is growing at a dramatic rate. However, SSL connection setup and processing can severely impair standard servers. There are several technologies available today to help offload servers from these computationally intensive tasks.

Linn Xia et al. [2] has proposed structured P2P load balancing systems, using DHT algorithms to distribute objects randomly among nodes results in unbalance load in each node. Numerous load balancing proposals exist in P2P networks, most of them only focus on namespace balancing and under the uniform assumption. In ideal condition, these techniques really work well, but ignoring the effect of heterogeneity and uneven in both of varying object loads and wide varying node capacity.

Jin-Ha Kim et al. [3] has proposed the session based scheme in cluster based web servers 'An SSL back end forwarding scheme in cluster-based web servers'. This paper represents the load distribution over the network with the effective SSL option. It also considers many security issues when the load distribution has made. It deals with dynamic content distribution compatibility.

Rahman. M.A [4] has proposed when the load is dynamic, certain load distribution systems fail to overcome the quick redistribution. Using DHT, the load distribution has been scheduled. Dynamic hash table maintains all details about the servers and sub-servers. Most basic DHT based peer-to-peer networks distribute objects among nodes in a way that tends to balance loads among the peers if the distribution of objects in the identifier space is uniform. The case is further complicated by the fact that, in a typical scenario, object loads and node capacities vary enormously. Additionally, a node load may vary greatly over time since the system can be expected to experience continuous insertions and deletions of objects, and continuous arrival and departure of nodes.

Ahmad Dalal'ah [5] has proposed a sliding strategy for load balancing. The strategy groups a certain number of adjacent nodes to perform a load balancing process. Upon the completion of a given period, the groups are to be rotated by shifting each group one position to the right, thus produces different groups. This strategy (sort of clustering) not only reduces the load balancing overheads, but also it could be utilized as a backbone by any load balancing strategy. The proposed load balancing strategy always converges, and tends to be in a steady state in a negligible processing time. In this paper, the load status and the locations of the nodes regarding the system's topology are irrelevant to load balancing process. The new algorithm can be always applied to any distributed system, even if it is heavily loaded, since the cost of scheduling is very low due to the highly reduced number of

messages. This is achieved by reducing dramatically the overheads incurred from attached information tables, message passing, job thrashing, and response time. The overheads stem from message passing and the scalability issues are among the main objectives of any load balancing system. To this aim, two methods of grouping the nodes are introduced, devised and tested. The first is to group the nodes in couples while the second one is to group the nodes into triples. The numerical results show that the overhead stem from computations is reduced dramatically in both methods. On the other hand, the number of messages are not any more an important issue, since it turns to be fixed with small number of messages as well as the utilization of the system is maximized.

III. Secure Socket Layer Based Back-End Forwarding Methodology: In a cluster-based data center or network server, all requests from clients to an application server are first passed to a distributor from a Web switch and then the distributor forwards each request to one of the application servers according to its distribution policy. The distribution in the application server should be done differently compared to the front-tier Web server in which a cache-aware distribution like Locality-Aware Request Distribution shows good performance. Especially due to the high overhead of the SSL protocol, the distributor in an application server should adopt a policy that minimizes the SSL overhead. Since the session reuse scheme, which is widely used in single Web servers, is very effective to reduce the SSL overhead, the researcher plan to exploit the session reuse scheme for the cluster-based application servers.

The distributor of SSL-Session algorithm maintains the client information to forward subsequent requests from the same client to the same application server. The advantage of the SSL-Session is that it avoids the unnecessary authentication and negotiation phase when a client tries to reconnect to the server. The disadvantage of this model is that it may cause load imbalance among the servers. If a server has clients whose requests are frequent and require much dynamic computation, the requests cannot be distributed to other lightly loaded servers, resulting in load skewness among the servers. The main drawback in the existing method is time latency. The server load could not be balanced due to unpredictable load of the proxy servers. The existing system fails to lookup path length and the number of heavy nodes encountered in each path. The metric of path length reflects the performance of the query forwarding scheme, and the metric of number of heavy nodes shows how the congestion control protocol avoids heavy nodes in direct traffic flow.

There are multiple servers installed in geographically located places and SSL-LB distributes the client's request to the appropriate server and reduces the pay load of the server. The SSL-LB has been modeled in this way only. The integrated Web and application server works as follows: The requests from client processes are distributed by the distributor to the servers. As soon as a server receives a request, it opens an SSL connection with the client. If the server has the client's previous session information, it skips the RSA asymmetric key operation. According to the request type (static or dynamic), the request is processed differently. If the request is for static content, it is serviced by the Web server module. Otherwise, it is forwarded to the application server module. The application server invokes the communication module to send the request to another node.

Representing main server and sub-servers is more important for modeling a load balancing system because the pre-assignment should be made before the navigation. Allocating servers helps to get and respond to the clients request quickly. This allocation will be done by the administrator of the system, who has the rights to represent the main server and the sub-servers. The module is to specify the IP addresses which are going to be used to identify the server and sub-servers. Assignment to a particular server might be based on a username, client IP address, or by random assignment. The server representation process has been made by getting IP address of the server and sub-servers and storing the IP address in the database by representing as proxy server.

IV. Results and Discussions: SSL-LB is an efficient load balancing algorithm in a distributed computer system. SSL-LB has been combining with RSA security algorithm to provide better security. Which differentiates the performance metrics and effective download between several existing approaches. The performance analysis considered by several parameters such as throughput, latency, coverage and security. While comparing these parameters with the existing one, the proposed system shows the efficiency by redirecting the client request to the available sub-server. The available

proxy's were filtered by the response time and the proxy server which responds quickly will be considered as a best server and the clients request allocated to that lightly loaded proxy server.

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

1) Latency
2) Throughput
3) Coverage
4) Security

1**) Latency:** In practice, hosts are added to a Network load balancing cluster which is 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.

2) 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). SSL-LB Load Balancing delivers incoming packets to the desired host, which results in lower response time and higher overall throughput.

SSL-LB Load Balancing scales performance by increasing throughput and minimizing response time to clients. When the capacity of a cluster host is reached, it cannot deliver additional throughput, and response time grows non-linearly as clients awaiting service encounter queuing delays. Adding another cluster host enables throughput to continue to climb and reduces queuing delays, which minimizes response time. As customer demand, throughput continues to increase; more hosts are added until the network's subnet becomes saturated. At that point, throughput can be further scaled by using multiple networks load balancing clusters and distributing traffic to them.

3) Coverage: Dealing the client requests efficiently even the server load capacity exceeds is more important for every load balancing scheme. But in the existing methods RR and SSL-Session are considering only a limited set of client request. The proposed SSL-LB algorithm provides better coverage compared to the existing methods.

4) Security: Sharing the files in the network makes every file available in the main server and the sub-servers, so that the main server and the sub-servers 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 using RSA security algorithm, so that security is higher than the existing schemes.

Table-1 shows the comparative measurement (in percentage) of three different algorithms. SSL-LB algorithms has less latency time, high throughput, better coverage and more security than the other two existing algorithms.

Metrics	SSL-LB (%) (Proposed)	SSL-Session (%)	Round Robin(%)
Latency	24	43	48
Throughput	44	25	20
Coverage	86	59	65
Security	87	67	50

Table-1: Comparative measurement of three different algorithms

Figure-1: Performance comparison of three different algorithms

Figure-1 shows impact and efficiency of the proposed SSL-LB algorithm. The proposed system model shows the performance advantages between the existing system models. The result defines the impact and efficiency of the proposed system.

V. Conclusion: The quality of load sharing is statistically determined by the number of client's requests and the client population fluctuates. The evenness of load balancing can be observed to vary slightly over time. The requests from various clients are gathered in the server side and the sub-

servers' load will be calculated using the SSL-LB algorithm. Depending on the response time of the sub-servers, the client request will be navigated to the main server or the particular sub-server. SSL-LB load sharing is a very simple load balancing approach but powerful load-balancing algorithm that delivers the highest possible performance and availability.

References:

1. Yuh-Ming Chin, Do Young Eun, 2008. Minimizing File Download Time in Stochastic Peer-to-Peer Networks. IEEE / ACM, 16(2) : 253-266.
2. Lin Xia, Han-Cong duan, Xu Zhou, Zhifeng Zhao, Xia-Wen Nie, 2010. Hetrogeneity and load balancing in structured P2P systems. International conference on communications, Circuits and Systems, p. 245-248.
3. Jin-Ha Kim, 2007. An SSL Back-End Forwarding Scheme in Cluster-Based Web Servers. IEEE transactions on parallel and distributed systems, 18(7):946-957.
4. Rahman. M.A, 2008. Load balancing in DHT based p2p networks. Electrical and Computer Engineering, International conference on Electrical and Computer Engineering ICECE-2008, p. 920-923.
5. Ahmad Dalal'ah, 2006. A Dynamic sliding load balancing strategy in distributed systems. The International Arab Journal of Information Technology, 3(2) : 178-182.
6. Al-Dahoud Ali, Mohammed A. Belal and Mohammed Belal Al-Zoubi, 2010. Load balancing of distributed systems based on multiple ant colonies optimization. American Journal of Applied Sciences, 7(3) : 428-433.
7. Khan Z., R.Singh, J.Alam and R.Kumar, 2010. Performance Analysis of dynamic load balancing techniques for parallel and distributed systems. International Journal of Computer and Network Security, 2(2):123-127.
8. 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.
9. 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.
10. 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.

*****