2009 10th International Symposium on Pervasive Systems, Algorithms, and Networks

Random Early Detection Web Servers for Dynamic Load Balancing

Chih-Chiang Yang, Chien Chen, and Jing-Ying Chen
Department of Computer Science
National Chiao Tung University, Hsiu-Chu, 30010, Taiwan
ycc.cs95g@nctu.edu.tw, chienchen@cs.nctu.edu.tw

Abstract- *Modern Web-server systems use multiple servers to handle an increased user demand. Such systems need effective methods to spread the load among web servers evenly in order to keep web server utilization high while providing sufficient quality of service for end users. In conventional DNS-based load balancing architecture, a Doman Name Server (DNS) dispatches requests to web servers based on their load status. Because web servers need to inform the DNS server about their load status from time to time, a so-called load buffer range is often employed to reduce the update frequency. Without care, however, using a load buffer range may result in load oscillation among web servers. To address this problem, we propose a Random Early Detection (RED) method with the intuition that the probability for a web server to become overloaded in near future is directly proportional to its current load. Simulation confirms that our method helps reducing the oscillation of the web server load significantly.*

Keywords: DNS-based, load buffer range, RED.

1. Introduction

In recent years, the number of people using Internet services has grown dramatically due to the rapid development of the Internet. To cope with the increasing user demand, it becomes a common practice nowadays to use multiple web servers to process user requests in parallel. However, if the user requests cannot be spread among web servers evenly such that some servers become overloaded while the others remain idle, the overall web servers' utilization will be dropped, resulting in poor and unstable quality of service for the whole system.

This uneven server load problem has been addressed by many researchers over the years. [1] classifies existing load balancing architecture into four classes, namely client-based, dispatcher-based [5], DNS-based [7], and server-based [3][4][8] load balancing architecture. In this paper we focus on the DSN-based load balancing architecture. In such architecture, web servers are usually placed in geographically decentralized areas, and a Doman Name Server (DNS) acts as a request dispatcher that dispatches requests to web servers. The advantage of this approach is that by considering the geographical relation between a client and each web server, the DNS can assign a web server with lower propagation delay to that client to provide better quality of service. In order to achieve load balancing, the DNS typically uses Round Robin scheduling to map different clients to different web servers in a logical cluster. [18] showed that the classic algorithms, such as Round Robin, are not adequate for the DNS scheduler. To improve the load imbalance issue, [7] proposed an adaptive time-to-live (TTL) policy in DNS-based architecture which assigns a different TTL value to each address based on client request rates. To resolve the issue of uneven domain load distribution, requests coming from popular domains will receive a lower TTL. In a dynamic environment, the algorithms using the detailed load information from the servers can achieve better load balancing, but at the cost of extra computation and communication. Unlike a traditional parallel/distributed system, web servers are geographically distributed, and the DNS cannot obtain their states too often to avoid network congestion and bandwidth waste. Therefore, a method that uses asynchronous feedback alarms and requires only limited state information from the overloaded servers had been proved more effective than those that use periodic feedback information from every server to make scheduling decision [18].

A conventional asynchronous feedback method for DNS-based load balancing architecture often sets a so-called *load buffer range* with low and high thresholds to decrease the state change frequency of a web server. If the load of a web server exceeds the high threshold, an overload alarm signal will be sent back to the DNS. DNS will then exclude this web server from further assignment of new requests. This web server will remain in an overloaded state until its utilization drops under the low threshold, then another asynchronous message will be sent to the DNS. The DNS will resume assigning the requests to this web server. Without care, however, setting the load buffer range improperly may result in load oscillation among web servers. To address this problem, we propose a random early detection (RED) method with the intuition that the probability for a web server to become overloaded in the near future is directly proportional to its current load. In our

978-0-7695-3908-9/09 $26.00 © 2009 IEEE
DOI 10.1109/I-SPAN.2009.44

simulation, we show that the oscillation of the web server's load can be reduced by using the concept of RED in the geographically distributed load balancing architecture.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 provides an overview of DNS-based load balancing methods. Section 4 illustrates a conventional load buffer range methods. And section 5 presets our random early detection method. Section 6 shows our simulation results. Conclusions and future work are given in Section 7.

2. Related Work

[2] discusses different load balancing techniques for Web-server systems and evaluates the performance of four specific load balancing schemes, that is, round-robin, connections, round-trip, and xmitbyte.

In this paper we are concerned with DNS-based load balancing architecture. The basic concept and operation of DNS are given in [12][13][14][15], and the common TTL value setting for DNS is discussed in [14][16][17] When load balancing is concerned, [7] proposes to assign a different TTL value to each address request by taking into account the capacity of the selected server and/or the request rate of the source domain of the request; [10] suggests that careful consideration is necessary when choosing DNS TTL values to balance responsiveness against extra client latency. In case the web servers are geographically distributed, [6] divides web servers into zones and considers the load of servers in each zone as well as the cost of transferring a job across zones to determine if there is benefit of executing a job across zones. Similarly, [3] considers the cost of transferring a job from one server to another to determine whether to execute a job in a local or remote server.

[11] presents Random Early Detection (RED) gateways for congestion avoidance in packet-switched networks, avoiding the global synchronization that results from many connections decreasing their windows at the same time.

3. DNS-based load balancing Architecture

The DNS-based load balancing architecture is illustrated in Fig. 1, in which clients are partitioned into several groups according to the local DNS (LDNS) servers they use, respectively. When a client wants to obtain a service from a web server with a particular domain name, he/she first sends the domain name resolution query to the LDNS server. After receiving a domain name resolution query, the LDNS server

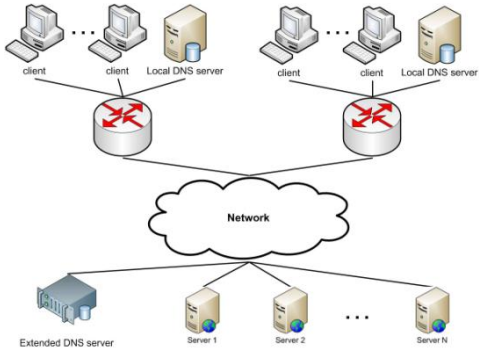

Fig. 1. DNS-based load balancing architecture

first checks to see whether there is a valid and unexpired IP address of that domain name. If so, the LDNS server sends the IP address to the client directly. Otherwise, the LDNS server would ask the root DNS server for the IP address of a DNS server (the Extended DNS server in Fig. 1 also called EDNS server) that is responsible for resolving that domain name; the LDNS server then forwards the domain name resolution query to the EDNS server to obtain a new mapping IP address and its associated TTL time. Finally, the LDNS server sends the new IP address to the client, and records the TTL time of this IP address. Before the TTL time expires, each domain name resolution query for the same domain name can be directly sent by the LDNS server without asking the EDNS server again.

The characteristics of DNS-based load balancing architecture are as follows:

- All service servers can be placed in a geographically distributed area.
- There is no direct geographical relationship between DNS server and service web servers.

In such architecture, one can exploit the geographical relationship between web servers and clients to minimize the query propagation delay for clients. Moreover, because of the existing mature master/slave architecture of DNS, slave DNS servers may periodically backup the data of the master server, and assist in apportioning the domain name resolution queries of the master DNS server. If the master DNS server fails, one of the slave DNS servers can take over the subsequent work for the master DNS server, therefore achieving high reliability.

On the other hand, in typical DNS architecture there is usually little or no information exchanged between the DNS server and web servers. Accordingly, conventional DNS-based load balancing methods usually use a random or round robin approach to perform simple load balancing; they are more likely to

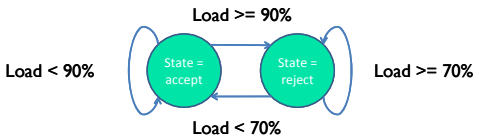

Fig. 2. State transition diagram of conventional load buffer range method

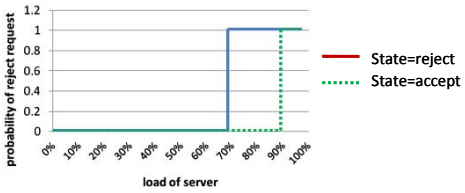

Fig. 3. State change of conventional load buffer range method

cause unbalanced load distribution among web servers. Therefore, we are motivated to consider how to use infrequent server state information to achieve a higher degree of load balancing among web servers.

4. Load Buffer Range Method

In DNS-based load balancing method, the DNS server distributes the load among servers in a round-robin manner, and the service server periodically sends its load status to the DNS server. Based on the load data collected from the web servers, the DNS server can skip the overloaded ones when dispatching requests. As previously mentioned, there is usually no direct geographical relationship between the DNS server and web servers, the web server should not send its state information to the DNS server too often in order to avoid congesting the network or wasting network bandwidth. For this reason, a conventional method usually defines a load buffer range (LBR) with low and high thresholds for each web server. The state transition diagram of the LBR example is shown in Fig. 2. As the example shows, before the load of a web server exceeds 90% (high threshold), the server is not overloaded. That is, the DNS server can assign new client requests to that web server. Once the load of that web server is greater than 90%, it enters into the overloaded state. A web server in overloaded state notifies the DNS server not to assign new client requests to that web server until its utilization return under 70% (low threshold). Fig. 3 shows the probability of the overloaded state against to the server load.

In this method, when there are not many service servers and the amount of requests is high, once one of the service servers is overloaded, it must keep its overloaded state until its load is under 70% and then notify DNS server to assign new client requests to that web server. During this period, the other web servers may need to share the additional 20% (90%-70%) load from that overloaded server. This may in turn cause other web servers to become overloaded, and so on, resulting in unstable service quality.

5. Random Early Detection Method

In order to solve the load oscillation phenomenon of web servers mentioned previously, we consider that the state of overload or under-load of a web server in the load buffer range should be a probability rather than definite, in order to avoid burdening the other web servers with too much load. Hence, we use the concept of random early detection (RED) method to determine the overload status of web servers probabilistically.

The RED idea is first presented in [11] for congestion avoidance in packet-switched networks. When the average queue size exceeds a preset threshold, the gateway drops or marks each arriving packet with a certain probability, where the probability is a function of the average queue length. It puts emphasis on avoiding the TCP global synchronization that results from each connection reduces the window to one and goes through Slow-Start in response to a dropped packet at the same time.

In [11], the RED gateway calculates the average queue size, which is compared to a minimum and a maximum threshold. When the average queue size is less than the minimum threshold, no packets are dropped. When the average queue size is greater than the maximum threshold, every arriving packet is dropped. When the average queue size is between the minimum and maximum thresholds, each arriving packet is dropped with probability p_a, where p_a is a function of the average queue length.

Applying the RED idea here in the context of DNS-based load balancing, the probability of a web server becoming overloaded is directly proportional to its current load. A line chart example of the probability of a web server becoming overloaded is shown in Fig 4. In this

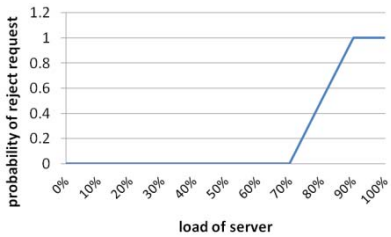

Fig. 4. State change of RED method

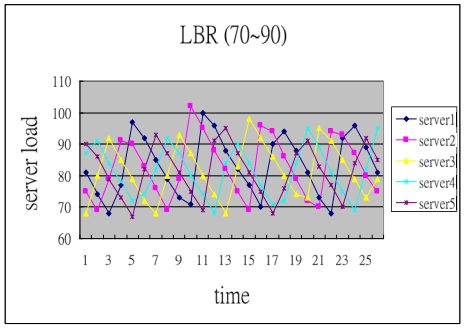

Fig. 5. Server load oscillation phenomenon of conventional load buffer range method

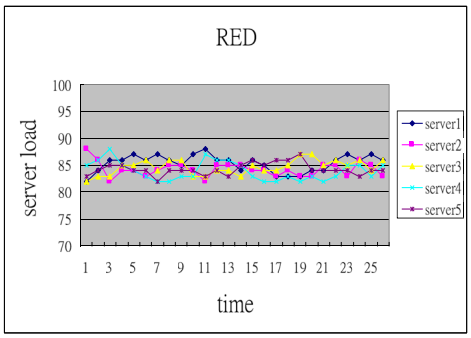

Fig. 6. Server load variation of RED method

example, the minimum threshold is 70% and the maximum threshold is 90%. When the load of a service server is less than 70%, its state should be under-load. When the load of a service server is greater than 90%, its state would be overloaded. Finally, when the load of a service server is between 70% and 90%, the probability of its state becoming overloaded is proportional to its current load.

6. Simulation

In our simulation, five web servers are placed in a geographically distributed area, and we control the overall amount of workload to about 85% of total server capacity.

The load oscillation phenomenon of the conventional load buffer range method is shown in Fig. 5. As we can see, the loads of those five servers increase to greater than 90% and then decrease to less than 70% by turns. As shown in Fig. 6, compared with conventional load buffer range method, using our RED method to probabilistically determine the state of the web servers can effectively raise the load balancing degree among web servers and smooth the load variation of each web server in the same request traffic, providing more stable Internet services.

We then decrease the load buffer range of conventional methods in order to observe the relationship between the range of load buffer and

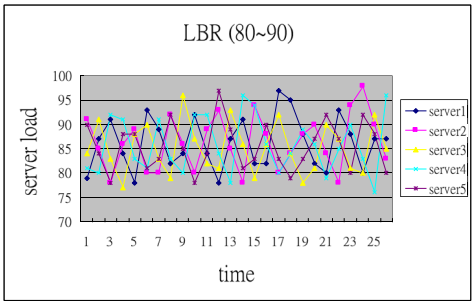

Fig. 7. Server load variation of load buffer range between 80% and 90%

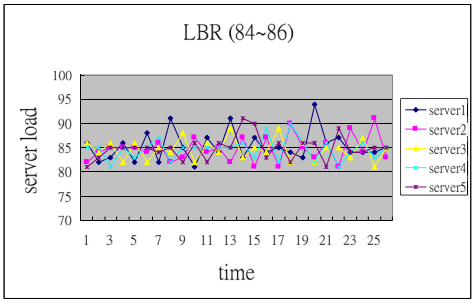

Fig. 8. Server load variation of load buffer range between 84% and 86%

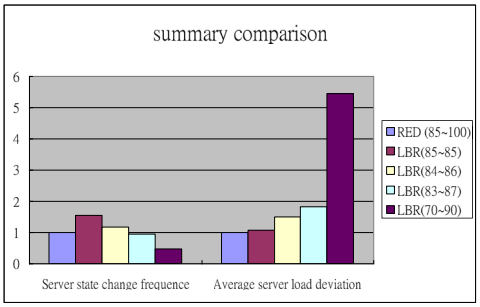

Fig. 9. Summary comparison RED and LBR methods

the standard deviation of server load. Fig. 7 and Fig. 8 are load buffer ranges from 80% to 90% and from 84% to 86% respectively. As we can see, reducing the load buffer range of the conventional method can diminish the degree of server load oscillation.

Finally, we should make a summary comparison of the server state change frequency and the average standard deviation of server load for RED method and the conventional methods with different load buffer range setting. Notice that the state change frequency indicates the asynchronous state messages sent by the web servers to the DNS. As shown in Fig. 9, even if we constantly reduce the load buffer range of the conventional method until it is zero, its load balancing degree will become closer to but still be slightly higher than the RED method's, and its server state change frequency has became 1.5

times of the RED method's at this time. Moreover, if we use the default setting (70%~90%) of the conventional method, although its sever state change frequency is half of the RED method's, but its average standard deviation of server load is greater than five times of the RED method's at this time.

7. Conclusion and future work

Because web servers can be placed in geographically decentralized area in DNS-based load balancing architecture, the states of web servers are not allowed to be obtained immediately to avoid congesting or wasting network bandwidth.

Compared with conventional two thresholds scheme, our RED method can use an acceptable server state change frequency to efficiently reduce the average standard deviation of web servers load to 1/5 of the conventional method's, smooth the load variation of web servers, and provide more stable quality of services. Moreover, in our simulation, no matter what we set the load buffer range of the conventional method to, its load balancing degree is still worse than our RED method's.

In the future, we will analyze the effect of different RED settings in order to invent an adaptive RED method which can depend on the request traffic to adjust the RED setting to achieve better load balancing.

References

[1] V. Cardellini, M. Colajanni, P.S. Yu, "Dynamic load balancing on Web-server systems," *IEEE Internet Computing*, Vol.3, No. 3, pp. 28-39, May-June 1999.

[2] H. Bryhni, E. Klovning, O. Kure, "A Comparison of Load Balancing Techniques for Scalable Web Servers," *IEEE Network*, Vol. 14 Issue 4, pp. 58-64, July-Aug 2000.

[3] M. Arora, S. K. Das, R. Biswas, "A De-centralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments," In Proceedings of Workshop on Scheduling and Resource Management for Cluster Computing, pp. 499-505, Vancouver, Canada, Aug 2002.

[4] W. Leinberger, G. Karypis, V. Kumar, R. Biswas. "Load balancing across near-homogeneous multi-resource servers," In Proceedings of the Ninth Heterogeneous Computing Workshop, pp. 61-70, Cancun, Mexico, May 2000.

[5] William Leinberger, George Karypis, Vipin Kumar, "Job Scheduling in the presence of Multiple Resource Requirements," In Proceedings of ACM/IEEE Conference on Supercomputing (SC '99), pp. 47-47, Portland, USA, Nov. 1999.

[6] Z. Zhang, W. Fan, "Web Server Load Balancing: A queuing analysis, " *European Journal of Operational Research*, 186(2), pp. 681-693, Feb 2008.

[7] M. Colajanni, P. S. Yu, V. Cardellini, "Dynamic Load Balancing in Geographically Distributed Heterogeneous Web Servers," In Proceedings of Int'l Conf. Distributed Computing Systems, pp. 295-302, May 1998.

[8] M. Aramudhan, and V. R. Uthariaraj, "LDMA: Load Balancing Using Decentralized Decision Making Mobile Agents", *Lecturer Notes on Computer Science*, Part IV, Vol. 3994, pp. 388-395, Springer-Verlag, 2006.

[9] M. Harchol-Balter. "Task Assignment with Unknown Duration." *Journal of the ACM*, 49(2) pp. 260–288, 2002.

[10] A. Shaikh, R. Tewari, and M. Agrawal, "On the Effectiveness of DNS-based Server Selection," In Proceedings of IEEE INFOCOM 2001, pp. 1801-1910, Anchorage, USA, April 2001.

[11] S. Floyd, and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Trans. On Networking*, Vol. 1, No. 4, pp. 397-413, August 1993.

[12] P. Mockapetris, "Domain Names – Concepts and Facilities," Internet Request for Comments 1034, Nov 1987.

[13] P. Albitz, and C. Liu, "DNS and BIND," *O'Reilly and Associates*, 1998.

[14] D. Barr, "Common DNS operational and configuration errors, " Internet Request for Comments 1912, Feb 1996.

[15] P. Cockapetris, "Domain names – implementation and specification," Internet Request for Comments 1035, Nov 1987.

[16] A. Kumar, J. Postel, C. Neuman, P. Danzig, S. Miller, "Common DNS implementation errors and suggested fixed," Internet Request for Comments 1536, Oct 1993.

[17] E. Cohen, H. Kaplan, "Proactive caching of DNS records: Addressing a performance bottleneck," In Proceedings of 2001 Symposium on Applications and the Internet, pp. 85-94, San Diego, USA, Jan. 2001.

[18] M. Colajanni, P.S. Yu, D.M. Dias, "Scheduling algorithms for distributed Web servers," In Proceedings of 17th IEEE International Conference on Distributed Computing Systems (ICDCS '97), pp. 169-176, Baltimore, USA, May1997.