Model of Load Distribution Between Web Proxy Servers Using Network Traffic Analysis

B. S. Renuka¹ · G. T. Prafulla Shashikiran¹

Published online: 28 March 2020
© The Author(s) 2020

Abstract

Proxy servers are installed to give information access services to requested client nodes or units individually. As of now, in the present framework intermediaries are confronting the performance bottleneck that is when numerous clients get to the Internet all the while since such servers have a constrained limit. Web servers often experience and face overload situations due to the extremely bursty nature of traffic caused on the Internet. Our concentration in the work is to propose successful load control instruments for web servers. A significant angle in the load control is to limit the work spent by the proxy server to distribute the load and pass on to the main servers. This research work proposes a technique for load balancing on proxy servers by investigating the load on the network between servers and proxy servers. The research investigation is based on calculating the bits per second movements of packet and sockets with respect to the ports such as HTTP/HTTPS and TCP. This proposed method increases the performance by reducing the load on the proxy servers.

Keywords Load balancing · Proxy servers · HTTP web protocol

Introduction

Load balancing is a process of distributing the overall load to the all requesting nodes or clients in the server system where current technologies are using as shown in Fig. 1. It is necessary to know the resource utilization and response time upon all rendering and network issues. Distribution of load in the cloud depends on the load balancing algorithm. One has to consider the network traffic and transmission of data through the network channel as well as resource distribution, which is otherwise called as the load distributing technique. A client that associates with the intermediary server, mentioning some administration, for example a document or module, site page or other asset accessible from an alternate server

that is proxy servers and the intermediary server assesses the solicitation as an approach to streamline and control its unpredictability [1].

Distribution of load optimally is the idea of initiating any load distribution algorithm, which includes maximum throughput, low response time and low overhead. These are the three main criteria while developing the load distribution algorithm. Load balancer on the proxy servers acts as an interface between the server and the client. Load balancing is mainly used for simplification, which means that when a client requests for some web page it routes to any server using any one of the algorithm, and hence, it acts as the single point of contact between the clients and the servers. Abstraction, failovers, responsiveness, error reporting, seamless recovery (if any one server goes down) scalability and re-usability through TCP multiplexing are its major advantages.

As a general rule, a self-arranging intermediary engineering dependent on self-ruling intermediaries can be contrasted with a straightforward market purchaser dealer condition, where the genuine purchaser goes about as a client who dependably pick a similar shop for every one of their solicitations (like pre-characterized intermediaries in internet browsers), yet the expansion of the market depends totally on the venders (independent proxies), and each shop

This article is part of the topical collection “Advances in Computational Intelligence, Paradigms and Applications” guest edited by Young Lee and S. Meenakshi Sundaram.

✉ B. S. Renuka
renuka@sjce.ac.in

G. T. Prafulla Shashikiran
prafulla9530@gmail.com

¹ Electronics and Communication Department, Sri Jayachamarajendra College of Engineering, JSS Science and Technology University, Mysuru, India

Fig. 1 Representation model of reverse proxy servers

has a constrained nearby stock like reserve, and the objective is to augment the client. A decent administration can be provided in two different ways that is either by having the mentioned thing in the nearby stock or by realizing the most appropriate approach to supply the thing. Also, every intermediary endeavors to pull in more demands (different intermediaries) by practicing on a specific classification of things (bunching). This choice is typically made dependent on the present specialization of the shop and the approaching solicitation design. The above merchant purchaser situation in a dynamic market is a very thing similarity for the objective to expand the hit rate for intermediary demands in a dispersed independent intermediary framework condition. Subsequently, expecting there is a successful method to look at and classify approaching. With regard to hashing calculations, a basic modulo work, for instance, can characterize closeness, over the mentioned URL. Be that as it may, such a basic arrangement is deficient [2].

were resolved locally by the cooperative proxy system. A high average hit rate means the system is capable of transferring the most needed web objects closer to the clients and average hop rate and the number of hops that are needed to resolve a request. Time taken to resolve the hop rate is when the average hop rate is lower. The action of forwarding a request between client/proxy and proxy/proxy and proxy/server constitutes a hop.

So here in this work, the aim is to reduce the load on the proxy servers by eliminating the running time of the load distribution algorithm by analyzing the network traffic of the channel in and around the proxy servers. In this work, we use a methodology called reverse proxy servers where incoming requests are taken care of by the intermediary, which cooperates in the interest of the customer with the service living on the server. The most well-known utilization of a turnaround intermediary is to give load adjusting to web applications and APIs. Web Proxy Server Service and Requests per Sec, this counter esteem speaks to the quantity of solicitations every second the intermediary server is assessing for the benefit of clients. This counter shows the work being done on the intermediary server, and qualities can shift generally

dependent on the kinds of need that clients demand. Web Security and Acceleration Server has numerous roots in proxy server [3].

Techniques Used in Load Distribution Algorithm

Among all the variety of algorithmic techniques like randomized distribution, least count, etc., round robin is the regular methodology used by the number of industries. The logic behind round robin is that for k number of servers and i number of requests it works in the order of $i \text{ Mod } k$, because for all $k + 1$ requests it goes to first server. The main redevelopment to be considered while following the approach is difficulty in handling the differing latencies, which actually affects the fallback on responsiveness. To overcome this [4] has come out with reduced latency with lognormal distribution and also while deriving this log normal is that latency cannot be zero at the same time.

HTTP/HTTPS request over TCP socket: The communication has to take place between a proxy server and the main server. So, the type of communication that usually takes place is through HTTP/HTTPS request over TCP socket. By investigating the size of the content in the http request header and analyzing the number of bps (bits per seconds) of data, the communication link is used. This part is elongated in the proposed version where the most part of it is used.

Related Work

The websocket handshake takes the advantage of the HTTP protocol in the establishing phase session [5]. The websocket upgrade request is a regular HTTP GET request with session endpoint specified as a request URI component as shown in Fig. 2 [6].

On long-running sessions, the impact of the initial handshake is diminished and the only observable traffic overhead is caused by the websocket data framing. When the given amount of payload data is transferred using the plain TCP protocol, these data are directly embedded as TCP payload. However, when transferred as a websocket frame, the TCP payload consists of both payload data and a websocket frame header [5]. The workload at each proxy server is estimated from the number of log records. For measuring the performance of the work conducting a set of experiments in the real environment and use a dataset simulated from the real traffic obtained from our university's proxy logs [7]. There are also many load balancing techniques used in several industries such as failover load balancing, optimal load balancing, proportional load balancing, saturation load balancing and search filter load balancing [8].

Fig. 2 Websocket over TCP sequence diagram

At the point when an ordinary HTTP request is made through a HTTP intermediary, practically these upcoming advances are influenced. Figure 2 demonstrates the correspondence example channel of a client making a proxied demand for www.abc.com. The association time is an ideal opportunity to interface with the intermediary server, and the following stage is for the customer to send the HTTP request to the intermediary, for example, the DNS time is an ideal opportunity to determine the proxy's hostname, as opposed to www.abc.com, and after that the intermediary needs to determine www.abc.com to an IP address, open an association with this IP address and forward the customers' HTTP request to the web server. The intermediary then advances the information back to the customer as it gets it. Generally, an expansion in holdup time would demonstrate that a web server did not have the assets to serve demands rapidly enough. Sadly, with an intermediary it is difficult to recognize this sort of issue from a moderate DNS supplier or lossy system prompting expanded increased connection times [3], yet from the customers' point of view, the first DNS time and interface time have now been gobbled up in the holdup, time which is never again conceivable to gauge these qualities autonomously (Fig. 3).

The distributed approach depends on a hashing calculation like the Cache Array Routing Protocol (CARP). The requested page is mapped to one intermediary in the intermediary exhibiting in a hashing framework and will either be settled by the nearby store or mentioned from the origin server. Hashing-based portions can be broadly observed as the perfect method to store website pages, and their area is pre-defined. Their real downside is indexability and poor adaptability [9]. The presentation of the framework under various sorts of load such as I/O, CPU, MEMORY is

Fig. 3 Proxied HTTP request breakdown

dependent on IOCM dynamic load adjusting calculation in a heterogeneous figuring framework. There are a number of unique burden adjusting strategies for group frameworks; their productivity relies upon topology of the correspondence arrangement that associates hubs. This result has built up as an effective load adjusting for I/O-, CPU- and MEMORY-concentrated tasks [10]. At the point when there is working framework conditions squid intermediary of linux stage show preferable execution over windows Average reaction time smallest time access is via a Linux server that is 21.9; 29.9; 20.9;48.2; 28.3 s by using Mozilla Firefox and 28.9; 25.3; 25.4; 32.9; 24.5 s using Internet Explorer [11].

Existing cooperative proxy systems can be organized in hierarchical and distributed. The hierarchical approach is based on the Internet Caching Protocol (ICP) with a xed hierarchy. A page not in the local cache of a proxy server is rst requested from neighboring proxies on the same hierarchy level. Root proxy in the hierarchy will be queried if requests are not resolved locally, and they continue to climb the hierarchy until the request objects are found. This often leads to a bottleneck situation at the main root server [12]. A TCP client simply ensures that a socket can be opened. You can configure the HTTP client to submit a valid HTTP request to the backend service. You can define HTTP GET, PUT, POST or DELETE operations. The response of the HTTP monitor call must match the configured settings.

Proposed Method

This methodology work is comprised of reducing the load on the proxy servers. Load means the actual algorithm such as round robin and other task scheduling algorithms that always run on the proxy server.

Multiple nodes are connected to multiple proxy servers as shown in Fig. 4. The information requested by each of the clients is unique, and if the requested information is not present in the cache of the proxy servers, then the communication between the proxy servers and the core servers plays a part. This is the area where our proposed method lies.

By considering the m balls and n bins as discussed in the above section and implementing in our scenario gives the probability of m hits on n servers. Let us consider it for m clients and n servers. Considering the below equation, we can derive it as follows. Let Y be the number of servers and X be the number of clients.

$$i, \text{ otherwise} \quad (1)$$

$$0, \forall \sum$$

Let $X_i = 1$, if server service is low key.

Each hit of request from the client on the proxy server is unknown. By taking the $E(x)$ for both the possible conditions (Fig. 5).

$$E(x) = E[X_i] = n(1 - 1/n)m \sum_{i=1}^n \quad (2)$$

If $m = n$, it practically does not exist. The client hit on the server can be mathematically determined as follows:

$$\text{if } n = m, E[X] = n(1 - 1/n)n$$

The above equation is the outcome of taking the probability function of X for both the cases in Eq. (1). At the mean time, it is also necessary to look at the value $(1 - 1/n)$ that

should not decrease as it increases the value of m otherwise. This equation suits in the case if the load distribution algorithm is randomized where client hits the proxy servers or proxy servers hit the core server.

Communication Between Proxy Server and Main Server

In this subsection, the covered part is the network communication between the main server and the proxy servers. When the request is arrived at a proxy server and if requested data, for example, from node 1 is not determined by the proxy server itself, then the request is transformed to the main server through TCP/IP HTTP socket where internally calculation takes place by examining the payload length of the HTTP header, which gives the exact size of the socket that is carried through TCP/IP. By measuring the number of bits per second (bps) of the communication network that serves from the main server with the particular proxy so as the load of the proxy is detected for the algorithm to run on the proxy server. The main idea of reducing the load on the proxy server is by initializing the algorithm to run on the proxy server when required, that is, only when too much of nodes are attached to particular proxy servers. Figure 6 [1] defines the same. Hence, the steps to conduct the proposed procedure are as follows:

Step 1: When client requests an information or raw web data, the request is sent to proxy server, which initially

Fig. 4 Network packet structure during the websocket session

a) Network packet structure during the *Websocket* handshake

Fig. 5 The flow of distribution of workload on various nodes

Fig. 6 Entities of network channel

examines the requested data or information availability in the proxy server itself in the form of cached data. Two things arise here: (a) if requested data are present in the proxy servers, the information is sent back to the client through the HTTP web protocol with status value of 200 and (b) if requested data are not present in the proxy server, the following tasks have to be implemented.

Step 2: From the above step, if the option is (b), then we investigate the size of the HTTP payload length on each communicating network between the proxy server and the main server. Product of payload length and network connectivity, which will be in bits per seconds, are calculated to get the value of strength of the network. So, to calculate the High Speed Packet Access (HSPA) the network between the proxy and the main server will be:

$$H(t) = \text{pay load length on HTTP port} \times \text{HSPA in bps} \quad (3)$$

Step 3: Every proxy server has a load balancer implicitly or explicitly running a task scheduling algorithm in it such as round robin and least count scheduling algorithm. Check for a condition with respect to the previous step whether the traffic caused in the channel is more than the given threshold value. If yes then start running the load distribution algorithm on the proxy servers, which eventually reduces the load and increases the performance of the proxy servers. To go more in detail with the above equation, below the packets captured the scenario for about 20 min as shown in Fig. 6 [7]. The packet length of high proximity is about to occur 40% in the overall scenario when compared to other packet lengths. From Eq. (3) HSPA is 8200 by considering the average overall count. $H(t) = 32.8\%$, which means that the 32% of bandwidth is absorbed by this particular server channel, and hence, with the lower generating keys around 32% is sent to the particular server.

Analysis and Discussion

The packet length is responsible for the lag in the network channel as it gives the payload length in the HTTP header, and hence, the maximum data transmission rate, which is called as burst rate, increases eventually for the maximum limit as shown in Fig. 6.

The number of point in contact by the servers with the other servers increases the tcp.errorlog due to the increase in number of three-way handshake with the each server and its network. Time To Live (TTL) is also taken care by not letting the connection to happen out of its maximum range. Figure 7 shows the increase in tcp.errorlog when the reachability by routing with other servers and the load is increased.

Network time out $H(t)$ increases when the servers are intervened with other proxy servers or core servers. As the

DNS routing does in the same manner, one server greets the other servers' request for the data exchange. As farther the connection goes on, load balancer loses its control on the particular request. Hence, that request leads to an inappropriate state. As shown in the figure below, the servers reach its internal collaborations (Fig. 8).

Considering the responsiveness as an important aspect and also considering the load distribution on the physical appliances and installing ESX servers, we create a virtual box out of that physical appliance when an experiment of 20 concurrent users is retrieving or uploading the data for the optimal result. The result is shown in Fig. 9 as an average response time. From the results obtained, it can be concluded that for any user the response time for a particular thread at any time is not constant and it depends on hits and throughput. When the load is increased or any injection of logs or packets is done to the physical appliance, the response time increases due to varying CPU utilization times, I/O activities and storage.

This work presents an experimental analysis to investigate the best possible method to introduce self-organizing within the network without incurring additional complexity and better response time. This counter lets measure how many users are currently connected to your proxy server's Web Proxy Service. Value does not always mean that users are actively using many browsers. Users might have the browser in the background on their desktop [13] (Fig. 10).

Conclusion

Proxy servers loaded with multiple clients/nodes tend to have a low performance factor with respect to response time, throughput, resource utilization and overhead. This proposed methodology gives an efficient load balancing technique that sends a request to a target proxy server through web HTTP and currently has the lowest workload among all proxy servers. The workload at each proxy server is estimated from the number of log records on the

Fig. 7 Entities of network channel

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
▼ Packet Lengths	166756	707.53	42	1494	0.0222	100%	1.6900	647.655
0-19	0	-	-	-	0.0000	0.00%	-	-
20-39	0	-	-	-	0.0000	0.00%	-	-
40-79	5498	50.06	42	78	0.0007	3.30%	0.5000	6574.748
80-159	64391	136.59	81	157	0.0086	38.61%	0.6000	7039.020
160-319	19893	207.28	160	319	0.0026	11.93%	1.1300	647.685
320-639	5722	432.91	320	635	0.0008	3.43%	0.1200	7021.983
640-1279	3315	914.28	640	1275	0.0004	1.99%	0.1200	7025.281
1280-2559	67937	1461.41	1280	1494	0.0090	40.74%	0.8200	1113.546
2560-5119	0	-	-	-	0.0000	0.00%	-	-
5120 and greater	0	-	-	-	0.0000	0.00%	-	-

Fig. 8 Increase in tcp.errorlog due to overhead **Fig. 9** Load distribution across servers at real-time configuration

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate
▼ HTTP Responses by Server Address	231				0.0000	100%	0.0200
▼ 192.168.0.1	230				0.0000	99.57%	0.0200
OK	230				0.0000	100.00%	0.0200
▼ 143.166.11.200	1				0.0000	0.43%	0.0100
KO	1				0.0000	100.00%	0.0100
▼ HTTP Requests by Server	2042				0.0003	100%	0.0800
▼ HTTP Requests by Server Address	2042				0.0003	100.00%	0.0800
▼ 239.255.255.250	2040				0.0003	99.90%	0.0800
239.255.255.250:1900	2040				0.0003	100.00%	0.0800
▼ 192.168.0.1	1				0.0000	0.05%	0.0100
192.168.0.1:1900	1				0.0000	100.00%	0.0100
▼ 143.166.11.200	1				0.0000	0.05%	0.0100
lyncdiscover.com	1				0.0000	100.00%	0.0100
▼ HTTP Requests by HTTP Host	2042				0.0003	100.00%	0.0800
▼ lyncdiscover.com	1				0.0000	0.05%	0.0100
143.166.11.200	1				0.0000	100.00%	0.0100
▼ 239.255.255.250	2040				0.0003	99.90%	0.0800
239.255.255.250	2040				0.0003	100.00%	0.0800

network channel. Load on the channel is determined using HTTP payload and bps movement of data. Load distribution algorithm runs only when the traffic is too high on the network channel and meets the given threshold value. As

scheduler runs when it is required, there is no overhead on the proxy servers and its ease of load distribution and performance of the proxy servers are increased.

Fig. 10 Average response time for different concurrent users

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Wikibooks. https://en.wikibooks.org/wiki/Intellectual_Property_and_the_Internet_Proxy_servers. Accessed 31 Mar 2019
- Liu J. Self-organized load balancing in proxy servers: algorithms and performance. *J Intell Inf Syst.* 2003;20:31–50.
- Ryan Braud. <https://blog.thousandeyes.com/measuring-performance-with-http-proxies/>. Accessed 12 Feb 2019
- Ulrich R, Miller J. Information processing models generating lognormally distributed reaction times. *J Math Psychol.* 1993;37:513–25.
- Khan R, Haroon M, Husain MS. Proceedings of 2015 global conference on communication technologies (GCCT 2015).
- Skvorc D, Horvat M, Srblic S. Performance evaluation of web-socket protocol for implementation of full-duplex web streams MIPRO 2014, 26–30 May 2014, Opatija, Croatia.
- Ngamsuriyaroj S, Rattidham P, Rassameeroj T, Wongbuchasin P, Aramkul N, Rungmano S. Workshops of international conference on advanced information networking and applications; 2011.
- Oracle fusion middleware deployment. <https://docs.oracle.com/cd/E2228901/html/821-1272/load-balblock.html#scrolltoc>. Accessed 25 Mar 2019
- Cohen J, Phadnis N, Valloppillil V, Ross KW. Cache array routing protocol V.1.1; 1997.
- Chandra PK, Sahoo B. Dynamic load distribution algorithm performance in heterogeneous distributed system for I/O- intensive task. In: National Institute of Technology Rourkela TENCON 2008–2008, TENCON 2008. IEEE Region 10 Conference 19–21 Nov. 2008.
- Parulian. An Analysis comparative of response time squid proxy on windows server and Linux server. 2016. <http://mpira.ub.uni-muenchen.de>. Accessed 12 Feb 2019
- Dykes SG, Jeery CL, Das, S. Taxonomy and design for distributed web caching. In: Proceedings of the Hawaii international conference on system science; 1999.
- <https://www.itprotoday.com/windows-78/measuring-proxy-server-performance>. Accessed 25 Mar 2019

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.