INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

TLWL LOAD BALANCING ALGORITHM FOR CLUSTER OF SIP SERVERS

P. VELAVAN, P. MUTHULAKSHMI

Assistant Professor, Department of ECE, Apollo Engineering College, Chennai.

Accepted Date: 22/11/2014; Published Date: 01/12/2014

Abstract: Session Initiation Protocol is a signaling protocol for telephone calls over IP.SIP is defined by the IETF and is gaining popularity. Unlike the H.323, SIP is designed specifically for the internet.SIP defines interfaces for establishing, modifying and terminating session with one or more participants in the VOIP environment. It facilitates development of telephony application. These facilities also enable personal mobility of users.SIP supports various facets of establishing and terminating multimedia communications like the user location to determine the location and end systems to be used for communication, user capabilities, to find and have control of the media and media parameters to be used for the communication, User availability, for determining the called party's willingness to engage in communication and other parameters like call setup, Call handling and teardown. The conversation can be IP to IP, PSTN to IP, and IP to PSTN. In a SIP environment along with the endpoint devices, five entities are required like proxy server, registrar server, redirect server, .Location server and Gateway.

Keywords: Session Initiation Protocol (SIP), Proxy Server.

Corresponding Author: MR. P. VELAVAN

Access Online On:

www.ijpret.com

How to Cite This Article:

P Velavan, IJPRET, 2014; Volume 3 (4): 355-364

PAPER-QR CODE

INTRODUCTION

Load balancing of servers by an IP sprayer can be implemented in different ways [4]. These methods of load balancing can be set up in the load balancer based on available load balancing types. There are various algorithms used to distribute the load among the available servers.

A. Session Initiation Protocol (SIP)

This is an IETF defined signaling protocol widely used for controlling communication session such as voice and video calls over Internet Protocol (IP)[1]. The protocol can be used for creating, modifying and terminating two-party (unicast) or multiparty (multicast) sessions. Sessions may consist of one or several media stream. Other SIP applications include video conferencing, streaming multimedia distribution, instant messaging, presence information, file transformer and online games.

The SIP protocol is an Application layer protocol], user datagram protocol (UDP), or stream control transmission protocol (SCTP)[2]. It is a text-based protocol, inc**orporating many elements of** the Hypertext Transfer protocol TTP) and the Simple Mail Transfer Protocol (SMTP).

B. Load balancing

This a computer networking method to designed to be independent of the underlying transport layer; it can run on transmission control protocol (TCP)[2]distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. Using multiple components with load balancing [4], instead of a single component, may increase reliability through redundancy. The load balancing service is usually provided by dedicated software or hardware, such as a multilayer switch or a Domain Name System server.

Jobs or customers arrive and require service that may be provided at one of several different stations. The associated routing problems concern how customers may be assigned to stations in an optimal manner. Much of the classical literature concerns a single class of customers seeking service from a collection of homogeneous stations. We argue that many contemporary application areas call for the analysis of routing problems in which many classes of customer seek service provided at a collection of diverse stations. This paper is the first to consider routing policies in such complex environments which take appropriate account of the degree of congestion at each service station. A simple and intuitive class of policies emerges from a policy improvement approach. In a numerical study, the policies were close to optimal in all cases.

C. Random Allocation

In a random allocation, the HTTP requests are assigned to any server picked randomly among the group of servers [1]. In such a case, one of the servers may be assigned many more requests to process, while the other servers are sitting idle. However, on average, each server gets its share of the load due to the random selection.

D. Round-Robin Allocation

In a round-robin algorithm, the IP sprayer assigns the requests to a list of the servers on a rotating basis. The first request is allocated to a server picked randomly from the group, so that if more than one IP sprayer is involved, not all the first requests go to the same server. For the subsequent requests, the IP sprayer follows the circular order to redirect the request. Once a server is assigned a request, the server is moved to the end of the list. This keeps the servers equally assigned. Better than random allocation because the requests are equally divided among the available servers in an orderly fashion. Round robin algorithm is not enough for load balancing based on processing overhead required and if the server specifications are not identical to each other in the server group [1]

E. Weighted Round-Robin Allocation

Weighted Round-Robin is an advanced version of the round-robin that eliminates the deficiencies of the plain round robin algorithm. In case of a weighted round-robin, one can assign a weight to each server in the group so that if one server is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the IP sprayer will assign two requests to the powerful server for each request assigned to the weaker one. This takes care of the servers in the group. This does not consider the advanced load balancing requirements such as processing times for each individual request.

The configuration of a load balancing software or hardware should be decided on the particular requirement. For example, if the website wants to load balance servers for static HTML pages or light database driven dynamic web pages, round robin will be sufficient. However, if some of the requests take longer than the others to process, then advanced load balancing algorithms are used. The load balancer should be able to provide intelligent monitoring to distribute the load, directing them to the servers that are capable of handling them better than the others in the cluster of server.

F. Call Join Shortest Queue

The prevalence of dynamic-content web services, exemplified by search and online social networking, has motivated an increasingly wide web-facing front end. Horizontal scaling in the Cloud is favored for its elasticity, and distributed design of load balancers [4]is highly desirable. Existing algorithms with a centralized design, such as Join-the-Shortest-Queue (JSQ), incur high communication overhead for distributed dispatchers.

A novel class of algorithms called Join-Idle-Queue (JIQ) for distributed load balancing in large systems was designed. Unlike algorithms such as Power-of-Two, the JIQ algorithm incurs no communication overhead between the dispatchers and processors at job arrivals. An extension of the basic JIQ algorithm deals with very high loads using only local information of server load.

II RELATED WORK

The existing system of Interposed request routing for scalable network storage explores interposed request routing in Slice new storage system architecture for high-speed networks. It incorporates network attached block storage. The Slice prototype uses a packet filter new proxy to virtualize standard Network File System (NFS) protocol, presenting to NFS clients a unified shared le volume with scalable bandwidth capacity. The Efficient support for P-HTTP in cluster – based Web server supports HTTP/1.1 persistent connections in cluster based request distribution [1]. We present two mechanisms for the efficient, content based distribution of HTTP/1.1 requests among the back-end nodes of a cluster server. We implement the simpler of these two mechanism, back end forwarding.

In the EQUILOAD system a load balancing policy for clustered web server a new strategy for the allocation requests clustered web server is developed based on the size distribution requested document. This strategy EQUILOAD manages to achieve a balanced load each of the back end server, parameters are obtained from analysis trace past data. The result show that EQUILOAD out performs random allocation, performs comparably or better than Shortest Remaining Processing time and join shortest Queue policies and maximizes cache hits the back end server

In the existing system state of the art in locally distributed web server system. The overall increase in traffic on the world wide web augmenting user perceived response times from popular websites especially conjunction with special event. The need to improve the performance of web based services produced variety of novel content delivery architecture. This system focuses on web system architecture and consists of multiple server nodes distributed over local area with one or more mechanism to spread client request among the nodes.

The system of TCP Connection Management Mechanisms for Improving Internet Server Performance investigates [2].

TCP connection management mechanisms in order to understand the behavior and improve the performance of Internet servers during overload conditions such as flash crowds. We study several alternatives for implementing TCP connection establishment, reviewing approaches taken by existing TCP stacks as well as proposing new mechanisms to improve server throughput and reduce client response times under overload. We implement some of these connection establishment mechanisms in the Linux TCP stack and

A. Transaction join shortest Queue

The Join the Shortest Queue scheduling policy [4] in which a newly arriving job is dispatched to the server with the fewest waiting jobs, is an easy to implement and highly effective load balancing scheme7]. For systems with a single server per queue, the policy is optimal in the sense of maximizing the discounted number of service completions in any specified time interval. It was also shown that minimizes the expected total time required to serve all jobs that arrived before some fixed time limit.

These systems present an accurate analytical model for evaluating the performance of the join the shortest queue (JSQ) policy. The system considered consists of N identical queues each of which may have single or multiple servers. A birth-death Markov process is used to model the evolution of the number of jobs in the system. The results show that this method provides very accurate estimates of the average job response times.

The associated routing problems concern how customers may be assigned to stations in an optimal manner *[9]*. Much of the classical literature concerns a single class of customers seeking service from a collection of homogeneous stations. We argue that many contemporary application areas call for the analysis of routing problems in which many classes of customer seek service provided at a collection of diverse stations. This paper is the first to consider routing policies in such complex environments which take appropriate account of the degree of congestion at each service station. A simple and intuitive class of policies emerges from a policy improvement approach. In a numerical study, the policies were close to optimal in all cases.

B. proxy server

Proxy server is an intermediary program that acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, possibly after translation, to other servers. A proxy interprets, and, if necessary, rewrites a request message before forwarding it[6]. It is useful to view Proxy Servers as SIP-level routers that forward SIP requests and responses. However SIP proxies employ routing logic that is typically more sophisticated than just automatically forwarding messages based on a routing table.

III PROPOSED SYSTEM

The session-oriented nature of SIP has important implications for load balancing. Transactions corresponding to the same call must be routed to the same server; otherwise, the server will not recognize the call. Session-aware request assignment (SARA) is the process where a system assigns requests to servers such that sessions are properly recognized by that server, and subsequent requests corresponding to that same session are assigned to the same server. A key aspect of our load balancer is that requests corresponding to the same call are routed to the same server [8]. The load balancer has the freedom to pick a server only on the first request of a call. All subsequent requests corresponding to the call must go to the same server. This allows all requests corresponding to the same session to efficiently access state corresponding to the session.

A. Architecture Diagram of load balancing SIP severs

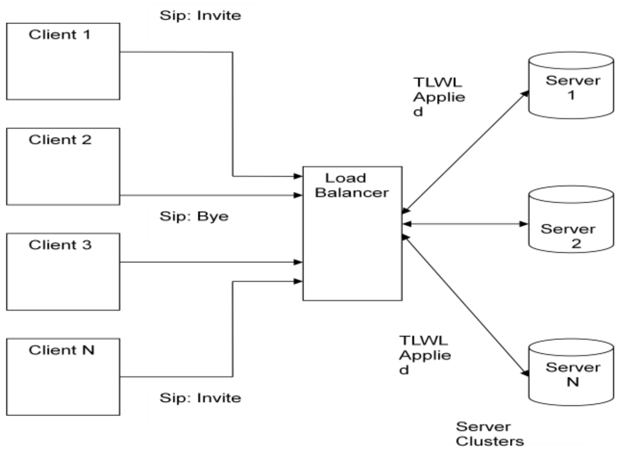

Figure1. Architecture of Load balancing server

Our new load balancing algorithms are based on assigning calls to servers by picking the server with the (estimated) least amount of work assigned but not yet completed. The concept

Of assigning work to servers with the least amount of work left to do have been applied. All responses from servers to clients first go through the load balancer which forwards the responses to the appropriate clients. By monitoring these responses, the load balancer [4] can determine when a server has finished processing a request or call and update the estimates it is maintaining for the work assigned to the server.

B. Transaction Least Work Left

Recent collapses of SIP servers in the carrier networks indicates two potential problems of SIP: the current SIP design does not easily scale up to large network sizes, and the built in SIP overload control mechanism cannot handle overload conditions effectively. This paper introduces several novel load-balancing algorithms for distributing Session Initiation Protocol (SIP) requests to a cluster of SIP servers. Our load balancer improves both throughput and response time versus a single node while exposing a single interface to external clients.

We present the design, implementation, and evaluation of our system using a cluster of Intel ×86 machines running Linux. We compare our algorithms to several well-known approaches and present scalability results for up to 10 nodes. Our best algorithm, Transaction Least-Work-Left (TLWL), achieves its performance by integrating several features: knowledge of the SIP protocol, dynamic estimates of back-end server load, distinguishing transactions from calls, recognizing variability in call length, and exploiting differences in processing costs for different SIP transactions.

By combining these features, our algorithm provides finer-grained load balancing than standard approaches, resulting in throughput improvements and response-time improvements of up to two orders of magnitude. We present a detailed analysis of occupancy to show how our algorithms significantly reduce response time.

1) Client Design and Request

In the first module the client side design is implemented using java FX technology. The clients are request to another client within a group. After the request is confirmed each other communicate vice versa. The formed group member details are shown in each client side. Based on the client group they have to communicate each other via load balancer and server.

2) Load Balancer Design and Server Design:

The load balancer is designed and communicates with the server clusters [10].All the servers are frequently communicated with the load balancer, based on the communication the load balancer allocate the work to the server. Initially the load balancer allocates the work to the server to own interest. If the server is finish the work, it will be send the feedback to the load balancer about work status, how many works left.

3) Client Server Communication using Load Balancer:

The client is communicating to the server through load balancer [3],[5]. So every communication is allocated to the server by the load balancer. If any of the servers is failed that status also update to the load balancer .If a server fails, the load balancer stops sending requests to the server.

If the failed server is later revived, the load balancer can be notified to start sending requests to the server again. A primary load balancer could be configured with a secondary load balancer that would take over in the event that the primary fails. In order to preserve state information in the event of a failure, the primary load balancer would periodically checkpoint its state, either to the secondary load balancer over the network or to a shared disk.

We have not implemented this failover scheme for this paper, and a future area of research is to implement this failover scheme in a manner that both optimizes performance and minimizes lost information in the event that the primary load balancer fails.

C. SIP Server Implementation

Development of a SIP Server application typically involves implementation of these layers:

1) SIP Stack layer

It has to have extra flexibility and customizability in the way transactions are processed because of the special needs of the proxy and the SIP Server in general.

2) SIP Server layer

This implements the standard SIP Server functionality.

3) Application layer

This implements all other aspects of the application (for example, service engine, billing module and database access).

D. Sequence Diagram:

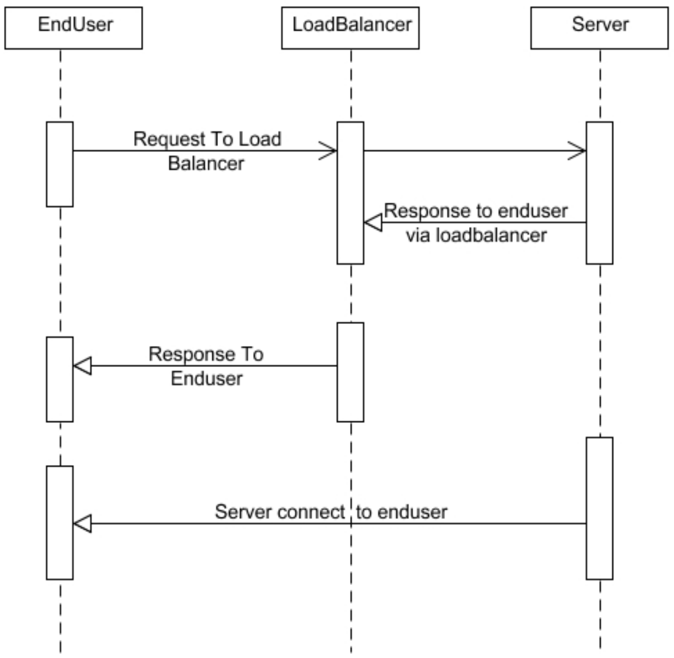

Figure 2 Sequence Diagram

IV. CONCLUSION

The proposed system uses a TLWL algorithm for load balancing in SIP server clusters. The TLWL algorithms result in the increased performance, both in terms of response time and throughput. This system is implementing in Java FX. The proposed system shows that by combining the SIP protocol, distinguishing transactions from calls the load balancing process can be improved greatly. In cases where SARA is not required the TLWL[4] algorithm can be used efficiently than the other load balancing algorithms developed based on Round Robin or hashing. The proposed system results in a reduced response time.

ACKNOWLEDGMENT

We thank our HOD and staff members of Apollo Engineering College for guiding us throughout our research.

REFERENCES

1. R. Fieldi ng , J. Getty s, J. Mogul, H. Frystyk, and T. Berners Lee, "Hypertext Transfer Protocol HTT P /1 .1 ," Internet Engineering Task Force, RFC 2 068, Jan. 1997.

2. M. Aron and P. D ruschel, "TCP implementation enhancements for improving Web server performance ," Computer Science Department, Rice University, Houston, TX, Tech. Rep. T R99-335, Jul. 1 999.

3. M. Aron, D. Sanders, P. Druschel, and W . Zwaenepoel, "Scalable content aware request distribution in cluster based network servers," in *Proc. USENIX Annu. Tech. Conf.*, San Diego, CA, Jun. 2000, pp. 323–336.

4. G. Ciardo, A . Riska, and E. Smirni, "EQ UIL OAD: A load balancing policy for clustered Web servers, " *Perform. Eval*, vol. 4 6, no. 2- 3, pp. 101–124, 2001.

5. H. Feng, V. Misra, and D. Rubenste in, "PBS: A unified priority based scheduler," in *Proc. A CM SIGM RICS* , San Diego, CA, Jun . 2 007, pp. 203–214.

6. V . Card ellini, E . Casalicchio, M . Cola janni, and P. S. Yu, "The state of the art in locally distributed Web server systems," *Comput. Surveys*, vo l. 34, no. 2, pp. 263–311, Jun. 2002.

7. J. Challenger, P. Dantzig, and A . Iyengar, "A scalable and highly available system for serving dynamic data at frequently accessed Web sites," in *Proc. ACM /I EEE Conf. Super comput.*, Nov. 1998, p p. 1–30.

8. G. Gold sz mid t, G. Hunt, R. K ing, and R .Mukherjee, "Network dispatcher: A connection router for scalable Internet services," in *Proc. 7t h In t. World Wide Web Con f.*, Brisbane, Australia, Apr. 1998, pp. 347–357.

9. Z. Fei, S. Bhattacharjee, E. Zegura, and M. A m ma r, "A novel server selection technique for improving the response time of a replicated service," in *P roc. I EEE INFOCOM*, 1 998, vol. 2, p p. 783–791.

10. M. Aron, P. Druschel, and W. Zwaenepoel, "Efficient support for PHTTP in cluster base d Web servers," in *Proc. USENIX Annu. Tech. Conf.*, Monterey, C A, Jun. 1999, pp. 185–198.