Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 7, July 2014, pg.817 – 822

RESEARCH ARTICLE

Implementation of Load Balancing Using SIP

Sandeep Kholambe, M.E. Student

Department of Computer Engineering, MET's Institute of Engineering Nashik, Maharashtra & University of Pune, India

sandeepkholambe@gmail.com

Abstract— The concept of server clusters for efficient and faster working has been extensively implemented through the use of Session Initiation Protocol (SIP). The load balancing for these clusters have been a major challenge in the better implementation of such systems. The requests that are received in a single session are sent to same load balancer to be serviced by same server. This paper implements and evaluates the load balancing on the server clusters. SIP Transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final response sent from the server to the client. The performance of the algorithm is measured through the attributes like response time taken to service the request and memory utilized for the execution of the algorithm. The proposed system is expected to significantly reduce response time which will be confirmed through the inputs and the recorded output. We have used the concept of threads which are treated as the load to the load balancer and use the load balancing algorithm for providing significantly better response time by distributing requests across the cluster.

Keywords— SIP, Load Balancing, Response Time, Memory Utilization

I. INTRODUCTION

In The Session Initiation Protocol [1] is a signaling protocol used media application.SIP is a protocol normally uses in instant messaging, Voice over IP, IPTV, voice and video conferencing. Wireless network providers are standardizing on SIP as the basis for the IP Multimedia System standard. The SIP protocol is that different transaction such as INVITE and BYE.A load balancer can make use of this information to make better load-balancing decisions that improve both response times. It is the first to demonstrate how load balancing is done with estimates of relative overhead for different requests. By using several novel algorithms for balancing load across SIP servers. In addition, the best performing algorithm takes difference of call lengths, different transactions.

- Call Join Shortest Queue: Tracks the number of calls allocated to each back-end server and allocate new SIP calls to the node.
- Transaction-Join-Shortest-Queue: Routes a new call to the server that has the few active transactions.
- Transaction-Least-Work-Left: Routes a new call to server that has the least work.

SIP is an application-layer control protocol [9] that can establish, modify, and terminate multimedia sessions such as Internet telephony calls. SIP can also invite participants to already existing sessions, such as multicast conferences. Media can be added to an existing session. SIP transparently supports name mapping and

redirection services, which supports personal mobility users can maintain a single externally visible identifier regardless of their network location. SIP supports five facets of establishing and terminating multimedia communications User location, User availability, User capabilities, User capabilities and Session management.

A central component is required to distribute worked across multiple server clusters. This mechanism is called load balancing and the device that does the load balancing is called the Load Balancer. A lot of researches have been done in HTTP [7] or File Service [2]. The advantage of this load balancer is to maintain sessions in which requests corresponding to the same session are sent to the save server until the session is completed. HTTP is a protocol for content access and management. SIP is a protocol for two or multiport session establishment.SIP is Peer to Peer while HTTP is essentially client server. Load balancing at highly accessed real Web sites is described in [5] and [12] the techniques deployed at the Web site can be used at other Web sites where it is desirable to provide significant dynamic content. A considerable amount of work has been done in the area of load balancing for HTTP requests [4]. Three main classes of architectures cluster-based Web system and Virtual Web cluster.

Client-side techniques for load balancing and assigning requests to servers are presented in [6] and [11]. Content-aware load balancing in which the load balancer examines the request itself to make routing decisions [3],[10]. Studied existing tool such as The SPEC SIP Infrastructure 2011 benchmark [14] is designed to evaluate a system's ability to act as a SIP server supporting a particular SIP application. OpenSIPS [15] is an Open Source SIP proxy/server for voice, video, IM, presence and any other SIP extensions. SIPp is [16] a performance testing tool for the SIP protocol. It includes a few basic SipStone user agent scenarios (UAC and UAS) and establishes and releases multiple calls with the INVITE and BYE methods.

II. SYSTEM ARCHITECTURE

Figure.1 shows User Agent Clients send SIP requests to load balancer, which then selects a SIP server to handle each request. The distinction between the various load-balancing algorithms presented in this paper is how they choose which SIP server to handle a request. Servers send SIP responses to the load balancer, which then forwards the response to the client.

SIP is used to establish, alter, or terminate media sessions. Once a session has been established, the parties participating in the session would typically communicate directly with each other using a different protocol for the media transfer, which would not go through our SIP load balancer.

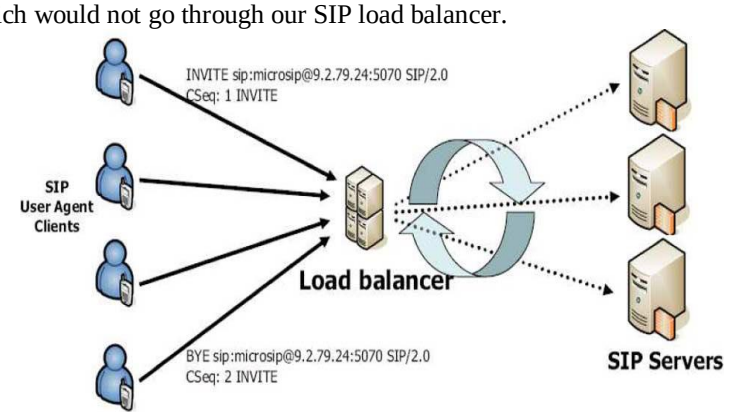

Fig 1:System Architecture

Working flow divided into client, load balancer server, two client connection for communication. Load balancer performs the following operation.

Start Operation: Then start number of client in each system. After running process start server, then start all clients on each system and display the message with time as show "Cluster server initialization and Server started" on server side and display "Initialization and connection established" on client side.

Invite Operation: Invite operation call by client system, it calls connect module for entering details of local and remote system data. To enter name and IP address of two clients. After entering data on first system display "Invitation Send" message with time and date of operation on log section, and other system display "Invitation Received" on log section. So the operation completes.

Accept Operation: Accept operation is performed on that system, which received invitation received message to accept for communication and Acceptance send with "Accept Ack".

Send Operation: In this section actual text entered and send to other system. Display the text message in Chat text section. Log section display the acknowledge message with time and date of operation "Text Message Send" and "Text Message Received" on both system.

Bye Operation: When client want to terminate the communication then Bye operation is performed. One system display "Bye Message Send" and other system display "Bye Message Received" with time and date of communication.

Enter Thread: To enter thread for creating load on server. Server display the response time, memory utilization, capacity for each thread.

Load balancer checks the request coming from which client and first it will check the load of each server, and then allocate the server those who have least work. If all servers have fulfilled the capacity then coming request is in queue. If the any client crashed and again wants to communicate then request goes to server if server are available then passed to it otherwise allocate another server for same client. All switching operation are performed by load balancer. In load balancing solution handles both inbound and outbound requests. The workload is distributed between servers that are prepared to balance user traffic. This can lead to a variety of problems, including limited communication between your server farms. Global load balancing should account for all IP traffic so user requests are sent to the proper location.

III. LOAD BALANCING ALGORITHM

This section describes proposed implementation. Figure 2 the structure of the load balancer. The rectangles represent key functional modules of the load balancer, while the irregular shaped boxes represent state information that is maintained. The arrows represent communication flows. The Receiver receives requests that are then parsed by the Parser.

The Session Recognition module determines if the request corresponds to an already existing session by querying the Session State, which is implemented as a hash table as described below. If so, the request is forwarded to the server to which the session was previously assigned. If not, the Server Selection module assigns the new session to a server using one of the algorithms described earlier. For several of the load-balancing algorithms we have implemented, these assignments may be based on Load Estimates maintained for each of the servers.

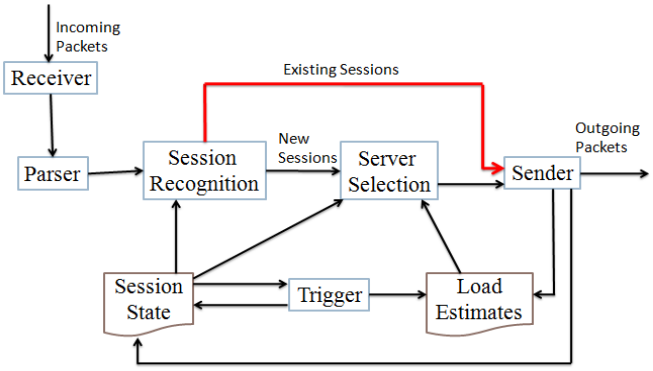

Fig 2:Load Balancer Structure

The Sender forwards requests to servers and updates Load Estimates and Session State as needed. The Receiver also receives responses sent by servers. The client to receive the response is identified by the Session Recognition module, which obtains this information by querying the Session State. The Sender then sends the response to the client and updates Load Estimates and Session State as needed. The Trigger module updates Session State and Load Estimates after a session has expired.

Load Balancing Algorithm steps are as follows:

- Create a Hash table (ht) object to hold Session-Ids
- Check the "Call-Id" in "ht" object
- If not found then "Assume he is new client"
  - If "Token is" INVITE then "select a new node from Linked list according to "Job Assigned"
  - Add the node to "Active Hash table"
- Update the Status of the Client-Id in Hash table.
- If he is the existing client and "INVITE" token is called update the status for the Node and record response time for INVITE.
- if ACK token is passed Update "Ack" Status
- If BYE token is passed then we are going to remove the request of the client from the server by decreasing the count value from nodes and move the entry to expire hash table.

IV. PROPOSED ALGORITHM

A. Load Balancer with Hash Algorithm

Proposed work modifies the traditional hash algorithm .

Input : The different kinds of sessions can be used text which is transported over a separate data plane protocol. To start transaction client server connection is established by define IP address on both UI. A SIP message sent from a client to a server, for the purpose of invoking a particular operation.

Process :SIP Transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final response sent from the server to the client. If the request is INVITE and the final response is a, the transaction also includes an ACK to the response. The ACK for a response to an INVITE request is a separate transaction. BYE transaction for terminating call.

- INVITE: The party that receives an INVITE request for the purposes of establishing a new session.
- BYE: For terminating BYE request to close transaction.
- ACK: Acknowledge between request and response.

Algorithm:

Let ht be the Hash Table, st be the store command, cid be the Call_Id, ll be the Linked List, x_i be the New Node, r be the record, rt be the response time, s be Status Ack, t be the token, req be the Request.

Begin

- Create the ht
- St(ht, cid)
- Check (cid, ht)
- If False, cid →New Client
    - If (t = INVITE)
        - Select (ll, x_i)
    - add(ht , x_i)
    - Update (ht, cid)
- If True Update(ht, cid)
    - Record(rt, INVITE)
- If Ack t=True Update(s)
- IF BYE= True remove(Server, req)
    - Move(Expired ht, req)
- END

Output :A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server. A server should be prepared to received requests on any IP address, port and transport combination on a SIP that is handed out for the purposes of communicating with that server. A server is a network element that receives requests in order to service them and sends back responses to those requests. Request is transferred by load balancer to least work server, a communication start with client till BYE transaction.

V. RESULT

A. *Threads*

A main thread listens for incoming messages from client to server. If the message is a request not matching any previous transaction, then a new thread is created to handle the new transaction associated with this message record system information. The thread persists as long as the transaction exists. Similarly, a process per message model can be defined that creates a new process for each incoming connection and message from client.

B. *Event Based*

Most of the blocking operations are made non blocking using events. A single thread handles events from a queue as well as messages from the listening socket. There is no locking or mutexes. Avoid the crossover of executing threads.

C. *Response Time*

Observation of system significantly differentiate in the response times of each threads from starting to end of the thread. Response time is define transaction between two system through SIP load balancer with invite, bye and acknowledgement scenario to each thread.

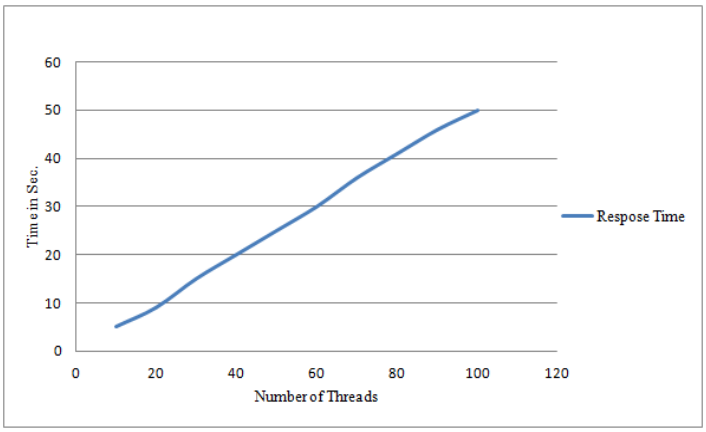

Fig 3: Number of thread against Response time

Figure.3 shows the graphical representation of threads and response time. As the number of threads increases the response time also increases linearly. This graph shows that the time taken to service the threads in load balancing scenario increases linearly as the number of threads increases similarly.

D. *Memory Utilization*

Memory require for execution of each thread in communication between client and load balancer server. Figure.4 shows the graphical representation of threads and memory utilization in transaction. Even though the threads increase, the memory utilization remains constant. The value taken is mode of the number of absolute made for a set of specified threads.

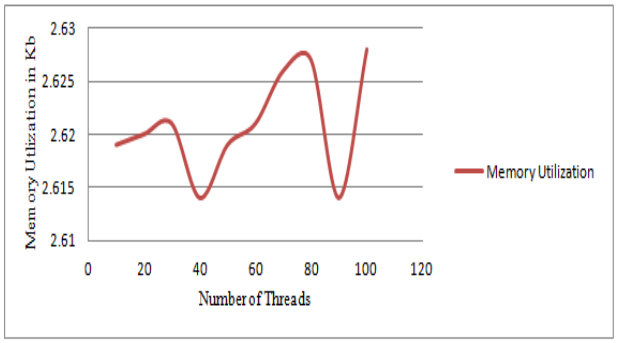

Fig 4: Number of thread against Memory Utilization

VI. CONCLUSIONS

The proposed algorithm results in the best performance, both in terms of response time and memory utilization. The most significant performance differences were in response time. For SIP applications that require good quality of service, these dramatically lower response times are significant. Trying to bypass the hardware limitations of holding a load balancer only in one server, by distributing the participants between several machines, is not an easy task, and to some extent, can affect the overall system transaction of the load balancer.

REFERENCES

[1] Hongbo Jiang, Arun Iyengar,, Erich Nahum, Wolfgang Segmuller, Asser N. Tantawi, and Charles P.Wright, "Design, Implementation, and Performance of a Load Balancer for SIP Server Clusters," IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 20, NO. 4, AUGUST 2012.

[2] D. C.Anderson, J. S. Chase, and A.Vahdat, "Interposed request routing for scalable network storage," in *Proc. USENIX OSDI*, San Diego, CA, Oct. 2000, pp. 259–272.

[3] M. Aron, P. Druschel, and W. Zwaenepoel, "Efficient support for P-HTTP in cluster-based Web servers," in *Proc. USENIX Annu. Tech. Conf.*, Monterey, CA, Jun. 1999, pp. 185–198.

[4] V. Cardellini, E. Casalicchio, M. Colajanni, and P. S. Yu, "The state of the art in locally distributedWeb-server systems," *Comput. Surveys*, vol. 34, no. 2, pp. 263–311, Jun. 2002.

[5] 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/IEEE Conf. Supercomput.*, Nov. 1998, pp. 1–30.

[6] Z. Fei, S. Bhattacharjee, E. Zegura, and M. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proc. IEEE INFOCOM*, 1998, vol. 2, pp. 783–791.
[7] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. M. Nahum, "Locality-aware request distribution in cluster-based network servers," in *Proc. Archit. Support Program. Lang. Oper. Syst.*, 1998, pp. 205–216.
[8] K. Singh and H. Schulzrinne, "Failover and load sharing in SIP telephony," in *Proc. SPECTS*, Jul. 2005, pp. 927–942.
[9] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session initiation protocol," Internet Engineering Task Force, RFC 3261, Jun. 2002.
[10] 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.
[11] D. Mosedale,W. Foss, and R.McCool, "Lessons learned administering Netscape's Internet site," *IEEE Internet Comput.*, vol. 1, no. 2, pp. 28–35, Mar./Apr. 1997.
[12] A. Iyengar, J. Challenger, D. Dias, and P. Dantzig, "High-performance Web site design techniques," *IEEE Internet Comput.*, vol. 4, no. 2, pp. 17–26, Mar./Apr. 2000.
[13] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee, "Hypertext Transfer Protocol—HTTP/1.1," Internet Engineering Task Force, RFC 2068, Jan. 1997.
[14] SPEC SIP Subcommittee "Systems Performance Evaluation Corporation (SPEC),", 2011 [Online]. Available: http://www.spec.org/specsip/.
[15] OpenSIPS, "The open SIP express router (OpenSER)," 2011 [Online]. Available: http://www.openser.org.
[16] R. Gayraud and O. Jacques, "SIP_p ," 2010 [Online]. Available: http:// sipp.sourceforge.net.

