ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Design and Implementation of Locally Distributed Web Server Systems using Load Balancer

R.Jayabal[*1], **R.Mohan Raj**[2]

[*1] M.E. Student, [2]Lecturer, Department of Computer Science and Engineering, St.Peter University,Avadi,Tamil Nadu,India

jayabal.mca35@gmail.com

Abstract

This Paper presents the design, implementation and evaluation of a load balancer for cluster-based SIP servers. Our load balancer performs session-aware request assignment to ensure that SIP transactions are routed to the proper back-end node that contains the appropriate session state. We presented three novel algorithms: CJSQ, TJSQ, and TLWL. The TLWL algorithms result in the best performance, both in terms of response time and throughput, followed by TJSQ.TJSQ has the advantage that no knowledge is needed of relative overheads of different transaction types.SIP applications that require good quality of service, these dramatically lower response times are significant. We showed that these algorithms provide significantly better response time by distributing requests across the cluster more evenly, thus minimizing occupancy and the corresponding amount of time a particular request waits behind others for service. TLWL-1.75 provides 25% better throughput than a standard hash-based algorithm and 14% better throughput than a dynamic round-robin algorithm..

Keywords: load balancing, Session Initiation Protocol (SIP), TLWL.

Introduction

The main objective of this paper to implement a load balancer, allocates the work to the clusters of SIP server using transaction least-work-left algorithm(TLWL).

Load Balancer

Network Load Balancing, a clustering technology included in the Microsoft Windows 2000 Advanced Server and Datacenter Server operating systems, enhances the scalability and availability of mission-critical, TCP/IP-based services, such as Web, Terminal Services, virtual private networking, and streaming media servers. This component runs within cluster hosts as part of the Windows 2000 operating system and requires no dedicated hardware support. To scale performance, Network Load Balancing distributes IP traffic across multiple cluster hosts. It also ensures high availability by detecting host failures and automatically redistributing traffic to the surviving hosts. Network Load Balancing provides remote controllability and supports rolling upgrades from the Windows NT 4.0 operating system. The unique and fully distributed architecture of Network Load Balancing enables it to deliver very high performance and failover protection, especially in comparison with dispatcher-based load balancers. This white paper describes the key features of this technology and explores its internal architecture and performance characteristics in detail.

Call-Join-Shortest-Queue (CJSQ):

CSJQ tracks the number of calls (in this paper, we use the terms call and session interchangeably) allocated to each back-end server and routes new SIP calls to the node with the least number of active calls.

Transaction-Join-Shortest-Queue (TJSQ):

TJSQ routes a new call to the server that has the fewest active transactions, rather than the fewest calls. This algorithm

Improves on CJSQ by recognizing that calls in SIP are composed of the two transactions, INVITE and BYE, and that by tracking their completion separately, finer-grained estimates of server load can be maintained. This leads to better load balancing, particularly since calls have variable length and thus do not have a unit cost.

Transaction-Least-Work-Left (TLWL):

TLWL routes a new call to the server that has the least work, where work (i.e., load) is based on relative estimates of transaction costs. TLWL takes advantage of the observation that INVITE transactions are more expensive than BYE transactions. On our platform, a 1.75:1 cost ratio between INVITE and BYE results in the best performance.

Session Initiation Protocol (SIP):

The Session Initiation Protocol (SIP) is a general-purpose signaling protocol used to control various types of media sessions. SIP is a protocol of growing importance, with uses in Voice over IP, Instant Messaging, IPTV, Voice Conferencing, and Video Conferencing. SIP has a number of features which distinguish it from protocols such as HTTP. SIP is a transaction-based protocol designed to establish and tear down media sessions, frequently referred to as calls. Two types of state exist in SIP. The first, session state, is created by the INVITE transaction and is destroyed by the BYE transaction. 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):

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. 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. In contrast, sessions are less significant in HTTP.

While SARA can be done in HTTP for performance reasons, it is not necessary for correctness. Many HTTP load balancers do not take sessions into account in making load-balancing decisions.

Voice over Internet Protocol (VOIP):

For systems running VOIP applications, the blade servers must not become overloaded by distributing the incoming calls. An imbalance of overloaded and under loaded servers results in dropped calls, jittery voice quality, or service interruptions. VOIP applications must remain available to users during periods of peak call volume. Incoming calls must be redirected if a server reaches peak capacity, or if a link fails, or if a server has been attacked. VOIP applications are also extremely time-sensitive-VOIP traffic must be guaranteed high priority using Quality of Service in order to minimize latency and maintain call quality.

System Analysis

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. 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. 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 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.

Transaction Least Work Left Algorithm:

The TLWL algorithm addresses this issue by assigning different weights to different transactions depending on their relative costs. It is similar to TJSQ with the enhancement that transactions are weighted by relative overhead; in the special case that all transactions have the same expected overhead, TLWL and TJSQ are the same. Counters are maintained by the load balancer indicating the weighted number of transactions assigned to each server. New calls are assigned to the server with the lowest counter. A ratio is defined in terms of relative cost of INVITE to BYE transactions. We experimented with several values for this ratio of relative cost. TLWL-2 assumes INVITE transactions are twice as expensive as BYE transactions and are indicated in our graphs as TLWL-2.

We found the best performing estimate of relative costs was 1.75; these are indicated in our graphs as TLWL-1.75. Note that if it is not feasible to determine the relative overheads of different transaction types, TJSQ can be used, which results in almost as good performance as TLWL-1.75.TLWL estimates server load based on the weighted number of transactions a server is currently handling. For example, if a server is processing an INVITE (relative cost of 1.75) and a BYE transaction (relative cost of 1.0), the server has a load of 2.75.

TLWL can be adapted to workloads with other transaction types by using different weights based on the overheads of the transaction types. In addition, the relative costs used for TLWL could be adaptively varied to improve performance. We did not need to adaptively vary the relative costs because the value of 1.75 was

ISSN: 2277-9655
Impact Factor: 1.852

relatively constant. CJSQ, TJSQ, and TLWL are all novel load-balancing algorithms. In addition, we are not aware of any previous work that has successfully adapted least work left algorithms for load balancing with SARA.

Problem Definition and methods:

As a cyber citizen, everyone has the security problem to maintain their private information. The private information is maintained in the several websites. With the services on the internet increased, as a member of different websites user may face the problem of setting the passwords. For the convenience of the memory user will set a simple password for all the internet services. All private information will be lost if the simple password is lost. If the user sets different password on different internet services, then the problem occurs when the user forget some of them when they have not been used for a long time.

Neighbors Node Discover:

Each and every client to know which neighbors clients are there in the network using of the neighbors node discover it shows how many clients is there in network, suppose new client arrive means it updated for the number neighbors client

Load Balancer Design and Server Design:

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.

Client Design and Request:

The load balancer is designed and communicates with the server clusters, 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.

Client –Server Communication using Load Balancer:

The client is communicating to the server through load balancer. So every communication is allocated to the server by the load balancer. If any of the server is failed that status also update to the load balancer

Data Flow Diagram

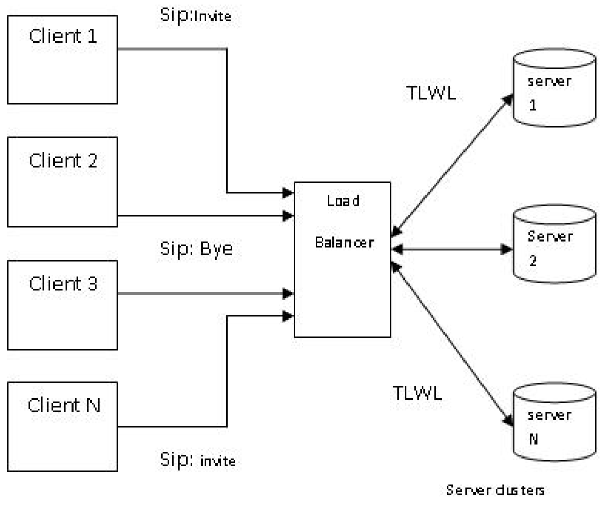

Fig 3.1: **Architecture Diagram**

- The client communicate to another client using of the server and with help of the load balancer in this load balancer using of the some of the load balancing algorithms CJSQ, TJSQ, and TLWL are implemented in a working load balancer for SIP server clusters.
- The server details to be updated to the load balancer
- The load balancer has the freedom to pick a server only on the first request of a call.
- Increase throughput and efficiency of load balancing.
- Dynamic estimates of back-end server load, exploiting differences in processing costs for different SIP transactions.

Load Balancer Architecture:

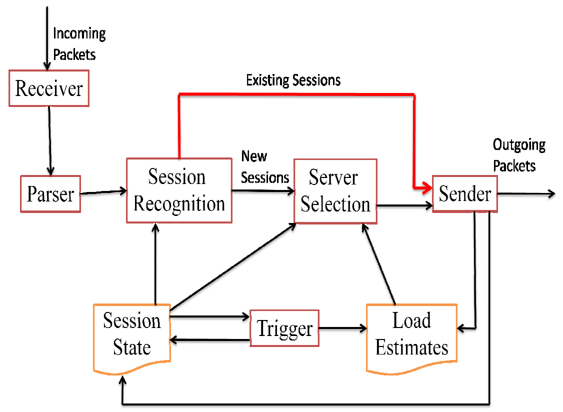

Fig 3.2: **Load Balancer Architecture**

Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). There are different notations to draw data flow diagrams defining different visual representations for processes, data stores, data flow, and external entities**.**

Dataflow Diagram Level 0

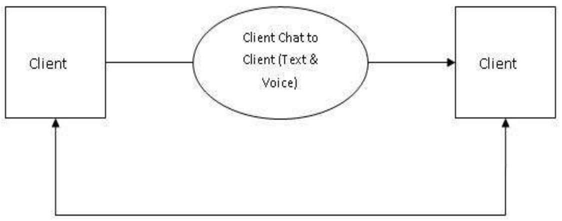

Fig 3.3: Dataflow Diagram Level 0

- The client communicates to another client using of client request to that particular client.
- Using of the client request only to communicate one client to another client.

Dataflow Diagram Level 1:

- Client send request to the load balancer then client communicated to the load balancer
- Load balancer directly communicate to server and the server details to be updated to the load balancer
- Client request to be transfer and distributed to the which server is free based on the load balancer algorithm Transaction Least Work Left to applied and select the server and to provide the service to client

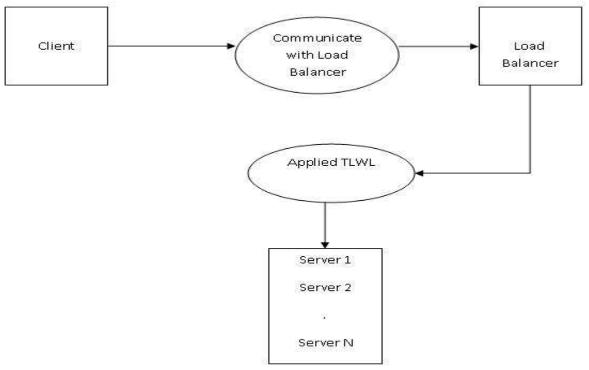

Fig 3.4: **Dataflow Diagram Level 1**

Dataflow Diagram Level 2:

- Client request to be transfer and distributed to the which server is free based on the load balancer algorithm Transaction Least Work Left to applied and select the server and to provide the service to client
- The server status to be updated to updated to load balancer then based on the client request to allocate to server services vice versa

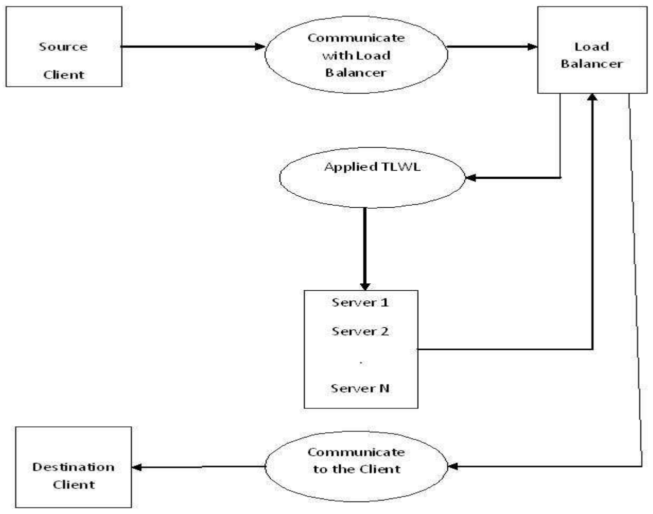

Fig 3.5: **Dataflow Diagram Level 2**

System Implementation

Our system design and control is targeted towards the implementation of the project. Though the implementation is not as crucial as system design it also requires something’s to take care. Blowfish cryptographic algorithm is the core technology used in this project to implement the password management system**.**

Blowfish Algorithm

Blowfish is a symmetric block cipher that can be effectively used for encryption and safeguarding of data. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for securing data. Blowfish was designed in 1993 by **Bruce Schneier** as a fast, free alternative to existing encryption algorithms. Blowfish is unpatented and license-free, and is available free for all uses.

Blowfish Algorithm is a **Feistel Network**, iterating a simple encryption function 16 times. The block size is 64 bits, and the key can be any length up to 448 bits. Blowfish is a variable-length key block cipher. It is suitable for applications where the key does not change often, like a communications link or an automatic file encryptor. It is significantly faster than most encryption algorithms when implemented on 32-bit microprocessors with large data caches.

Feistel Networks

A Feistel network is a general method of transforming any function (usually called an F- function) into a permutation. It was invented by Horst Feistel and has been used in many block cipher designs. The working of a Feistal Network is given below:

- Split each block into halves
- Right half becomes new left half
- New right half is the final result when the left half is XOR'd with the result of applying *f* to the right half and the key.
- Note that previous rounds can be derived even if the function *f* is not invertible.

The Blowfish Algorithm:

- Manipulates data in large blocks
- Has a 64-bit block size.
- Has a scalable key, from 32 bits to at least 256 bits.
- Uses simple operations that are efficient on microprocessors.

Employs precomputable subkeys.

On large-memory systems, these subkeys can be precomputed for faster operation. Not precomputing the subkeys will result in slower operation, but it should still be possible to encrypt data without any precomputations.

Consists of a variable number of iterations.

For applications with a small key size, the trade-off between the complexity of a brute-force attack and a differential attack make a large number of iterations superfluous. Hence, it should be possible to reduce the number of iterations with no loss of security (beyond that of the reduced key size).

Uses subkeys that are a one-way hash of the key.

This allows the use of long passphrases for the key without compromising security. Uses a design that is simple to understand. This facilitates analysis and increase the confidence in the algorithm. In practice, this means that the algorithm will be a Feistel iterated block cipher.

Conclusion

This Paper presents the design, implementation and evaluation of a load balancer for cluster-based SIP servers. Our load balancer performs session-aware request assignment to ensure that SIP transactions are routed to the proper back-end node that contains the appropriate session state. We presented three novel algorithms: CJSQ, TJSQ, and TLWL. The TLWL algorithms result in the best performance, both in terms of response time and throughput, followed by TJSQ.TJSQ has the advantage that no knowledge is needed of relative overheads of different transaction types.SIP applications that require good quality of service, these dramatically lower response times are significant. We showed that these algorithms provide significantly better response time by distributing requests across the cluster more evenly, thus minimizing occupancy and the corresponding amount of time a particular request waits behind others for service. TLWL-1.75 provides 25% better throughput than a standard hash-based algorithm and 14% better throughput than a dynamic round-robin algorithm.

Our results are influenced by the fact that SIP requires SARA. However, even where SARA is not needed, variants of TLWL and TJSQ could be deployed and may offer significant benefits over commonly deployed load-balancing algorithms based on round robin, hashing, or response times. Using of SIP will develop Video chatting in the futures.

References

[1] 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. This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination. JIANG et al.: LOAD BALANCER FOR SIP SERVER CLUSTERS 13

[2] M. Aron and P. Druschel, "TCP implementation enhancements for improving Webserver performance," Computer Science Department, Rice University, Houston, TX, Tech. Rep. TR99-335, Jul. 1999.

[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] 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.

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

[6] 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. Supercomputers., Nov. 1998, pp. 1–30.

[7] G. Ciardo, A. Riska, and E. Smirni, "EQUILOAD: A load balancing policy for

clustered Web servers," Perform. Eval., vol. 46, no. 2-3, pp. 101–124, 2001.

[8] D. Dias,W. Kish, R.Mukherjee, and R. Tewari, "A scalable and highly available Web server," in Proc. IEEE Compcon, Feb. 1996, pp. 85–92.

[9] 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.

[10]H. Feng, V. Misra, and D. Rubenstein, "PBS: A unified priority-based scheduler," in Proc. ACM SIGMETRICS, San Diego, CA, Jun. 2007, pp. 203–214.

[11]M. Harchol-Balter, M. Crovella, and C. D. Murta, "On choosing a task assignment policy for a distributed server system," J. Parallel Distrib. Comput., vol. 59, no. 2, pp. 204–228, 1999.

[12]V. Hilt and I. Widjaja, "Controlling overload in networks of SIP servers," in Proc. IEEE ICNP, Orlando, FL, Oct. 2008, pp. 83–93. [18] IBM, "Application switching with Nortel Networks Layer 2–7 Gigabit Ethernet switch module for IBM Blade Center," 2006 [Online]. Available: http://www.redbooks.ibm.com/abstracts/redp3589.html?Open

[13]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.

[14]E. Nahum, J. Tracey, and C. P. Wright, "Evaluating SIP proxy server performance," in Proc. 17th NOSSDAV, Urbana–Champaign, IL, Jun. 2007, pp. 79–85.

[15]Foundry Networks, "ServerIron switches support SIP load balancing VoIP/SIP traffic management solutions," Accessed Jul. 2007 [Online]. Available: http://www.foundrynet.com/solutions/sol-app-switch/solvoip- sip

[16]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.

[17]C. Shen, H. Schulzrinne, and E. M. Nahum, "Session initiation protocol (SIP) server overload control: Design and evaluation," in Proc. IPTComm, Heidelberg, Germany, Jul. 2008, pp. 149–173.

[18]K. Singh and H. Schulzrinne, "Failover and load sharing in SIP telephony," in Proc. SPECTS, Jul. 2005, pp. 927–942.

[19]SPEC SIP Subcommittee "Systems Performance Evaluation Corporation (SPEC),", 2011 [Online]. Available: http://www.spec.org/specsip/

[20]OpenSIPS, "The open SIP express router (OpenSER)," 2011 [Online]. Available: http://www.openser.org

[21]L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappola, "RSVP: A new resource reservation protocol," IEEE Commun. Mag., vol. 40, no. 5, pp. 116–127, May 2002.