Development of Software Dispatcher Based Load Balancing Algorithms for Heterogeneous Cluster Based Web Systems

Prof. Gautam J. Kamani, Dr. N. N. Jani, Dr. P. V. Virparia

Abstract: Once the web site becomes a popular and the access frequency from a large domain of user increase then single web server may not be able to handle high volume of incoming traffic. To provide a better service to all the clients, they need a fully replicated web server clusters. In such an environment, one of the most important issues is that of server selection (and load balancing). In web cluster, load balancing process done at Open Systems Interconnection layer 4 (Hardware Load Balancer) and layer 7 (Software Load Balancer) with typical static and dynamic load balancing approaches. In this paper we present the performance analysis of round robin, random, least connection and new execution time load balancing algorithms at layer 7 in heterogeneous web cluster. The main purpose of this paper is to help in design of new algorithms in future by studying the behavior of various static and dynamic load balancing algorithms.

Keywords: Web Cluster, load balancing, Static load balancing, Dynamic load balancing.

1. INTRODUCTION

The continuing growth of the World Wide Web (WWW) is placing increasing demands on popular Web servers. It is crucial to achieve the scalable performance of web servers as far as data traffic in WWW is concern. Many sites are now using distributed Web servers systems (i.e., groups of machines) to service the increasing number of client requests, as a single server cannot handle the workload. The distribution of the web server allows handling more requests and increasing the server throughput.

Load balancing is technique that distributes the load among multiple servers. Incoming client requests must be distributed in some fashion among the machines in the distributed Web server systems that not only should this distribution balance the load on the available servers, but also it must do so in a manner that does not increase the latency for the Web user. The other benefit of server load balancing is its ability to improve application availability. If an application or server fails, load balancing can automatically redistribute end-user service requests to other servers within a server farm or to servers in another location.

2. CLUSTER BASED WEB SYSTEMS

A cluster-based Web system refers to a collection of server machines that are housed together in a single location, that are interconnected through a high-speed network. Cluster based web systems also known as “Web Cluster” or “Web farm” means the collection of all the servers.

In Web cluster architecture, the server nodes are only visible by single virtual IP address. Thus, the authoritative DNS server for the Web site always performs a one-to-one mapping by translating the site name into the Virtual IP address, which corresponds to the IP address of a dedicated front-end node or web switch.

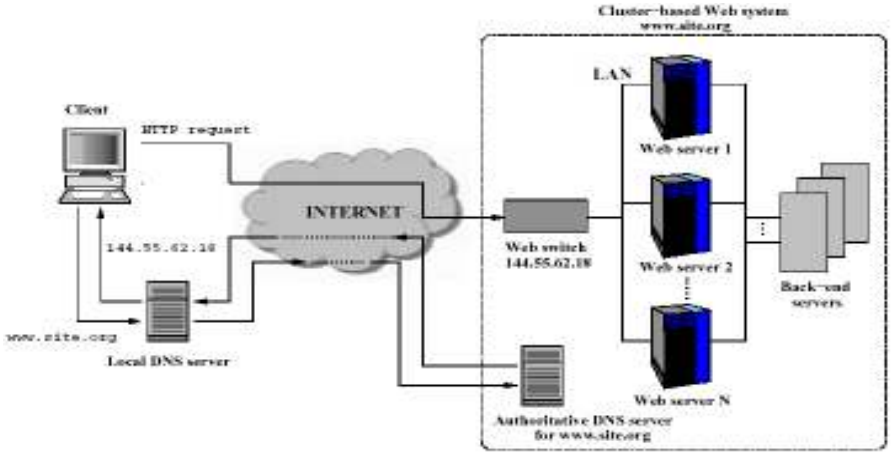


Figure 1: Cluster Web Architecture

The front-end node or Web switch receives all inbound packets that clients send to the Virtual IP address, and routes them to some Web server node. The dispatcher is able to uniquely identify each server in the cluster through a private address that can be at different protocol levels depending on the proposed architecture. The existing dispatcher-based architectures typically use simple algorithms for the selection of the Web-server because the dispatcher has to manage all incoming flow and the amount of processing for each request has to be kept to minimum. However, it is possible to integrate this architecture with some more sophisticated assignment algorithms proposed for distributed Web-server systems having a centralized dispatcher with full control on client requests.

3. LOAD BALANCING TECHNIQUES

In general, load balancing solutions are of two main types: hardware-based load balancer (Transport-level load balancing) and software-based load balancer (Application-level load balancing).

Hardware-based load balancers are specialized hardware boxes that include application-specific integrated circuits customized for a particular use. Hardware-based load balancers are often used for transport-level load balancing. In general, hardware-based load balancers are faster than software-based solutions. Their drawback is their cost. It is very high compare to software load balancer and Hardware-based load balancers are typically more difficult to develop, they often run on proprietary architectures, such as the Cisco CSS and Foundry's Serverlron.

Software-based load balancers are often used for Application-level load balancing. Software-based solutions run directly in the application that uses the application payload to make load balancing decisions. Software-based load balancers run on standard operating systems and standard hardware components such as PCs. Software-based load balancers are typically easy to develop because the coding resources for a widely used OS are easily available. This can help shorten code and new-feature turnaround. Sever-based load balancers are typically flexible in what they can do. New features can be rolled out swiftly, and the machines themselves can take on new and creative ways of performance monitoring, as well as other tasks. It is cheaper than hardware-based load balancer but it has high over head compare to hardware-based load balancer.

4. LOAD BALANCING ALGORITHMS

Load balancing can be achieved through static load balancing algorithms and dynamic load balancing algorithms with the help of clusters of heterogeneous web server systems. Some static and dynamic algorithms are given below:

[i] Round Robin and Random Algorithm

Round Robin algorithm is a simple and most popular algorithm. Round robin algorithm uses an available server list. This algorithm alternatively assigns to web request to the web servers in order. With equal workload round robin algorithm is expected to work well although it is easy to implement. Pseudo code for round Robin algorithm:

nSystem : total available server

Serverlistarr[cnt] : array of available server with port number

totRequest: counter for http request

totRequest=totRequest+1

If nSystem=1 then

webSystemHost=serverlistar[0].host name

webSystemPort=serverlistar[0].port number

Else

idx=totrequest % nSystem;

webSystemHost=serverlistar[idx].host name

webSystemPort=serverlistar[idx].port number

End if

Random algorithm is an alternative algorithm of Round robin algorithm. It is also a static load balancing algorithm. Random algorithm has an available server list. Pseudo code for random algorithm:

nSystem : total available server

Serverlistarr[cnt] : array of available server with port number

ranNum : random number

totRequest: counter for http request

totRequest=totRequest+1

ranNum= Generate Random number (Range from 0 to nSystem-1)

webSystemHost=serverlistar[ranNum].host name

webSystemPort=serverlistar[ranNum].port number

[ii] Least Connection Algorithm

This algorithm assigns a web request to the server with the least number of established connections. The least-connection algorithm is simple dynamic load balancing algorithm; because it needs to count live connections for each server dynamically. Pseudo code for Least connection algorithm:

nSystem : total available server

Serverlistarr[cnt] : array of available server with port number

serConn[cnt] : array of active connection counter for available server

totRequest: counter for http request

totRequest=totRequest+1

// find least active connection server

mincon= serConn first element value

for i= 1 to nSystem-1 increment by 1

if mincon>serConn[i] then

mincon=serConn[i]

idx = i

endif

endfor

webSystemHost=serverlistar[idx].host name

webSystemPort=serverlistar[idx].port number

serConn[idx]=serConn[idx]+1

Decrease value 1 from active connection counter serConn when request is completed

[iii] Execution Time Load Algorithm

Execution Time Load algorithm is fully dynamic algorithm. This algorithm assigns a web request to least execution time loaded server. It is more suitable algorithm for heterogeneous cluster web systems because different configuration server systems require different execution time for same web resources.

nSystem : total available server

Serverlistarr [cnt] : array of available server with port number

curLoad[cnt] : array of existing current load for available server

exeFilename[cnt]: array of available web resource file name

exeTime[srv][cnt]: array of execution time require for all resources for available server

reqFilename: web request file name

totRequest: counter for http request

totRequest=totRequest+1

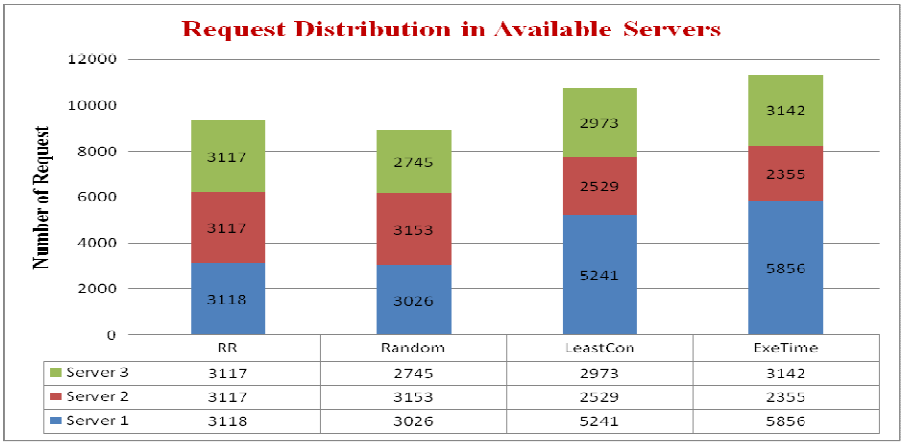
```
// find minimum load server
minload= curLoad first element value
for i= 1 to nSystem-1 increment by 1
    if minload>curLoad[i] then
        minload=curLoad[i]
        idx = i
    endif
endfor

// find request file execution time for minimum loaded server
for i= 0 to size(exeFilename)-1 increment by 1
    if exeFilename [i] = reqFilename then
        fileidx = i
    endif
endfor
webSystemHost=serverlistar[idx].host name
webSystemPort=serverlistar[idx].port number
curLoad[idx]=curLoad[idx]+exeTime[idx][fileidx]
Decrease execution require time from assign server current load counter
when request is completed
```

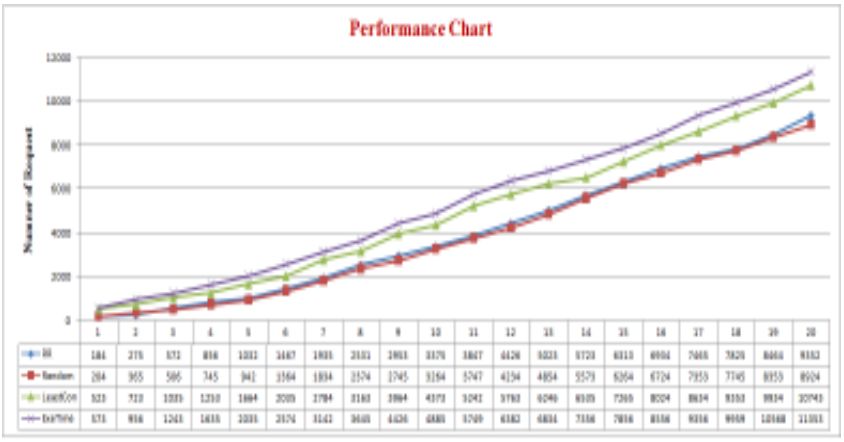
5. EXPERIMENTAL SETUP & RESULT GRAPH

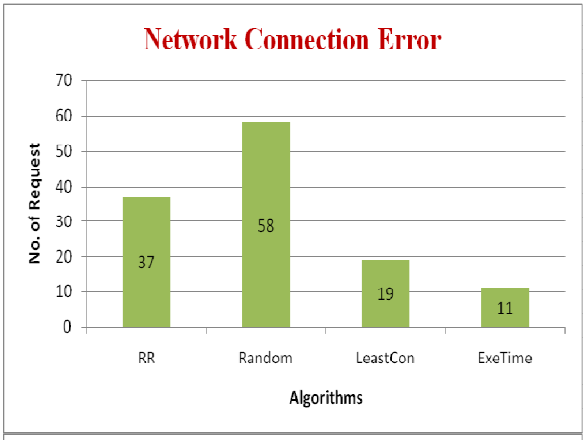
Our experiments are based on the system that consists of one or several duplicated web servers with different configuration and one system with load balancing algorithms that act as a request dispatcher. The request dispatcher is in charge of accepting web requests and scheduling one of the servers to execute the web requests.

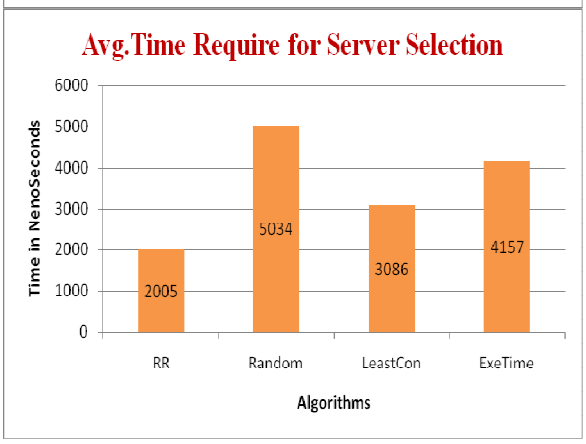
We have performed our tests on a 100Mbps LAN network. All systems are connected D-Link 24 port unmanageable switch via UTP cat-6 cable and our LAN is isolated from other networks. Our test includes THREE systems as replicated web servers, ONE system for request dispatcher and 20 client systems. Following table has the hardware and software configurations of these systems:

System	CPU	RAM	Operating System	Web Server / Software
Server – 1	Intel Core i3 3.10GHz	4 GB	Window 7	IIS 7.0
Server – 2	AMD Dual core 4400+ 2.30 Ghz	2 GB	Window 7	IIS 7.0
Server – 3	Intel Core2 Duo 2.66 GHz	2 GB	Window Vista	IIS 7.0
Dispatcher	AMD Dual core 4400+ 2.30 Ghz	2 GB	Window Vista	Java 6
Client	AMD Dual core 4400+ 2.30 Ghz	2 GB	Window XP	-




Performance Chart


Network Connection Error
No. of Request
70
60
50
40
30
20
10
0
37
58
19
11
RR
Random
LeastCon
ExeTime
Algorithms


Avg.Time Require for Server Selection
Time in NenoSeconds
6000
5000
4000
3000
2000
1000
0
2005
5034
3086
4157
RR
Random
LeastCon
ExeTime
Algorithms

6. RESULT ANALYSIS

	Round Robin	**Random**	**Least Connection**	**Execution Time Load**
Algorithm Type	Static	Static	Dynamic	Dynamic
Load Distribution	Equal	Randomized	As per active Connection	As per execution time load
Performance	Good	Good	Better	Better than LC Algo.
Network Error	Medium	High	Low	Less compare to LC Algo.
Overhead (Server Selection)	Low	High	Medium	Greater than LC Algo. But Less than Random Algo.

7. CONCLUSION

In this paper, we have studied the static and dynamic load balancing algorithms such as round robin, random, least connection and execution time load and implemented at layer 7 by developing software dispatcher in JAVA. We have measured the load distribution, average server selection time (over head) and the performance of the algorithms. From the result chart, we can say that dynamic algorithms performance is better than static algorithms. Execution Time Load (ETL) algorithm provide comparatively highest level of services because of the intelligent server selection. ETL algorithm has lesser network error compare to Least Connection and much less than compare to random and round robin algorithms. The only limitation of ETL algorithm is marginal larger overhead in

respect of server selection. Therefore execution time load algorithm is suitable for heterogonous web server farm.

8. REFERENCES:

1. Valeria Cardellini, Emiliano Casalicchio, Michele Colajani, Philip S Yu (2001). The State of the Art in Locally Distributed Web-server Systems, IBM Research Report, pp. 10-12, 26-41
2. Valeria Cardellini, Michele Colajani, Philip S Yu(1999). Dynamic Load Balancing on Web-Server System, IEEE Internet computing, Vol. 3, pp. 10-15
3. Michele Colajani, Philip S Yu, D. M. Dias (1998). Analysis of Task assignment polices in scalable distributed web systems, IEEE Transactions on Parallel and Distributed Systems, Vol. 9, pp. 7-13
4. Tony Bourke (2001). Server Load Balancing, Sebastopol, CA, O'REILLY, pp. 13-22
5. Hossien Bidgoli (2004), The Internet Encyclopedia, Vol. 2,John Wiley & Sons, Hoboken, New Jersey, pp. 499-511
6. Willy Tarreau (2006). Making application scalable with load balancing. Information Website: http://1wt.eu/articles/2006_lb/
7. http://www.ibm.com/software/network/dispatcher
8. http://www.cisco.com
9. http://www.foundrynet.com/pdf/wp-server-load-bal-web-enterprise.pdf

AUTHOR'S PROFILE:

Prof. Gautam J. Kamani: Asst. Prof, College of Agricultural Information & Tech., Anand Agril. University, Anand

Dr. N. N. Jani: Dean Faculty of Computer Science, Kadi Sarva Viswavidyalaya(University),Gandhinagar,

Dr. P. V. Virparia: Associate Professor, Dept. of Computer Science, S.P. University, Vallabh Vidyanagar