International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

WEB SERVER SCALABILITY OF WR DISPATCHING APPROACHES IN DISTRIBUTED WEB SERVER SYSTEMS

P.K. Kumaresan1, B. Sundaramurthy2

1 Professor & Vice Principal, Department of Computer Science and Engineering, Vinayaka Missions Kirupananda Variyar Engineering College, a constituent College of Vinayaka Missions Research Foundation Deemed to be University, Salem.

2 Associate Professor, Department of Computer Science and Engineering, Vinayaka Missions Kirupananda Variyar Engineering College, a constituent College of Vinayaka Missions Research Foundation Deemed to be University, Salem.

Abstract- Once the web site becomes a popular then single web server may not be able to handle high volume of incoming traffic. In order to achieve web server scalability, more servers need to be added to distribute the load among the servers. Scalability and availability can be provided by distributed web server architectures that assign the client request among the multiple server nodes. Load sharing and Load balancing techniques are applying for assign the request among the multiple web servers. In this paper we will review the distributed web architecture, request dispatching approach and dispatching algorithms for distributed web server systems.

1. INTRODUCTION

The continuing growth of the World-Wide Web is placing increasing demands on popular Web servers. Data traffic in the worldwide web (WWW), it is crucial to achieve the scalable performance of web servers. The overall performance and resource utilization can be improved by spreading document requests among a group of web servers systems called **distributed web server systems.**

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 system operates as a distributed web server. The server module is running on several machines and the manager module controls the server operation. The manager module itself can be distributed, to avoid bottleneck in high network loads. The distribution of the web server allows handling more requests and increasing the server throughput.

Distributed Web server systems, provide load balancing, cooperative caching and the effective use of geographic distribution of the server machines.

Load balancing is technique that distributes the load among multiple servers. Load balancing applies to all types of servers (application server, database server).

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. DISTRIBUTED WEB SYSTEM ARCHITECTURE

Distributed architectures can be differentiated depending on the name virtualization being extended at IP level. Basically two types of distributed web server systems architectures are there.

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

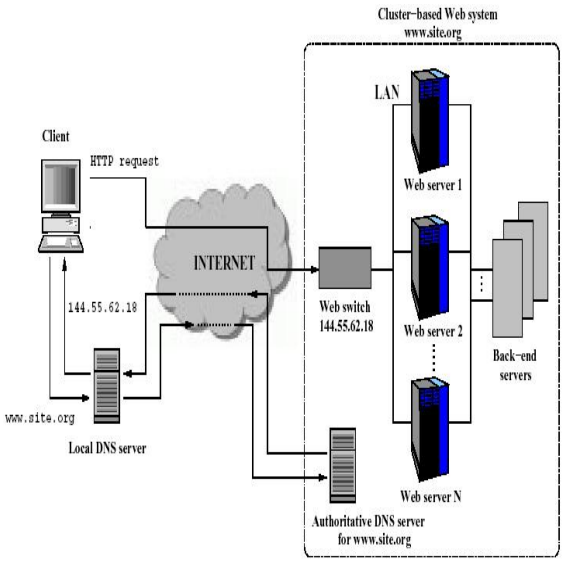
1. Cluster based web systems
2. Distributed web systems

2.1 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.

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.

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. In such a way, it acts as the centralized dispatcher of a distributed system with fine-grained control on client requests assignments.

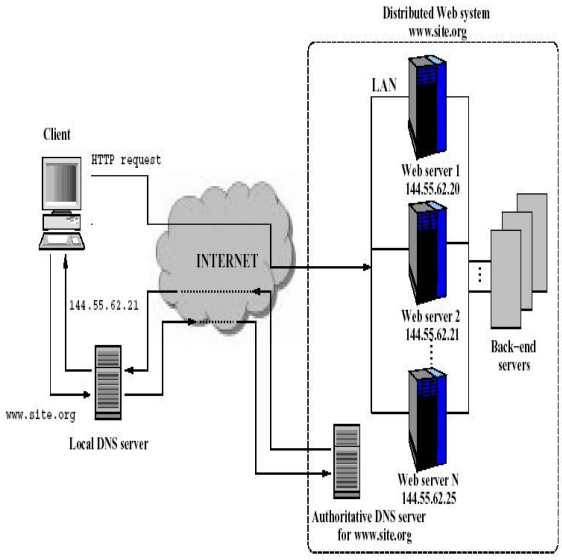


Web cluster architecture is only use for locally distributed web systems. It does not use for geographically distributed web systems.

2.2 Distributed web systems

A distributed Web system consists of distributed server nodes that assign a separately unique IP addresses. An IP address may be visible to client applications. This architecture is use for locally and geographically distributed web systems.

Unlike the cluster-based Web system, this architecture does not have a front-end node or web switch, so the client request assignment to a target Web server is typically carried out during the address resolution of the Web site name by the DNS mechanism.



2.3 Request Dispatching Approach

Load balancing is critical in managing high performance Web-server clusters. Request load must be spread across the Web-server nodes, so as to reduce the response time and provide WWW users with the best available quality of service. Load balancing among the servers can be achieved by several approaches with different degrees of effectiveness. There are four different approaches to distribute the incoming requests.

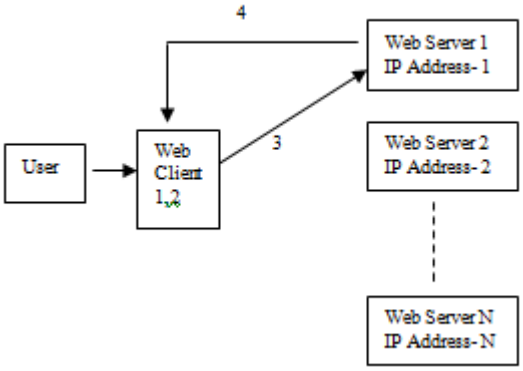
International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

1. Client based approach
2. DNS based approach
3. Dispatcher based approach
4. Server based approach

Client based Approach

In client based approach, The Web clients can play an active role in the request routing, if we assume that the Web clients know the existence of the replicated servers of the Web-server system.

The client request routing is achieved through a Java applet which is executed at the client side each time a user requests an access to the distributed Web-server system. As the applet knows the IP addresses of all server nodes in the Web cluster.



In above Figure shows the simplicity of the Web clients based approach.

1 : Received the user request

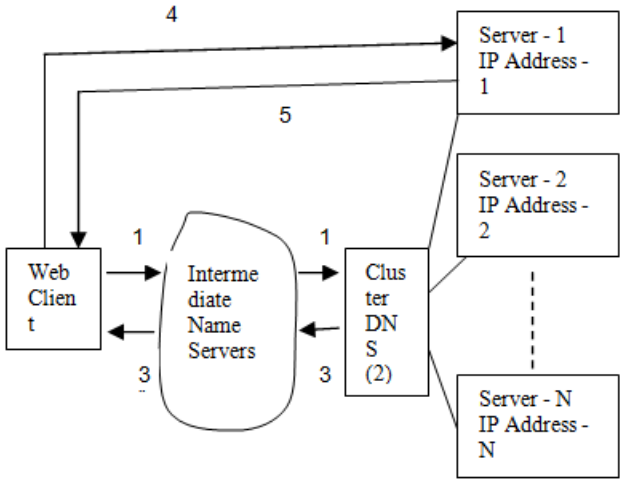
2 : The Web client is able to select a node of the cluster and, once resolved the address mapping,

3 : Submits the request to the selected node.

4 : This Web-server is responsible for responding to the client.

DNS based Approach

In DNS based approach the responsibility of spreading the requests among the servers is delegated to the cluster DNS, that is the authoritative DNS server for the domain of the cluster nodes. Through the translation process from the symbolic name (URL) to IP address, the cluster DNS can select any node of the Web-server cluster.



1 : Address Request URL
1" : Address request reaches the DNS Cluster
2 : Web server IP Address & TTL selection
3 : Address mapping (URL -> IP address -1) using DNS cluster
3" : Address mapping (URL -> IP address -1) using Name server
4 : Document Request to (IP address -1)
5 : Document Response from (IP address -1)

Dispatcher based Approach

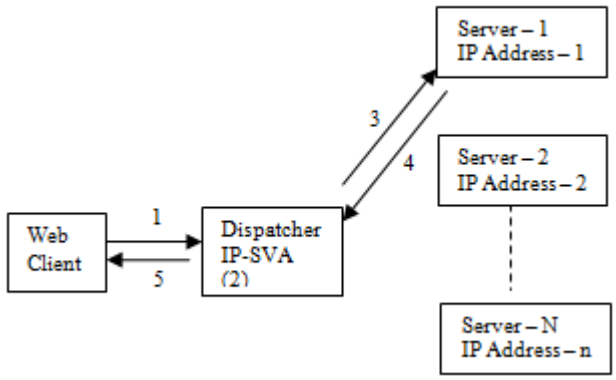
This approach provides the Web-server cluster with a single virtual IP address (IP-SVA). It is visible. In fact, this is the IP address of a dispatcher that acts as a centralized scheduler and in contrast to the DNS, has complete control on the routing of all client requests.

To distribute the load among the Web-server nodes, 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.

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

However, it is possible to integrate this architecture with some more sophisticated assignment algorithms pro-posed for distributed Web-server systems having a centralized dispatcher with full control on client requests.



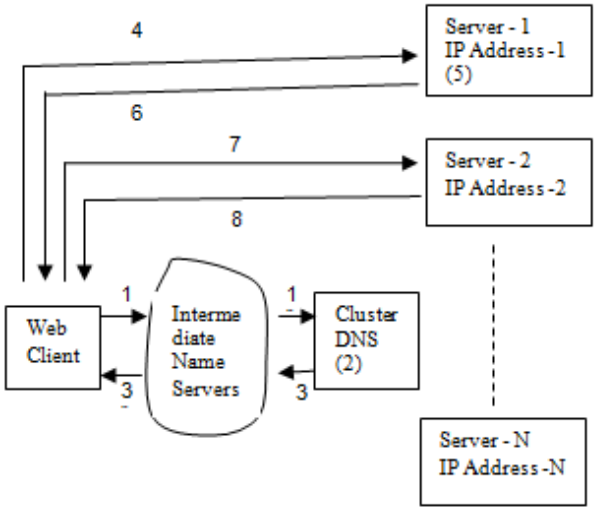
1
: Document request (IP-SVA)
2 : Web server selection
3 : Request passes to selected web server.
4 : Response document send to IP-SVA
5 : Response document send to Web Client

Server based Approach

The server-based techniques use a two-level dispatching mechanism: client requests are initially assigned by the cluster DNS to the Web-servers; then, each server may reassign a received request to any other server of the cluster.

Unlike the DNS-based and dispatcher-based centralized solutions, the distributed scheduling approach allows all servers to participate in balancing the load of the cluster through the request (re-)assignment mechanism.

The server-based proposals differ in the way of the redirection decision is taken and implemented. HTTP redirection and Packet redirection mechanism use in server base approach.



1 : Address Request URL
1" : Address request reaches the DNS Cluster
2 : Web server IP Address & TTL selection
3 : Address mapping (URL -> IP address -1) using DNS cluster
3" : Address mapping (URL -> IP address -1) using Name server
4 : Document Request to (IP address -1)
5 : Web server selection
6 : HTTP redirection
7 : Document request to (IP address -2)
8 : Document response from (IP address -2)

4. DISPATCHING ALGORITHMS

There are several dispatching algorithms for distribute the load among distributed web server systems. Main purpose of dispatching algorithm is selecting the best server that gives the quick response to client and balance the load. Dispatching algorithms are classified in different manner such as content blinding and content aware, load sharing and load balancing, static and dynamic etc… some algorithm are explain here.

Round Robin Algorithm:

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

It is a most popular and simple algorithm. Round robin algorithm uses an available server list. That assigns a task in circular way. RR algorithm use for homogeneous server farms.

Step 1: START
Step 2: Initialize **i** to **0** (i is static variable)
Step 3: Initialize **n** to **number of server** (n is constant variable)
Step 4: k=i % n
Step 5: select the k+1 web server.
Step 6: set i=i+1.
Step 7: STOP.

As an example, let us assume that S1, S2 & S3 are three web server systems. Then first request assign to S1, second request assign to S2, third request assign to S3 and fourth request assign to S1 and so on…

This algorithm presents a drawback: they are non-deterministic. This means that two consecutive requests from the same user will have a high chance of reaching two different servers. If user contexts are stored on the application servers, they will be lost between two consecutive requests. And when complex session setup happens (e.g.: SSL key negotiation), it will have to be performed again and again at each connection.

Least Loaded Algorithm

Least Loaded algorithm use a server load index for dispatching a request. A dispatch a request to the fewest loaded server. Updates the server load indexes file at regular interval or frequently.

This algorithm presents drawbacks: they are non-deterministic. It increases the network traffic because their updates the load indexes file for current load.

Step 1: START
Step 2: Receive the request from client
Step 3: Select the least loaded web server from least index file and dispatch the request.
Step 4: receive the response from server.
Step 5: STOP

Client Affinity algorithm using Cookie insertion:

In client affinity algorithm, Instead of assigning each new connection to a server only on the basis of the server state regardless of any past assignment, consecutive connections from the same client can be assigned to the same server for either performance or functional reasons.

As an example of performance reasons, consecutive SSL connections from the same client are assigned to the same server during the life span of the SSL key, so as to avoid time- and resource-consuming operations for SSL key negotiation and generation.

Cookie insertion method is best for client affinity. In this method inserting the server's identifier in a cookie added to the response. This way, the dispatcher has nothing to learn, it will simply reuse the value presented by the user to select the right server.

A major drawback is that when the server fails, the user has no easy way to escape, and keeps trying to reach the dead server.

Step 1: START
Step 2: Receive the request from client
Step 3: check the **Server Identifier** Cookie variable in HTTP Request Header
  If it is available then
    Select that variable value as a web server and dispatch request.

    Set newnode variable value is FALSE.
  Else
    Select least loaded server and dispatch the request.

    Set newnode variable value is TRUE.
  End if
Step 4: receive the response from server.
Step 5: if newnode=TRUE then
    Insert the **Server identifier** cookie variable in HTTP

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

Response Header
End if
Step 6: send the response to client
Step 7: STOP

Content Aware Algorithm:

Content aware algorithm assigns a request using the request content information. Content aware policy algorithms operate at HTTP level. The request URL information use for different purpose such as To use specialized server nodes to provide different Web services (specialized servers), such as streaming content, dynamic content, and to partition the Web content among the servers.

It might sound very common, but it is still rarely done. On many applications, about 25% of the requests are for dynamic objects, the remaining 75% being for static objects. The easiest solution consists in relying on a reverse proxy cache in front of the server farm, which will directly return cacheable contents to the users without querying the application servers. The cleanest solution consists in dedicating a lightweight HTTP server to serve static contents and other server serves the dynamic contents.

Server selections depend on the request content type. If content is static then select a web server that provide static information. If content is dynamic then select a web server that provide the dynamic content.

Step 1: START
Step 2: Receive the request from client
Step 3: Check the request content type
If request content is static then
Select a static content server
Else
Select a dynamic content server
End if
Step 4: Send the request to selected server
Step 5: receive the response from server
Step 6: send the response to client
Step 7: STOP

5. CONCLUSION

Replication of information among multiple Web servers is necessary to support high request rates to popular Web sites. In this paper, we have studied distributed Web systems architectures, request dispatching approach and algorithms. Load sharing and balancing is important in request dispatching. Load balancing system to take decisions according to existing current servers load such as least loaded. Load sharing system only distributes the load among web systems such as round robin, client affinity and content aware. The dispatching approach depends on the web systems architecture and dispatching algorithms selection depends on the content distribution.

6. REFERENCES

1. Valeria Cardellini, Emiliano Casalicchio, Michele Colajani, Philip S Yu "The State of the Art in Locally Distributed Web-server Systems"
2. Valeria Cardellini, Michele Colajani, Philip S Yu, "Dynamic Load Balancing on Web-Server System"
3. Michele Colajani, Philip S Yu, D. M. Dias "Analysis of Task assignment polices in scalable distributed web systems"
4. Willy Tarreau, "Making application scalable with load balancing"
5. Dan Mosedale , William Foss , Rob McCool, Lessons Learned Administering Netscape's Internet Site, IEEE Internet Computing, v.1 n.2, p.28-35, March 1997 [doi>10.1109/4236.601086]
6. Chad Yoshikawa , Brent Chun , Paul Eastham , Amin Vahdat , Thomas Anderson , David Culler, Using smart clients to build scalable services, Proceedings of the annual conference on USENIX Annual Technical Conference, p.8-8, January 06-10, 1997, Anaheim, California
7. Michael Baentsch , Lothar Baum , Georg Molter , Steffen Rothkugel , Peter Sturm, Enhancing the Web's Infrastructure: From Caching to Replication, IEEE Internet Computing, v.1 n.2, p.18-27, March 1997 [doi>10.1109/4236.601083]
8. Thomas T. Kwan , Robert E. McGrath , Daniel A. Reed, NCSA's World Wide Web Server: Design and Performance, Computer, v.28 n.11, p.68-74, November 1995 [doi>10.1109/2.471181]
9. Michele Colajanni , Philip S. Yu , Daniel M. Dias, Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems, IEEE Transactions on Parallel and Distributed Systems, v.9 n.6, p.585-600, June 1998 [doi>10.1109/71.689446]
10. D. M. Dias , W. Kish , R. Mukherjee , R. Tewari, A scalable and highly available web server, Proceedings of the 41st IEEE International Computer Conference, p.85, February 25-28, 1996
11. Singhai , S. -B. Lim, The SunSCALR Framework for Internet Servers, Proceedings of the The Twenty-Eighth Annual International Symposium on Fault-Tolerant Computing, p.108, June 23-25, 1998

International Journal of Computer Science
Scholarly Peer Reviewed Research Journal - PRESS - OPEN ACCESS
ISSN: 2348-6600
http://www.ijcsjournal.com Volume 6, Issue 1, No 4, 2018. ISSN: 2348-6600
Reference ID: IJCS-345 PAGE NO: 2304-2310

12. D. Mosedale, W. Foss, and R. McCool, "Lessons Learned Administering Netscape's Internet Site," IEEE Internet Computing, Vol. 1, No. 2, Mar.-Apr. 1997, pp. 28–35.
13. C. Yoshikawa et al., "Using Smart Clients to Build Scalable Services," Proc. Usenix 1997, Usenix Assoc., Berkeley, Calif., Jan. 1997.
14. M. Baentsch, L. Baum, and G. Molter, "Enhancing the Web's Infrastructure: From Caching to Replication," IEEE Internet Computing, Vol. 1, No. 2, Mar.-Apr. 1997, pp. 18–27.
15. T.T. Kwan, R.E. McGrath, and D.A. Reed, "NCSA's World Wide Web server: Design and Performance," Computer, Vol. 28, No. 11, Nov. 1995, pp. 68–74.
16. M. Colajanni, P.S. Yu, and D.M. Dias, "Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems," IEEE Trans. Parallel and Distributed Systems, Vol. 9, No. 6, June 1998, pp. 585–600.
17. D.M. Dias et al., "A Scalable and Highly Available WebServer," Proc. 41st IEEE Computer Soc. Int'l Conf., IEEE Computer Soc. Press, Los Alamitos, Calif., Feb. 1996, pp. 85–92.
18. R.J. Schemers, "lbmnamed: A Load Balancing Name Server in Perl," Proc. 9th Systems Administration Conf., Usenix Assoc., Berkeley, Calif., Sept. 1995.
19. M. Beck and T. Moore, "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," Proc. 3rd Workshop WWW Caching, Manchester, England, 1998.