DOI 10.29042/2018-3023-3030

Helix Vol. 8(1): 3023- 3030

NDLB: Nearest Dispatcher Load Balancing approach for Web Server Cluster

Kadiyala Ramana[1], M.Ponnavaikko[2] and A.Subramanyam[3]

[1] Research Scholar, Department of Computer Science and Engineering, SRM University, India, [2] Provost, Vinayaka Missions University, Chennai – India, [3]Dean, Annamacharya Institute of Technology and Sciences (Autonomous), Rajampet – India

***Email:** ramana.it01@gmail.com*

Received : 20th October 2017, Accepted : 11th November 2017, Published : 31st December 2017

Abstract

With the growing popularity of web based applications, the primary and consistent resource in the infrastructure of World Wide Web are web server clusters. Overtly in dynamic contents and database driven applications, especially at heavy load circumstances, the performance handling of clusters is a solemn task. A novel distributed web server system NDLB (Nearest Dispatcher Load Balancing) is proposed in this paper which uses both DNS and Dispatcher to forward the client requests efficiently to the servers in a user transparent way. This system conquers superior response time than other distributed web server architectures and also poises loads between servers within the clusters effectively. However, the NDLB architecture is accessible and more indulgence in both Dispatcher and DNS; Moreover, if the cluster capability is less than the request rate it offers a load balancing architecture.

Keywords: world wide web, DNS, Dispatcher, response time, load balancing, web server, cluster

1. Introduction

The volume of the information available online and services available for the internet users increased with the blast of the world wide web. The thriving of information and various service demands has made a sensational pressure on the World Wide Web (WWW) infrastructure. To serve a large number of client request they need advanced web server systems. Because of their scalability, availability and cost-effectiveness distributed web server cluster architectures became more popular instead of using one web server, which has high processing capabilities.

In 1995, the number of internet users was less than 1% in the world population, whereas today it is 40%. In 2016, there were 3.5 billion internet users while in 2005 there were 1.02 billion internet users [1]. With the fast growth of internet traffic, most popular websites need to scale up their server capacities. The popular way to provide a list of alternative, or equivalent mirrored servers at different locations. The mirrored servers are not transparent to the users and it is hard to provide load balancing and fault-tolerance [2]. The technique which is used to redistribute the workload from loaded servers to idle servers in order to improve the performance is called Load balancing. The most promising approach to handle popular web sites is to maintain a virtual single interface and to use a distributed architecture. A web cluster is known to be a compilation of servers which works jointly as a solitary articulate system for providing highly & scalable web services. It relies on load balancing techniques where it shares service traffic efficiently between its back-end servers and visibly to the clients. The scalability is termed as the capacity in system measurement where to meet the escalating demands as service traffic. The capacity of the system is determined based on the support of number of parallel connections of servers per second without affecting of momentous queuing delay in the interior infrastructure.

By taking advantage of the server redundancy, load balancing techniques improves the system availability [3]. The ability of a server to provide endless services over time is called Availability and it is deliberated as uptime percentage. When a cluster server declines or abort, the load will routinely redistribute with slight or refusal brunt laying the service among other available services.

The servers in the Web server cluster are not essentially situated in the equivalent site and they will be located in diverse biological locations. In proxy servers they are all located at different locations. Because of the rapid increase of Internet, the broadcast time is an important recital factor in network service.

In web cluster, load balancing involves a several major concerns. The primary concern is measurement of work load. In different applications, workload has different meanings. In web services, the client request is a basic building block of load balancing and its response lively connections is a simple server load index [4].

Request distribution policy and mechanism are the two additional core issues in the load. For each incoming request from the clients, the load balancing policy will determine the target server allocation policy competently and evidently to

clients. Numerous load balancing mechanisms and policies are available with diverse characteristics.

2. Existing Architectures

The classification of the existing architectures is made into five classes, based on which component dispatches the incoming client request between servers. In these five classes, the first one requires modification of software at client side and in remaining four one or more elements will be affected in the web server cluster. The five approaches are

- Client based approaches
- DNS based approaches
- Dispatcher based approaches
- Server based approaches
- Dynamic Dispatcher based approaches (Uses both DNS and Dispatcher)

2.1. Client based Approaches

In these approaches, web client entity (web browser or proxy servers at client) will be responsible for selection of server. No processing will be done at server side for selection of the same. The dispatching of client requests to various replicated servers will be done by using client software.

Web clients: In replicated web-server architecture, all the web clients are aware of existence of the servers. Netscape's Approach [5] and Smart Clients [6] are the two schemes used for selection of server at client side.

Client's DNS resolver: For I2-DSI System, Beck and Moore [7] proposed this scheme. In this, at client side they have used a DNS resolver which issues probes for the servers and choose the server based on earlier access information or response time from the client.

Client-Side Proxy: The proxy server is similar to web client which redirects client request to web server nodes. Baentsh et. al proposed an approach, which incorporate server replication and caching [8]. In client-side proxy, by implementing Web Location and Information Service they record replicated URL addresses and redirects requests to the selected server.

The above approaches reduce the load on servers by perform dispatching at client side. However, the deficiency was limited applicability because the user must know that the architecture is distributed.

2.2. DNS based approaches

In these approaches, an authorized DNS is used at server side which maps the domain name to an IP Address of any one server in the cluster by using numerous scheduling strategies. The selection of the server will be done by the server-side DNS which does not suffer from the problems that is faced by Client-based approaches. The authorized DNS have limited control over the requests which reach the server cluster. To control network traffic between client and DNS so many caching techniques (Like web browsers, DNS Resolvers, Intermediate Name Servers etc.) will be used.

DNS not only provides IP addresses of the server nodes, it also includes a validity period called Time-To-Live (TTL) value in name resolving process. When this value expires, the mapping request is sent to the authorized DNS otherwise it resolved by any of the caching techniques mentioned above. No one can set this value as low or zero because it doesn't work for non-cooperative name servers and caching at client side. This will increase the network traffic and becomes bottleneck to itself. Some of the DNS based approaches are elucidated in [9] and [10]. Based on the scheduling algorithms which are used for selection of server and values of TTL the DNS based approaches are classified as the below.

Constant TTL algorithms: Based on the server and client state information (location. Load etc.) these algorithms are classified as System stateless algorithms [11], Server state based algorithms, Client state based algorithms [12] and server & client state based algorithms.

Dynamic TTL algorithms: In these algorithms, the TTL value is dynamically change when URL is mapping to an Address [9]. These are classified as Variable TTL algorithms and Adaptive TTL algorithms.

In all the above approaches when replicated object change from one place to another, this requires change in mapping. Hence all the approaches mostly support static replication schemes rather than dynamic replication schemes. These approaches also have limited control among requests because of mapping which is performed at different levels. Because of packet size limitations in UDP, these approaches cannot handle beyond 32 web servers for a public URL [10].

2.3. Dispatcher-based approaches

These approaches provide full control to the server-side entity over client requests. In these, the DNS will return the address of a dispatcher, which dispatch the client requests to one of the servers available in the cluster. At server side the dispatcher acts as a Centralized scheduler, which controls all client request distribution. This approach is much more transparent because for outside world it looks like a single IP address. These mechanisms were characterized as Packet single rewriting [13], Packet double rewriting [14] and packet forwarding [15]. Various Dispatcher based approaches are elucidated in [16] and [12].

In this approach dispatcher is the single decision entity. Whenever the request rate increases rapidly, it will lead to bottleneck at the dispatcher. Furthermore, this will system will fail because of its centralized nature. The performance also degrades because of modification and rerouting of each request through the dispatcher.

2.4. Server-based approaches

In these approaches, dispatching will be done at two-levels. First at Cluster DNS then at each server (the request received to any of the servers in the cluster if it is required). The problem of Client request non-uniform load spreading and inadequate control of DNS was solved using this approach. Some of the Server based approaches are elucidated in [17], [13] and [18].

These approaches increase the latency time observed by the clients because of its redirection mechanisms.

2.5. Dynamic Dispatcher Based Approaches

This approach is based on DNS and Dispatcher. DNS Server will initially communicate with the server and converts URL to an IP Address. One Dispatcher is associated to all of the web servers available in the cluster. And every Dispatcher is associated to the Dispatcher Selector. Each Dispatcher comprises of a Load Collector, who gathers the load of every web server and an Alarm Monitor, who monitors the Load and provisionally stops the services of web server whose load is very high. Every server comprises a Load Checker and a Request Counter who computes and directs the information about load on web server.

In this approach, first the client request will be sent to DNS. The DNS will forward the request to Dispatcher Selector who forwards the request to the Dispatcher having minimum loaded web server in the cluster. Dispatcher analyses the Load collector which receives the data from the Load checker, Request counter and also checks the Alarm monitor component for the least loaded web server among all the servers in the cluster. Dispatcher forwards the load information about the minimum loaded server to the Dispatcher selector. Dispatcher selector forwards the IP address of minimum load Dispatcher to the DNS who return this client. Then Client sends the request to the web server and get response directly from the web server [19].

3. Proposed Architecture

The proposed architecture has been designed in a way such that it yields better response time, throughput and number of requests served in a better way when compared to the existing approaches discussed above.

3.1. Design

Figure 1 depicts a typical design of distributed internet server model projected during this work. In this design, one Virtual IP address is allotted to the web server cluster, which is the IP address of the dispatcher. This is able to recognize every server in cluster using a private address and redistributes the work load between the servers based on random algorithm. Moreover, the selected web server sends the response directly to the client.

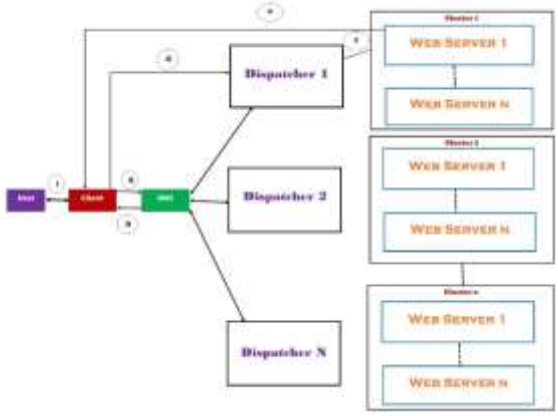

Figure 1: Distributed Web Server Model

4. Implementation Setup

Implementation of the experimental test bed with both software and hardware configurations as explained below.

4.1. Hardware Setup

The web server cluster consists of 10 computers configured as follows. One computer is used as DNS, two computers are used as dispatchers, 5 computers are used as web servers and 2 computers as clients. Two web servers are under control of one dispatcher and remaining three web servers under another dispatcher. To provide transparency to the clients, one Virtual IP address is used for each dispatcher. The web server, each has an Intel i5-4590S 3.0GHz CPU with 4 GB of DDR RAM. The dispatcher is an Intel i5-4030 302GHz CPU with 4 GB of DDR RAM.

4.2. Software Setup

Client-Side Software

To scrutinize the performance of the proposed system, a JMeter testing tool has been taken as a load testing tool for measuring and analyzing the performance of various services, with a focus on internet applications. JMeter is designed for testing for web applications and further extended to test the other functions.

Apache JMeter is used to test the performances of both dynamic and static resources. It is also used to simulate an overloaded web server, object or network to test its strength and investigate overall performance under different load types.

DNS Software

As discussed earlier, DNS-based schemes for load-balancing require that DNS returns the IP address of server or cluster, based on the state information.

Current application of the domain name server (BIND) provide such support. It supports random and round-robin selections of IP address.

Server Software

All the server machines will run apache web server. But one could use any other software without necessitating any change in the architecture. In addition to the web server, also execute another process that gathers state information like load averages, Memory and CPU utilization, number of server processes running and number of active connections to handle client requests etc.

Dispatcher Software:

Dispatcher is responsible for dispatching requests within the cluster. Depending on the scheme, it can take into account loads on various servers and previous request rate of the clients, to choose a particular server. It also keeps a table of client's IP addresses and port number so that successive requests from one client can be sent to the same server.

4.3. Pseudo Code

Client_Module:

```
{
/* creates and forwards the client request*/
Client.request(Ureq);
}
```

DNS_Module:

```
nDispatcher : total available dispatcher
Dispatcherlistarr[cnt] : array of available
dispatcher with IP Address
Calculate RTT by sending probes to all
dispatcher for every 2 mins
for each dispatcher in list in ascending order of
rtt
      if(available capacity of dispatcher >
request rate of client)
{
            reduce available capacity of cluster by
client request rate
            return(dispatcher IP address);
      }
```

Dispatcher_Module:

```
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
```

5. Implementation Results

In the subsequent sections, the results of experiments with JMeter tool are explained. The load is changed and the server cluster CPU Utilization, Average Response Time, throughput, Number of requests served and Error Rate for the proposed approach are measured. Results of all the existing approaches are provided for comparison purpose.

5.1. CPU Utilization:

CPU utilization is mainly used to estimate the system performance, which is the sum of work handled by the Central Processing Unit. Figure 2 shows how the CPU Utilization varies for DNS based [20], Dispatcher based [21], Dynamic Dispatcher based [19] and NDLB approaches as the number of client requests increases from 10000 to 30000 for JMeter workload. As the number of client requests are increased the CPU Utilization begins to decrease because of web server CPU reaches the maximum utilization which starts the queuing.

Based on the generated number of client requests and served requests the percentage of CPU Utilization has been calculated as shown in Table 1. For NDLB approach the CPU Utilization starts at 99.03% for 10,000 requests and it decreases to 75.28% for 30,000 requests. The DNS based, Dispatcher based and Dynamic Dispatcher based Approaches do not perform as well as NDLB Approach. For 10,000 clients request the CPU utilization for DNS based approach is 85.51%, for Dispatcher based approach 89.92% and for Dynamic Dispatcher based approach 95.23%. For 30,000 clients request the CPU utilization for DNS based approach is 61.16%, for Dispatcher based approach 64.45% and for Dynamic Dispatcher based approach 69.77%. So, the higher CPU Utilization is provided by NDLB Approach.

Table 1: Comparative analysis of CPU Utilization with the proposed NDLB approach

Number of Requests Generated	CPU Utilization (%)			
	DNS based Web Server System	Dispatcher based Web Server System	Dynamic Dispatcher based Web Server System	Nearest Dispatcher based Web Server System
10000	85.51	89.92	95.23	99.03
20000	61.88	78.86	81.65	87.54
30000	61.16	64.45	69.77	75.28

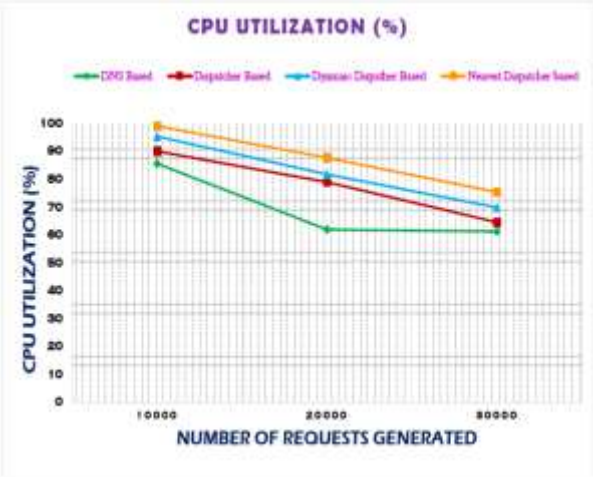

Figure 2: Comparative analysis of CPU Utilization with the proposed NDLB approach

5.2. Average Response Time:

Response time is the aggregate sum of time it takes to reply to a request for service. Figure 3 shows how the Average Response Time varies for DNS based, Dispatcher based, Dynamic Dispatcher based and NDLB approaches as the number of client requests increases from 10000 to 30000 for JMeter workload. As the number of client requests are increased the Average Response Time begins to increase because of web server reaches the maximum utilization which starts the queuing.

Based on the generated number of client requests and served requests the Average Response Time has been calculated as shown in Table 2. For NDLB approach the Average Response Time starts at 340ms for 10,000 requests and it increases to 1136ms for 30,000 requests. The DNS based, Dispatcher based and Dynamic Dispatcher based Approaches do not perform as well as NDLB Approach. For 10,000 clients request the CPU utilization for DNS based approach is 555ms, for Dispatcher based approach 459ms and for Dynamic Dispatcher based approach 371ms. For 30,000 clients request the CPU utilization for DNS based approach is 1487ms, for Dispatcher based approach 1351ms and for Dynamic Dispatcher based approach 1254. So, the less Average Response Time is provided by NDLB Approach.

Table 2: Comparative analysis of Average Response Time with the proposed NDLB approach

Number of Requests Generated	Average Response Time (msec)			
	DNS based Web Server System	Dispatcher based Web Server System	Dynamic Dispatcher based Web Server System	Nearest Dispatcher based Web Server System
10000	555	459	371	340
20000	1130	903	855	739
30000	1487	1351	1254	1136

Figure 3: Comparative analysis of Average Response Time with the proposed NDLB approach

5.3. Throughput

Throughput is a quantity of how many units of work are being handled. within the case of load testing, this is normally hits per second, also referred to as requests per second. Figure 4 shows how the Throughput varies for DNS based, Dispatcher based, Dynamic Dispatcher based and NDLB approaches as the number of client requests increases from 10000 to 30000 for JMeter workload. As the number of client requests are increased the throughput begins to increase.

Based on the generated number of client requests and served requests the throughput has been calculated as shown in Table 3. For NDLB approach the throughput starts at 152.4 requests/second for 10,000 requests and it increases to 219.8 requests/second for 30,000 requests. The DNS based, Dispatcher based and Dynamic Dispatcher based Approaches do not perform as well as NDLB Approach. For 10,000 clients request the CPU utilization for DNS based approach is 130.5 requests/second, for Dispatcher based approach 137.3 requests/second and for Dynamic Dispatcher based approach 147.8 requests/second. For 30,000 clients request the CPU utilization for DNS based approach is 177.6 requests/second, for Dispatcher based approach 185 requests/second and for Dynamic Dispatcher based approach 203.5 requests/second. So, the high throughput is provided by NDLB Approach.

Table 3: Comparative analysis of Throughput with the proposed NDLB approach

Number of Requests Generated	Through Put (Req/Sec)			
	DNS based Web Server System	Dispatcher based Web Server System	Dynamic Dispatcher based Web Server System	Nearest Dispatcher based Web Server System
10000	130.5	137.3	147.8	152.4

20000	138.9	174.6	180.2	192.7
30000	177.6	185.0	203.5	219.8

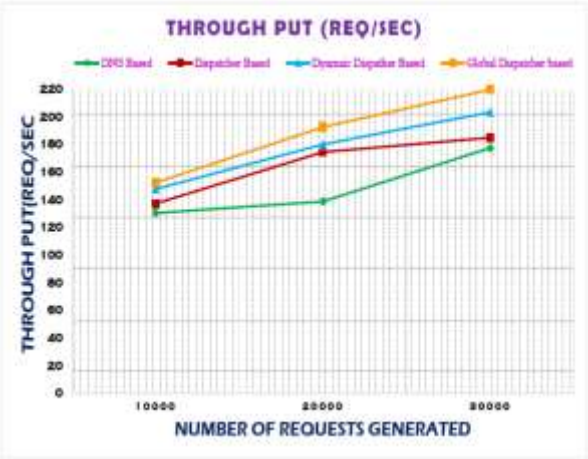

Figure 4: Comparative analysis of Throughput with the proposed NDLB approach

5.4. Number of Requests Served

A universal and generally accepted definition of performance is to observe the system output that characterizes the number of successfully served requests from a total of input requests. Figure 5 shows how the Number of requests served varies for DNS based, Dispatcher based, Dynamic Dispatcher based and NDLB approaches as the number of client requests increases from 10000 to 30000 for JMeter workload. As the number of client requests are increased the Number of requests served begins to decrease.

Based on the generated number of client requests, Number of requests served has been calculated as shown in Table 4. For NDLB approach the Number of requests served starts at 9903 requests for 10,000 requests and it decreases to 22585 requests for 30,000 requests. The DNS based, Dispatcher based and Dynamic Dispatcher based Approaches do not perform as well as NDLB Approach. For 10,000 clients request the Number of requests served for DNS based approach is 8551 requests, for Dispatcher based approach 8992 requests and for Dynamic Dispatcher based approach 9523 requests. For 30,000 clients request the Number of requests served for DNS based approach is 18349 requests, for Dispatcher based approach 19336 requests and for Dynamic Dispatcher based approach 20931 requests. So, the high Number of requests served is provided by NDLB Approach.

Table 4: Comparative analysis of Number of Requests Served with the proposed NDLB approach

Number of Requests Generated	Number of Requests Served			
	DNS based Web Server System	**Dispatcher based Web Server System**	**Dynamic Dispatcher based Web Server System**	**Nearest Dispatcher based Web Server System**
10000	8551	8992	9523	9903
20000	12376	15771	16329	17507
30000	18349	19336	20931	22585

Figure 5: Comparative analysis of CPU Utilization with the proposed NDLB approach

5.5. Error Rate

Error Rate is a noteworthy metric because it measures "performance failure" in the application. It tells us how many failed requests are happening at a certain point in time of our load test. In many load tests, this climb in Error Rate will be extreme. This speedy rise in errors says us where the target system is stressed beyond its capability to deliver acceptable performance.

Figure 6 shows how the Error Rate varies for DNS based, Dispatcher based, Dynamic Dispatcher based and NDLB approaches as the number of client requests increases from 10000 to 30000 for JMeter workload. As the number of client requests are increased the Error Rate begins to increase.

Based on the generated number of client requests and Number of requests served, Error Rate has been calculated as shown in Table 5. For NDLB approach the Error Rate starts at 1.97% for 10,000 requests and it increases to 24.72% for 30,000 requests. The DNS based, Dispatcher based and Dynamic Dispatcher based Approaches do not perform as well as NDLB Approach. For 10,000 clients request the Error Rate for DNS based approach is 14.49%, for Dispatcher based approach 10.08% and for Dynamic Dispatcher based approach 4.77%. For 30,000 clients request the Error Rate for DNS based

approach is 38.84%, for Dispatcher based approach 35.55% and for Dynamic Dispatcher based approach 30.23%. So, the less Error Rate is provided by NDLB Approach.

Table 5: Comparative analysis of Error Rate with the proposed NDLB approach

Number of Requests Generated	Error Rate (%)			
	DNS based Web Server System	Dispatcher based Web Server System	Dynamic Dispatcher based Web Server System	Nearest Dispatcher based Web Server System
10000	14.49	10.08	4.77	1.97
20000	38.12	21.14	18.35	12.46
30000	38.84	35.55	30.23	24.72

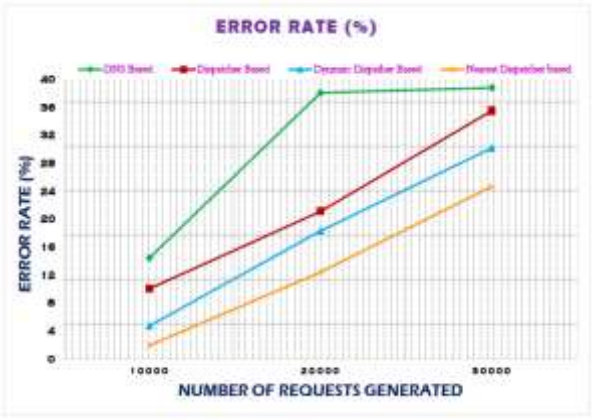

Figure 6: Comparative analysis of CPU Utilization with the proposed NDLB approach

7. Conclusion

In this paper a novel approach is proposed for dynamic load balancing with both DNS and Dispatcher. DNS calculates the round-trip time to the dispatcher of each cluster and forwards the IP address of the Dispatcher, to the client which has the low round trip time. Dispatcher selects the appropriate server in the cluster using random method. A model web server cluster was employed and equipped with the proposed algorithm. The investigational results attained from the JMeter tool confirm the enhancements in clusters performance in terms of CPU utilization, Error Rate, Average Response Time, Number of Requests served and Throughput in contrast to the DNS, Dispatcher and Dynamic Dispatcher based approaches. This approach also provides availability and scalability when compared to the existing approaches.

REFERENCES

[1] http://www.internetlivestats.com/internet-users/

[2] Kadiyala Ramana and M.Ponnavaikko, "Web Cluster Load Balancing Techniques: A Survey", International Journal of Applied Engineering Research, Volume 10, Number 19, pp 39983-39998, 2015

[3] Cheng Zhong Xu, Scalable and Secure Internet Services and Architecture (Chapman & Hall/Crc Computer & I), Chapman & Hall/CRC, 2005

[4] C. Xu, "Scalable and Secure Internet Services and Architecture", Chapman &Hall/CRC, 2005.

[5] D. Mosedale, W. F., and M Cool, R. "Lessons learned administering Nets ape's site". Internet Computing Vol. 1 No. 2 (March-April 1997), 28-35.

[6] Yoshilakawa, C., Chun, B., and Eastham, P. "Using smart clients to build scalable services", Proceedings of Usenix 1997 (January 1997).

[7] Beck, M., and Moore, T. "The Internet-2 Distributed Storage Infrastructure project: An architecture for Internet content channels". 3rd Int'l WWW Caching Workshop, Manchester, UK (June 1998). http://www.ahe.ja.net/events/workshop/18/mbeck2.html.

[8] Baentsh, M., Baum, L., and Molter, G. "Enhancing the Web's Infrastructure: From Caching to Replication". Internet Computing Vol. 1. No. 2 (March-April 1997), 18-27

[9] Colajanni, M., Yu, P. S., and Cardelini, V. "Dynamic load balancing on geographically distributed heterogenous web servers". IEEE 18th Int'l Conference on Distributed Computing systems (May 1998), 295-302.

[10] Cardelini, V., Colajanni, M., and Yu, P. S. "Dynamic load balancing on web server systems". IEEE Internet Computing, vol 3, no 3 (May-June 1999), 28-39.

[11] Kwan, T. T., McGrath, R. E., and Reed, D. A. "NCSA's World Wide Web server: Design and performance". IEEE Computer, no. 11 (November 1995), 68-74.

[12] Cisco Systems Inc. "Distributed Director White Paper". http://www.cisco.com/warp/public /cc/cisco/mkt/scale/distr/tech/d_wp.htm, 1997

[13] D. Sanghi, P. Jalote, P. Agarwal, N. Jain, and S. Bose, "A testbed for performance evaluation of load-balancing strategies for Web server systems," Software—Practice and Experience, vol. 34, no. 4, pp. 339–353, 2004.

[14] Anderson, E., Patterson, D., and Brewer, E. "The Magi router: an application of fast packet interposing". http:// s.berkeley.edu/~eanders/projects/magi router/osdi96-mrsubmission.ps.

[15] G.D.H. Hunt, G.S. Goldzsmit, R. K., and Mukherjee, R. "Network Dispatcher: A connection router for scalable internet services". Proceedings of 7th Int'l World Wide Web Conference (April 1998).

[16] Damani, O., Chung, P., and Kintala, C. "ONE-IP: Techniques for hosting a service on a cluster of machines". Proceedings of 41st IEEE Computing Society Int'l Conference (February 1996), 85-92.

[17] Andersen, D., Yang, T., Holmedahl, V., and Ibarra, O. H. "SWEB: Towards a scalable World Wide Web-server on multi computers". Proc. of 10th IEEE Int'l Symp. on Parallel Processing, Honolulu (April 1996), 850-856.

[18] Akamai Inc. "How FreeFlow Works". http://www.akamai.com/service/howitworks.html.

[19] Harikesh Singh, Dr. Shishir Kumar "Dispatcher Based Dynamic Load Balancing on Web Server System, International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September, 2011.

[20] Y. S. Hong , J. H. No , S. Y. Kim, "DNS-Based Load Balancing in Distributed Web-server Systems", Proceedings of the The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06), p.251-254, April 27-28, 2006

[21] Pao, T. L., & Chen, J. B., "The scalability of heterogeneous dispatcher based web server load balancing architecture", In Proceedings of the 7th international conference on parallel and distributed computing, application and technology, pp. 213–216, 2006.

