International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.2, Issue No.5, pp : 376-382 1 May. 2013

A Fault-Tolerant Based Load Balancing Model in Distributed Web Environment

Shyam Bhati[1], Sandeep Sharma[2], Karan Singh[3]

[1,3]Department of CSE, [2]Department of ECE, School of ICT, Gautam Buddha University, Greater Noida, U.P., India

[1]shyam.thakur031@gmail.com, [2]sandeepsharma@gbu.ac.in, [3]karan@gbu.ac.in

Abstract—*Internet users are increasing exponentially since the last decade which drastically increases the internet traffic. This gives us a challenge to manage the requests from the client on the web server. In the distributed environment the number of client requests web services are increasing, the quality of service in terms of response time is decreasing. Distributing the load of a single Web Server into a cluster of distributed Web Servers is one solution for same problem. Load balancing is an effective way to handle the load of a single Web Server to multiple Web Servers. In this paper we are discussing some of the load balancing techniques and their associated problems further we propose a model of fault-tolerant based load balancing in distributed web environment.*

Keywords—Web Server, Load balancing, Web-server cluster, Web service, Distributed web environment.

I. INTRODUCTION

At present, there is a tremendous demand of high performance internet web services. Single web server can't provide high performance web services. Load of the standalone web server must be divided into cluster of web servers to achieve reliable and scalable web services. In contrast, distributed solutions are a natural and popular way to improve the reliability, scalability, efficiency, and availability of a system [20]. Distributed web servers have several advantages such as higher computing power and fault tolerance. Complexity involved in scheduling and balancing the load of individual servers are the disadvantages of distributed web servers. How to create good quality web services to meet the increasing load demand has become an urgent issue. This demand can be attributed to the following [18].

1) Scalability: When the service load increases, the system can be expanded to meet demand, without reducing the quality of service.

2) Availability: Even if there is some hardware and software failure, the entire system of services must be available 24 hours a day, 7 days a week.

3) Manageability: The entire system may be physically large, but it should be easy to manage.

4) Cost-effectiveness: The entire system implementation is economical, easy to pay.

How to solve these problems, in the past, we considered some factors such as improving the server CPU speed, , improving the server operating system and increasing memory capacity. But only by improving the performance of the hardware in single web server system, we could not really solve the problem, because if we improve the performance of a single web server system to a certain limit, its price increases automatically, and the single web server performance of the system is always limited. In general, a PC server can support many concurrent accesses, more advanced server can support a large no of concurrent accesses, but that capability cannot meet the requirements of larger sites either. When some major event occurs, the web server access will dramatically increase, resulting in web server bottlenecks. Multiple servers must used to provide web services, and web site requests should be distributed to these servers to share in order to provide the ability to handle a large number of concurrent services [18]. But how to realize reasonable allocation of the web server to multiple web servers becomes a problem, therefore load balancing mechanism came into picture. To handle the increasing load of a popular web site, we should deploy it in multiple web servers, to ease the burden of a single web server. This paper is organized as follows: Section 1 discusses the overview of load balancing. Section 2 discusses distributed architecture for Web servers. Sections 3 cover the mechanisms that can be used to route requests in distributed Web systems. Section 4 explains how to evaluate fault tolerant distributed web system. Section 5 describes the proposed model of fault-tolerant and efficient load balancing model in distributed web server environment. Section 6 concludes the paper and addresses some pressing issues for future research.

II. ARCHITECTURE OF DISTRIBUTED WEB SERVERS

In distributed web systems, Web browser is a client running on client machine, that is responsible for implementing all the interactions with the Web server, including generating the requests, transmitting them to the server, receiving the results from the server, and presenting them to the client machine. A distributed web-server system needs to appear as a single web server to the outside world, so that users need not to learn about the names or locations of the distributed web servers. The client application may be aware about some mechanism used to dispatch requests among the multiple servers. In this survey, we assume that the clients do not need any modification to interact with the scalable Web system. From these premises, the basic Web system architecture considered in this survey consists of multiple server nodes, grouped in a local area with one or more mechanisms to spread client requests among the nodes and, if necessary, one or more internal routing devices. Each Web server can access all site information, independently of the degree of content replication. The Web system requires also one authoritative Domain Name System (DNS) server for translating the Web-site name into one or more IP addresses [21]. A high-level view of the basic architecture is shown in Figure 1. A router and other network components belonging to the Web system could exist in the way between the system and the Internet. However, modern distributed web system consists also of back-end nodes that typically act as data servers for dynamically generated information. User makes a request to a Web server, by either typing a URL or clicking on a link, for example, http://www.abc.com/ttl/index.html. First, the client (browser) extracts the symbolic site name (www.abc.com) from the requested URL and asks, its local domain name server to find out the IP address corresponding to that name. The local name server obtains the IP address by querying the site's authoritative name server. The client establishes a TCP connection with the server or device corresponding to the IP address, returned by the local domain name server. After the completion of the TCP connection, the client sends the HTTP request (using the GET method) for /ttl/index.html to the Web server that sends the requested object back. Depending on the HTTP protocol used for the client/server interactions, the subsequent requests can use the same TCP connection (HTTP/1.1) or different TCP connections (HTTP/1.0) to the same server [21].

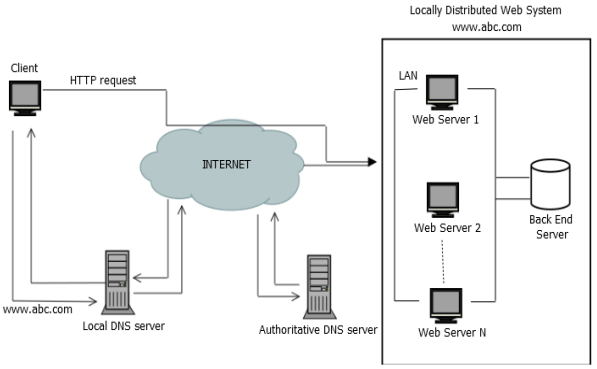

Fig. 1 Distributed Web Server architecture [20]

III. LOAD BALANCING MECHANISMS

In this section we have discussed some load balancing techniques for distributed web systems. We have gone through several research papers on load balancing, but we selected some of them to discuss in this survey.

Web Server Routing Mechanisms

A client request can be redirected to another node by implementing some routing mechanisms by the Web servers. Specifically, we consider the triangulation mechanism implemented at the TCP/IP layer, and HTTP redirection and URL rewriting mechanisms that work at the application layer. Some previous routing mechanisms that typically have a centralized activation, but Web server mechanisms are distributed. Indeed, when a new request arrives from the Web switch, each Web server can decide to serve it locally or to redirect it, although some centralized coordination can be used [21].

A. Triangulation

The triangulation mechanism is based on packet tunnelling [22], when this routing is activated; the client continues to send packets to the first contacted server even if the request is actually serviced by a different node. The first contacted node is responsible to route client packets to the different node at the TCP/IP layer, by encapsulating the original datagram into a new datagram. The second node recognizes that the data-gram has been rerouted and responds directly to the client. Subsequent packets from the same client pertaining to the same TCP connection continue to reach the first contacted node, which reroutes them to the second server until the connection is closed [21].

B. HTTP Redirection

The HTTP protocol standard, starting from version 1.0, allows a Web server to respond to a client request with a 301 or 302 status code in the response header that informs the client to resubmit its request to another node [23]. The built-in HTTP redirection technique supports only per URI based redirection. The status code 301 is returned only when the requested

resource has been assigned a new permanent URI and any future reference to this resource will use the returned URI. The status code 302 is returned only when the requested resource resides temporarily under a different URI [21]. Figure 2 shows the flow of requests and responses when HTTP redirection is activated. HTTP redirection allows content-aware routing, because the first server receiving the HTTP request can take into account the content of the request when it selects another appropriate node. The main drawback is that this mechanism consumes resources of the first contacted server and adds an extra round-trip time to the request processing, as every HTTP redirection requires the client to initiate a new TCP connection with the destination node [21].

C. URL Rewriting

URL rewriting is a more recent technique to implement Web server rerouting. When redirection is activated, the first contacted node changes dynamically the links for the embedded objects within the requested Web page so that they point to another node [24].

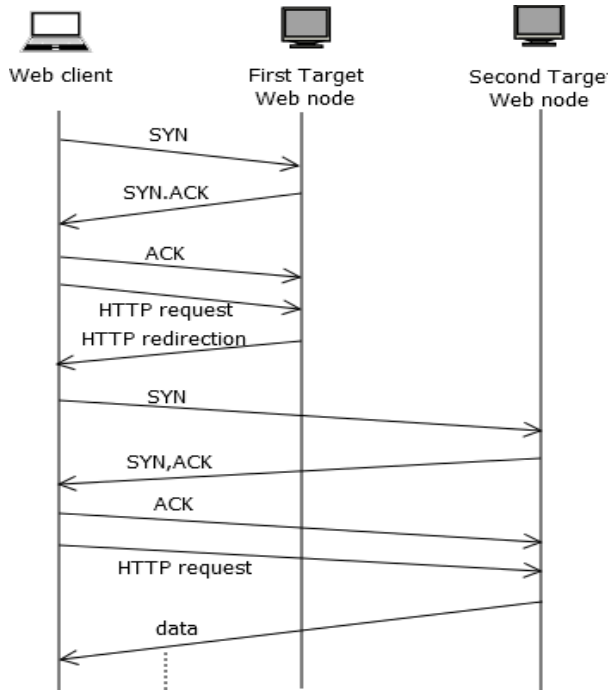

Fig. 2 HTTP Redirection

The drawback of URL rewriting is that it introduces additional load on the redirecting Web node because each Web page has to be dynamically generated in order to contain the modified object references. Furthermore, it may cause a considerable DNS overhead, as an additional address resolution is necessary to map the new URL into the IP address of the second target node. It has been demonstrated that address lookup might take substantially longer than network round-trip time [25]. The URL rewriting has some advantages over the other solutions. For example, this approach can be handled completely at the user level, and the redirection mechanism can redirect further communications of a client to a specific Web server with one request redirection only [21].

Comparison of load balancing Mechanisms

DNS-based routing determines the server destination of client requests during the address resolution phase that is typically activated at most once for each Web session. The request distribution among the Web nodes is very coarse because all client requests in a session will reach the same server. Moreover, address caching at intermediate name servers and client browsers further limits the necessity of contacting the A-DNS. Setting TTL to low values allows for a more fine-grained request distribution, but it could make the A-DNS a potential system bottleneck and increase the latency time perceived by users. Because of the presence of non cooperative name servers that use their own lower bounds for the TTL, the range of the applicability of this solution is limited or not well predictable. Although initially conceived for locally distributed architectures, DNS-based routing can scale well geographically. The popularity of this approach for wide-area Web systems and for Content Delivery Networks is increasing due to the seamless integration with standard DNS and the generality of the name resolution process, which works across any IP-based application. A solution to DNS problems is to add a second-level routing carried out by the Web servers through a rerouting mechanism operating at the TCP or HTTP layer. The main disadvantage of triangulation is the overhead imposed on the first contacted server, as it must continue to forward client packets to the destination node. That is to say, the triangulation mechanism does not allow the first server to completely get rid of the redirected requests. Moreover, as triangulation is a content-blind routing mechanism, it requires full content replication, and does not allow fine-grain dispatching when the Web transaction is carried out through an HTTP/1.1 persistent connection. Unlike triangulation-based solutions, application-layer mechanisms, such as HTTP redirection and URL rewriting, do not require the modification of packets reaching or leaving the Web-server system. This allows the server to take into account the requested content in the dispatching decision, thus providing also fine-grain rerouting.

Table I : A Summary of Load Balancing Mechanisms In Distributed Web Environment [20]

	DNS	Triangulation	HTTP redirection	URL rewriting
Dispatching	Content-blind	Content-blind	Content-aware	Content-aware
Dispatching Granularity	Session	TCP connection	Page/object	Page/object

Client/server data flow	Direct	Triangular	Redirection	Redirection
Overhead	None	Operations at the first contacted server	Round-trip times	Server operations address lookups
Supported net applications	HTTP,FTP	HTTP,FTP	HTTP	HTTP

The HTTP redirection is fully compatible to any client software; however, its use limits the Web based service to HTTP requests only and increases network traffic, because any redirected request needs two TCP connections prior to be serviced. There is another trade-off when comparing URL rewriting and HTTP redirection for multiple requests to the same Web site. The former mechanism requires additional DNS look-ups but no request redirection after the first one, while the latter solution might save address lookup overheads at the price of multiple request redirections. Application-layer mechanisms can avoid ping-pong effects that occur when an already rerouted request is further selected for reassignment. A cookie can be set when the request is redirected for the first time, so prior to decide about reassignment the server inspects if a cookie is present or not. Besides the use of cookies, other means for session identification can be used (e.g., session encoding in URLs), either to identify browsers that do not support cookies or to avoid the misuse of long-living cookies. The triangulation mechanism does not suffer from these side effects, because the destination node can deduce if the request has already been rerouted by simply inspecting the source packet address. Table 1 outlines and compares the main characteristics of the routing mechanisms for distributed web systems [21].

D. DNS Based Load Balancing Mechanism [1]

A DNS based adaptive algorithm is being discussed here. In this technique distributed web servers are arranged in multiple logical ring connections. One virtual URL name is given to the cluster web server that is one of possible approach to handle ever increasing requests to loaded web sites.

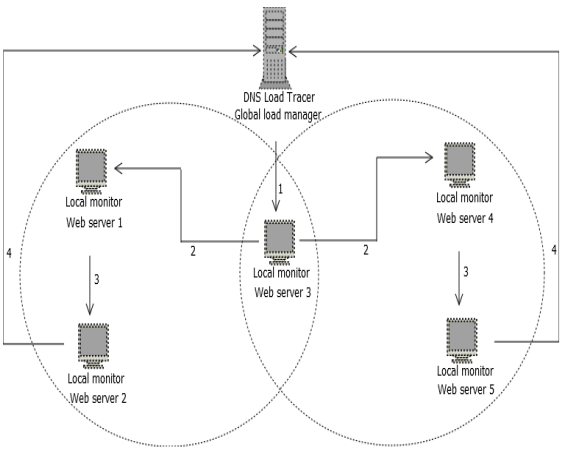

Fig. 3 DNS based load balancing architecture

This scenario consists of three entities: the client, the domain name server (DNS) and the Web- server. DNS resolves all initial address resolution requests from local gateways. Each client session can be characterized by one address resolution and several Web page requests. At first, the client receives the address of one Web-server of the cluster through the DNS address resolution. The client submits several HTTP requests to the Web server on the same connection. In order to select the least loaded Web-server, the DNS could use some scheduling policy to balance the load among several Web-servers to avoid becoming overloaded. The DNS systems assign the client requests in a round robin manner among the Web-servers. The limited control and non-uniformity of client requests from the clients require more sophisticated DNS scheduling algorithm..The DNS can collect various kinds of data from the Web servers such as history of server state, the number of active server connections, detailed processor loads, resource utilization and server cache hits. In this paper, they focus on proposing a simple scheme for collecting the load information from the Web servers that are arranged in the multiple logical ring connections. The server load information is the CPU utilization over a short interval. In this paper, we consider the maximum and minimum of the CPU utilization over a short interval. In figure 3, they show the main software components needed to implement the proposed distributed Web-server cluster system. The software components include the server load monitor, the load collector and the redirection module.

Figure 3 shows the architecture of the DNS load balancing mechanism and flow of the request redirection in distributed web environment. The server load collector begins load collection process by placing initial load values and IP-addresses in a load collecting message and sending it to its neighbours in the logical rings. When a Web-server receives a load collecting message from its neighbour, it compares the values of the current maximum load and the minimum load with its own load value. If the arrived current maximum value is less than its own, then it replaces the current maximum load with its own. If the arrived current minimum value is greater than its own, then it replaces the current minimum with its own and sets the corresponding IP-address in the message. If the load collector receives the

message from the neighbour server, then it broadcasts this message to all
the Web-servers to announce the most heavily loaded server and the server with the least load. The load collector sends a load collecting message to its neighbours periodically. In this paper, they use HTTP redirection for reassigning the client request on the Web server with the least CPU load. Moreover, we used the request generator, Load Cube in order to generate client requests. The experimental results show that the proposed scheme achieves better performance than the default load balancing scheme based on the Round-Robin policy.

E. RTSLB algorithm [14]

In this paper, they propose the Real Time Server statistics based Load Balancing (RTSLB) algorithm that uses four factors to redirect the request for an object to a Web server. The four factors are:
1) The weighted metric of cache hits on different Web servers,
2) CPU load of the Web server,
3) The server response rate (SRR) and
4) The number of client requests being handled by the server.
Their experiments show that the RTSLB algorithm outperformed three algorithms, namely:
1) a random load distribution algorithm,
2) A round robin load distribution algorithm, and
3) A competitive learning algorithm
In the RTSLB algorithm, the decision for redirection is made as follows. Initially, the presence of the requested objects is checked in the cache of each Web server. If the object is not present in any of the caches, a Web server is randomly chosen for redirection. If the object is present in just one of the server's caches, the redirection is made based on the load on the server i.e. the number of connections that are being currently served on a given server. If the load on the selected Web server is above a preset threshold then the request is redirected randomly to one of the remaining servers. In the event of a cache hit on more than one server, the redirection is made based on the CPU loads and the available SRR of each of the Web servers. Figure 4 shows flow of the RTSLB algorithm and steps of a request redirection.

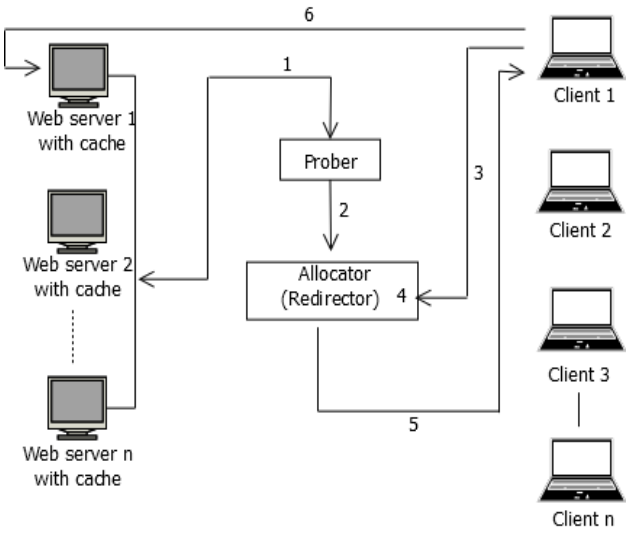

Fig. 4 Flow of RTSLB algorithm

A 60 percent weight is assigned to the CPU load and a 40 percent weight is assigned to the SRR of the servers under consideration. The RTSLB algorithm scores over the other three algorithms because it varies its distribution policy not only on the basis of the cache contents but also by considering the load experienced by the individual servers. The RTSLB algorithm outperforms the other three algorithms or matches them in all simulations.

F. WLC algorithm [17]

WLC stands for Weighted Least Connections algorithm; it is based on two more parameters than RTSLB algorithm. In this algorithm author considered following parameters to balance the load among web servers in distributed environment.
Active number of connections 2) Utilization rate of CPU 3) Disk and memory 4) the numbers of current processes and the response time of the server.

IV. Fault Tolerant Mechanism in Distributed Web Environment

Several approaches have been proposed to implement fault-tolerant Web services using multiple replicated servers. In such an approach, there are one primary node and one or more backup nodes. They deployed multiple dispatchers because a dispatcher may become a single point of failure. The authoritative name server distributes client requests to a dispatcher and then passed to one of WASs connected to the corresponding dispatcher node [13]. In passive replication, as shown in Figure 5-(a), the primary dispatcher redirects all client requests and sends heartbeat timestamp messages to the backup node periodically. If a backup node finds failure in the primary node, it takes up the primary's role through IP aliasing. Figure 5-(b) describes a simple diagram for active

replication, where all dispatchers handle client requests assigned to them and exchange heartbeat messages for failure detection. They evaluate the fault-tolerant distributed Web systems with multiple dispatchers, which are registered in the local authoritative name server. Client requests arrived through the public IP address are distributed to a dispatcher using one of load control rules, and then passed to one of WASs connected to the dispatcher. They used active no of connections to balance the load of web servers.

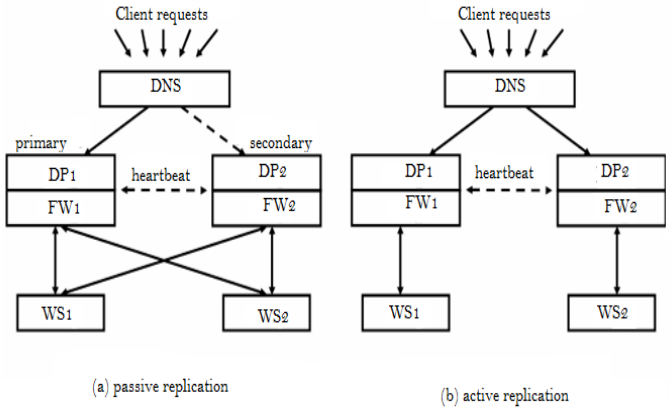

Fig. 5 Replicated Dispatchers [13]

It is also necessary to synchronize clocks for failure detection using timeouts in the synchronous systems. Each heartbeat message contains the dispatcher's timestamp value and a backup dispatcher synchronizes its local clock with the received time-stamp value using our clock synchronization algorithm. In this system, a packet filtering firewall, the most basic type of firewall, is deployed for the purpose of reducing the impact of packet filtering. It inspects each incoming packet and passes the packet into the internal network or drops it according to the list of access control rules. The experimental results show that active replication scheme achieves better performance than passive replication scheme in the case for dispatcher failures.

V. Model Architecture of Fault Tolerant Based Load Balancing Mechanism

We studied different-2 techniques in various research papers. After understanding all these techniques, we are proposing a model of fault tolerant and efficient load balancing mechanism. In this proposal we are discussing model architecture and those parameters on which load has to be balanced. In figure 6 there are more than one load balancers connected to DNS and Web servers also. DNS is responsible to parse the client requests to any one of the load balancer. Then load balancer with the help of prober and redirector will dispatch client requests to the least loaded server. In this model load balancer is combination of prober and redirector. Prober searches the least loaded server and Redirector dispatches the client request to the selected Web server.

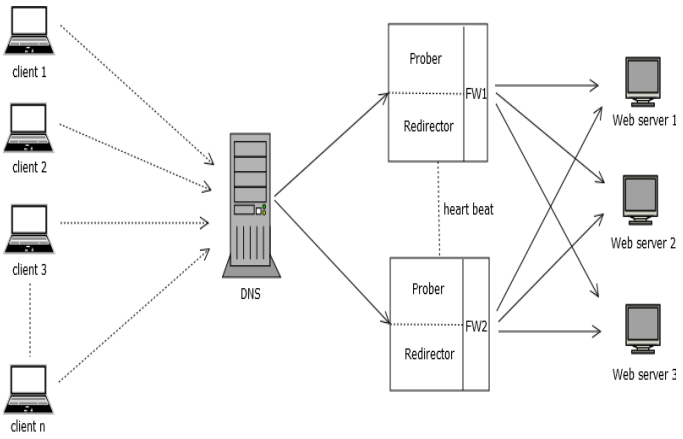

Fig. 6 Proposed model of fault-tolerant based load balancing mechanism

Because a load balancer may be the single point of failure, so we took two load balancers, these load balancer works on active replication scheme and both are working at the same time. Prober selects the least loaded Web server based on six parameters.

1. Web server cache hit
2. Active no of connections
3. CPU load
4. Server Response Rate
5. Resource utilization
6. Failure rate
7. No. of processes

First four parameters are discussed in RTSLB algorithm [15]. If we design and implement RTSLB algorithm based on seven parameters, so the performance may increase. In our system, a packet filtering firewall, the most basic type of firewall, is deployed for the purpose of reducing the impact of packet filtering. It inspects each incoming packet and passes the packet into the internal network or drops it according to the list of access control rules.

VI. Conlusion

Much effort has been devoted in recent years to improve the scalability of systems supporting Web sites. In this paper, we have analyzed dispatching algorithms that are suitable for locally distributed Web systems. At present the Web is playing the most important interface for accessing web services and applications, and Web cluster architecture will be in demand for Web-based information systems, Web hosting centres, and Application Service Providers. One research path is in the direction of combining performance with security and fault-tolerance, and

accessibility from different client devices, all topics that are still seen as separate issues in Web clusters. Finally, we observe that most of the topics and algorithms analyzed in this paper change completely if we assume that the multiple servers of the content provider are distributed over the world rather than grouped in a local area. Our proposed model is based on seven parameters and we deployed more than one load balancers that make our model fault-tolerant based mechanism. In future we shall come up with its implementation and efficient results.

Acknowledgme
nt

The authors thank the management of Bapuji Institute of Engineering and Technology, Davangere for providing the necessary facilities to undertake the above work.

REFER
ENCES

i. *Y. S. Hong, J. H. No and S.Y. Kim, DNS-Based Load Balancing in Distributed Web-server Systems, Proceedings of the Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems and Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA'06).*

ii. *Toshiya Fujii, Tadashi Dohi, Statistical Failure Analysis of a Web Server System, International Conference on Availability, Reliability and Security, 2009.*

iii. *LI Li, TIAN RuiXiong, YANG Bo, GAO ZhiGuo, A Model of Web Server's Performance-Power Relationship, International Conference on Communication Software and Networks, 2009.*

iv. *Michael Grottke, Lei Li, Kalyanaraman Vaidyanathan, and Kishor S. Trivedi, Analysis of Software Aging in a Web Server, IEEE TRANSACTIONS ON RELIABILITY, VOL. 55, NO. 3, SEPTEMBER 2006.*

v. *Mikael Andersson, Introduction to Web Server Modeling and Control Research, TECHNICAL REPORT Publication code: CODEN: LUTEDX(TETS-7211)/1-27/(2005)&local 26 October 28, 2005.*

vi. *Michele Colajanni,Philip S. Yu,, and Daniel M. Dias, Analysis of Task Assignment Policies in Scalable Distributed Web-Server Systems, ieee transactions on parallel and distributed systems, vol. 9, no. 6, june 1998.*

vii. *Maurice Castro, Michael Dwyer and Michael Rumsewicz," Load balancing and control for distributed World Wide Web servers", Proceedings of the IEEE International Conference on Control Applications Kohala Coast-Island of awai'i, Hawai'i. USA - August 22-21, 1999.*

viii. *Ben Chung-Pun Ng and Cho-Li Wang, Document Distribution Algorithm for Load Balancing on an Extensible Web Server Architecture.*

ix. *Paul Ezhilchelvan, Mohammad-Reza R.Khayyambashi, Graham Morgan and Doug Palmer, Measuring the Cost of Scalability and Reliability for Internet-Based, Server-Centered Applications.*

xxv. *c. Press, Los Alamitos, CA, 85–94, 2001*

x. *Valeria Cardellini,Michele Colajanni, Member, and Philip S. Yu, Request Redirection Algorithms for Distributed Web Systems, ieee transactions on parallel and distributed systems, vol. 14, no. 4, april 2003 355.*

xi. *Hiroshi Yokota, Shigetomo Kimura, Yoshihiko Ebihara, A Proposal of DNS-Based Adaptive Load Balancing Method for Mirror Server Systems and Its Implementation, Proceedings of the 18th International Conference on Advanced Information Networking and Application (AINA'04).*

xii. *Yibei Ling, Shigang Chen, Xiaola L i n, Towards Better Performance Measurement of Web Servers, ICICS-PCM 2003 15-18 DsembaZ003SinBapon.*

xiii. *Y. S. Hong and J. H. No, In Han, Evaluation of Fault-tolerant Distributed Web Systems, Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'05).*

xiv. *Deepak C. Shadrach, Kiran S. Balagani, Vir V. Phoha, A Weighted Metric Based Adaptive Algorithm for Web Server Load Balancing, Third International Symposium on Intelligent Information Technology Application, 2009*

xv. *Yan Yang, Fei Yang, Ailixier Aikebaier, Performance Evaluation of an Optimized Load Allocation Approach for Web Server Clusters, IEEE 24th International Conference on Advanced Information Networking and Applications Workshops, 2010*

xvi. *Navid Aghdaie and Yuval Tamir, Performance Optimizations for Transparent Fault-Tolerant Web Service*

xvii. *Yang Jiao and Wei Wang, Design and Implementation of Load Balancing of Distributed-system-based Web Server, Third International Symposium on Electronic Commerce and Security, 2010*

xviii. *Jiani Guo and Laxmi Narayan, Load Balancing in a Cluster-Based Web Server for Multimedia Applications, ieee transactions on parallel and distributed systems, vol. 17, no. 11, november 2006*

xix. *Savio S.H. Tse, Approximate Algorithms for Document Placement in Distributed Web Servers, ieee transactions on parallel and distributed systems, vol. 16, no. 6, june 2005*

xx. *V. Cardellini, E. Casalicchio, M. Colajanni, and P.S. Yu, The State of the Art in Locally Distributed Web-Server Systems, ACM Computing Surveys, vol 34, no. 2, pp. 263-311, June 2002.*

xxi. *Aversa, L. And Bestavros, A. Load balancing a cluster of Web servers using Distributed Packet Rewriting, In Proceedings of the 19th IEEE International Performance, Computing, and Communication Conference (Phoenix, AZ, Feb.). IEEE Computer Soc. Press, Los Alamitos, CA, 24–29., 2000*

xxii. *Berners-Lee, T., Fielding, R., And Frystyk, H. 1996. Hypertext Transfer Protocol—HTTP/1.0. RFC 1945.*

xxiii. *Li, Q. And Moon, B. Distributed Cooperative Apache Web server. In Proceedings of the 10th International World Wide Web Conference (Hong Kong, May). ACM Press, New York, 555–564, 2001*

xxiv. *Cohen, E. AND Kaplan, H. Proactive caching of DNS records: Addressing a performance bottleneck, In Proceedings of the Symposium on Applications and the Internet (San Diego, CA, Jan.). IEEE Computer So*