"A LOAD BALANCING MODEL BASED ON REPLICAS OF A SERVER TO ENHANCE PERFORMANCE AND RELIABILITY"

[1]SHWETA K. TARAR
P.G Department of Computer Science & Technology, D.C.P.E, H.V.P.M, Amravati, India
shweta_tarar@rediffmail.com
[2]NISHA S. PUJARI
P.G Department of Computer Science & Technology, D.C.P.E, H.V.P.M, Amravati, India
nishapujari@outlook.com
[3]DIVYANI D. LANDE
P.G Department of Computer Science & Technology, D.C.P.E, H.V.P.M, Amravati, India
divyanilande1993@gmail.com
[4]PROF. N. J. PADOLE
P.G Department of Computer Science & Technology, D.C.P.E, H.V.P.M, Amravati, India
njpadole@gmail.com

ABSTRACT: *Nowadays uncountable web applications exist on World Wide Web containing large information and resources. Any company or organization which hosts their services on web obviously has service users expecting server performance and server reliability. The question of issues related to these parameters arises when network traffic within company's website increases. Sometimes, this increase in network traffic continues until, ultimately, web server crashes as server is over processing capacity. Server load balancing needs to be performed to handle high network traffic which decrease web service load time and increases performance. The work presented here is server load balancing method using replicas of server, server side small software/program and DNS redirection centralize server, enhancing the server performance, availability and reliability.*

Keywords: Network Traffic, Server Load Balancing, Replicas of Server, Redirection Centralize Server, DNS *etc.*

. .

1. INTRODUCTION

With the increase in the web application on World Wide Web the number of users using the services of also increases on each web applications. In spite of large number of users of any particular web site, service user expects quick services, good performance and availability. Quick services means a user must get a response within a time for dynamic responses from server side such as from server administrator. Good performance can be think off as a web page loading time and resource accessibility time, lesser the time better the performance. Availability in the sense of whatever the situation is there on server side the server must be able to deliver the available services demanded by its users. All these issues go well and fulfilled until there are limited number of user/request/network traffic processed by server. As soon as the network traffic goes on increasing the burden on server grows. Each request adds on the overall processing capacity of server that it is capable of. Sometimes, this increase in network traffic continues until, ultimately, web server crashes as server is over processing capacity. This crash of server or over burden of number of requests a server can handle results in unavailability of server resources and services, impacting bad performance for users experiencing slow information access, network downtime or failed connections [3]. One of the solutions for such problems is server load balancing. [5] Load Balancing is distributing incoming service request across available multiple resources. The complications and performance issue comes when multiple service requests are for same resource or resources.

Here we present a real time example of network downtime and slow information access because of high network traffic straining a server. Sant Gadge Baba Amravati University, Amravati geographically covers the western Vidarbha belt (i.e., five districts – Amravati, Akola, Yavatmal, Buldhana and Washim) of Maharashtra State, India. The University has ten faculties which includes Arts, Commerce, Sciences, Medicine, Ayurved, Education, Social Science, Law, Home Science, and Engineering & Technology. The semester examination of every faculty comes almost in same period. The result of every faculty is displayed on university website. As the result of all faculties comes in same period probably lakhs of university students hits the website, resulting in high network traffic towards university server. The high network traffic makes the server resources to load very slowly and unavailable sometimes. The mark sheet takes 10-20 minutes to load or even loads only half of the mark sheet sometimes or doesn't load at all sometimes. The resultant of this is problems faced by students of university. The following figure depicts the above problem.

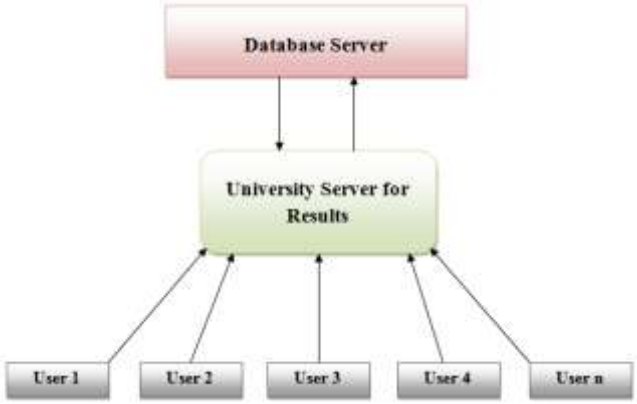

Figure 1: Increase in Network Traffic Load of Server

In the above figure all the student users of university hit university website probably at same time making server overloaded with its processing capacity than it is capable of, resulting server underperform, slow loading of resources and probably making server services unavailable sometimes. Taking the above stated issue as a motivation a server load balancing mechanism needed to improve the overloaded server performance one of the aspect to be worked. We need such server load balancing mechanism which must be cost effective in terms of hardware and software. Without using very costly hardware such as routers maintaining alternate routes to pages, costly distributed servers or multi-layer switch we need a mechanism server load balancing. Software needed is such that there should be less processing with each incoming request at low cost. With this requirement an efficient server load balancing mechanism can be develop and implemented to achieve a good server performance. Another needful aspect is availability, which cannot be achieving without the additional hardware, but is achievable with low cost hardware may include some additional server machines. To keep hardware cost we can replace server machines with personal computers acting as servers.

Based on the above discussion our idea is to implement generalize server load balancing mechanism for any web application using replicas of server to achieve availability and a redirection centralize server for load balancing to enhance performance of the overall web application when multiple service requests are for same resource or resources.

2. LITERATURE REVIEW

As we need to balance the load of web application, the structure of web application plays an important role in categorizing the server load balancing mechanism. As we know web applications are based on client server architecture, two important terms are involved are client and server. Focusing on these two entities we categorized load balancing mechanism in two categories.

- Load Balancing from Client Side
- Load Balancing from Server Side

2.1 Load Balancing from Client Side

This is the approach of forwarding the client request to any replicated servers which are transparent to client. The server selection process can be done by client itself, provided on browser or client side proxy machines. This approach violates one of client server based web application design issue transparency at URL level. So to use us approach is not a good idea.

2.2 Load Balancing from Server Side

This is approach does the load balancing at server side, which can be further categorized based on how replicated server is selected. These categories are Any of Replica Approach and DNS based Request Dispatcher Approach [2].

- **Any of Replica Approach**

Here client request is initially given to any of the replicas of server at randomly. Each replicated server here must have some mechanism to check its own status of total load, and has to make decision whether to serve the request or to forward to another available replicated server as it itself is overloaded.

- **DNS based Request Dispatcher Approach**

Here the incoming request is delegated to main DNS server. In turn main DNS server can select any of the replicated servers based on different scheduling polices. This approach requires calculating/to get load on each replicated server, so how to calculate/get load? After getting exact load from all replicated servers, on which replicated server does the main DNS server must forward the incoming request?

2.3 Load Balancing Algorithms

In the DNS based Request Dispatcher Approach there is need to decide on which replicated server the incoming request is to be forwarded. This requires what sort of load balancing algorithm is to be used. Here are some load balancing algorithms.

1. Round Robin: Incoming requests are forwarded across replicated servers sequentially.
2. IP Hash: The clients IP address is use to determine which replicated server receives the incoming request.
3. Least Connections: Incoming request is forwarded to the replicated server having minimum current load.

2.4 Load Status of Replicated Servers

In both the approaches how to calculate/get load of each replicated server is common issue. However, load balancing method Round Robin and IP hash doesn't require to calculate/get load of replicated servers. If it is required the load of replicated server can be calculated/get by three ways.

1. Each replicated server has the policy to give their load status to main DNS server periodically.
2. Main DNS server will ask for load status to replicated servers on each incoming request.

3. Main DNS server may have database table maintaining incoming and outgoing of request.

2.5 Approach Selection

Exactly which approach to use, is probably situation dependent. But one can differentiate based on complex processing burden on both the DNS and Replicated Servers. When it comes to decision making the system is considered as smart and very obviously smart systems need to perform complex task. So the Any of Replica Approach needs to make decision of whether to serve the incoming request or to forward it to another available replicated server. The important point here is to have criteria for overloaded server. And if every replicated server is overloaded then what is the policy to handle the incoming request. So using Any of the Replica Approach might tends in designing of complex software. Now, DNS based Request Dispatcher Approach where main DNS server somehow has to calculate/get load on all replicated server on each incoming request and has to make decision on which server to forward the incoming request. In spite of two complex task in second i.e. DNS based Request Dispatcher Approach, we can use this approach by handling complex task at software level in simple manner.

3.PROPOSED SYSTEM

As discussed in the previous section, as per our needs and to keep the things simple it is decided to implement DNS based Request Dispatcher Approach using Least Connections. So it is need to calculate/get load of all replicated servers, this real time status of replicated servers will be achieved by using third way Main DNS server have database table maintaining incoming and outgoing of request. After getting the load incoming request is forwarded to the replicated server having minimum current load.

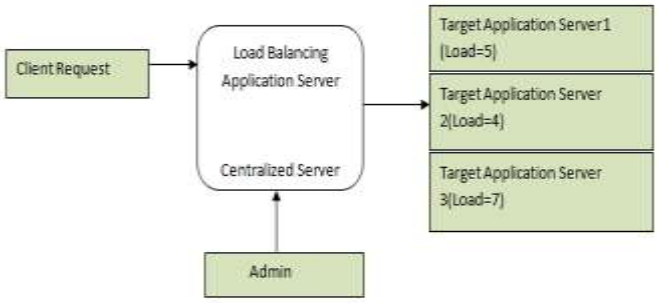

Figure 2: System Architecture

A single server is replaced with a group of servers, from which one will be the centralized server and other will be the replicated processing servers. Here there is one important issue if centralized server goes down whole system goes down. Therefore, special attention is given to centralized server; it is kept very low loaded. Its task is only to accept the request, go on hold, check replicated server status, and forward the request to least loaded server as shown in above figure ensuring the high performance of load balancing system implemented. [4] If one of the replicated server goes down the load balancer/ centralized server forward request to remaining replicated servers, ensuring availability of the resources and services of the servers. In this manner, a load balancing system performs the following functions:

- Distributes client requests or network traffic efficiently across available replicated servers.
- Ensuring high performance, availability and reliability by forwarding incoming requests only to replicated servers that are available.

4. CONCLUSIONS AND FUTURE SCOPE

A load balancing system here implemented acts as the "traffic analyzer" in the way of your main DNS server which routes client requests across all replicated servers capable of serving the incoming requests in a manner that maximizes performance in terms of loading time, processing capacity utilization and ensures that no one server is overloaded or underloaded and ensures the reliability and availability as more than one replicated servers are available to serve the incoming requests. Later improvements are possible in this system such as implementing round robin algorithm for load balancing purpose, implementation of IP Hash algorithm for load balancing for replicated servers in geographical [1] topology or making the system flexible to add or subtract replicated servers on demand.

5. REFERENCES

[1] Gaochao Xu *et. al.*, "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud", IEEE Transaction of Cloud Computing, Volume:18,Issue:1,pp.34-36, Feb, 2013.
[2]Philip S. Yu *et al.*, "Dynamic Load Balancing on Web-server Systems".
[3]http://www.radware.com/Resources/server_load_ balancing . aspx
[4] https://www.nginx.com/resources/glossary/load-balancing/
[5] https://en.wikipedia.org/wiki/Load_balancing_(computing)