Web Scalability: Using Server Virtualization, Caching and Load Balancing

Siddique Asma[1] Patil Amit[2] Mirsinge Ibad[3] Tulve Shabab[4] Prof. P.S Lokhande[5]
[1,2,3,4]Student [5]Assistant Professor
[1,2,3,4,5]Department of Computer Engineering
[1,2,3,4,5]AIKTC, University of Mumbai, Mumbai, India

Abstract— Web applications are the main source of information exchange of businesses over the globe. The main problem over the globe about the web applications is the service of the particular server. As a server has its own capacity to handle some amount of particular clients, there is always a possibility the number of Clients may get increased, at this time most of servers are unable to respond. This paper gives a purposed solution for Servers to handle more clients even when its capacity exceeds without scaling it horizontally that is (without increasing hardware). Management of clients requests on a web Server is managed In order to increase the capacity on a server and hence it gets capable to handle more clients. Architecture for scaling web servers is uses application level scaling which makes it flexible to be applied on any other servers.

Key words: Scaling, Web Server Scaling, Caching, Load Balancer, Application Level Scaling, Scalability

I. INTRODUCTION

Now-a-days E-commerce sites are increasing rapidly. Almost all selling and buying is done on E-commerce sites. Since these type of business is growing fast the system which is responsible for this type of business should also be improved so as to be compatible with increasing number of users. It is important to improve the system so as to maintain its performance. If the performance of this type of system is lowered then it will lead to the losing a customers. Losing customers means loss in business. In E-Commerce one hour of site failure results leads to loss of millions of dollars. Web applications typically undergo maintenance at a faster rate than other systems; this maintenance often consists of small incremental changes [1]

This influenced us to introduce scalability in these systems so as to improve the system and making system relevant to user. Scalability is essential term to achieve the best performance from the system. Scalability can be defined as "The potential or capability of a system to handle more loads without increasing the response time of request". In Web Scalability the capacity of Servers to handle a particular number of users request is increased, through this scalability is achieved. A Web application can be differentiated from a Web site based on the "ability of a user to affect the state of the business logic on the server"[2]

II. LITERATURE REVIEW

For Web Scalability we have observed and studied following.

A. Scaling using Load Balancer:

When multiple web servers are present in a server group, the HTTP traffic needs to be evenly distributed among the servers. In the process these servers must appear as one web server to the web client, for example an internet browser. The load balancing mechanism used for spreading HTTP requests is known as IP spraying. The equipment use for IP spraying is also called the load dispatcher or network dispatcher or simply the load balancer. In this case the sprayer intercepts each HTTP request, and redirects them to a server in the server cluster.[5]

Depending on the type of the sprayer involved, the architecture can provide capability, load balancing and fail over requirements. Load balancing of servers by an IP sprayer can be implemented in different ways. These methods of load balancing can be set up in the load balancer based on available load balancing types. There are various algorithms used to distribute the load among the available servers. Algorithms: Random Allocation, Round Robin, Weighted Round robin.

B. Scaling using Caching

A powerful concept for reducing the delay caused in processing request and saving bandwidth is caching web resources. Here the resources refer to web pages. We find caches at Web browsers, organization proxy server caches, Internet service providers, content delivery networks (CDNs), and Web servers. A cache at the server side is mainly used to reduce the time required for processing a request. Server side caching is the act of caching data on the server. Data can be cached anywhere and at any point on the server that makes sense. It is common to cache commonly used data from the database to prevent hitting the database every time the data is required. We cache the results from competition scores since the operation is expensive in terms of both processor and database usage. It is common to cache pages or page fragments so that they don't need to be generated for every visitor.

1) Cache Hit Ratio:

A cache-hit ratio is the number of times the database found something in cache divided by the number of times it looked for some object in the cache. The higher the ratio, the more effective the cache is at improving performance. Taking into account the cache hit ratios; the frequency of reference f to Web documents is inversely proportional to the rank r, which is measured in terms of the document's popularity. The most popular document has r = 1, the second most popular has r = 2, and so on.

This relationship, called Zipf's law, states that: f = k/r . . . where k is a constant.

Types of Caching Used:

- Object caches: are used to store objects for the application to reuse. These objects are either come from a database directly or are generated through data computation.
- Memcached: It is a high-performance, distributed memory object caching system, generic in nature, but

intended for use in speeding up dynamic web applications by alleviating database load.

– Reverse Proxy Cache: Reverse proxy caches is a strategy for web application caching. While object cache are usually used to cache database objects, reverse proxy cache are used to cache the result of web server e.g. web, DNS and other network lookups. Reverse proxy caches reduce loads on web servers and improve response time to user requests, facilitating scalability.

C. Scaling Database:

The database is the most common bottleneck in web applications, since a lot of reads and writes occur at the database level, and hence the primary focus in this book section is on scaling them. A scalable database is one that performs well under increasing traffic and dataset.

1) Methods for Database Scaling

a) Replication
- Database replication using the master-slave model
- Multi-master Replication Model
- Replication Delay and Consistency

b) Partitioning
- Round Robin Partitioning
- Hash Partitioning
- Range Partitioning
- Vertical Partitioning
- Horizontal Partitioning

2) Short falls in Existing System:

- Multiple Web servers are assigned using horizontal scaling technique and appropriate load balancing technique is applied i.e DNS load balancer.
- Cache operations are costly, hence existing system larger amount of cache is assigned for the repeated request of data.

3) How to Overcome:

- In Proposed method we use server virtualization to create virtual server of single server and the applying load balancing technique in order to scale.
- Using Zipf's Law and accurate calculations of the demanded page, It will be stored in the cache

III. Problem Statement

We are Scaling web servers on application level not by using horizontal or vertical scaling techniques.

IV. Objective and Scope

A. Objective:

The objective of the proposed system is to scale a web server on application level that is to use minimum hardware and scale the requests on server side. The proposed system would have the capability to handle clients request more than its original limit. Since available solutions in market are to increase the Hardware components another solution is to increasing the number of server machines.

B. Project Scope:

- Server will be capable of handling more concurrent user then conventional server.
- Caching gives Boost to handle more concurrent user.
- Web security at client side of cookies.
- Simultaneous updating of cache.
- More space required for cache operation.

V. Proposed System Architecture

A. Basic System Architecture:

A typical Web-based electronic commerce system has a three-tier architecture: the Web server, electronic commerce application server, and database system[3]**.** Basic system follows three tier Architecture. Server is divided according to Web Server, Application Server and Database Server. The Web server is a process that handles requests from users and returns the requested web pages. The application server contains the business logic and accesses the database for information.

Basically the work of the Architecture is, A user asks any request to a server, in Server the Web server Process the request and forwards it to the application server. Application Server has access to database of the Server, through accessing the data it returns the information requested by the client

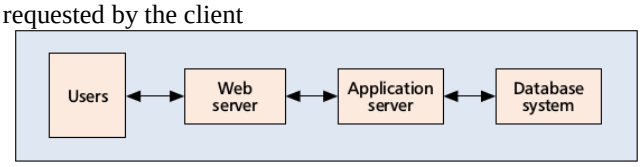

Fig. 1: Basic 3-tier Architecture

B. Proposed Architecture:

The System is divided into multiple components in which uses the three tier architecture along with some modifications.

1) Load Balancer:

On server side this is the first component to receive the users request. At a particular second it receives thousands of clients requests. The main purpose of this load balancer is to divide the requests among different web servers. Load balancer uses the technique such as DNS Load balancing. In this proposed system Architecture the load balancer will use this technique and the algorithm used will be Round Robin or Weighted Round Robin Along with Load balancer there is a counter which counts the number of concurrent users according to their respective ip address. Calculating these algorithms load balancer will forward the request to application server.

2) Web Server:

Main work of web server is process HTTP or any other protocol request, The primary function of a web server is to store, process and deliver web pages to clients. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP)[wiki]. In this Architecture the web server requests for web pages to the application server which are stored in database. If the request is found on the fragment it will retrieve from that fragment.

3) Application Server:

An application server, according to our definition, an application server exposes business logic to client applications through various protocols, possibly including HTTP [javaworld]. Here an application server will receive the request from the web server, It has all access to the

database of the server. Application server will process the db request and will send reply to the web server.

4) Payment Gateway:

This is a third party gate way provided on internet for safer mode of payment.

5) Database Server:

A database server is a computer program that provides database services to other computer programs or computers, as defined by the client–server model[wiki]

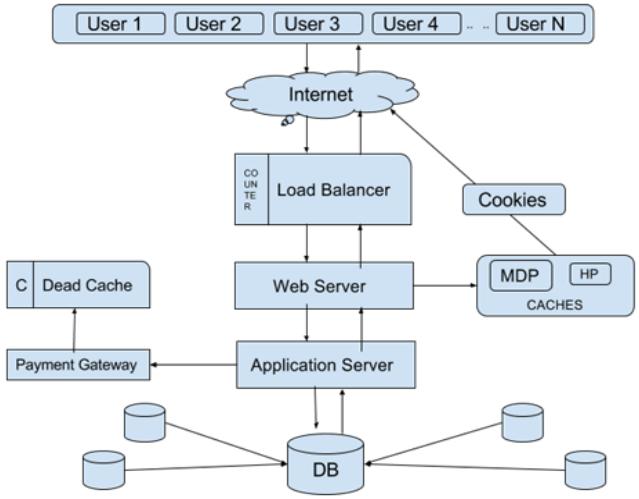

Fig. 2: Proposed System Architecture

6) Caches:

- MDP-Most Demanded Pages: In this cache recent most demanded page will be stored. Calculations will be done before assigning the caches to gain maximum hit ratio.
- HP- Home Page: This Home page cache will be created if the servers capacity and the capacity of MDP cache is exceeded. The user will be redirected to home page in such scenarios.
- Cookies: When the users request is redirected from the MDP cache, some information will be sent to users along with the page in form of cookies. This technique will allow reducing the calculation for next page on the server.
- Dead Cache: When user want to enter into the payment mode to payment gateway user goes into inactive state or dead state. In this protocol the server forwards the user session to the dead cache along with the counter. Advantage of this cache, the counter will count the users gone for payment and the server will take new users requests at mean time.

VI. TECHNICAL DETAILS

A. Methodology:

According to proposed architecture the system is divided into different components.

1) Scaling Method Flow:

- Distribute the server into multiple virtual servers to achieve clustering.
- Uses DNS load balancer to distribute the load on virtual servers.
- Appropriate Caching technique is used to create caches to reduce load.
- Data base is distributed into replica or fragments to avoid bottle neck and to reduce response time.

2) Description of Methodology

a) Load Balancing Algorithms used to handle requests on Servers.

DNS based Load Balancing Technique: DNS-based load balancing represents one of the early server load balancing approaches. The Internet's domain name system (DNS) associates IP addresses with a host name. If you type a host name (as part of the URL) into your browser, the browser requests that the DNS server resolve the host name to an IP address. The DNS-based approach is based on the fact that DNS allows multiple IP addresses (real servers) to be assigned to one host name, as shown in the DNS lookup example in Listing. DNS is an efficient solution for global server load balancing, where load must be distributed between data centers at different locations. Often the DNS-based global server load balancing is combined with other server load balancing solutions to distribute the load within a dedicated data center. Algorithms: Randomized Distribution Round Robin Weighted Round Robin

3) A Local Server will be distributed into Virtual Servers.

a) Technique:

- Set your hostnames or setup OS to recognize your local websites.
- Create a folder for the website.
- Setup Apache to serve multiple sites.

4) Creating Caches of Frequent used Data.

As stated in the book A Fresh Graduate's Guide to Software Development Tools and Technologies caching of web paged is achieved by mathematical calculations by zipf's principle.

a) Caching techniques:

- Object Cache.
- Memcached.
- Reverse Proxy cache
- Content Delivery Network.

5) Data fragmentation method is applied.

When particular piece of data is frequently accessed by the users at a particular site, it is more feasible to fragment that piece of information and store that copy of fragment at that site rather than storing the whole information including which is used very often. Thus the information can be retrieved more easily from that site, also the time required for retrieval decreases as there is less data stored to search for. This decreases the size of data store data each site and also speeds up the accessibility. Also it reduces the bottle neck attack.

a) Techniques:

- Replication
  - Database replication using the master slave method.
  - Multi-master Replication model.
  - Replication delay and consistency.
- Partitioning
  - Vertical
  - Horizontal

B. Project Requirements:

1) Software Requirements

- Technology : PHP
- Web Technologies : Html, JavaScript, CSS
- Database : Mysql
- Web Server : APACHE

2) Hardware Requirements

- Processor : Intel
- RAM : 1GB

VII. MARKET POTENTIAL

A. Market Potential of Project:

There is no proper solution in current market to overcome the scalability issue.

Other solutions are to scale vertical or horizontal which are time consuming, costly, and high maintenance operations

Another solution is cloud computing in which web servers are hired on timely basis.

Solution purposed in this paper is scaling web servers by managing the requests among the server in order to reduce the response time and thereby increasing the capacity of then server

B. Competitive Advantage of Project-

1) Previously web Scaling was to improve the hardware capacity or increasing the number of machines in web server.
2) This type of scaling is often costly and increases the maintenance when it is implemented.
3) This project provides software scaling in order to improve scalability
4) Software web Server scaling will provide reduce maintenance cost, reduce Hardware cost, easy maintainability.

VIII. CONCLUSION AND FUTURE SCOPE

A. Conclusion:

Scalability is an important constraint in this century as several businesses are implemented on web in order to spread it throughout the globe. We have presented a new approach and several techniques for scaling web server. The new technique differs from existing technique in that with the use of minimum caches and improving load balancing technique we can achieve high request handling capacity of a server. Cache operations are costly, so when the load on the server is less than its capacity this operations should be preserved, we divided the throughout architecture into two layers, this provides to reduce costly operations. This solution we purposed in paper will be optimal as it uses a bit of hardware that is cache and software in load balancer will be used in optimal and efficient web server scaling.

IX. FUTURE SCOPE

- There is no proper solution in current market to overcome the scalability issue.
- Other solutions are to scale vertical or horizontal which are time consuming, costly, and high maintenance operations
- Another solution is cloud computing in which web servers are hired on timely basis.
- Solution purposed in this paper is scaling web servers by managing the requests among the server in order to reduce the response time and thereby increasing the capacity of then server

ACKNOWLEDGEMENT

We would like to thanks our guide Prof. P.S. Lokhande for giving his valuable guidance who also provided expertise that greatly assisted the research.

REFERENCES

[1] E. Kirda, M. Jazayeri, C. Kerer, and M. Schranz, "Experiences in Engineering Flexible Web Services," IEEE MultiMedia, vol. 8, no. 1, pp. 58-65, Jan. 2001.
[2] J. Conallen, Building Web Applications with UML. Addison-Wesley, 2000.
[3] Gregor v. Bochmann,"Scalability of Web-Based Electronic Commerce Systems" July 2003.
[4] Daniel A. Menascé, "Scaling website through caching", George Mason University menasce@cs.gmu.ed June 2003
[5] Sameena Naaz "Load Balancing Algorithms for Peer to Peer and Client Server Distributed Environments." June 2012.
[6] [wiki] www.wikipedia.org
[7] [java world]www.javaworld.com

