CLUE Based Load Balancing in Replicated Web Server

Prachi Shukla
Department of Technology
Savatribai Phule Pune University, Pune, India
prachi.shukla2706@gmail.com

Anup Kumar
Department of Computer Science
TIT-Excellence, Bhopal, India
anupkumar0072000@yahoo.co.in

Abstract—It has been seen that the number of users in (web server) are increasing every now and then. This results in bottleneck which may lead to crashing of web servers or reduces the quality of service. So, there is a need of efficient and robust algorithm for load balancing. We have proposed an algorithm that gives better result and is capable of balancing the load on data centre subjected to the key feature. As Load balancing is the critical task in the technical era and the traffic in the web is increasing exponentially, we propose a solution which deals with core load balancing to resolve this trending issue. Here we have designed an intelligent sensing algorithm that will help the server to distribute the load according to the availability of server, location and then redirect the request to comparatively most available server. Distribution of load is done according to the availability of the system instead of round robin fashion. Availability of a server is computed according to current trifle load, current critical load and a deviation value. Depending upon the computed load: the most available, available and least available server are found. Finally, the request is dispatched to the nearest server which is highly available.

Index Terms—Clue, Cluster, Data Centre, Load Balancing, Replicated Server.

I. Introduction

Load balancing refers to a situation when the load on each server is almost equal. To initiate this we first distribute the load on the servers and then check their current load for any kind of further distribution (distribution here means assigning a particular server to incoming request).If loads are equally distributed then it is balanced. To find the equal distribution is really difficult because equal has many meaning in technical interface(equal distribution can be defined as number of requests assigned with respect to load of that server or the type of response a particular data centre is serving or the overall computation it requires to respond a client).So, a number of factors together tell us about the load of server like bandwidth, response time ,size of data transfer, type of content, latency of a server and many more [2].Here we are discussing about an intelligent sensing algorithm that will help the server to distribute the load according to the availability of system, type of requested data and size of element. It also guarantees the quality of service. At very initial stage, assuming all the servers in a data centre are available/all are equally loaded with the consideration that there are n servers for same web site, the incoming request will be assigned to the server in sequential manner i.e. 1 to 1,till nth request is dispatched to nth server. Now when n+1 request comes, the proposed algorithm comes in action, i.e. it starts finding the least loaded nearest server so as to connect it with the next incoming request to give effective response time.

II. PRELIMINARIES

A. Server Load Balancing

Server load balancing distributes service requests across a group of real servers and makes those servers look like a single big server to the clients [1]. Often dozens of real servers are behind a URL that implements a single virtual service. In a widely used server load balancing architecture, the incoming request is directed to a dedicated server load balancer that is transparent to the client. Based on parameters such as availability or current server load, the load balancer decides which server should handle the request and forwards it to the selected server[4]. To provide the load balancing algorithm with the given input data, the load balancer also retrieves information about the server's health and load to verify that they can respond to traffic.

III. Related Work

Few people have explored this area and have done the critical task using various schemes like -

1. Round-Robin Allocation: In which the IP sprayer assigns the requests to a list of the servers on a rotation based phenomena. The first request is allocated to a server picked randomly from the group and IP sprayer follows the circular order to redirect the request. Once a server is assigned a request, the server is moved to the end of the list.

2. Weighted Round-Robin Allocation: Weighted Round-Robin is an advanced version of the round-robin that eliminates the deficiencies of the plain round robin algorithm. In case of a weighted round-robin, one can assign a weight to each server in the group so that if a server is capable of handling twice as much load as the other, the powerful server gets a weight of 2(double). In such cases, the IP sprayer will assign more requests to the powerful server.

978-1-5386-5956-4/18/$31.00 ©2018 IEEE
DOI: 10.1109/CSNT.2018.22

3. Weighted Balanced: It assigns more traffic to a faster link or less traffic to a connection with a bandwidth cap. Set a weight on the scale for each connection and outgoing traffic will be proportionally distributed according to the specified ratio[1][2].

4. Priority: It routes traffic to your preferred link as long as it is available. Arrange the connection priority order, and traffic will be routed through the healthy link that has the highest priority in the list. Lower priority links will only be used if the current connection fails.

5. Least Used: This helps us to choose the better connec-tion with freer bandwidth. Traffic will be directed to the link with the most available bandwidth among the selected connections [1].There are some more techniques for balancing the load but most of them are focused on single issue.

IV. Problem Statement

Although easy to implement, round robin DNS has problematic drawbacks, such as those arising from record caching in the DNS hierarchy itself, as well as client-side address caching and reuse, the combination of which can be difficult to manage [3]. Round robin DNS should not solely be relied upon for service availability. If a service at one of the addresses in the list fails, the DNS will continue to hand out that address and clients will still attempt to reach the inoperable service. Round robin algorithms do not take into account heterogeneity of the system.

Also, it may not be the best choice for load balancing on its own since it merely alternates the order of the address records each time a name server is queried. There is no consideration for matching the user IP address and its geographical location, transaction time, server load, network congestion, etc[4]. Round robin DNS load balancing works best for services with a large number of uniformly distributed connections to servers of equivalent capacity [1]. Otherwise, it just does load distribution [3].

V. Proposed Solution

A. System Design

The basic principle is that network traffic is sent to a shared IP in many cases called a virtual IP (VIP), or listening IP. This VIP is an address that is attached to the load balancer. Once the load balancer receives a request on this VIP, it will make a decision on where to send the request. This decision is normally controlled by a load balancing method, a Server health check or next generation device which is based on a rule set. The request is then sent to the appropriate server and the server will produce a response [4]. The response will be sent back to requesting node from chosen server in lesser response time.

Client request is typed in browser and sent to the server. The http request is having domain name which is ultimately requesting a resource/server.

The content which is needed is available in more than 1 server (node).

Master server computes range of availability on all candidate servers using current trifle load(ctl), current critical load(ccl) and a deviation value along with their load value.

The master server directs the request to appropriate server.

Appropriate server means the server ready to serve the request and is nearest node where the upcoming request can be dispatched.

B. Solution

CLUE gives a clue, at which server the current request should move so that request reaches to data center quickly and response time is less.

C. Functionality

The key functionality of this system is handled by an intelligent algorithm CLUE. CLUE stands for”Cloud Look up Enabling”. The working of CLUE is defined as follows

(A). It checks the availability of the system.

(B). It redirects the request to the appropriate server.

(C).Initiates the load balancing according to the state of server and location.

(D). It handles the single point failure in case, the master server act up then the responsibility is transferred to adjacent peer to carry out the load balancing.

(E).Perform both external and internal load balancing. External is to scale out/ scale up [1] and internal is balancing within cluster or data centre.

Considering that there are few server having identical site within them like server 1, 2, 3, 4, 5 all are the custodian for any one like - google.com/google.co.in/youtube.com etc. Now according to their busyness we are rating them. The numerical rating is load of each server, as the request is passed from DHT after finding that which data centre /cluster is going to reply then this CLUE will run to find the highly available one and to record the busiest one.

Here we are assuming the value of load = 10 * x such that x is the actual value of load that ranges from 0.1 to 1.0 in general and can exceed 1.0 in extreme situation.

So, according to the above convention we have load value within a range of 1-10.Load can be defined in terms of:

1. Current load is denoted by load on node(Ln) that is for node 1,2,3,4,5 in the form of L1, L2 etc.

2. Minimum load (other than 0) that any node possess in working condition, is named as current trifle load(ctl) for ex - ctl = 3. This is going to be a clue for the least busy node or most available server.

3. Maximum load (other than 10) that any node possess in working condition, is named as current critical load(ccl) e.g ccl = 9. This is going to be a clue for the most busy node or unavailable server.

4. To scale the algorithm we have a variance, to adjust the rigidness of above values; which is denoted by theta(th) = 1.By adding or subtracting theta we can vary min/max values.

D. Algorithm

```
N = No. of server in cluster.
Efficient outcome can be seen when the no. of requests >N
Clue ( )
clue;
int N = 5;int ctl = 2;int ccl = 10;int theta = 1;
 //Above values are assumed, it can differ from server to server
//N = Total no.of node/server till which the loop will run
//ctl = min value of load
//ccl = max value of load
//theta = variance
//Ri is current request
//Ln is load of n th server
//r = Total no. of request to be dispatched.

for (i =0 ; i<r ; i++)
{
  for(n =0;n<N;n++)
    {
      if(flag of Ri !=0)
        {
          if((Ln =< ctl)) OR (Ln = ctl+theta))
              {
                clue       =       00;
                request dispatched to nth server; set flag for Ri
                = 0;
              }
          else if((Ln>ctl) && (Ln<ccl))
              {
                clue = 01;
                request dispatched to nth server; set flag for Ri
                = 0;
              }
          else if((Ln>ccl) && (Ln<=(ccl+theta))
              {
                clue       =11;
                request dispatched to nth server; set flag for Ri
                = 0;
              }
          else
              {
                set flag for Ri = 1;
                external load balancing required
              }
        }
  else
      {
        Display assigned;
      }
    }
}
```

Further,clue holds the value of each node for every request and then uses it for internal load balancing after periodic time interval.

E. Implementation

The figure depicts logical topology of replicated servers arranged in ascending order of their geographical position with their respective loads status associated with them[4][5],here geographical position means distance from client in ascending order eg.P0 is nearest, P1 is second nearest then P2,P3,P4....as per the diagram. We are interested in determining the avail-ability of server which is closest to client with least delay. While implementing, the geo position of server is maintained by adding a delay factor to every server and initial loads were assumed. The code is implemented in java for both CLUE and ROUND ROBIN then a comparison is done using response time.

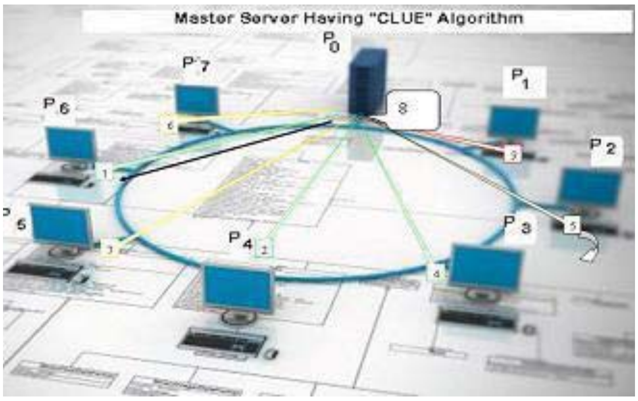

Fig. 1. Clue based Load Balancing

F. Analysis

CLUE fills the loop hole of round robin, weighted round robin, priority assignment, by sensing and then redirecting. It handles scalability very well as load increases. It is also scal-able in a way, that if the kind of service provided by the node is changed or if we want to run the same algorithm in another cluster providing different services/different capacity then just change the value of ctl and ccl like for a cluster providing videos, then just by increasing the two values (ctl and ccl) the same can be implemented.

It is flexible enough as the variance slides the constrain of trifle point and also by changing the value of variance one can get different range of busyness and availability of system. By comparing the proposed solution with already implemented technique we found that there is significant difference in response time. In graph; No. of request increases with time and shows that the time taken by clue is far compressed and stable although at some point it may same as round robin but average performance of algorithm is quiet better, as in proposed method the incoming request is assigned by sensing the load, location and variance factor thus resulting in intelligent assignment.

Comparative graph is drawn by testing N no of request and their respective response time for eg. request on x axis at 20th position has a response time around 100 ms with clue 800ms with round robin. Noticeably for 1st, 4th,and 18th response time is same irrespective to the algorithm used but when we summarize the pattern of response time is significantly less than other. Moreover there is a state when the algorithm jumps out of loop which means the given resources are unable to handle the traffic, thus it clues for external load balancing which means either replace existing server with another more powerful one or add more server.

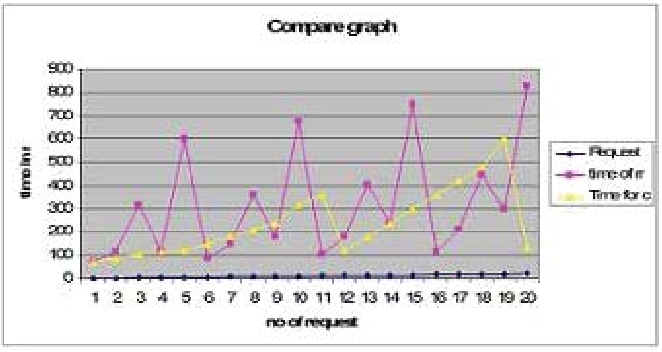

Fig. 2. Comparative Graph

VI. CONCLUSION

This Literature proposes CLUE based Load balancing algo-rithm which first looks into the cloud and then rationally enables load balancing. The attributes of server like geograph-ical position and availability are consider ,to ensure high throughput and low delay for any request coming to a cluster. It also consists of variance that helps to slide the boundary value of load which makes this algorithm flexible. We have designed and implemented the proposed solution in a system and evaluated the efficiency of this load scheduling scheme based on execution time. According to the analysis, the result ensures that this is a dynamic algorithm that predicts the most available, available and least available server which justifies the reason to connect the incoming request to the specific (appropriate) server. To the best of our knowledge, this solution provides a new load sharing policy to carry load balancing in replicated web server. Further handling external load balancing could be one of the future work

REFERENCES

[1] Misikir Eyob Gebrehiwot,Samuli Aalto, and Pasi Lassila, “ Energy efficient load balancing in web server clusters,” 29th International Teletraffic Congress, 2017.

[2] XU Zongyu1,WANG Xingxuan1, “A Modified Round-robin Load-balancing Algorithm for Cluster-based Web Servers,”,July 2014.

[3] Zongming Fei,Samrat Bhattacharjee,Ellen W. Zegura,Mostafa H. Am-mar, “A Novel Server Selection Technique for Improving the Response Time of a Replicated Service,”INFOCOM ’98. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. II, , 1998.

[4] Agung B. Prasetijo,Eko D. Widianto ,Ersya T. Hidayatullah, “Performance Comparisons of Web Server Load Balancing Algorithms on HAProxy and Heartbeat,” 3rd Int. Conf. on Information Tech., Computer, and Electrical Engineering, october 2016.

[5] J.Amudhavel, U.Prabu, P. Dhavachelvan, N. Moganarangan, V.Ravishankar, R.Baskaran,“Non-Homogeneous Hidden Markov Model approach for Load Balancing in Web Server Farms ,” Proceedings of 2015 Global Conference on Communication Technologies,2015.