An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems

Gurasis Singh[1], Kamalpreet Kaur[2]

[1]Assistant professor,Department of Computer Science, Guru Nanak Dev University College, Jalandhar ,India
[2] Lecturer, Department of Computer Science and Engineering, Govt. Polytechnic College For Girls, Jalandhar, India

---***---

Abstract - *Web servers are the basis of almost every activity that uses the internet. There must be a web server in the path from client to service. With load balancing of servers, users no longer experience network downtime, slow information recovery, or aborted connections. In this paper, we propose an improved WLC algorithm that maintains the load among various servers by preventing requests being transmitted to the new real server. When web requests are continuously assigned to only a new real server more than the maximum continuous allocation number(C), the proposed algorithm excludes the new real server from activated real server scheduling list and deactivates it. . Finally after C-1 allocation round times, the new real server is activated and included into the server scheduling list. The proposed algorithm balances the load among real servers by avoiding overloads of the new real server.*

Key Words: Web Servers, load balancing, web cluster system.

1. Introduction

Modern high-traffic websites must handle thousands of parallel requests from clients and return the factual text, images, video, or application data, all in a fast and definite manner. To cost effectively scale to appropriate these high volumes, modern computing practice generally desires adding more servers. A load balancer is a hardware or software tool designed to transmit traffic across different servers [1]. It gives much more technical control over where traffic goes and how it is managed. Load balancers can also be set up in a highly available manner so that no single point of failure occur. It is defined as efficiently distributing incoming traffic across a bunch of backend servers, reffered as server pool.

A load balancer acts as a "traffic cop" present in front of real servers and transmitting user requests among all the real servers that can attain those requests in a manner that increases speed and capacity utilization and assures that no server is overworked, which could weaken the performance [1,2]. If one server goes down, the load balancer transmits the traffic to the remaining real servers. When a new server is added to the cluster of servers, the load balancer automatically start sending requests to the new server.

A Load balancer allows users to intelligently share traffic to a single Internet Protocol (IP) across multiple servers using a number of variant protocols [3]. This means that the load can be distributed among various nodes, instead of being assigned to a single server in order to increase the performance during times of high load. It increases the reliability of web application and allows clients to produce the application with high performance. If any server node fails, the requests are programmatically assigned to other nodes without any interference of service.

2.Classification of Load Balancing Algorithms

In various business Information Technology (IT) infrastructures, many network paths exist to direct user access to internal and external networks. With load balancing of servers, users no longer experience network downtime, slow information recovery, or aborted connections[5]. By providing various different routes to the requested pages through distributed requests of server, the mechanism called as load balancing of servers provides clients with definite access to the web application. If one server loses functionality, this fail over system provides a backup path. By providing the web application availability among business networks, organizations would achieve advanced infrastructure support to enhance the high level performance. There are various mechanisms that can be used to effectively balance the clients requests among a bunch of servers. Various load balancing mechanisms have different benefits. The choice of load balancing algorithm depends on the type of application or service being served and the servers and network status, at the time of the client request. To service new requests, the algorithms can be used in combination to examine the best server. When the load is low, the simple load balancing method is used. But more complex methods are used in case of high load to evenly distribute the client requests under network and service stress[6].

Load balancing Algorithms are classified into the following subtypes:

A. Round Robin

In the round robin algorithm, the processes are divided mainly between all the processors. Every new process is assigned to new processor in the order of round robin. The round robin algorithm is expected to give high performance, with more reliability. However when the user requests required unequal processing time round robin algorithm will be less efficient as some nodes can become overloaded while some remains idle.

B. Weighted Round Robin

It is similar to the round robin algorithm in such a way that the distributing manner remained cyclical by which client requests are provided to the nodes. In the configuration of the load balancer, user provides "weights" parameter to each and every node. The node with the higher weight have the ability to handle more no of client requests. For example, If server 1 is more capable and is having more specifications than server 2.Server 1's ability is 4x more than that of server 2's capacity, then user can assign a weight of 4 to server 1 and weight of 1 to the server 2. So when users requests come, the first 4 requests will be assigned to server 1 and the 5th request to server 2. If more requests comes, the similar procedure will be followed in such a way that, the 6th, 7th, 8th, 9th request will go to server1, and the 10th request to Server 2, and so on. Fig.2.2 shows the working of weighted round robin.

C. Least Connection Algorithm

The least-connection scheduling algorithm directs network connections to the server with the least number of established connections. It is one of the dynamic load balancing scheduling algorithms; as it needs to count the no of connections for every server to calculate its load. The load balancer maintains the connection number of each and every server, increases the connection number when a new connection is send off to it, and decrease the connection number when the connection finishes. Least connection algorithm transmit a web request to the server that has least web connection number.

D. Weighted Least Connection

The weighted least-connection scheduling is a superset of the least-connection scheduling, in which you can assign a performance weight to each real server. The servers with a higher weight value will receive a larger percentage of active connections at any one time [12]. The default server weight is one, and the IPVS Administrator or monitoring program can assign any weight to real server. In the weighted least-connections scheduling algorithm, a new network connection is given to a server which has the minimum ratio of the number of current active connections to its weight.

In Weighted least connection scheduling algorithm (WLC), various different performance weight no's can be given to each and every server. The Weighted least connection scheduling algorithm does to least connection algorithm what weighted round robin algorithm does to round robin algorithm. That is, it introduces a "weight" that is based on the specifications of every server.

3. Web servers as load balancer

A. Nginx web server

Nginx is a web server software, that can be configured to be a very powerful and simple load balancer to enhance the availability of servers and its performance. The Nginx web server is a very capable software as a load balancer, as Ngnix uses an event-driven and non-threaded architecture, it can surpass web servers like Apache web servers. In configuration Nginx acts as a single entrance point to a distributed web application that is working on different and real servers. Nginx load balancing is one of the most relavant mechanism that can ensure redundancy, and it is comparatively easy and quick in setup and configuration.

Using aproxy when the demand of serving a single website outgrow the capabilities of a single machine is very helpful. Also, there are web frameworks like Seaside and Ruby On Rails Mongel server, that employ applications on framework determined web servers while these single purpose servers contributes to the powerful application services, there are not appropriate for hosting entire application. In these cases, Nginx load balancer is used to pass only the necessary requests to the web application server and providing a secure environment.

The following load balancing mechanisms are supported in Nginx load balancer:

1) Round robin: In this technique the requests are assigned to the application servers in a round-robin fashion.

2) Least connected: In Least connected technique the next request is assigned to the server having the least number of active connections.

3) Ip-hash: A hash function is used to determine which server should be selected for the next request, based on the IP address of client.

B. HAProxy Web Server

HAProxy, stands for High Availability Proxy. It is a common open source software load balancer and that can be run implemented on solaris, linux, and FreeBSD. It is mainly used to enhance the performance and to increase the reliability of a server environment by distributing the workload among various servers. HAProxy Web Server can be used in many high-profile environments, including imgur, instagram, GitHub, and twitter. There are many concepts and terms that are important when discussing load balancing and proxying.

1) Access Control List (ACL)

In case of load balancing, ACLs are needed to test some condition and perform an action like selecting a server, or blocking a request, based on the test result. The use of ACLs allows flexible traffic forwarding on the bases of various factors such that the number of connections to a backend and the matching of pattern.

2) Backend

A backend is a group of servers that are responsible for the receiving of the forwarded requests. The Backends are mainly defined in the HAProxy configuration's backend part.

Most commonly, a backend can be defined by the load balance algorithm that is used and the combined list of ports and servers.

A backend can have many servers present in it, adding more servers to the backend increases the potential load ability by sharing the load among different servers

4. IMPROVED WEIGHTED LEAST CONNECTION SCHEDULING ALGORITHM

Load balancing is one of the most essential area pertaining to research in the domain of Manet's. In this work, Load balancing protocols are surveyed and classified in three categories which are single path based protocols, multi path based protocols and clustering based load balancing routing protocols. We have grouped and summarized these protocols and then compared them with respect to our described comparison framework.

A. Server Selection Algorithm

Server selection is one of basic approaches to improve the quality of distributed network services over the Internet. Server selection algorithms fall into four categories namely DNS, Router, Server-side and Client-side algorithms. Client-side selection is necessary when the group of servers is heterogeneous in nature or broadly dispersed across the network. The server-side selection methods rely upon the group of servers. Group of servers contains the members with same resources and a local network which is shared. In clusters, the primary concern is balancing request load across servers: when resources are equal and the load is balanced, a client receives similar response times from all servers.

One popular approach for load distribution uses DNS aliasing. A site is assigned multiple IP records in the local DNS table. Upon receiving a translation request, the DNS Bind program selects from among these records in a round robin fashion (DNSRR). While simple, DNS-RR offers only crude load balancing and is often combined with a server-side mechanism. In server side mechanisms, clients send requests to a dispatching module that tracks current load conditions and assigns servers accordingly. The dispatcher may direct a client to the chosen server via an HTTP Redirect. Alternatively, the dispatcher may change IP addresses on all incoming packets to route them to the appropriate servers. This algorithm requires the dispatcher keep track of all the site's TCP streams, earning it the name TCP router.

B. Overload Algorithm

Overload algorithm uses maximum series assignment number (C) in order to avoid concentrated web request assign to new added real server. C is a maximum web request assignment number of times within that web requests can be continuously assigned to a real server. If web requests are assigned to a real server more than C, the real server's state is set to 'deactivated state' and excepted from activated real server list and after C-1 times web requests, the real server is set to 'activated state' and is included in activated real server list. Each real server is assigned a performance weight. Performance weight is defined as a constant number which means capacity of each real server, as different servers are having different specifications, like CPU, RAM etc .

Overload algorithm receives a selected real server and the number of activated real servers as parameters. If there is only one real server in activated real server list, real server selection algorithm resumes since a web request cannot be continuously assigned to same real server. Overload reduce algorithm is executed in normal operation mode or overload mode. Normal operation mode is the state in which load distribution scheduling is executed normally and periodically checks web requests concentration on one real server. Overload mode is the state in which web requests are assigned to maximum web request assignment number overload state allocated as maximum series assignment number to real server that web request is equal, reduce overload.

```
IWLC_Schedule (A[], n)
Input  :  A[0:n-1] : Arrangement of real server
 n  :  Numbe of arrangement element
output  :  least  :  Selected real server
{
  var  least  :  Save of selected real server
  var  active  :  Number of server to be vitality state
   for ( i = 0; i < n; i++ )
   {
        if ( A[i].flag == DISABLE )
       {
          A[i];  Real server in scheduling exception
        }
    if ( least.conns / least.weight > A[i].conns / A[i].weight
)
      {
          least = A[i];
      }
    active++;                /* New add part */ }
    Overload (least, active);      /* New add part */
    return least;
 }
```

Fig. Real server selection algorithm

```
Overload (s, n)
 s : Selected real server
n : Number of activated real servers
 {
 var old_s : Previously Selected Real Server
 var count : Series assignment number
var flag : Overload flag
var L : Maximum series assignment number
if ( flag == 1 )      /* Overload mode */
{
  if ( --count < 1 )
   {
      old_s.flag = ENABLE;
      flag = 0;
   }
   return;
}

if ( n < 2 )         /* Normal mode */
{
  return;
}
 if ( old_s != s )
{
 old_s = s;
count = 1;
return;
}
if ( ++count >= C )
{
 count = C - 1;
 old_s.flag = DISABLE;
 flag = 1; }
}
```

FIG. OVERLOAD ALGORITHM

5. Simulation And Results

Docker container is used to create virtual servers. Docker is an open source platform for developing, shipping, and running web applications. Docker provides users the ability to package and run the web application in a loosely isolated environment referred as container. Containers are do not need the extra load of a hypervisor, but run directly within the kernel of host machine. This means users can run more containers on a given hardware combination than if they are using virtual machines. We can even run Docker containers within host machines that are actually virtual machines. A Dockerfile is a type of script that has collections of commands and instructions that will be always automatically executed in the particular sequence in the docker environment for the building of new docker images.

We analyze the performance of configured docker container in an aspect of number of user requests and web tasks. To evaluate the effectiveness of the existing algorithms in terms of average execution time of each server and number of requests assigned to different servers, simultaneous requests are sent to the web application, based on php technology. The simultaneous requests are sent by "node js" scripting language, in order to get huge traffic on the web server. Nginx then transmits the requests on three virtual servers created by Docker Container.

A. Performance Evaluation of Existing Algorithms

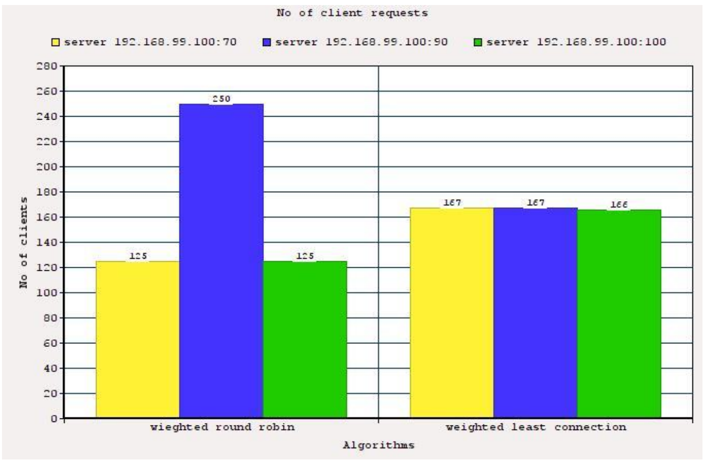

Fig. Number of client requests

The above figure shows the comparison of weighted round robin and weighted least connection on the basis of number of client requests. The server with port number 90 has more weight so more requests are assigned to it as compared to other servers, in weighted round robin, whereas in weighted least connection the client requests are assigned to the server having least connection per weight of that particular server.

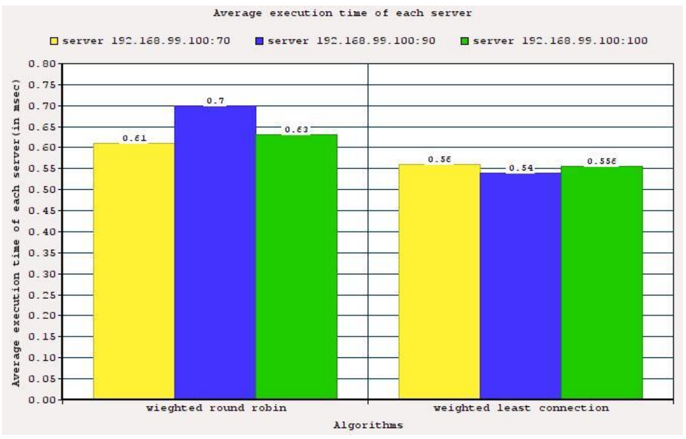

Fig. Average execution time of each server(in msec)

The above figure shows the average execution time of each particular server created by docker container. Nginx is given 500 simultaneous requests one at a time and is configured to weighted round robin and weighted least connection and the average execution time is observed.

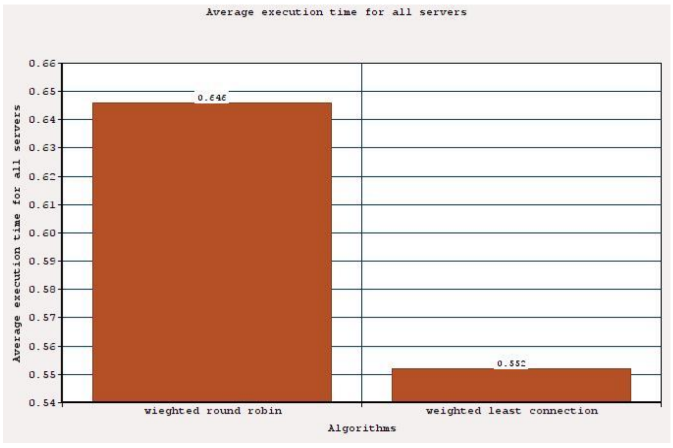

Fig. Average execution time of all the three servers (in msec)

The Average execution time of all the three servers in weighted round robin observed to be more than the average execution time for all servers in weighted least connection.

B. Performance Evaluation of Proposed Algorithm

The proposed IWLC algorithm is compared to the previous WLC algorithm. When a new real server is added to a web cluster system with many simultaneous users, previous WLC scheduling algorithm assigns web requests to only the new real server, and makes load imbalance among real servers. IWLC scheduling algorithm is proposed that maintains load balance among real servers by avoiding web requests being assigned to only a new real server. When web requests are continuously assigned to only a new real server more than the maximum continuous allocation number(C), the proposed algorithm excepts the new real server from activated real server scheduling list and deactivates the new real server.

Initially, three virtual servers on port 70, 90 and 100 are given a total of 500 requests and is subdivided on the basis of weighted least connection. Whenever the fourth server will come, with simultaneous requests, weighted least connection algorithm will assign maximum requests to the newly added server. Applying IWLC algorithm by taking C= 100 means that any server can get only 100 continuous requests after then that server is excluded from the activation server list and other servers are assigned the next requests.

The new server added is having the IP address 192.168.99.100.110. Fig.4.8 shows the comparison of WLC and IWLC algorithm, by taking fourth virtual server in the actual activated server list, with 500 more requests. Initially 500 requests are assigned to all the three servers after then another 500 requests are assigned with a new server added into the list. As shown in the figure more requests are assigned to fourth server added in WLC algorithm, as compared to the IWLC and the average execution time for all the servers is also reduced in case of IWLC as shown in Figures below.

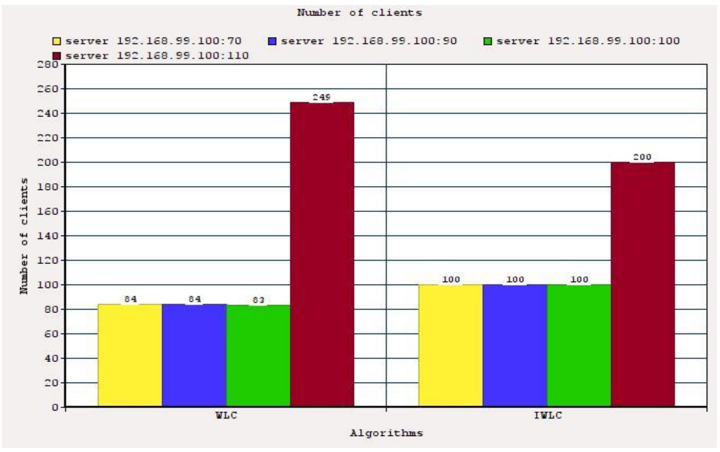

Fig. Number of client requests assigned

The Average execution time of all the servers is shown in the below figure, when a new server on port number 110, is added into the list which is assigned a total of 249 requests. The average execution time for the fourth server is more in WLC as compared to IWLC.

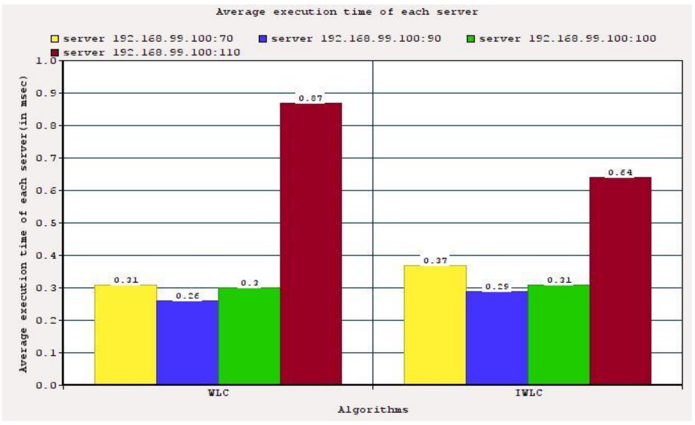

Fig. Average execution time of each server

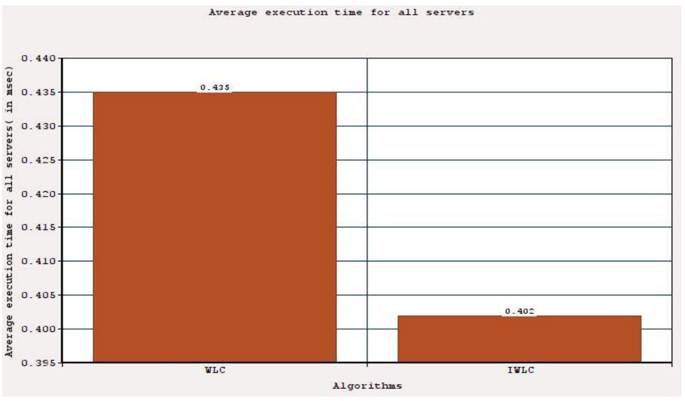

Fig. Average execution time for all servers

Figure above shows that when the fourth server is added into the activation list with simultaneous 500 requests, the average execution time for all servers in IWLC is less than the averge execution time for all the servers in case of WLC scheduling algorithm.

6. Conclusion

We proposed the improved WLC algorithm that maintains load balance among real servers by avoiding web requests being assigned to only a new real server when the real server is newly added to a activated real server list. The WLC load balancing algorithm, in which a throughput weight is assigned to real servers and the least connected server is selected for processing web requests, is mainly used for web cluster systems. When a new real server is added with many simultaneous clients to a web cluster system, the previous WLC scheduling algorithm makes load imbalance among various real servers by assigning web requests to the new real server added. The proposed algorithm consists of two algorithms namely, real server selection algorithm and overload reduce algorithm. In proposed algorithm, a real server is selected by real server selection algorithm and the selected real server is passed to overload reduce algorithm as a parameter. If web requests are assigned to a real server more than C times, the real server is set to be deactivated state and excepted from activated real server list in overload reduce algorithm and after C-1 allocation round times, the new real server is included into real server scheduling list by activating it. As a result, proposed algorithm maintains load balance among real servers by avoiding overloads on the new real server.

REFERENCES

[1] Kim, S.C., Lee, Y.: System Infrastructure of Efficient Web Cluster System to Decrease the Response Time using the Load Distribution Algorithm. Journal of Korea Information Science Society C(10), 507–513 (2012)

[2] Aversa, L., and Bestavros, A. Load balancing a cluster of web servers using distributed packet rewriting. In Proc. of IEEE International Performance, Computing and Communications conference (2010), pp. 24–29.

[3] Balasubramanian, J., Schmidt, D. C., Dowdy, L., and Othman, O. Evaluating the performance of middleware load balancing strategies. In Proc. of IEEE International Conference of Enterprise Distributed Object Computing (2009), pp. 135–146.

[4] Kwon, O.Y.: Cluster system abstract, Technical. Policy same native place, Korea Institute of Science and Technology Information news (2010)

[5]. Yun, C.G.: A Load Distribution Technique of Web Clustering System based on the Real Time Status of Real Server. Journal of Korea Information Science Society A(5), 427–432 (2005)

[6]. Kuzmin, A.: Cluster Approach to High Performance Computing. Computer Modeling & New Technologies 7(2), 7–15 (2011)

[7]. Shao, Z., Jin, H., Chen, B., Xu, J., Yue, J.: HARTs: High Availability Cluster Architecture with Redundant TCP Stacks. In: Proc. of the International Performance Computing and Communication Conference (IPCCC), pp. 255–262 (2012)

[8]. Zhang, W.: Linux Virtual Server for Scalable Network Services. Ottawa Linux Symposium (2008)

[9]. Santana, C., Leite, J.C.B., Mossé, D.: Load forecasting applied to soft real-time web clusters. In: Proceedings of the 2013 ACM Symposium on Applied Computing (2010)

[10]. Zhou, H., Hu, Y., Mu, D., Hu, W.: Proportional Delay Differentiation Service and Load Balancing in Web Cluster Systems. In: INFOCOM IEEE Conference on Computer Communications Workshops (2014)

[11] L.Wenzheng and S.Hongyan, "A novel algorithm for load bal ancing in cluster systems," in 14th International Conference on Computer Supported Cooperative Work in Design, 2010.

[12] S.Sharma, S.Singh, and M.Sharma, "Performance analysis of load balanc ing algorithms," in World Academy of Science, Engineering and Technology, 2008.

[13] Z. Xu and R. Huang, "Performance study of load balancing algorithms in distributed web server systems," in CS213 Parallel and Distributed Processing Project Report.