A State Of Art Survey For Web Server Performance Measurement And Load Balancing Mechanisms

Omid H. Jader, Subhi R. M. Zeebaree, Rizgar R. Zebari

Abstract: Nowadays, the massive load on the internet by the demanders and diversity of web applications, the web servers have become crucial. Therefore, many related companies and web developers try to generate powerful structures and efficient systems for web servers in order to satisfy internet users and the web servers from being overworked. Furthermore, in big web-based companies due to enormous number of clients, one server could not handle all the incoming requests and some of them be rejected. Hence, the idea of cluster server and load balancing methods been provided to tackle the problem. In this paper, twenty up-to-date references have been depended for reviewing different studies who addressed web server performance and load balancing algorithms in the last half-decade, to compare their capabilities and provide an efficient platform to build web-based system structures.

Index Terms: Web Server, Apache, Performance Measurement, Nginx, Load Balancing, HAproxy, and Cluster.

1. INTRODUCTION

Recently, demands on the services and information over the internet become more and more through the continuous expansion of web applications. We can assume that users can obtain most of their daily life information from the internet accurately and quickly as well as all these services are web-based and carried on servers [1]. Therefore, Web Servers receive client's requests, process them and deliver the responses. In addition, the dependency of users upon different electronic fields in both public and private sectors based on the web makes high pressure on the servers. As well as, efficiency and effectivity of web servers determine the attraction ratio by organizations and web developers such as precise and fast response to the clients [2], [3], [4].

On the other hand, the performance of web servers immediately affected by the load directed to them. Therefore, controlling the load in the web servers is crucial. Hence, Designing and proposing an efficient system that could handle a huge load is vital for researchers. Furthermore, the main cause of the overload is the enormous and continuous rise of client's requests to the services provided by the servers. So that, this extra load causes the servers to be failed and languish to provide services exactly. Accordingly, the server damage overload affects the organization's customer attraction, decrease the income and lose reputation [5], [6]. Improving the hardware configuration of the server provide high availability of services and resources [7], [8]. Furthermore, another identical solution is establishing a cluster server, which is widely used in organizations in order to ensure the responsiveness of almost the requests by the cluster server [9], [10].

Alternatively, utilizing a group of the servers to gain resource persistency and availability requires a mechanism to fairly balance and distribute the incoming load among servers. Moreover, this mechanism is helpful to alleviate the traffic toward the server's cluster [11]. Nowadays, balancing the load is an indispensable way in web technology to establish and provide a robust service [12], [13]. Meanwhile, load balancing techniques enhance the performance of cluster servers by improving traffic management, decreasing time to respond requests, raising the throughput and minimize the heavy load on servers in the cluster [14], [15]. The rest of this paper organized as follow; section two deals with the web servers and performance measurements, including a review of last four years references. In the third section, a background about load balancing and various load balancers with the verity of balancing algorithms addressed and the most recent proposed algorithms about load balancing reviewed. While, fourth and fifth sections are the discussion and conclusions for the comparison of addressed reviewed references.

- *Omid H. Jader, is currently pursuing master degree program in Information System Engineering in Erbil Polytechnic University, Kurdistan Region-Iraq, PH-+9647504192232. E-mail: omid.jader@epu.edu.iq*
- *Dr. Subhi R. M. Zeebaree, Duhok Polytechnic University, Kurdistan Region-Iraq, PH-+9647504332160. E-mail: subhi.rafeeq@dpu.edu.krd*
- *Rizgar R. Zebari, , Duhok Polytechnic University, Iraq, PH-+9647504763915. E-mail: rzgarz11@gmail.com*

2 WEB SERVERS

In the last few years, web server plays a vital activity in information transaction over the internet in the globe as result of huge number of websites in plenty fields of the life in both private and public sectors. Web server refers to serving customer's requests to varied types of information in the form of documents and applications that held on the server side and accessed by the demanders using web browsers [16]. In addition, HTTP (hypertext markup language) is a protocol that transfers request and responses between clients and web servers. When a user asks for a file an HTTP request will send toward the server for that file and the server explores inside its database and answer the request either if exist or not [17].

Generally, web servers are divided into two major processing structure either process-driven or event-driven. In the first case (for example apache and IIS web servers) the incoming request might be served by multiple thread or process during the request completion. Meanwhile, that process cannot be assigned to the other requests at the same time. Reversely, in event-driven architecture, the incoming requests are partitioned into several stages called events. Each stage might be calculated by many processes in which when a stage is completed the processes can assign to other events of another request. So, in this approach, several requests are processed concurrently. The examples of this case are Nginx, Node.JS, and Lighttpd web servers [18], [19], [20], [21]. As well as, for bulk web-based companies that have thousands or millions of

users it is crucial to have a robust and more efficient server architecture. Mostly, though the high rate of concurrent connections to the server, some of the requests are rejected and failed to respond, this cause customer dissatisfaction [22]. On the other hand, it's incredible that how much the web-based users become more, attend to the server performance get obligatory. Obviously, with a good configuration web server, more load can handle compared to a powerless server setting [7]. Furthermore, observing the server performance is a cardinal topic and cover a wide area. High computation and effective web servers raise the user's satisfaction and hardware revenue due to unnecessary to extra servers [2]. Alternatively, to alleviate the congestion and perform better performance under massive user's requests, a group of servers is provisioned to get out of failure. In cluster server, each server is a node and the entire nodes are performed and represented as a single server which participates with the same resources like storage, IP and applications. Whenever a server in the cluster become down, the remained servers perform it's a job with the same previous service automatically [14]. As well as, data between the main and secondary servers are consistent in a synchronization and asynchronization manner [5]. In addition, there are various web servers with different architecture in the market used by the web site's owners. The web server performance, scalability, and flexibility determine the market participating ratio. According to the latest survey of the Netcraft (an internet service company) performed in 24th December 2018, the Apache web server still is the leader of the market share in most metrics due it's historic attribute. But, IIS has the highest market share ratio in terms of the total number of sites, Nginx is in second and the third is Apache. Nginx web server got a rapid growth in domain numbers, while Apache and IIS gradually decreased. As well as, it is considered that Nginx progressed by 39.9% during the last 12 months in computer facing metric. While Apache and Microsoft IIS increased slightly with 4.98% and 1.93 respectively [23].

2.1 Performance Measurement Parameters

It is a fact that measurement parameters indicate the efficiency and availability ratio of a system or a process [24]. Meanwhile, with utilizing measurements observation and assumption operations can be done effectively to ensure the web server health. The hardware characteristics of web servers combined and controlled continuously (such as reliability, availability, scalability resource usage etc.). In this paper, we describe the most popular metrics which are used in the literature.

Throughput

It is a vital key in the assessment of process performance. It refers to the served or processed requests at a given time interval. Usually, it is used to evaluate the load amount of a server. In another word, it shows the transmission speed of data by testing the adaption of bandwidth in the server [11], [20].

Response Time

Refers to identify the spent time of processing data from the server to the client [11]. Or, how much a server takes to find the requested file and send back to the destination such as data querying from a server's database [20]. Latency is another word for response time and it is the summation of waiting and replying time [9].

CPU Utilization

Is the percentage of CPU consuming when dealing with a specific load. When the number of user's requests increases the CPU usage is increase similarly.

Memory Utilization

It is the amount of used RAM during processing the incoming load by the server machine.

Request per second

Refers to the request numbers that are processed by the server in a second. The higher the performance the better the number of requests responded in a second [9].

Failed requests

In some time, due to heavy load over the web server a part of incoming requests cannot be served and then canceled. This is determined by the failed request measurement.

Concurrency

Refers to analyze the web server performance by measuring the number of accepted requests simultaneously. How the machine has much more concurrency feature the request failure ratio will fall down.

2.2 Web Server Literature

This paper, reviewed several studies of the last few years that worked on the web server performance. In [16], G. Liu et al designed a 10/40GB network to evaluate the performance of different popular web servers namely Apache, Nginx and Lighttpd in big data analyze environment. They depended on throughput as a performance metric. In addition, the network consisted of a high featured server with installing the aforementioned web servers and a 40GB Network Interface Controller (NIC) adapter. Then 5 files of different sizes created inside the server (file sizes: 1K, 4K, 16K, 64K, and 256K). The server machine connected to a 10/40GB switch with fiber optic cable, besides 5 clients that support 10GB network connected to the switch with 10GB links. The clients sent requests to the saved files simultaneously. Its conducted that all the web servers support 40GB network and Nginx and Lighttpd outperformed than apache to small files (less than 64KB) while increasing the number of concurrent requests. Whereas, the performance of apache got better results when requested to large files (ex: 64 KB). In [1], Rizgar et al proposed an analysis study to evaluate the performance and availability of IIS 10.0 and Apache 2 web servers under various attacks like SYN and HTTP attacks. Linux Ubuntu 16.04 and windows server 2016 operating systems were used for Apache and IIS respectively. The evaluation process performed in three stages without attacks, with SYN and HTTP attacks. IIS in CPU utilization outperformed apache in the first situation, but the second web server is a little bit faster than the first one in terms of response time. The average CPU usage of IIS was upper than apache as well as the average response time not influenced in apache web servers with SYN attacks. Finally, in contrast to the previous state the performance of IIS was better during HTTP attacks. In [7], A. Barzu et al applied several tests to show the effect of hardware scalability on the web server performance. The server's response and the process time is measured with modification in device configuration when handling a large number of requests. The modifications were adding server machine cores and raising the capacity of RAM. They indicated

that adding the CPU cores affects the improvement of response time and processing time of the server significantly. Whereas scalability of RAM had no impact on the response and process time, but it minimized the number of failed requests in such a way that the entire requests are processed without the need to resend. In [20], L. P. Chitra and R. Satapathy analyzed a comparison study between Node.JS and the traditional IIS. The network throughput is used to evaluate the performance of both web servers in different scenarios. Firstly, the test is implemented by sending requests to a simple page which was hosted on both Node.JS and IIS. The results indicated that the throughput of Node.JS is greater and also incremented by the increasing number of users. In the other scenario, the Input/Output (I/O) intensive state tested by developing a web page on the tow web servers that returns the requested data from a database. The results shown that the performance of Node.JS is higher than IIS, even it is enhanced by adding a number of users with compare to IIS in terms of I/O operations. Finally, they analyzed both servers in terms of CPU intensive by requesting a computational web page that contains a calculating operation. In this situation, the IIS is outperformed than Node.JS. As result, it is concluded that Node.JS web server is efficient and capable for I/O operations but not for CPU operations. In [25], J. Chen and W. Cheng analyzed the web traffic according to HTTP protocol and surveyed over the most popular web servers to obtain the main reasons that influence the web server performance. They listed the name of about 100 web servers based on their highest number of HTTP requests. After evaluating the time of web servers that take to response the requests for the ten of the highest web servers, the results illustrated that Nginx, Apache, and IIS are the best web servers in sequence and they said that it is necessary to enhance the web server requirements with the improvement of internet technology. In [18], Prakash et al proposed an analytical study to indicate the scalability, efficiency, and responsiveness of Apache and Nginx by measuring, RAM usage, error rate, and response time. Response time and memory usage of Nginx web server were lower than the apache in both tests, especially when the number of requests exceeded to 7000 requests per second, in apache web server both measurements rise dramatically. Reversely, the error rate of apache web server was less than Nginx and as the number of requests increased the error rate of apache was better than Nginx. So, it is concluded that the scalability and responsiveness obtained in Nginx are finer, but in terms of efficiency, it degrades in heavy load. In [26], C. Chen et al designed a queuing model to analyze the web server performance based on several web page size and a various number of users. The response time and bandwidth were used as metrics of performance. They increased the number of users from 1-100 and the webpage size from 10-500KB. the average response time for a 10KB webpage for each request per user was 3.77ms, while with the same count of users for 500KB webpage the average time of server response exponentially increased which recorded by about 1442.07ms per each request. Alternatively, the average dedicated bandwidth to each user was about 2765 KB per second when requested to a 10KB webpage, as well as for 500KB webpage, the average bandwidth felt down to 1513 KB per second. It means, from a range of 10-500KB and incrementing a user per time the average bandwidth reduced about 2.37 percent for each user. According to the results, the service rate will be changed by modifying the webpage size and bandwidth rate. Also, the number of requests will increase by raising the webpage size, but the request numbers will be reduced when the amount of data is more than the system capacity for example for 400 to 500 KB of data.

3 LOAD BALANCING TECHNOLOGY

It is the fact that the overload is the most dangerous disease faced by the web servers which cause request failure. Load balancing techniques are the best solution to tackle this problem over the web servers [11]. It works as a transmitter between the client's requests and the web servers by delivering the incoming requests to all the nodes inside the cluster in a way that enhances the speed and resource usage and to guarantee cost effective and scalability. Meanwhile, it is used to provide high performance by preventing each server from being overload inside the cluster. Furthermore, the load balancer is a software or hardware mean built to balance the traffic over different servers and brilliantly shares client load to one real IP among several nodes [19]. Additionally, load balancing is a method to distribute the traffic over several links equally toward a server pool to ensuring optimal traffic direction, decrementing time to respond requests, incrementing throughput and skip overwork [14], [27]. Load balancing avoids the barriers in front of using various applications, homogeneous and heterogeneous cluster web servers [22]. Also, load balancing of web servers is a crucial need in web technology to assures the optimal quality of service by evaluating the outputted performance metrics from the server status and balance the load based on the performance measurement results [12]. Alternatively, over recent years the popularity of distributed systems has led to problems with the need for server hardware and load balancing which makes extra costs for companies and causes the quality of service [28].

3.1 Categorizing Balancing Algorithms

Several mechanisms of load balancing are presented and followed by researchers that each has various advantages and can be used to efficient distribution of the incoming load among the cluster server. According to the provisioned applications, services, server hardware, and network situation, the load scheduling algorithm is selected. The goal is efficiently to maintain and consuming the available system capabilities and to optimize the service performance to satisfy the demanders. Based on the previous studies, the scheduling algorithms are classified into two strategies, dynamic and static. In static load balancing mechanisms, the client request distribution performs depend on the previous configuration without considering the amount of load and current connection numbers of each server inside the server pool. It means the predesigned information of the system is necessary in this strategy for example number of nodes per cluster or the predetermined weight of each node [29]. As well as, this method is more suitable with homogeneous web cluster environment and it is easy to apply and acceptable to use, therefore static algorithms widely implemented in cluster servers for load balancing. Otherwise, the drawbacks of this strategy will appear when realized to bound function web pages for instance login or shopping card web pages because in this case when a new page visits, again the users have to log in as well as this strategy has weakness concurrency and its only capable in limited situations. For example round robin, ratio, weighted round robin, etc. [11], [22], [30]. On the other hand, the dynamic scheduling algorithm works based on the real-time evaluation of the server's performance like accepted connections and a load volume of any server nodes. So,

according to the feedback of the load information, the algorithm tries to distribute the new incoming requests to the most proper server among others. Mostly, under various circumstances, the dynamic load balancing algorithms provide finer results than the static algorithms and this method can tackle the problem of bounded functional web pages (sessions). Whereas, dynamic load balancing algorithms need more CPU and RAM utilization to evaluate and compute the current information of all servers inside the cluster. The algorithms that consider this strategy are least connection, least load, weighted least connection, etc. [11], [22], [29], [31].

3.1 Load Balancing Literature Review

In [27], R. Li et al proposed a system for web application depended on container technology. They designed a dynamic weighted least connection to balance the load among web servers based on Nginx. The performance metrics computed in this study were CPU, memory and network utilization as well as the ratio of memory used by the container software. Two modules are generated, one for achieving parameters from servers and calculating each server weight, the other for Nginx configuration. Then, the load of each node measured by the summation of all the four metrics to determine the weight per node. After that, the Nginx module compared the weight with a predefined threshold, if it was greater the weight is changed based on a specific formula, otherwise, it remained as it was. The experiment performed with one Nginx web server (load balancer) and 5 real servers with different hardware and software specifications. Docker 1.13 installed on each real server which held the web application using Nginx and PHP container. Then, the iperf tool used to calculate the bandwidth between the servers and load balancer as well as the ab command to send 10000 concurrent requests by 100 users. Finally, 5 as a threshold for weight change and 3 as time interval selected as the best. After that, the response time of the developed dynamic weighted least connection algorithm performed to balance the load and compared with round robin and least connection algorithm. The findings showed the more performant of the developed DWLC which was faster than RR and least connection algorithms by 52.6 and 46.4 percent respectively when almost the requests are completed. Z. wen et al in [22], proposed a dynamic observation load balancing approach based on Nginx mechanism. The algorithm relied on a queuing theory by using M/M/1 style. According to this approach, both load balancer server and cluster server used First-In First-Out model in their queuing model to handle the incoming requests. They assumed the amount of load as the main parameter of the dynamic observation algorithm which directly affects the weight modification of the server cluster. As well as, the response time of each server had a direct impact on the queuing length on that server. The suggested system performed with one load balancer and 3 servers with different hardware and software configuration to represent as a heterogeneous cluster server. The users load generated with LoadRunner tool to provide high concurrency of requests. Additionally, the proposed load balancing algorithm tested with several metrics and compared the obtained performance measurements with the IP hash algorithm and weighted round robin algorithm. The results indicated that the suggested dynamic observation load balancing algorithm was more efficient and feasible by achieving the lowest request failure rate and response time with compare to the other two algorithms. In [32], A.D. Usman et al suggested a 2-step load balancing strategy to fairly distribute the load in a heterogeneous wireless network in the base of network bandwidth and load. In the first step, Monte Carlo's method used to list the networks load randomly. Then, the new list passed across the second step and applied Load Leveling method using brute force technique to arrange and divide the load into the upper and lower average. The designed system tested with 10 to 1000 mobiles and the obtained results compared to that of least connection algorithm. It is proven that the 2-step load balancing algorithm is better than LCA with about 12.4%. Also, the resource usage of the network and the service quality for the mobiles were more significant by using the suggested algorithm. In [29], F. Mbarek and V. Mosorow performed a comparison test among several load balancing algorithms in heterogeneous web servers. Ten VMs used for clients and three web servers that installed apache and Nginx. As well as, the load balancer was HAproxy and the used balancing algorithms were RR, static RR, and least connection algorithm. The performance metrics were throughput, resource usage, connection numbers per second and request number per second. As result, the test proved that the RR algorithm has more operable in heterogeneous cluster than static RR and least connection algorithms. In [14], H. handoko et al designed a reliable and high available MarianDB Galera cluster with HAproxy load balancer and virtual redundancy protocol by using Keepalived tool. The designed structure executed virtually in a cloud-based environment. Transaction per second and response time are measured as performance analysis of the system based on two types of queries; simple mode (read-only) and complex mode (delete, update, read and write). They considered two situations; no balancing and load balancing with using round robin algorithm. The findings revealed that utilizing load balancer and VRRP achieved much better results when a simple query mode executed. Whereas, in complex mode, there was not a valuable impact because in this mode any change to data in each server has to be constant in the database as well as need synchronization to the other nodes inside the cluster. In [11], swandika et al argued that selecting the best balancing algorithm is a challenge for web servers that require limited functionalities by different users like login pages (session required). That is why they implemented a load balancing technique on a Software Defined Network (SDN) for determining a suitable balancing algorithm in order to manage and distribute the load over cluster server. Least connection and IP hash algorithms tested in this study because the selected algorithms can deal with sessions for the connected uses. Additionally, three metrics used to test the performance of each algorithm which are memory usage, throughput, and time to respond. Finally, the authors indicated that the IP hash algorithm provides more optimal outcomes than least connection algorithm based on the aforementioned metrics. The findings were the average of 189.2 MB, 1.23 Mbps and 61ms for IP hash while 203.8MB, 1.11Mbps and 73ms for least connection with corresponding to the provided metrics sequentially. In [9], Sahand et al proposed a system to evaluate the load balancer efficiency in a private cloud environment among several web servers. A physical server with the Proxmox platform is used and 6 virtual machines are created with Ubuntu OS, three of them as web servers with the same hardware specifications and the other 3 VMs as the load balancer. Two of the load balancers used HAproxy in order to consider high availability and the remaining one is installed Nginx. They used Apache Benchmark to generate the load and least connection algorithm as load

balancing algorithm. The study tested in several scenarios; without the load balancer, with the load balancer in heavy load and low load as well as check the load balancer failure rate and replace with the alternative one. Generally, it is proven that to prevent overload on the webservers, using load balancer tool are necessary. As well as, the response time of HAproxy was lower than Nginx toward the incoming requests and could accept more requests under heavy load status. Reversely, CPU usage of Nginx was lower than HAproxy. After that, it is considered that when the main server is down the other alternative server could take the main role in around 2 seconds. But during this time some of the requests may be lost. In [19], G. singh and K. kaur proposed a developed weighted least connection algorithm to balance the load over the existing system as well as when a new server added to the cluster server. The proposed method applied based on two main algorithms. First one was the Server Selection Algorithm to determine the state of web servers and select the real server from the list of active servers that could accept the requests. The second was an overload algorithm to avoid sending requests to the new added by using a limited number of requests (X). They consumed Nginx as the load balancer and Docker container platform to evaluate the performance of the used algorithms. Furthermore, the Overload method gets an active server as a parameter from the result of the selection method. Then, the number of current connections of the selected server compared with X, if it exceeded, the server will be deactivated and stop from accepting any other requests. After that, when the load of the stopped server became X-1, it will be converted to the active list of servers and could accept requests. In addition, the improved WLC compared with the existing WLC and round robin algorithm. The results indicated that improved WLC provide better performance than other algorithms in terms of load distribution and average execution time. In [33], A. A. Abdelltif et al presented a load balancing application module based on SDN in a cloud environment to enhance the response time and resource usage. The module consisted of a system observer, a dynamic load balancing algorithm and a service categorization module which applied on an SDN controller and the cluster server. OpenFlow switch used to connect the server pools to the controller. The system observer gathered feedback from the servers and sent to the dynamic load balancer. As well as, the requests accepted by the service categorization module to identify requests services and send them to the balancer module. Furthermore, the load of each server computed according to resource utilization (CPU and Memory) and bandwidth. The proposed system applied to different VMs using OpenStack and HTTPref tool used to create the load toward the servers. Request per Second, response time and reply time are used to measure the performance of the Dynamic load balancing algorithm module. The results of the proposed method showed better performance in all the aforementioned metrics with compare to HAproxy. In [12], Dr. Mustafa Elgili argued the efficiency of three different load balancing algorithm; Round Robin, least connection and least loaded algorithm. The study used OPNET tool to simulate the proposed system structure. A load balancer placed between 8 HTTP servers and 112 clients. It is indicated that the CPU usage of web servers will be higher while using the least loaded algorithm with compare to the other two algorithms that had the same rate of CPU consuming. As well as, based on balancing the load among servers, the least loaded method distributed the requests with high difference among servers. Although, Round robin distributed the load approximately in equal except the first server which accepted the highest number of requests with compare to the other 7 servers. Alternatively, the least connection method divided the load most fairly than the other two methods. In [10], Dr. Joi and Christian proposed a powerful, scalable, and flexible infrastructure for service availability in large companies which is a key feature to make services operable to the customers full hours per day. Although, in this system, a mixed load balancer (from HAproxy and DNS) consumed for distributing the client's requests over the cluster server. Additionally, both load balancers used Round Robin algorithms. Hence, the balancing procedure implemented twice for each request toward the web servers, firstly by the DNS and then balanced by HAproxy. The system was easy to configure and obtained high availability of 99.9 percent. In [34], A. B. Prasetijo et al proposed a system that contains three web servers (installed Ubuntu OS and Apache web server) and two HAproxy servers to distribute the load among servers. As well as, heartbeat technique used to ensure the availability of web servers. Three load balancing algorithms (Round Robin, least connection and source) are tested in the study. Then, the system tested in two states. A certain number of requests are sent to web servers using Httperf tool without existing load balancers. It is noticed that the requests not equally distributed and a large number of requests failed especially by the first server because most of the requests directed to it. In the second scenario, the two HAproxy load balancer allocated and both of them acted as master and slave. When a HAproxy server felt down the second one became master (in only 10 ms) and had continued to accept requests and forward them to web servers. Several performance metrics considered during the use of balancing algorithms. Generally, the least connection algorithm played a better role in terms of throughput, response time, number of connections and the total amount of failed requests. But, Round Robin was the best when the system faced high traffic. In [35], D. sharma and V. B. Aggarwal designed an architectural model for load distribution over several numbers of web servers based on their weights. An Apache HTTP load balancer placed between client’s requests and web servers. The servers had different specifications which all installed Apache Tomcat software and hosted a web application. Moreover, the load balancer had distributed the requests to web servers according to a predefined weight per each web server. A server with high specification had higher weight and could receive more load. Subsequently, the algorithm tested with real-time requests from 100 to 10000 and two different load balancer Apache and Nginx. The results showed that the total processing time of Nginx load balancer is better than the apache load balancer.

4 DISCUSSION

Two main comparison-directions can be presented according to analyzing the previous studies related with the web server performance measurement and load balancing mechanisms illustrated in sections 2 and 3. Table 1 illustrates a comparison of the reviewed papers about web servers. This table produces previous works of seven closest researches with five important compared-features. While, table 2 represents a comparison of reviewed papers about web server Load Balancing techniques. This table produces previous works of thirteen closest researches with seven important compared-features.

TABLE 1: *COMPARISON OF THE REVIEWED PAPERS RELATED TO WEB SERVERS.*

Research	Web servers	Study Goals	Performance Metrics	Measurement tool	Results
[1]	Apache 2 and IIS 10.0	Web server performance analysis during HTTP and SYN flood DDoS attacks	Response time CPU Usage	Jmetter	IIS performance was better during HTTP attacks, while Apache outperformed IIS with SYN attacks
[16]	Apache Nginx Lighttpd	Performance evaluation of web servers in 10/40 Gb Ethernet network	Throughput	Apache-bench (apid)	with small files the performance of Nginx and Lighttpd was better, but Apache achieved better performance with large files
[20]	Node.js And IIS	Performance analysis of web servers for various services	Throughput	Jmetter 2.13	Node.js had more throughput in respect to I/O intensive services, whereas for CPU intensive services, IIS outperformed
[7]	Node.js	The impact of hardware scalability upon server performance	Response time, Processing time, total processing time and failure rate	jHipste r	Number of CPUs had direct impact on the response time, while capacity of RAM reduced the number of failed requests and has less impact on the time
[25]	Survey on top ten web servers	Determine the main reasons of progressing web servers	Response time	_____	The new web server's users like Nginx, Tengine, cafe and so on, are increasing
[18]	Apache Nginx	To analyze the performance of process driven and event driven web servers	Response time Memory usage Error rate	Httperf	The responsiveness and scalability of Nginx was higher, while apache was better in terms of request failure
[26]	Apache	Analyzing the response time of web servers with different web page sizes	Response time Bandwidth	Webserver Stress tool	With the maximum webpage size, the number of requests increased and the average user's bandwidth minimized.

TABLE 2: *COMPARISON OF THE REVIEWED PAPERS ABOUT LOAD BALANCING.*

Research	Study Goals	Web servers	Load balancer	LB Algorithms	Performance Metrics	Measurement tool	Results
[33]	Design a LB model and compare with HAproxy.	Not mentioned.	Proposed model, and HAproxy	Proposed method and HAproxy algorithm	Response time, Request/sec.	HTTPerf	The proposed balancing mechanism outperformed HAproxy.
[22]	Design a dynamic LB algorithm	Jboss	Nginx	IP-hash, WRR, Proposed algorithm.	Response time, Throughput, Request Failed rate.	LoadRunner	Proposed algorithm improved server performance efficiently.
[11]	Evaluate LCA and IP-hash in SDN environment	Apache	POX controller	IP-hash, LCA.	Response time, Throughput, Resource usage.	Iperf, GNU wget, Htop.	IP-hash a superior the LC Algorithm.
[19]	To improve WLC algorithm	Docker container to create web servers	Nginx	WRR, WLC, Improved WLC.	Execution time, Request number.	Node.js script to send concurrent requests.	The improved algorithm outperformed the others

[9]	To evaluate load balancer efficiency in Cloud.	apache	HAproxy, Nginx.	LCA.	Response time, Request/sec, CPU usage, Failed requests.	Jmeter, apache benchmark.	In request serving HAproxy was faster, while in terms of CPU usage Nginx outperformed.
[14]	Analyzing high availability and reliability of MarianDB cluster.	MarianDB server.	HAproxy	RR	Transaction/sec, Response time.	sysbench	MarianDB Galera cluster performed better with the use of VRRP and load balancer.
[27]	Presenting a dynamic WLC balancing algorithm.	Nginx	Nginx	RR, LCA, Proposed DWLC algorithm.	Response time.	Iperf, Apache bench.	The proposed algorithm outperformed than RR and LCA.
[29]	Test LB algorithms in dissimilar web cluster	Apache, Nginx.	HAproxy.	RR, Static RR, LCA.	Request/sec, resource usage, throughput, connection/sec	_____	RR algorithm achieved better performance in heterogeneous web servers.
[32]	To tackle LB problems in heterogeneous Wireless Net.	_____	Designed load balancer.	Proposed algorithm, LCA.	Index	Jaines fairness index.	. The Jaines fairness index of proposed algorithm was more convenient than LCA and better QoS.
[10]	Design a robust and available infrastructure using LB and DNS method.	HAproxy backend.	HAproxy	RR	Throughput	_____	The designed system provided 99.905% of operation availability
[35]	To design and evaluate a new dynamic LB algorithm.	Apache Tomcat	Apache httped, Nginx	Proposed dynamic algorithm	Total processing time	_____	The algorithm obtained better results with Nginx.
[12]	Evaluating the efficiency of LB algorithms.	_____	_____	RR, LCA, and Least loaded	CPU usage, Connected number	OPNET to simulate the design	LCA outperformed the other algorithms.
[34]	To provide a high available LB architecture.	Apache	HAproxy	LCA, RR, and Source.	Connection rate, Throughput, Response time, and Failed connections	HTTPerf	Least connection algorithm had better results than the others.

5 CONCLUSION

Depending on the previous comparison illustrated in section (4), it can be concluded in web server part that [1] accomplished best results with most suitable design related to the web server performance analysis during HTTP and SYN flood DDoS attacks. They used two different web servers with (response time, CPU Usage) for performance measurement and Jmetter measurement tool. They proved that IIS performance was better during HTTP attacks, while Apache outperformed IIS with SYN attacks. Also, on the load balancing part the improved WLC and developed DWLC algorithms attained better performance. Both algorithms depended on the weight of the web server node for distributing the load among them and Nginx mechanism as load balancer. The DWLC algorithm performance has been evaluated under 10000 concurrent requests and the main metric where depended was the response time. The proposed algorithm was faster by 52.6 and 46.4 percent compared to round robin and least connection algorithms respectively. In addition, the improved WLC algorithm achieved better performance than of the WLC and round robin algorithm in term of load distribution and average execution time.

REFERENCES

[1] R. R. Zebari, S. R. M. Zeebaree, and K. Jacksi, "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers," in 2018 International Conference on Advanced Science and Engineering (ICOASE), 2018, pp. 156-161.

[2] D. Kunda, S. Chihana, and S. Muwanei, Web Server Performance of Apache and Nginx: A Systematic Literature Review. 2017, pp. 43-52.

[3] R. K. Ibrahim, S. R. M. Zeebaree, and K. F. S. Jacksi, "Survey on Semantic Similarity Based on Document Clustering," Advances in Science, Technology and Engineering Systems Journal, vol. 4, no. 5, 2019 2019.

[4] K. Jacksi, S. R, and N. Dimililer, "LOD Explorer:

Presenting the Web of Data," (in en), International Journal of Advanced Computer Science and Applications, vol. 9, no. 1, 2018 2018.
[5] E. Konidis, P. Kokkinos, and E. Varvarigos, "Evaluating Traffic Redirection Mechanisms for High Availability Servers," in 2016 IEEE Globecom Workshops (GC Wkshps), 2016, pp. 1-5.
[6] K. Jacksi, N. Dimililer, and S. Zeebaree, A SURVEY OF EXPLORATORY SEARCH SYSTEMS BASED ON LOD RESOURCES. 2015.
[7] A. Barzu, M. Carabas, and N. Tapus, "Scalability of a Web Server: How Does Vertical Scalability Improve the Performance of a Server?," in 2017 21st International Conference on Control Systems and Computer Science (CSCS), 2017, pp. 115-122.
[8] S. Zeebaree and K. Jacksi, "Effects of Processes Forcing on CPU and Total Execution-Time Using Multiprocessor Shared Memory System," ijcert, vol. 351, pp. 275-279, 2015/04/30/ 2015.
[9] S. K. Saeid and T. Ali Yahiya, "Load Balancing Evaluation Tools for a Private Cloud: A Comparative Study," ARO-THE SCIENTIFIC JOURNAL OF KOYA UNIVERSITY; Vol 6, No 2 (2018), 2018 2018.
[10] J. E. C. d. l. Cruz and I. C. A. R. Goyzueta, "Design of a high availability system with HAProxy and domain name service for web services," in 2017 IEEE XXIV International Conference on Electronics, Electrical Engineering and Computing (INTERCON), 2017, pp. 1-4.
[11] I. P. A. Suwandika, M. A. Nugroho, and M. Abdurahman, "Increasing SDN Network Performance Using Load Balancing Scheme on Web Server," in 2018 6th International Conference on Information and Communication Technology (ICoICT), 2018, pp. 459-463.
[12] M. Elgili, Load Balancing Algorithms Round-Robin (RR), Least-Connection and Least Loaded Efficiency. 2017.
[13] A. Hasso, K. Jacksi, and K. Smith, "Effect of Quantization Error and SQNR on the ADC Using Truncating Method to the Nearest Integer Bit," in 2019 International Conference on Advanced Science and Engineering (ICOASE), 2019, pp. 112-117, Zakho - Duhok, Iraq: IEEE.
[14] H. Handoko, S. M. Isa, S. Si, and M. Kom, "High Availability Analysis with Database Cluster, Load Balancer and Virtual Router Redudancy Protocol," in 2018 3rd International Conference on Computer and Communication Systems (ICCCS), 2018, pp. 482-486.
[15] L. M. Haji, S. R. M. Zeebaree, K. Jacksi, and D. Q. Zeebaree, "A State of Art Survey for OS Performance Improvement," Science Journal of University of Zakho, vol. 6, no. 3, pp. 118-123, 2018/09/30/ 2018.
[16] G. Liu, J. Xu, C. Wang, and J. Zhang, "A Performance Comparison of HTTP Servers in a 10G/40G Network," in Proceedings of the 2018 International Conference on Big Data and Computing, 2018, pp. 115-118, 3220216: ACM.
[17] K. Jacksi and S. Abass, "Development History Of The World Wide Web," International Journal of Scientific & Technology Research, vol. 8, pp. 75-79, 2019/09/01/ 2019.
[18] P. Prakash, R. Biju, and M. Kamath, "Performance analysis of process driven and event driven web servers," in 2015 IEEE 9th International Conference on Intelligent Systems and Control (ISCO), 2015, pp. 1-7.
[19] G. Singh and K. Kaur, "An Improved Weighted Least Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems," International Research Journal of Engineering and Technology (IRJET), vol. 5, no. 3, p. 6, 2018 2018.
[20] L. P. Chitra and R. Satapathy, "Performance comparison and evaluation of Node.js and traditional web server (IIS)," in 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), 2017, pp. 1-4.
[21] J. Davis, A. Thekumparampil, and D. Lee, "Node.fz: Fuzzing the Server-Side Event-Driven Architecture," in Proceedings of the Twelfth European Conference on Computer Systems, 2017, pp. 145-160, 3064188: ACM.
[22] Z. Wen, G. Li, and G. Yang, "Research and Realization of Nginx-based Dynamic Feedback Load Balancing Algorithm," in 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2018, pp. 2541-2546.
[23] Netcraft, "December 2018 Web Server Survey," 2018 2018.
[24] S. Neamat, "Factors Affecting Project Performance in Kurdistan Region of Iraq," International Journal of Advanced Engineering Research and Science, vol. 4, no. 5, pp. 1-5, 2017 2017.
[25] J. Chen and W. Cheng, "Analysis of web traffic based on HTTP protocol," in 2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2016, pp. 1-5.
[26] C. Chen, G. Lin, Y. Lin, H. Song, and Y. Bai, "Performance measurement and queueing model of Web servers with a variation of Webpage sizes," in 2015 International Symposium on Next-Generation Electronics (ISNE), 2015, pp. 1-4.
[27] R. Li, Y. Li, and W. Li, "An Integrated Load-balancing Scheduling Algorithm for Nginx-Based Web Application Clusters," Journal of Physics: Conference Series, vol. 1060, p. 012078, 2018/07// 2018.
[28] N. Rathore, "Performance of Hybrid Load Balancing Algorithm in Distributed Web Server System," Wireless Personal Communications, vol. 101, no. 3, pp. 1233-1246, 2018/08/01/ 2018.
[29] F. Mbarek and V. Mosorov, "Load balancing algorithms in heterogeneous web cluster," in 2018 International Interdisciplinary PhD Workshop (IIPhDW), 2018, pp. 205-208.
[30] K. Jacksi, "Design and Implementation of E-Campus Ontology with a Hybrid Software Engineering Methodology," Science Journal of University of Zakho, vol. 7, no. 3, pp. 95-100, 2019/09/30/ 2019.
[31] M. Jin, C. Wang, P. Li, and Z. Han, "Survey of Load Balancing Method Based on DPDK," in 2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS),

2018, pp. 222-224.
[32] A. D. Usman, A. M. S. Tekanyi, and M. Abdulkarim, "Two-Step Load Balancing Scheme for Fairness Improvement in HetNets," NIGERIAN JOURNAL OF TECHNOLOGICAL DEVELOPMENT, vol. 15, no. 2, p. 6, 2018 2018.
[33] A. A. Abdelltif, E. Ahmed, A. T. Fong, A. Gani, and M. Imran, "SDN-Based Load Balancing Service for Cloud Servers," IEEE Communications Magazine, vol. 56, no. 8, pp. 106-111, 2018 2018.
[34] A. B. Prasetijo, E. D. Widianto, and E. T. Hidayatullah, "Performance comparisons of web server load balancing algorithms on HAProxy and Heartbeat," in 2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), 2016, pp. 393-396.
[35] D. Sharma and V. B. Aggarwal, "PERFORMANCE EVALUATION OF DYNAMIC LOAD BALANCING SYSTEM BY USING NUMBER OF EFFECTIVE PARAMETERS," International Journal of Advanced Research in Computer Science, vol. 8, no. 7, p. 4, 2017 2017.