Computer Engineering and Intelligent Systems
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.8, No.2, 2017

Web Server Performance of Apache and Nginx: A Systematic Literature Review

Douglas Kunda[1*] Sipiwe Chihana[2] Muwanei Sinyinda[2]
1.School of Science Engineering and Technology, Mulungushi University, PO box 80415 kabwe, Zambia
2.School of Science Engineering and Technology, Mulungushi University, PO box 80415 kabwe, Zambia

Abstract
Web Server performance is cardinal to effective and efficient Information communication. Performance measures include response time and service rate, memory usage, CPU utilization among others. A review of various studies indicates a close comparison among different web servers that included Apache, IIS, Nginx and Lighttpd among others. The results of various studies indicates that response time, CPU utilization and memory usage varied with different web servers depending on the model used. However, it was found that Nginx out performed Apache on many metrics that included response time, CPU utilization and memory usage. Nginx performance under these metrics showed that its memory (in case of memory) does not increase with increased requests. It was concluded that though Nginx out performed Apache, both web servers are powerful, flexible and capable and the decision of which web server to adopt is entirely dependent on the need of the user. Since some metric such as uptime (the amount of time that a server stays up and running properly) which reflects the reliability and availability of the server and also landing page speed was not included, we propose that future studies should consider uptime and landing page speed in the testing of web server performance.
Keywords: web server, web server performance, apache, Nginx

1. Introduction

The use of Information Communication Technology today cannot be completed without the use of high performance web servers. Due to increase in number of websites, the demand is for accurate, faster, continuous & attractive access to web content (Isha & Vikram, 2015). A Web server is a server that is responsible for accepting HTTP requests from web clients and serving them HTTP responses, usually in the form of web pages containing static (text, images etc.) and dynamic (scripts) content (Ljubuncic, 2011). In other studies, a Web server is a file server connected to its clients via the Internet (Lijun & Qiuyu, 2014). The server can be a part of the network, more precisely the part of the distributed network. The server program is a program that provides its services to the client program that resides either in the same computer or on another computer connected through the network. With the popularization of Internet, Web applications play an important role in the fields of E-Commerce information searching and sharing and such like (Guangzhu & Shujuan, 2009). The main service offered by a web server is most importantly to have access to the documents stored at the web server to web clients. These documents are typically Hypertext Markup Language (HTML) documents but may also be graphics or plain ASCII files. Documents stored at the web server are typically accessed by web clients using a web browser. For Web application system, when the user sends a request from the browser to the results returned, a lot of links are in the middle. If the performance test results don't meet the design or business needs and a performance problem exists (Yu & Xia, 2016).

This paper however reviews performance of the web servers specifically the Apache and Nginx Web Servers. The biggest players in the web server business, Apache and IIS, have had the field to themselves for a long time. Now, however, they have to contend with a few seriously capable upstarts, the most prominent of which is Nginx (pronounced 'engine-x'). Nginx's popularity is mainly due to being open-source and having the desirable combination of high performance and low resource consumption. It is important to note that Nginx is most often compared to Apache due to its similar open-source philosophy According to (Netcraft, 2016) the two web servers are market leaders with market share as of September 2016 of 46.11% and 10.12% respectively.

1.1 An overview of Web Servers

As earlier alluded to the main function of the web server is to service requests for files on the local file system. A client contacts the web server over the network, sends the requests containing the name of the file required and then the server responds with the contents of the file if it exists, the basic operation of this process can be depicted from the figure below.

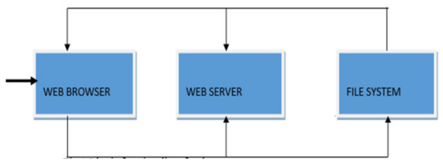

Figure 1: Basic Web Server Functionality

These days, a majority of applications are web-based. Web servers play a pivotal role in such environments as they handle all user requests. Any slowdown in the web server tier will adversely affect the user experience. As the performance of web applications is getting ever more important, as the number of users and competitors is still increasing, for this reason, recent research has comprehensively explored methods for evaluating or optimizing the performance of a Web server (Xianghua, et al., 2013). Web server performance refers to the ability of the Web server responding to users' requests, which directly affects the users' feeling (Zhaoyang & Wei Wang, 2010). Therefore an ideal Web Server is expected to be of high performance. High-performance Web servers reduce the hardware cost of meeting a given service demand and provide the flexibility to change hardware platforms and operating systems based on cost, availability, or performance considerations (Prakash, et al., 2015).

When considering the performance of a web server, several performance measures are of interest. These include resource utilization and response time. Resource utilization of the CPU and local disk in the web server and of the network which denote the percentage of time in which each resource is busy processing jobs. Response time on the other hand refers to the time when a client initiates a request by opening the TCP connection until the response is received. Response time is simply the delay observed from the point-of-view of a client and is highly variable since it depends largely on the size of the document that has been requested.

The use of web servers has mainly been based on expected performance where high performing web servers have a higher market share than the low performing web servers as highlighted by (Netcraft, 2016). Also high performance web servers not only boost confidence to the user but also gives the specific web server an upper hand in terms of competitiveness. This paper therefore discusses the various performance metrics of web servers focusing on the two market share web server leaders namely; Apache Web server and Nginx.

2. Web Server Performance

An appropriate evaluation model of Web servers play a very important role in performance analysis, evaluation and capacity planning (Yin, et al., 2008). A number of studies have been conducted in relation to web server performance. Such studies have been based on different models of analysis. This section discuses web server performance measures and the performance of Apache and Nginx web servers under different performance metric measures.

2.1 Memory Usage and Response Time

In examining the relative performance of each web server (Apache, Nginx and Lighttpd) to see how they compare head-to-head, (Dreamhost, 2016) conducted a web server performance tests on the three web servers. The test methods used ApacheBench (an HTTP server benchmarking tool). In each test, 25,000 requests were made for a 5k PNG file locally from a VPS (Virtual Private Server) to remove potentially variable network conditions from the equation and each test was run with different numbers of concurrent requests to gauge performance at different levels of usage. (Consider that it's very common for browsers to allow up to 6 concurrent connections per single user you have browsing a site. Therefore 10 users browsing your site at the same time would amount to approximately 60 concurrent connections). The metrics under consideration where memory usage and request per second. On memory usage, (Dreamhost, 2016) contended that it is important to measure, especially on a VPS where your memory usage has a hard cap and raising it costs you additional money. The results on memory usage showed that both Lighttpd and Nginx came out as clear leaders in this test. The disparities however were so huge that it had to do with how Apache handles scaling with more incoming requests. To handle additional requests, it spawns new threads (i.e., processes). As more and more connections come in, more and more Apache processes are spawned to handle them. This causes memory usage to grow fairly quickly. The results of the test on memory usage is shown below

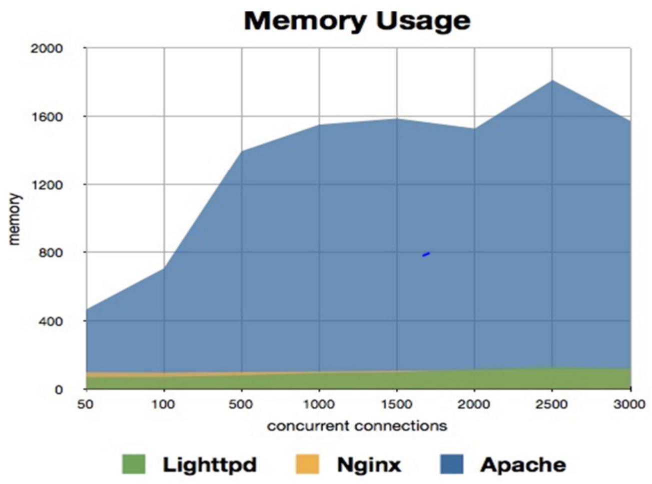

Figure 2: Memory Usage Comparison
Source: (Dreamhost, 2016)

In comparison, you see that Nginx and Lighttpd both have fairly static memory usage. Lighttpd actually increases more over time (and purportedly has issues with memory leaking), whereas Nginx stays fairly static across the board from start to finish. On request per second, (Dreamhost, 2016) noted that this is essentially a measure of how fast the server can receive and serve requests at different levels of concurrency. The more requests they can handle per second, the more able the server is to handle large amounts of traffic. The results showed that Nginx clearly dominates in the raw number of requests per second it can serve. At higher levels of concurrency, it can handle fewer requests per second, but still can serve double what Lighttpd does (which was already doing nearly 4x what Apache was able to do). The results of request per second are shown in the figure below.

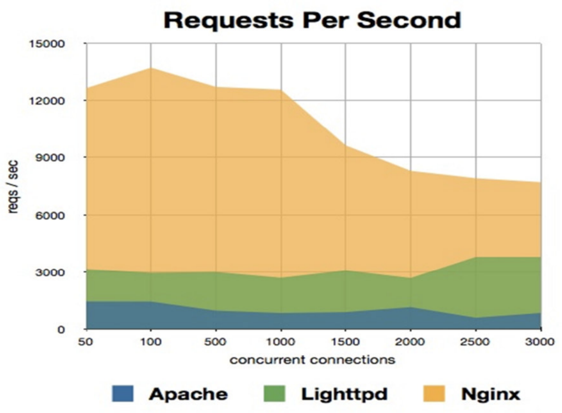

Figure 3: Request Per Second Comparison
Source: *(Dreamhost, 2016)*

It was concluded that Apache supports a larger toolbox of things it can do immediately and is probably the most compatible across all web software out there today. Furthermore, most websites really don't get so many concurrent hits as to gain large performance/memory benefits from Lighttpd or Nginx.

Performance of web servers has been in many cases attributed to software aging. In a related study using response time as a performance metric, in their study on performance modelling of Apache Web Server affected by aging, (Jing & Kishor S., 2011) used Response time (RT) as a customer-affecting metric to detect the onset of software aging. This was in line with Avirtzer et al (2006) who proposed several software aging detection and rejuvenation algorithms based on the measured variation of RT. (Jing & Kishor S., 2011) used Apache web server 1.3.14 in their experiment where the Apache web server was structured as a pool of workers. To assess the impact of MaxRequests per Child and MaxClients, the parameter that can speed up the accumulation of side effects of residual defects, i.e., MaxRequests per Child was adjusted and the parameter was set to zero, which meant no child processes will ever be killed. The other parameter, i.e., MaxClients was set to 250. The connection rate of the data set varies from 350 per second to 390 per second during a 14-day period to represent different workload levels. From this first experiment, they concluded that under this setting the capacity of the web server was about 390 requests per second. In the second experiment, (Jing & Kishor S., 2011)model the Apache Queuing Model according to the Apache working mechanism where the transactional service rate *μ(t)* was defined to be a monotone non increasing function of (*t*). The service time was denoted by the r-stage Erlang distribution, service discipline is FCFS, arriving transactions are queued and the buffer size is K and the queuing model was presented as M/Er/1/K. The results of their experiment showed that the performance of SARAA and SRAA are better than the CLTA. It was then concluded that the optimum performance can be obtained by adjusting the control parameters of bucket number, bucket depth, and sample size.

2.2 Dynamic User Workloads

In analyzing web server performance under dynamic user workloads, (Peña-Ortiz, et al., 2012) analysed the effect of using a more realistic dynamic workload on the web performance metrics. In their study the authors evaluated a typical e-commerce scenario and compared the results obtained using different levels of dynamic workload instead of traditional workloads. Testbed models of TPC-W and the Dweb were used in the same framework. TPC-W in this case is a transactional web benchmark that models an online bookstore environment, which is a representative e-commerce system and the TPC-W specification defines a full website map for the on-line bookstore that consisted of 14 unique pages and their navigation transitions. Dweb was used to define more realistic cases where customers take decisions according to their own navigation objectives and the results of the visited pages (contents retrieved), and change their navigation roles generating dynamic navigations patterns as occurs in the real web. This behaviour according to (Peña-Ortiz, et al., 2012) can be easily defined and generated as a dynamic workload by using Dweb and GUERNICA, which later in the experiment replaced the TPC-W. The setup of their experiment was two-tier configuration that consists of an Ubuntu Linux Server back-end and an Ubuntu Linux client front-end tier. The back-end runs the TPC-W server part, whose core is a Java web application deployed on the Tomcat web application server. Requests to static content of this web application, such as images, are served by the Apache web server, which redirects requests for dynamic content to Tomcat. TPC-W web application generates this type of content by fetching data from MySQL database. On the other hand, the front-end tier is able to generate the workload either using conventional or dynamic models.

Both web application and workload generators are run on the SUN Java Runtime Environment 5.0. The performance metrics were divided into two sides; the customer's side and the server side. On the client side, the metrics were response time and number of requests per page whereas on the server side, On the server side, the platform collects the server performance statistics required by the TPC-W specification (CPU and memory utilization, database I/O activity, system I/O activity, and web server statistics) as well as other optional statistics. These metrics allow a better understanding of the system behaviour under test and permit to check the techniques used to improve performance when applying a dynamic web workload. Experimental results showed that, when a more dynamic and interactive workload is taken into account, performance indexes can widely differ and noticeably affect the stress borderline on the server. The object requests rate decreased between 45% and 60%, but dynamic workloads also changed the request nature, which became more dynamic. This change implied an important growth by 15% in the number of executed queries by the database with a significant increase in their dynamic nature that led the database to be an overloaded application. Consequently, the CPU utilization increased by 30% consolidating the processor as the main performance bottleneck. (Peña-Ortiz, et al., 2012) Concluded that considering user's dynamic navigations on workload characterization has a stronger impact on the system performance metrics than modelling changes in the role, because dynamism is more present in the navigations than in modelling changes.

In a related study on web server performance analysis, the authors in their study used histogram workloads, (E.Hernández-Orallo & J.Vila-Carbó, 2009) observed that Capacity planning relies mainly on two models: a workload model and a performance model. The workload model captures the resource demands and workload intensity characteristics. The performance model is used to predict the performance of a web server system based on the system description and the workload model. The first step in any performance evaluation model is to understand and characterize the workload. Real workloads have a large number of elements, so the information needed to describe the workload must be reduced. In other words, a workload model must capture the most relevant characteristics of the real workload. (E.Hernández-Orallo & J.Vila-Carbó, 2009)thus developed the histogram workload model characterized by the number of HTTP requests on the server during a pre-established time period (sampling period) and the web server performance model which dependent on response time (response time of an HTTP request from when it comes to the server until it is sent back to the sender) and probability rejection (the long-term ratio of the number of the HTTP requests rejected to the number of the total input transactions). In ensuring applicability of their model, experiments were conducted to evaluate the performance of a web server using a given real workload with several clients. This network consisted of several Pentium III1.2 GHz/256 Mbytes boards running Linux (Kernel 2.6.23hrt). These boards are referred to as Blade1, Blade2, etc. Blade 1 contains the Web Server (Apache 2.2.10). The other three blades are clients that reproduce the workload.

Each board is connected to two Ethernet networks: a system network and a test network. The system network is a conventional Ethernet and is used for OS generated traffic (NFS and some other services). The test network uses a switched Ethernet and is only used for the workload. This way, OS-generated traffic does not interfere with web traffic workload at all. Using Tsung, they calculated that the mean number of TPS of this configuration is 193. The experiment was aimed at comparing the response time distribution obtained using our histogram model with the one obtained from real experiments. The HTTP server parameters were the following: throughput R = 193 TPS, HTTP transaction buffer B = 300 and Rout ¼ 100 Mb=s. For the histogram model, we used the DISCA distribution: that is, the transactions histogram A, the response size distributions S and the processing time P. To obtain the real response time distribution, (E.Hernández-Orallo & J.Vila-Carbó, 2009) replayed the HTTP workload log using the Tsung tool and obtained the response time from the Apache server logs. The mean response time on the Apache was 0.0402 s with a maximum of 2.75 s. With our model the results had a mean of 0.0458 s and a maximum of 1.81 s. The distribution of the response time was in the range 0– 0.5 s. Their experiment showed that, the real response times are a slightly lower than the model results, although the error is very low and the curves are very similar. They concluded that the model is accurate compared with real results and that it improves the results of classic queuing models.

2.3 Hardware assisted-virtual machines

Where performance may be attributed to be based on hardware assisted-virtual machines, (Shao, et al., 2009) conducted an experiment in which a Dell PowerEdge 1950 server was used to host all the guests invoked by Xen 3.1.0. The server had 4 Intel Xeon 5110 1.60GHz processors (8 cores totally), 2GB RAM, 120GB 7200 RPM SATA disk and 1Gbps Ethernet link. Domain0 was also installed Fedora core 8 (x86_64 version), and configured with 1GB memory. The HVM guest had a bridged network access to the outside, and Apache 2.2.3 was employed to provide web service. Shao and De Zhong (2009) also employed httperf to measure the performance. During the experiments the CPU usage inside and outside the guests was recorded. The inside CPU usage rate was obtained by using top alike utilities, while the outside CPU usage rate, which reflected the real processor resource utilization status of a guest, was obtained by instrument *Xentop*. One VCPU for both domain0 and the HVM guest was first configured. Three kinds of workloads, each retrieving a fix size file: 1KB, 4KB and 40KB are generated to test the web server hosted by the HVM guest. The HVM guest with more virtual processors (i.e., 2 VCPUs) was configured secondly and the preceding experiment was repeated. The results from these experiments showed that the maximum throughputs of the web server were 475, 300, and 83 requests per second respectively. The HVM inside CPU usage data showed that the VCPU usage reaches almost 100% when the load turns high enough while the outside CPU usage rate of domain0 showed that it can reach 80%~90% during the performance peak. This, according to (Shao, et al., 2009) means that domain0 is not the bottleneck in this experiment since the HVM guest has reached the saturation status antecedently. HVM with 2 VCPU showed that the throughput can reach 650, 450, and 115 requests per second respectively. Compared with the data in HVM with 1 VCPU, the performance is improved nearly 50%. Similarly, it was observed that the inside CPU usage of HVM guest reaches 100%, while the outside CPU usage rate of domain0 is also close to 100% during the performance peak. It was then concluded that when the HVM is configured with 2 VCPU, both domain0 and HVM guest can be fully utilized.

2.4 Dynamic and static content

A study by (He, et al., 2009) on bare PC web server performance on workloads for dynamic and static content designed a bare PC Web server that could serve both dynamic and static content. Measurements of response time, connection time and throughput for workloads containing requests for dynamic and static content indicated that the server had better performance than the Apache and IIS Web servers. To determine the capacity (i.e., maximum number of supported requests/sec) of the extended Web server design when serving only static Web pages a gigabit Ethernet switch was used and response and connection time respectively for the IIS and bare PC servers for up to 2000request/sec with a 3593 byte file. The connection time (after the handshake) is dominated by the delay for the GET request and the disconnection time. They found that the capacity of the bare PC server is 6000 requests/sec compared to 3000 requests/sec for the IIS and Apache servers. However, the response time of the bare PC server only remains stable until 5000 requests/sec and rapidly increases thereafter. The connection time for the server showed similar behaviour except that it only increases slightly after 5000 requests/sec because of the small file size. At capacity, the CPU utilization of the IIS and Apache servers reached 99%. In contrast, the bare PC server CPU utilization was respectively 41% and 82% for3000 and 6000 requests/sec and approximately linear for up to 5000 requests/sec. In studying the performance of the bare PC Web server when serving only dynamic Web pages with a dataset size of 211 bytes, the original PHP file was 469 bytes, and the actual data returned by the DB server consisted of 11 bytes within a 414-byte TCP Payload. The Web server reformatted the PHP file resulting in the 211- byte HTTP file (plus a 151-byte HTTP header) that was transferred to the client. The capacity of the bare PC Web server is 1000 dynamic page requests/sec with a corresponding response time of 66.7 ms but its performance starts to degrade when the request rate exceeds 800 requests/sec. In contrast, the capacity of the Apache and IIS servers was 600 requests/sec and the corresponding response times were 1812 ms and 2189 ms respectively with a significant increase in response and connection times after 400 requests/sec. The server response and connection times for up to 400 dynamic page requests/sec.

The study established similarity in performance of the bare PC and IIS servers. This was because the MySQL server requires the bare PC server to open a new TCP connection for each request whereas the SQL server maintains persistent TCP connections to the IIS server. They also observed that the CPU utilization of the Apache and IIS servers reached 99% at 600 requests/second (with 427 threads at capacity) and 500 requests/sec respectively, while the bare PC server's CPU utilization was only 17.7% for 600 requests/sec and 29% at capacity. They thus concluded that that the bare PC Web server has significant better performance and lower CPU utilization than the Apache and IIS servers for workloads consisting of requests for dynamic and static content.

2.5 Responsiveness, scalability and efficiency

Performance may also be attributed to event of process commonly known as process driven or performance driven web servers. (Prakash, et al., 2015) conducted a performance analysis of process driven and even driven web servers on Apache and Nginx using three matrices, namely; responsiveness, scalability and efficiency where they defined response time, memory usage and error rate as key factors for responsiveness, scalability and efficiency respectively. Their experiment consisted of two client machines and two servers. Servers were hosted on the VMware ESXi hypervisor as a Virtual Private Server. The underlying machine had configuration, i5 CPU with 3.30 GHz, 4GB Memory (RAM), Intel graphic media accelerated HD graphics card and 320GB, 5400 RPM Hard Disk and used Httperf as a tool for measuring web server performance as it provides a flexible facility for generating various HTTP workloads and for measuring server performance. In their test, Prakash et al (2015) involved initiating a total of 5OOO TCP connections and on each connection one HTTP call was performed. In their performance measurements, they tested performance of web servers by hosting the website on VPS and installing benchmark application on client machines. They also focused more on response time, primary memory usage and error rate for efficiency. Two clients were used to send the requests. The tests were carried out by increasing the number of requests from each client during which they measured the memory usage of the server machine. In each test, they calculated the response time and the memory usage. An automated script was written to send requests continuously to the server. The results on both Apache and Nginx showed that as the number of requests increased the memory usage also increased. Using the base line memory of 221 MB which was being used by the other process, it was found that when requests start hitting the server, the memory usage increased and the memory consumption was found to be more in Apache than in Nginx as a result scalability could not be achieved in Apache but in Nginx. In using response time, it was found by Prakash et al that the response time of the Nginx was increasing linearly up to 5000 requests and response time became twice afterwards whereas the response time of the Apache increased linearly up to 5000requests and response time increases to thrice the previous time. The response time of the Nginx is 50% much better as compared to Apache. This experiment showed that Nginx provides better performance in terms of responsiveness and scalability, while its efficiency degrades in heavy load. It was concluded therefore that Nginx web server provides an effective tool to reduce unnecessary memory usage and makes it cost effective to VPS customers.

2.6 Network Bandwidth, Static and Dynamic files

In a related study, (S.Dabkiewicz, 2012) on Web server performance analysis conducted an experiment on the performance of two web servers Apache 2.4.1 and Nginx 1.1.15 and every server got a Debian Squeeze 64-bit installation with standard tools and a SSH-server for remote login. To be able to expect results (S.Dabkiewicz, 2012)added that it was necessary to measure things like network bandwidth and the maximum of connections which the server should be able to handle resulting on the available bandwidth. In this experiment, Iperf was a tool used to measure the network bandwidth which resulted in a network bandwidth of 941 Mbps. The Apache HTTP server benchmarking tool (ab) was be used to test the performance of a web server where both static and dynamic files were used. For static file he used a small HTML-file with a size of 91 bytes and furthermore an image file with a file size of 55 Kilobytes and for dynamic part, he used PHP, although PHP was not really a dynamic language, the file was parsed at the server and then send to the client, which made it a bit dynamic. The results in static showed that apache performed better than one Nginx worker with 7212 requests per second against 7367 requests per second with Apache 2 event. But when increasing the workers in Nginx, it reached 7742 requests per second with 4 workers, which was 500 requests per second more than Apache can handle. On image file, the image file had a file size of 56858 bytes, it was found that the network was the bottleneck, and only 2.068 hits could be expected, which was almost reached during the test. All web servers reached about 2000 requests per second.

In dynamic, the result of the hello worki-file had a file size of 11 bytes with the maximum hits per second of about 10.5 million. These results showed that the Apache web server performs much better than the

Nginx server. 1873 requests per second with Nginx vs. 5142 request per seconds with the event mpm from Apache. Dabkiewicz (2012) observed that even with using more workers in Nginx, the maximum requests which were processed is 3501 request per seconds with 4 workers which is 1600 request per seconds less than the Apache 2 worker server. It was then concluded that Nginx performs very well on static files comparing to Apache as it could handle about 500 requests per seconds more than Apache and that using dynamic files one sees that Apache works much better than Nginx, even with more workers enabled. 1600 request per second is a lot, when assuming that a browser handles about 10 requests per domain. The CPU load was at both server around 30 % during the PHP hello test. The results are found to be consistent with those by (Yong & Masahiko, 2015). He however noted that the main reason of performance imbalance problem is that the high-spec client uses too much of web server resource and network bandwidth.

2.7 Server Architectures (Concurrency Workload)

(Fan & Wang, 2015) On performance comparison of web servers with different architectures studied how different server architectures asynchronous and thread-based impact the responsiveness of web servers under high concurrency workload. Through extensive measurements of a standard web server benchmark (Apache Bench), that the web servers with asynchronous architecture can achieve much better tail latency than the thread-based version due to their robustness to handle high concurrency workload. Fan and Wang’s experiment adopted the standard Apache Bench (AB) to test the performance of different architectures of web servers. Concretely, Apache Bench sends a fixed total number of HTTP requests to the web server under testing with specified concurrency level. They tested five popular open source web servers which were categorized into two groups; asynchronous web server group (Tomcat-NIO, version 7, Nodejs version 0.12.7 and Nginx version 1.4.6) and the thread-based server group (Tomcat-BIO, version 7 and Apache version 2.4.7). The experiment also consisted of one client machine and one web server machine. Each had 2.5 GHz Xeon six-core CPU, 16 GB of RAM, a 10,000 rpm SCSI disk, and four one-gigabit Ethernet cards. Only one CPU core was enabled in BIOS to remove the multi-core complexity. Both the client and the server runs a 3.13.0 Linux kernel. In comparing the average response time of two representative web servers: Tomcat-NIO and Tomcat-BIO while increasing the workload concurrency from 100 to 3200, the average response time of the two servers was quite similar at each workload concurrency, it was found that there was no significant impact of different server architectures on web server performance. For example, at workload concurrency 800, the average response time for Tomcat-BIO was about 350ms while 400ms for Tomcat-NIO. However, the performance difference between the two web servers became clear when more detailed response time analysis was applied. It was found that at workload concurrency 800, the 95th percentile response time was just about 200ms for Tomcat-NIO while over 1 second for Tomcat-BIO. From this experiment, (Fan & Wang, 2015) concluded that the set of experiments showed that different server architectures have significant impact on server tail latency and not the average response time. In configuring Apache and Nginx to serve the same computational tasks as for the other three web servers with the help of additional module PHP. They found that Nginx outperforms Apache under all the workload concurrency and concluded that the Nginx throughput also decreases under very high workload concurrency (e.g., 3200), suggesting non-trivial concurrency overhead as well.

2.8 Webpage sizes and number of users

(Chen, et al., 2015) Analysed web server performance using performance measurements and queuing model with variations of webpage sizes. In their study, (Chen, et al., 2015) measured and modelled the system response times of Web servers with a variation of Webpage sizes and different numbers of users. A queuing model was used as the representation of physical measuring and obtained the simplified relationship between Webpage size and number of users. Their study made use of two computers as performance measurement system. The measurement was started by user clicking the server and waits for the response and when the server responds, the user request for a new service. The test interface was adjusted and set it to 20 minutes with those parameters. When the system response time increased, the number of clicks decreased. With simulation parameters of the numbers ranging from 1 to 100, in increments of 1, the users click randomly all the way. The system recorded every 10 seconds within a click time, the time to the first byte, the time to connect and the time for the DNS. When they took the Webpage sizes ranging from 10 KB to 500 KB, in increments of 10 KB, there were variations in the system response times based on the difference in the Webpage size. If the measurement of the Webpage size is 10KB, and there are 1-100 users, the system response time averaged about 3.77 milliseconds. With the same number of users, a 500KB Webpage averaged a system response time of about 1442.07 milliseconds for each click. A 10KB Webpage size is 50 times smaller than a 500KB Webpage size, but the average system response time was 382 times less. When considering average clicks of each user per time as the arrival rate and Average Bandwidth/Total Webpage Size of Clicks requests/sec) as a measure of the service rate with Webpage sizes of 10 KB-500KB, it was found that if the arrival rate is higher than the service rate, the system response time immediately increased. When the measuring data show a great deal of increment, the bandwidth, on the contrary, was reduced. With Webpage of 460 KB, it was found that the average arrival rate was not changed, but when the number of users was 66, the service rate suddenly increased to 0.08 and then gradually decreased. It was also observed at that time that the bandwidth rose to 4521 KB/s from 942 KB/s, and that the data quantity rose to 56 MB from 53 MB. The average system response time declined to 105 ms from 512 ms, but immediately after the system response time presented the increment of the index, and at 100 users it reaches 1920 ms. Chen et al (2015) thus concluded that both the Webpage size and the bandwidth can be related to the service rate where clicks can be related to the arrival rate and that as the Webpage size increases, the total number of messages will also increase, but when the quantity of data is greater than the system capacity, which is from 400KB to 500KB, the number of messages will be reduced.

3. Open Issues

Following a thorough review on different studies in this paper with regards to web server performance, a comparative performance analysis can be deduced between Apache and Nginx. The performance measures considered in this study are; response time (under static files and dynamic files), memory usage, CPU utilization (under dynamic user loads and variation of web page sizes with different number of users) and Dynamic as well as Static content.

In light of this study, it has been reviewed that, on response time Nginx is much better as compared to Apache as it provides better performance in terms of responsiveness and scalability, while its efficiency degrades in heavy load. Under static files, it has been established that apache performed better than one Nginx worker with 7212 requests per second against 7367 requests per second with Apache 2 event. But when increasing the workers in Nginx, it reached 7742 requests per second with 4 workers, which was 500 requests per second more than Apache can handle. Additionally, in dynamic files it has been found that Apache web server performs much better than the Nginx server. 1873 requests per second with Nginx vs. 5142 request per seconds with the event mpm from Apache. However, this study has found that Nginx clearly dominates in the raw number of requests per second it can serve. At higher levels of concurrency, it can handle fewer requests per second, however it was found that that the Nginx throughput decreases under very high workload concurrency (e.g., 3200).

Under memory usage however we could again see that Nginx came out as clear leaders in the test. This could however be seen with how apache handles scaling with more incoming requests. To handle additional requests, it spawns new threads (i.e., processes). As more and more connections come in, more and more Apache processes are spawned to handle them. This in the end caused memory usage to grow fairly quickly, Nginx however was seen to be providing an effective tool to reduce unnecessary memory usage.

In considering CPU utilization, this study found that Nginx offers low CPU utilization which does not necessarily increase with increased workload while CPU utilization increased in Apache with increased number of requests for both dynamic and static content.

Table 1: Summary of Web Server Performance review-Apache and Nginx.

Author	Web Server	Response time	Memory Usage	CPU Utilization
(Dreamhost, 2016)	Apache	Handles less requests per second at high concurrency	Increases with increase in requests	
	Nginx	More requests per second even at high concurrency	Does not increase with increase in requests	
(Jing & Kishor S., 2011)	Apache	Handles 350-390 requests per second.		
(S.Dabkiewicz, 2012)	Nginx	Under static files-single worker, it handles 7212 requests per second.		
		With four workers, it can handle 7742 requests per second.		
		Under dynamic files, it handles 1873 requests per second		
	Apache	With single worker, it can handle 7367 requests per second		
		With four workers, it can handle 7242 requests per second.		
		Under dynamic files, it can handle 5142 requests per second.		
(Fan & Wang, 2015)	Apache	Under dynamic files, it can handle 5142 requests per second.		
	Nginx	Outperforms under all workload concurrency though its throughput decreases under high workload concurrency		
(Prakash, Biju, & Kamath, 2015)	Apache	5000 requests per second	Increases with increased requests	
	Nginx	Increases by 50% more requests per second than Apache	Increases with increased requests	
(He, Karne, Wijesinha, & Emdadi, 2009)	Apache	6000 requests per second		At capacity, CPU utilization is 99% and 82% at bare PC server
(Peña-Ortiz, et al., 2012)	Apache			CPU utilization increases with workload
	Nginx			CPU utilization does not increase with workload

Conclusion

Web server performance analysis can be conducted using different models/tools e.g. apache benchmarking tool (ab) and among the many factors that could be considered on performance measurement are response time under static files and dynamic files, memory usage, CPU utilization under dynamic user loads and variation of web page sizes for different number of users. Although the studies reviewed have showed performance of other web servers, the focus of this study was the performance metrics of Apache and Nginx. It can therefore be concluded from this study that Nginx performed better than Apache based on major performance measures that include response time, CPU utilization and Memory usage. Both Apache and Nginx are however powerful, flexible, and capable. Deciding which server is best for you is largely a function of evaluating your specific requirements and testing with the patterns that one expects to see.

In the studies reviewed, some metric such as uptime and landing page speed were not considered for performance measurement. Uptime which is the amount of time that a server stays up and running properly reflects the reliability and availability of the web server. On the other hand landing page speed is very critical and average customer's expectation is for a page to load in the blink of an eye. If this expectation isn't met, they are likely to move away quickly often to a competitor's site. Testing the load time of a landing page increases chances of making a good first impression of the server .We therefore, propose that future studies should consider uptime and landing page speed in the testing of web server performance.

Acknowledgment
The authors would like to thank the University Management and School of Science Engineering and Technology of Mulungushi University Zambia for their profound support.

References

Chen, C. et al., 2015. Performance Measurement and Queueing Model of Web Servers with a Variation of Webpage. *Next-Generation Electronics (ISNE),* pp. 1-4.

Dreamhost, 2016. *Dreamhost.* [Online] Available at: www.dreamhost.com [Accessed 2016].

E.Hernández-Orallo & J.Vila-Carbó, 2009. web server performance analysis using histogram workload models. *computer networks,* pp. 2727-2739.

Fan, Q. & Wang, 2015. Performance Comparison of Web Servers with Different Architectures: A Case Study using High Concurrency Workload. *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies,* pp. 37-42.

Guangzhu, J. & Shujuan, J., 2009. A Quick Testing Model of Web Performance Based on Testing Flow and its Application. *Web Information Systems and Applications Conference,* pp. 57-61.

He, L., Karne, R., Wijesinha, A. & Emdadi, A., 2009. A Study of Bare PC Web Server Performance for Workloads with Dynamic and Static Content. *2009 11th IEEE International Conference on High Performance Computing and Communications.,* pp. 1-6.

Isha, A. & Vikram, B., 2015. A Brief Survey on Web Application. *International Journal of Latest Trends in Engineering and Technology (IJLTET),* may, 5(3), pp. 367-375.

Jing, Z. & Kishor S., T., 2011. PerformanceModeling of ApacheWeb Server Affected by Aging. *2011 Third International Workshop on Software Aging and Rejuvenation,* pp. 1-6.

Lijun, Z. & Qiuyu, Z., 2014. The Overtime Waiting Model for WebServer Performance Evaluation. *2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery,* pp. 229-232 .

Ljubuncic, I., 2011. *Apache Web Server Complete Guide.* s.l.:www.dedoimedo.com.

Netcraft, 2016. *September 2016 Web Server Survey.* [Online] Available at: https://news.netcraft.com

Peña-Ortiz, R., Gil, J., Sahuquillo, J. & A.Pont, 2012. Analysing web server performance under dynamic user workloads. *Computer communications,* 36(4), pp. 386-395.

Prakash, P., Biju, R. & Kamath, M., 2015. Performance Analysis of Process Driven and Event Driven Web Servers. *IEEE Sponsored 9th International Conference on Intelligent Systems and Control (ISCO)2015,* pp. 1-7.

S.Dabkiewicz, 2012. Web Server Performance Analysis. *Lia project,* pp. 1-14.

Shao, Z., Jin, H. & Zhang, D., 2009. A Performance Study of Web Server Based on Hardware-assisted Virtual Machine. *The 7th IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2009, Rabat, Morocco, May 10-13, 2009,* pp. 1-4.

Xianghua, X., Tingting, X. & Yuyu Yin, J. W., 2013. Performance evaluation model of Web servers based on response time. *2013 IEEE Conference Anthology,* pp. 1-5.

Yin, Z. M., Xiao, Z. & H. Wang, 2008. A Web Performance modelling based on methodology learning form date. *international conference for young computer scientists,* pp. 1285-1291.

Yong, J. & Masahiko, T., 2015. Web server performance enhancement by suppressing network traffic for high performance client. *APNOMS,* pp. 448-451.

Yu, Y. & Xia, J., 2016. Analysis and Research on the Performance. *Analysis and Research on the Performance Optimization of web Application System in High Concurrency Environment,* pp. 1-6.

Zhaoyang, Q. & Wei Wang, Z. L., 2010. *Web Server Optimization Model Based on performance analysis.* s.l., IEEE, pp. 1-4.