**Software Engineering Program**

**School of Science and Computer Engineering**

**University of Houston Clear Lake**

**Spring-2016**

**SWEN 5233, Section 01 - Software Architecture**

**Project Milestone 2**

**Part 2**

**Web Server**

**Mohan Ajay Kumar Yaramati - Apache**

**Sudha Mounika Jaripiti - Nginx**

**Dr. Sheena Judson Miller**

**05/02/2016 11:59AM**

# Introduction:

Web server is a technology which is associated and transfers the client request through Text Transfer Protocol. It processes it and frames with a desired response based on client requirement. Web page basically consists HTML information which has images, style and other scripts. The main function of the web server is storing web pages, process and to deliver the pages as response to clients.

The Nginx server is another frame to Apache web server. As it experiences the problems in Apache web server, the Igor Sysoev has come up with a new idea of web server called Nginx web server. At present webserver is used almost 29.8% more than the host server. The main idea of developing the server is to achieve greater accuracy.

Apache is popular from longtime and the support for the server is fair. There is large library with different parties like first and third party which calls Apache with other software.

In general Apache will have support from third party as its value and time is available. Administrators are likely to work with Apache as many people start in shared hosting scenarios.

Nginx has increased support just because many adopt it for performance. Earlier, it was difficult to experience documentation of English language in Nginx because the fact was that most of earlier development and documentation are in Russian, the documentation has filled and plenty of resources on Nginx site and third parties.

The third party applications, documentations and support is more available and in some cases it gives a choice between auto configuration for Apache and Nginx. Even with no support Nginx will work alternative and is usually straight forward as long as the project itself documents the requirements.

On same hardware Apache has more performance when compared with Nginx. The event loop replaces thread per connection and solves the problem. In Nginx by adding a single reverse proxy to single application server. An additional lag request and response flow from reverse proxy server to application server but these communications have low latencies.

The features of concurrency in web servers is likely to be so high, as web servers use request without any delay. The attributes can be discussed regarding the features and their performance. These features are required for serving large number of requests with easy and fast accessibility which links each module to serve the request fast and with more accuracy

# Overview

Apache is built for support in different languages. The mod\_python and mod\_php Apache modules are used to process the PHP and perl code in processor of Apache. Mod\_python is more efficient as it does not have to load for each request. Apache might be preferred for application as Apache does not use Common Gateway Interface

In regards with concurrency, Nginx has been written specifically to address the performance of webserver related to Apache. As a scaling, the site manage is greater in volumes of traffic. It can be deployed as web server. The apache serves in this drop in solution and acts as network load. The effect of this is to restore the performance of Apache to levels of administration.

The performance and scalability of Nginx starts from event driven architecture. It significantly differs from Apache server in processing the threads. In Nginx the process has a capability to handle thousands of HTTP. This can give an output related to high implementation that results in scalability and performance.

Using Nginx gives consistency within each service. The distribution of web application can be used as a mix of many application components together, hence they cannot combine a number of different technologies and platforms. The lightweight nature of the Nginx server is much better decision. Therefore, both Apache and Nginx both have their own place but Nginx is clearly in dominant. Nginx has its own place in the architecture as the ideal application platform.

# Diagram and Description

Apache web server has a measured architecture with a center component that characterizes the most fundamental usefulness of a web server and various modules which executes the progressions of preparing a HTTP ask for, offering handlers for one or a greater amount of the stages. Concurrency exists just between various steady indistinguishable procedures that serve approaching HTTP asks for on the same port. Modules are not actualized as isolated procedure in spite of the fact that it is conceivable to fork youngsters or to participate with other free procedure to handle a period of preparing a request.

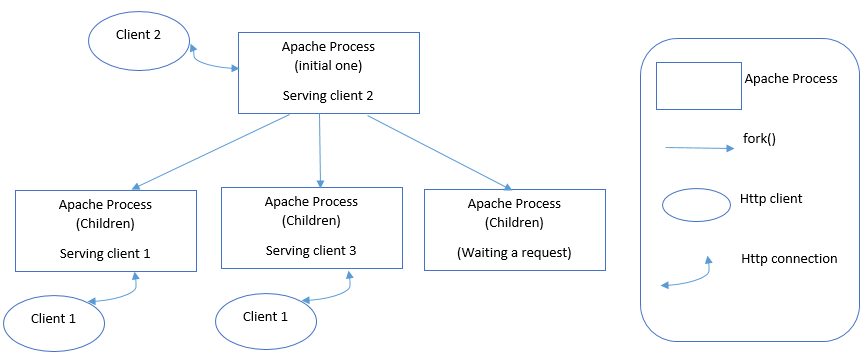


Figure 1 Apache Concurrency [7]

Ngnix has almost-unlimited concurrency. It transforms worst case traffic to best case and maximizes application utilization. It has improved application availability and capacity with advanced techniques like A|B testing.

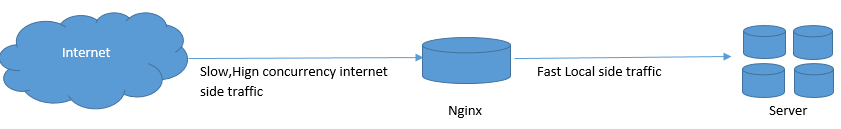


Figure 2 Ngnix Concurrency

## Web server performance in memory usage

This is imperative to measure particularly on a Virtual Private Server where your memory use has a hard top and raising it costs you extra cash. It needs to do with how Apache handles scaling with all the more approaching requests. To handle extra demands, it brings forth new processes. As more connections come in, more Apache procedures are produced to handle them. This causes memory use to become decently fast.

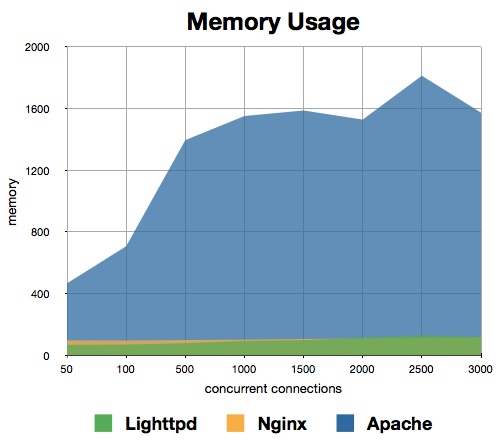


Figure 3 Memory Usage [3]

## Efficiency of web servers in responding

This is basically a measure of how quick the server can get and serve demands at various levels of concurrency. The more demands they can deal with every second, the more capable the server is to handle a lot of activity. Here's the means by which the servers think about in this coliseum. Nginx plainly commands in the crude number of solicitations every second it can serve. At more elevated amounts of concurrency, it can deal with less demands every second, yet at the same time can serve triple what Nginx does.

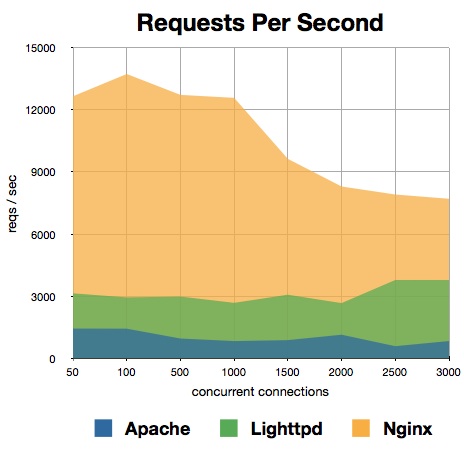


Figure 4 Request Per Second [3]

# Comparison

One major contrast among Apache and Nginx is the real way is

## Handle associations and activity.

This gives perhaps the most important contrast in the way that they react to various activity conditions.

Apache gives an assortment of multi-processing modules that manage how customer requests are taken care of. Fundamentally, this permits heads to swap out its association taking care of design effectively. Apache gives an adaptable architecture to picking diverse association and request taking care of algorithms. The decisions gave are basically an element of the server's development and the expanding requirement for concurrency as the web scene has changed.

Nginx went onto the scene after Apache, with more attention to the concurrency issues that would confront destinations at scale. Utilizing this learning Nginx was planned from the beginning to utilize an asynchronous, non-blocking, event-driven association taking care of algorithms. Each of the associations took care of by the worker are placed inside the occasion circle where they exist with different associations. Inside the loop, occasions are prepared asynchronously, permitting work to be taken care of in a non-blocking way. At the point when the association closes, it is expelled from the loop. This style of association preparing permits Nginx to scale incredibly far with constrained resources.

## Static vs Dynamic Content

As far as real world use-cases a standout amongst the most well-known correlations between Apache and Nginx is the path in which every server handles demands for static and element content.

Apache servers can deal with static substance utilizing its ordinary record based strategies. The execution of these operations is primarily an element of the MPM strategies depicted previously. Apache can likewise handle dynamic substance by implanting a processor of the dialect being referred to into each of its laborer occasions. This permits it to execute dynamic substance inside the web server itself without relying on outer segments. These dynamic processors can be empowered using progressively loadable modules. Apache's capacity to handle dynamic substance inside implies that arrangement of element preparing has a tendency to be less complex. Correspondence does not should be composed with an extra bit of programming and modules can undoubtedly be swapped out if the substance prerequisites change.

Nginx does not have any capacity to process dynamic substance locally. To handle PHP and different solicitations for element content, Nginx must go to an outside processor for execution and sit tight for the rendered substance to be sent back. The outcomes can then be transferred to the customer. For managers, this implies correspondence must be designed amongst Nginx and the processor more than one of the conventions Nginx knows how to talk. This can confuse things in some way particularly when attempting to envision the quantity of associations with permit, as an extra association will be utilized for every call to the processor. This strategy has some favorable circumstances too. Since the dynamic translator is not inserted in the laborer handle, its overhead may be available for element content. Static substance can be served in a straight-forward way and the translator may be reached when required. Apache can likewise work in this way, however doing as such evacuates the advantages in the past segment.

## Modules

Both Nginx and Apache are extensible through module frameworks, however the way that they work vary altogether.

Apache's module framework permits you to powerfully stack or empty modules to fulfill your necessities over the span of running the server. The Apache center is constantly present, while modules can be turned on or off, including or expelling extra usefulness and guiding into the primary server. Apache utilizes this usefulness for an extensive assortment undertaking. Because of the development of the stage, there is a broad library of modules accessible. These can be utilized to modify a percentage of the center usefulness of the server, for example, mod\_php, which inserts a PHP translator into every running specialist. Modules are not constrained to preparing dynamic substance, in any case. Among different capacities, they can be utilized for revising URLs, verifying customers, solidifying the server, logging, reserving, pressure, proxy, rate restricting, and encoding. Dynamic modules can develop the center usefulness extensively without much extra work.

Nginx likewise actualizes a module framework. It is very unique in relation to the Apache framework. In Nginx, modules are not progressively loadable, so they should be chosen and incorporated into the center programming. For some clients this will make Nginx a great deal less adaptable. This is particularly valid for clients who are not open to keeping up their own particular ordered programming outside of their conveyance's traditional bundling framework. While dispersions bundles have a tendency to incorporate the most generally utilized modules, on the off chance that you require a non-standard module, you will need to manufacture the server from source yourself. Nginx modules are still extremely helpful however, and they permit you to manage what you need out of your server by just including the usefulness you mean to utilize. A few clients additionally may consider this more secure, as subjective parts can't be guided into the server. Notwithstanding, if your server is ever placed in a position where this is conceivable, it is likely traded off as of now. Nginx modules permit a large number of the same abilities as Apache modules. For example, Nginx modules can give proxy bolster, pressure, rate restricting, logging, reworking, geolocation, validation, encryption, spilling, and mail usefulness.

# Conclusion:

Subsequently after going over the advantages and confinements of both Apache and Nginx, you may have a superior thought of which server is more suited to your necessities. In any case, numerous clients find that it is conceivable to influence every server's qualities by utilizing them together. The ordinary setup for this association is to put Nginx before Apache as an opposite intermediary. This will permit Nginx to handle all solicitations from customers. This exploits Nginx's quick preparing pace and capacity to handle expansive quantities of associations simultaneously.

For static substance, which Nginx exceeds expectations at, the documents will be served rapidly and specifically to the customer. For element content, for occurrence PHP records, Nginx will intermediary the solicitation to Apache, which can then process the outcomes and return the rendered page. Nginx can then pass the substance back to the customer. This setup functions admirably for some individuals since it permits Nginx to work as a sorting machine. It will handle all solicitations it can and go on the ones that it has no local capacity to serve. By eliminating the request, the Apache server is requested that handle, we can mitigate a portion of the obstructing that happens when an Apache process or string is involved.

This design likewise permits you to scale out by including extra backend servers as essential. Nginx can be arranged to go to a pool of servers effortlessly, expanding this current setup's flexibility to disappointment and execution.

Both Apache and Nginx are intense, adaptable, and skilled. Choosing which server is best for you is to a great extent an element of assessing your particular prerequisites and testing with the examples that you hope to see. There are contrasts between these ventures that have an undeniable effect on the crude execution, abilities, and the usage time important to get every arrangement up and running. Nonetheless, these typically are the consequence of a progression of exchange offs that ought not be coolly rejected. At last, there is nobody size-fits-all web server, so utilize the arrangement that best adjusts to your goals.

# References:

[1] Nginx *Nginx vs. Apache: Our View of a Decade*[Online]. Available https://www.nginx.com/blog/nginx-vs-apache-our-view/

[2] Anturis N*ginx vs. Apache*[Online]. Available https://anturis.com/blog/nginx-vs-apache/

[3] Dream Host Wiki *Web Server Performance Comparison*[Online]. Available http://wiki.dreamhost.com/Web\_Server\_Performance\_Comparison

[4] DigitalOcean *Apache vs Nginx:*  *Practical Consideration*[Online]. Available https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations

[5] Nginx *Maximizing PHP 7 Performance with NGINX, Part II: Multiple Servers and Load Balancing*[Online]. Available https://www.nginx.com/blog/maximizing-php-7-performance-with-nginx-part-ii-multiple-servers-and-load-balancing/

[6] DZone *Maximizing Drupal 8 Performance with Nginx, Part I*[Online]. Available https://dzone.com/articles/maximizing-drupal-8-performance-with-nginx

[7] Slideshare *Apache Web Server*[Online]. Available http://www.slideshare.net/IvanGeorgeArouje/apache-webserverarchitecture