Apache and Nginx

Comp531 Paper
NetId: hx12
Name: Haoyuan Xia

Abstract

In this paper, I will introduce and compare two popular web servers, Apache and Nginx. I will firstly give an overview of these two web servers and discuss some features of them. Following are comparisons between Apache and Nginx in a variety of aspects. Finally, I will make a conclusion.

Overview

Apache and Nginx are the two most common open source web servers in the world. Together, they are responsible for serving over 50% of traffic on the internet. Both solutions are capable of handling diverse workloads and working with other software to provide a complete web stack.

While Apache and Nginx share many qualities, they should not be thought of as entirely interchangeable. Each excels in its own way and it is important to understand the situations where you may need to reevaluate your web server of choice.

Features

Apache

The Apache HTTP Server was created by Robert McCool in 1995 and has been developed under the direction of the Apache Software Foundation since 1999. It is the web server component of the popular LAMP (Linux, Apache, MySQL, PHP) stack. Though there are many other web stack components these days (e.g., *NodeJS*, rich clients JS frameworks, various cloud services, etc.), LAMP still remains very popular. Because of this popularity, Apache benefits from great documentation and integrated support from other software projects.

Apache is often chosen by administrators for its flexibility, power, and widespread support. It is extensible through a dynamically loadable module system and can process a large number of interpreted languages without connecting out to separate software.

Nginx

In 2002, Igor Sysoev began work on Nginx as an answer to the C10K problem, which was a challenge for web servers to begin handling ten thousand concurrent connections as a requirement for the modern web. The initial public release was made in 2004, meeting this goal by relying on an asynchronous,

event-driven architecture to handle massive amounts of connections. This architecture makes handling high and fluctuating loads much more predictable in terms of RAM usage, CPU usage, and latency.

Nginx has grown in popularity since its release due to its light-weight resource utilization and its ability to scale easily on minimal hardware. It excels at serving static content quickly and is designed to pass dynamic requests off to other software that is better suited for those purposes.

Nginx is often selected by administrators for its resource efficiency and responsiveness under load. Advocates welcome Nginx's focus on core web server and proxy features.

Performance

Static Content

Apache servers can handle static content using its conventional file-based methods. The performance of these operations is mainly a function of the MPM methods described above.

Nginx is about 2.5 times faster than Apache based on the results of a benchmark test running up to 1,000 concurrent connections. Another benchmark running with 512 concurrent connections, showed that Nginx is about two times faster and consumed a bit less memory (4%).

Clearly, Nginx serves static content much faster than Apache. If you need to serve a lot of static content at high concurrency levels, Nginx can be a real help.

Dynamic Content

Apache can also process dynamic content by embedding a processor of the language in question into each of its worker instances. This allows it to execute dynamic content within the web server itself without having to rely on external components. These dynamic processors can be enabled through the use of dynamically loadable modules.

Nginx does not have any ability to process dynamic content natively. To handle PHP and other requests for dynamic content, Nginx must pass to an external processor for execution and wait for the rendered content to be sent back. The results can then be relayed to the client. However, this method has some advantages as well. Since the dynamic interpreter is not embedded in the worker process, its overhead will only be present for dynamic content.

Modules

Apache

Apache's module system allows you to dynamically load or unload modules to satisfy your needs during the course of running the server. The Apache core is always present, while modules can be turned on or off, adding or removing additional functionality and hooking into the main server.

Apache uses this functionality for a large variety tasks. Due to the maturity of the platform, there is an extensive library of modules available. These can be used to alter some of the core functionality of the server, such as *mod_php*, which embeds a PHP interpreter into each running worker.

Nginx

Nginx also implements a module system, but it is quite different from the Apache system. In Nginx, modules are not dynamically loadable, so they must be selected and compiled into the core software.

For many users, this will make Nginx much less flexible. This is especially true for users who are not comfortable maintaining their own compiled software outside of their distribution's conventional packaging system. While distributions' packages tend to include the most commonly used modules, if you require a non-standard module, you will have to build the server from source yourself.

Security

Both projects have an excellent security track record for their C-based code base. The NGINX code base, however, is significantly smaller by several orders of magnitude, so that is definitely a big plus from a forward-thinking security perspective.

There is vulnerability reporting available for Apache 2.2 and 2.4. NGINX also has a list of recent security advisories. Apache offers configuration tips for DDoS attack handling, as well as the *mod evasive* module for responding to HTTP DoS, DDoS, or brute force attacks. You can also find helpful resources for dealing with DDoS threats on the NGINX blog.

Conclusion

According to the comparisons before, we can see that both Apache and Nginx are powerful, flexible, and capable. Deciding which server is best for users is largely a function of evaluating specific requirements and testing with the patterns expected to see.

There are differences between these projects that have a very real impact on the raw performance, capabilities, and the implementation time necessary to get each solution up and running. However, these usually are the result of a series of tradeoffs that should not be casually dismissed. In the end, there is no one-size-fits-all web server, so use the solution that best aligns with specific objectives.

References:

[1] Soledad, Nell David S (2009). "Eastern Apache Wizardcraft", *Mythical papers of the University of Cebu* (No.14). Philippines: University of Cebu Press.

[2] Seymour, Deni J. (2009a) "Nineteenth-Century Apache Wickiups: Historically Documented Models for Archaeological Signatures of the Dwellings of Mobile People", Antiquity 83(319):157–164.

[3] Seymour, Deni J.(2009b) "Evaluating Eyewitness Accounts of Native Peoples along the Coronado Trail from the International Border to Cibola", New Mexico Historical Review 84(3):399–435.

[4] Foster, Morris W; & McCollough, Martha. (2001). "Plains Apache", in R. J. DeMallie (Ed.), Handbook of North American Indians: Plains (Vol. 13, pp. 926–939). Washington, D.C.: Smithsonian Institution.