A Systematic Review of Web Server Technologies

Brixton Hamilton

Abstract

The choice between different web server technologies is an increasingly relevant problem that users must face. This article will outline the features and functionalities that web servers perform, the differences between them, focusing on Apache, IIS, Jetty and Nginx, and how those relate to the requirements of different users. I will analyse the current trends in web server technologies, and suggest how they might continue to change in the future.

As the internet rapidly grows and evolves, so must the web server technologies we use to host the websites on it. From merely tens of thousands of sites predominantly using NCSA server technology shortly after the birth of the internet, as of November 2016 there are 171 Million active sites[1], and over time the ever-changing needs of these sites will need to be met by more suitable web servers. The role of the web server in isolation is clear-cut, it must handle HTTP requests. However, outside this it must work with database systems (such as MySQL or MongoDB), serve dynamic content using server-side scripting languages (e.g. PHP or ASP .NET), and deal with the operating system on which it runs (most commonly Linux). Many different web server technologies exist, and when choosing between them, compatibility with the other elements of your server architecture is important. For the past 20 years, the clear-cut leader for active websites has been Apache, an adaptable, open-source server, usually run alongside Linux, PHP and MySQL as part of a ‘LAMP Stack’.

![Web server developers: Market share of active sites](image)


The reasons for Apache’s dominance over the market are many, but flexibility plays a key role. Apache is highly customizable, with added functionality via ‘modules’, allowing the user to adapt it to suit their needs, from varying sizes and types of website. It is also free and open-source, which makes it very accessible to the masses, complete with thorough first and third-party documentation. Finally, it provides a high level of security, making it a good all-round choice. For those with more
specific requirements, however, there are plenty of alternatives. Nginx is relatively new in comparison, performance-based and fast gaining representation. IIS is Microsoft’s own server architecture, and Jetty is a java-based web server. In comparing the features and benefits of different web server architectures, and how they might suit the needs of different users, I will focus primarily on these four. A comparison of some selected features is shown below.

<table>
<thead>
<tr>
<th>Server</th>
<th>Language</th>
<th>Windows</th>
<th>Linux</th>
<th>HTTPS</th>
<th>ASP.NET</th>
<th>CGI</th>
<th>FastCGI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>C</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>IIS (Microsoft)</td>
<td>C++</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Jetty</td>
<td>Java</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Nginx</td>
<td>C</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Wikipedia – Comparison of Web Server Software [3]

Beyond the basic requirements, almost all web servers share a list of common features and functionalities, one of which is the ability to process HTTPS requests. This creates a secure channel between the client and the server, providing authentication and confidentiality for requests passed. This is paramount when sending and receiving sensitive data, such as banking information.

The first discrepancy between these servers is the operating systems on which they can run. Apache and Nginx can be run both on Windows and UNIX-like operating systems (notably Linux and OS X). This covers most of the commonly used operating system structures, which make them more versatile. IIS, however, can only be run on Windows, and comes packaged with Windows 8.1 and higher. This is slightly more limiting to users when selecting an Architecture for their web server. Jetty, however, is the most versatile of the listed servers, as it runs on the JVM which is designed to be run on almost any operating system or device. As the internet of things grows, using java-based web servers lends itself very well to connecting embedded systems to the outside world.

For production of dynamic content, a language must be chosen for server-side scripting. Apache and Nginx are again very versatile, allowing multiple languages to be used, the most common of which is PHP. For IIS, Microsoft use ASP .NET, a server-side web application framework allowing code in any .NET language. Jetty uses Java Servlets which handle dynamic JSP pages, allowing object-oriented Java code to be included in the pages.

CGI, or Common Gateway Interface, allows the server to interface with these methods of dynamic page generation. Apache, IIS and Jetty all have the capability to use this feature, although some alternatives are included, such as Apache’s mod_php module. Nginx, however, doesn’t use CGI at all, opting to implement FastCGI, a more efficient variation which reduces computational overhead. This overhead is an important flaw in CGI, as if the web server is handling tens of thousands of requests, CGI creates a new process for each request, and this can add up to a very large amount of wasted processing power. FastCGI provides more scalability, providing FastCGI processes that can each handle multiple requests.
Another similar issue that can arise with Apache, IIS and Jetty is that they create a new thread for every client. This can cause blocking and consumes more resources creating separate processes. Nginx, however, implements an event-driven architecture, which is asynchronous, non-blocking, and single-threaded. A master process creates multiple worker processes, which can each handle multiple requests and responses to the server. This allows resources used by the server to be allocated and released dynamically, which in turn increases efficiency. This leads Nginx to being more suitable when high performance is important, such as high traffic websites with thousands of static requests. This is evident in the graph shown, where among the top 1000 visited websites, Nginx is more than twice as popular as its nearest competitor, Apache. This is a stark contrast to the data when taking into account all websites.

Another feature that sets Nginx apart from the rest is it is designed around the capability to be used as a reverse proxy. This allows caching of static requests, and allocation of requests to one host to multiple servers. It can also perform as a load-balancer, spreading requests out between multiple servers to ensure none of them have to handle too many at the cost of performance. This makes it even stronger a choice for a high-traffic site. These functions are also possible using apache modules, and are included with IIS.

For the majority of the internet, the choice in web-servers is increasingly becoming a two-horse race between Nginx and Apache, and it is clear that the undisputed reign of Apache is slowly coming to an end. The percentage of active sites that it represents has dropped over 10% [5] in the past five years, and shows no signs of that changing in the near future. The difference in benefits between the two is very easy to understand. Apache brings usability and versatility to the table, making it the preferred platform for the majority of smaller users. It has over 20 years of documentation, and most features work ‘out of the box’. Nginx on the other hand, requires a lot more configuration, is much newer, and is a Russian product, meaning some of the documentation needs to be translated. All these things become less important when performance is your primary concern, as Nginx is the clear-cut winner. As time goes on and Nginx becomes more established and user-friendly, we are likely to see it become even more dominant for the highest-traffic websites, and gain significant ground overall.

One thing that is important to consider is the impact on managed hosting services on these trends. A lot of less experienced users will opt into paying a third-party to host their website, often removing control of which Server architecture is chosen. If these services switch between technologies, a drastic amount of the market can shift in turn. The GoDaddy Group is the largest web hosting provider, and uses Apache for the majority of their sites [6]. Were this to switch in the future, we could see a significant shift in the current paradigm.
The market is not without new contenders, with LiteSpeed web server gaining 40 Million new sites in November alone, a growth of 740% [7]. LiteSpeed focuses on performance, similar to Nginx, while maintaining full compatibility with several Apache features [8], allowing easy switching between the two. This could allow it to significantly grow in the future, and fill the gap left by users switching away from Apache as it becomes more outdated.

In conclusion, it is clear that the market is wide-open for either an established technology or a new one to take-over, if it can pull ahead of the competition. Within the next few years, we could easily see a complete shift away from Apache’s 20-year reign. Most of the popular servers have very similar functionality, but there are more choices than ever before, allowing the room for individual servers to set themselves aside from the others, and command a higher market share.

References


General:


Newcastle University. CSC3422 Lecture Slides

1445 words including abstract, excluding tables and references