PHP web serving study

Performance report

by Stefan "SaltwaterC" Rusu

Date: May - June 2010
Contact: http://saltwaterc.net/contact
or admin [at] saltwaterc [dot] net

Hardware Configurations

Web Server: AMD Phenom II X4 940, ASUS M3A78 Pro, Western Digital WD5000AAKS, 4x Geil Ultra 2 GiB RAM DDR2 PC2-6400

Database Server: AMD Athlon II X2 240e, ASUS M4A78-AM, Western Digital WD5000AAJS, 2x A-DATA 2GiB RAM DDR2 PC2-6400

Test Machine: AMD Turion 64 X2 TL-50 on ASUS A6M-Q030, Hitachi Travelstar 7k100 80GB, 2x Corsair VS 1GiB RAM DDR2 PC2-5300

Network: ASUS WL500g Premium V1, 100Mbps CAT 5 UTP cables

Software Configurations

OSes: Windows Web Server 2008 R2, Ubuntu Server 10.04 LTS x86_64

Test machine OS: Ubuntu 10.04 LTS x86_64

Web Servers: IIS 7.5, Apache 2.2.14 + prefork MPM, worker MPM, and event MPM, Cherokee 1.0.2, Hiawatha 7.2, lighttpd 1.4.26, LiteSpeed Enterprise & LiteSpeed Standard 4.0.14, nginx 0.7.65

PHP Stack: Zend Server Community Edition 5.0.1 + PHP 5.3.2

MySQL: 5.1.41

IIS used the FastCGI SAPI through Named Pipe. Most of the settings were defaults. The rewrite rules were imported from the .htaccess files.

Apache + prefork MPM used the mod_php5 SAPI.

Apache + worker & event MPM used the FastCGI SAPI through mod_fcgid (reconfigured FastCGI binary that usually is used by the Zend Server GUI).

Cherokee, Hiawatha, lighttpd and nginx used a Zend Server binary compatible PHP-FPM (FastCGI Process Manager) custom build.

LiteSpeed used a Zend Server binary compatible PHP LSAPI custom build.

Under both platforms, the Zend extensions (Optimizer+, Data Cache, Debugger) were disabled. The configurations were mostly the same. All of the web servers actively used their URL rewrite support during certain tests.

All the tested web servers had their logging capability enabled, thus the OS/filesystem performance was also part of the actual test.

Except Apache + prefork MPM + mod_php5, all the web servers used 10 PHP upstreams processes.

Under Windows, the Zend Optimizer+ extension was replaced by WinCache 1.0.1.

Under Ubuntu, except mod_php5, all the connections to the external PHP runtime were made through UNIX sockets.

All the servers that ran under Ubuntu used the same configuration ini files. The Zend Optimizer+ bytecode accelerator was dropped in favor of XCache 1.3.0 that usually performed better. The original reason was Zend Optimizer+'s explicit refusal to work under LSAPI. None of the web servers that used PHP-FPM have built in support for FastCGI process spawning, therefore using this solution for having a dedicated PHP daemon was the right thing to do.

LiteSpeed Tech implemented its own PHP SAPI, the LSAPI, a FastCGI like IPC method that provides better performance, or at least it is marketed as such.

Apache was aided by mod_cache and its memory backend in order to speed up the static objects serving.

The web servers used static content compression, if possible dynamic content compression. PHP's compression support wasn't used.

Besides the first PHP stack tests, Apache JMeter was the only tool that did the web server stress testing. JMeter used 25 threads (clients) with 0 ramp up period for 1000 loops. That's 25,000 requests per each file. For generating a test scenario, the HTTP proxy functionality of JMeter was used in order to capture the request headers for all the requested objects. Cherokee's JMX files were used as templates for the other web servers testing. The reason for using these files is purely alphabetical as originally Apache's results were saved under the 'LAMP' directory, while the first data involved just IIS, Apache, and nginx. During the tests I took the decision to evaluate more web servers.

1000 loops / 25000 requests were executed for better accuracy. Due to the actual size and work involved in this test, multiple concurrency levels weren't benchmarked. Higher concurrency wasn't possible in certain tests (such as Drupal and Joomla) due to high number of objects and CPU bottleneck onto the test machine. The available network bandwidth was rarely a concern during the tests as the bottleneck usually was located somewhere else.

Some random details

If you're questioning the authenticity of this test, there are about 6.9 GiB of XML dumps aka test results. Even compressed there's still a large amount of data, but it can be provided. The configuration details are smaller though. This is just a preliminary, performance report. A full article with all the juicy details is forthcoming. The article would be more about the experience with the web servers and less about performance. I will do better benchmarks, but with less web servers involved. This experience acts as filter for future tests.

Drupal and Joomla were used because they are the definition of poor performance, thus great for benchmark tools. Even the default themes are the opposite of any decent web performance book since they load a lot of static objects. Don't get me started about the coding.

The tests

Synthetic Tests

1) The Free Web Hosting test is mostly a mathematical test. It actually benchmarks the PHP stack performance, without any server involvement.
2) phbench is another PHP stack test that uses more functionality from the PHP reference in order to provide a complex alternative to the Free Web Hosting test. Currently this application is under development. So far it provides just minimal functionality. This benchmark was developed by me. The result is an average of five runs.
3) Stress testing for the Free Web Hosting code. Basically a PHP script that doesn't return any response body, but it computes the same mathematical operations as the 1st test.
4) Stress testing for a small Kohana Framework application that reads all the posts from a WordPress installation, filters them through the HTML Purifier library for XSS issues, then displays the text.

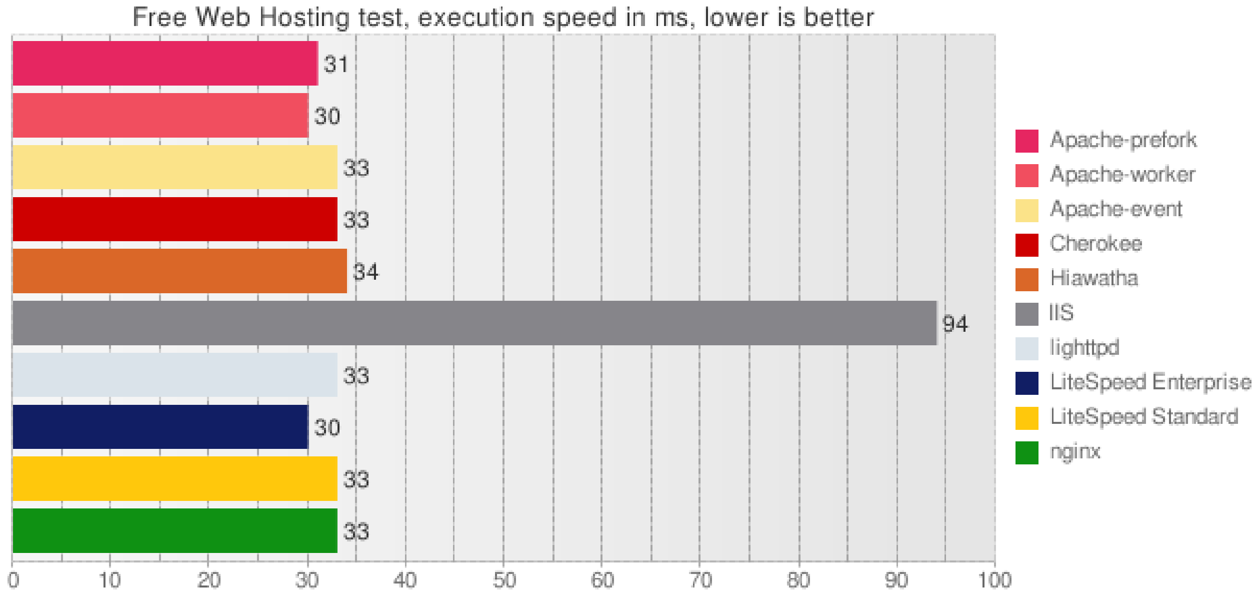
Application Tests

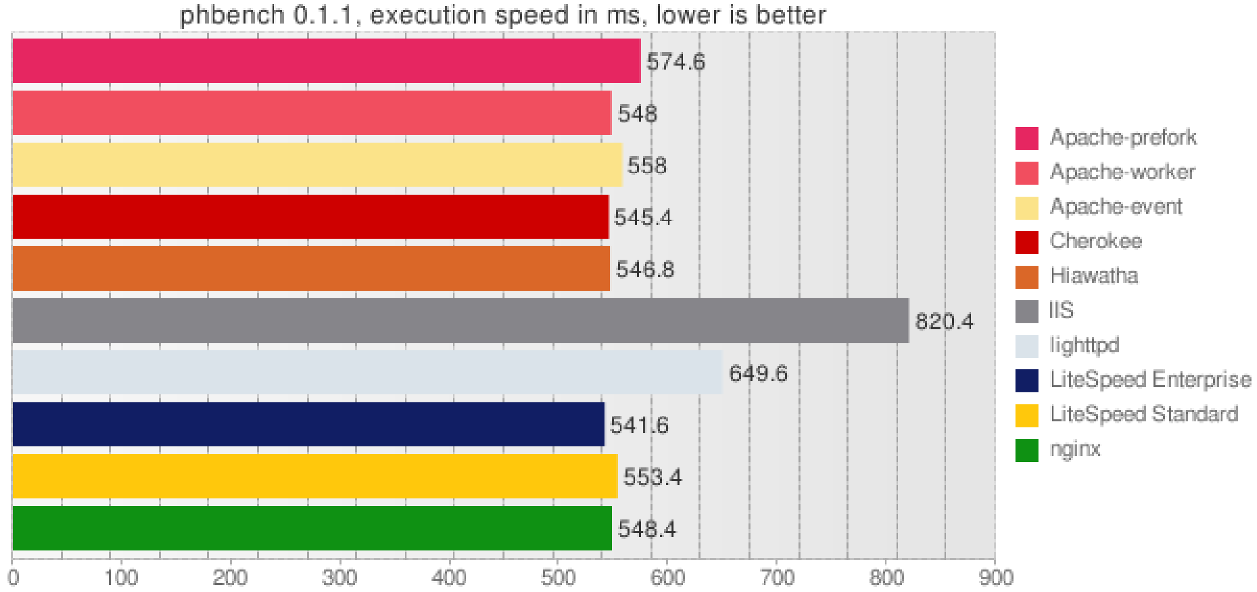
Drupal, Joomla and WordPress were used as test applications. There were three distinct tests for each application:

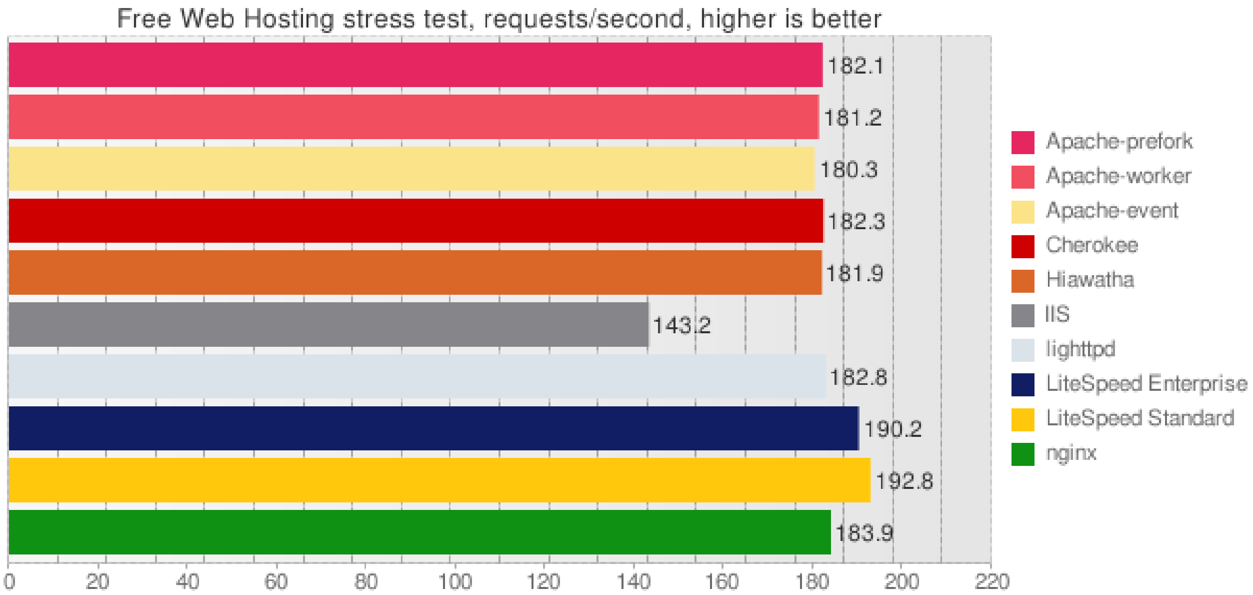
1) Home page stress test
2) Page/Article stress test
3) Static content stress test

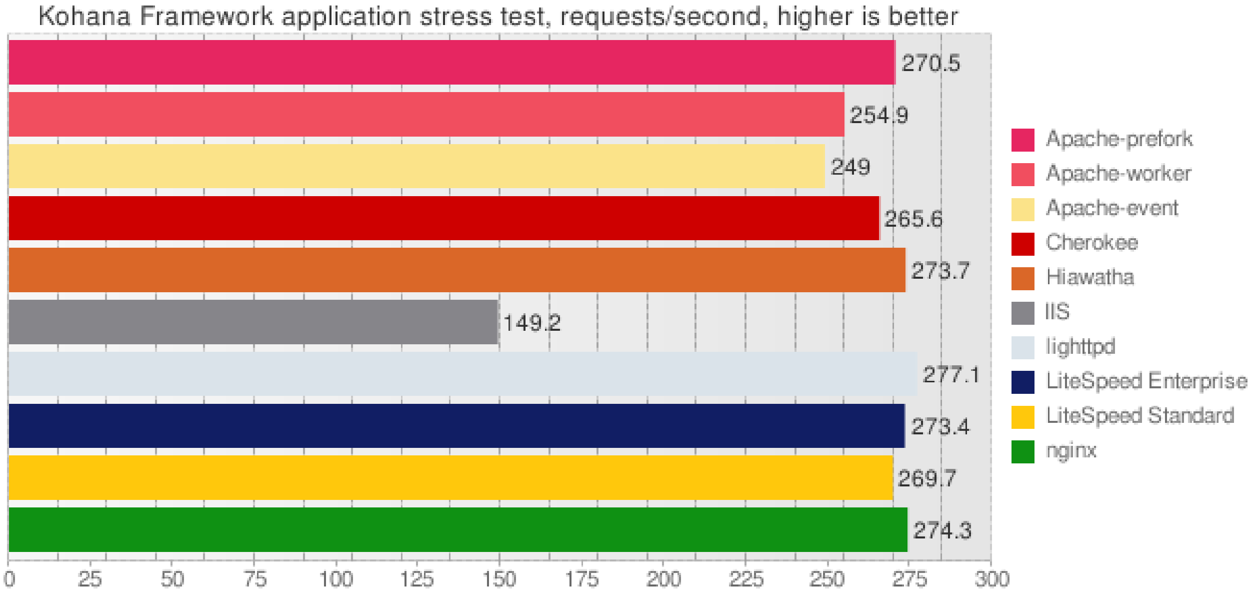
The results

Synthetic Tests

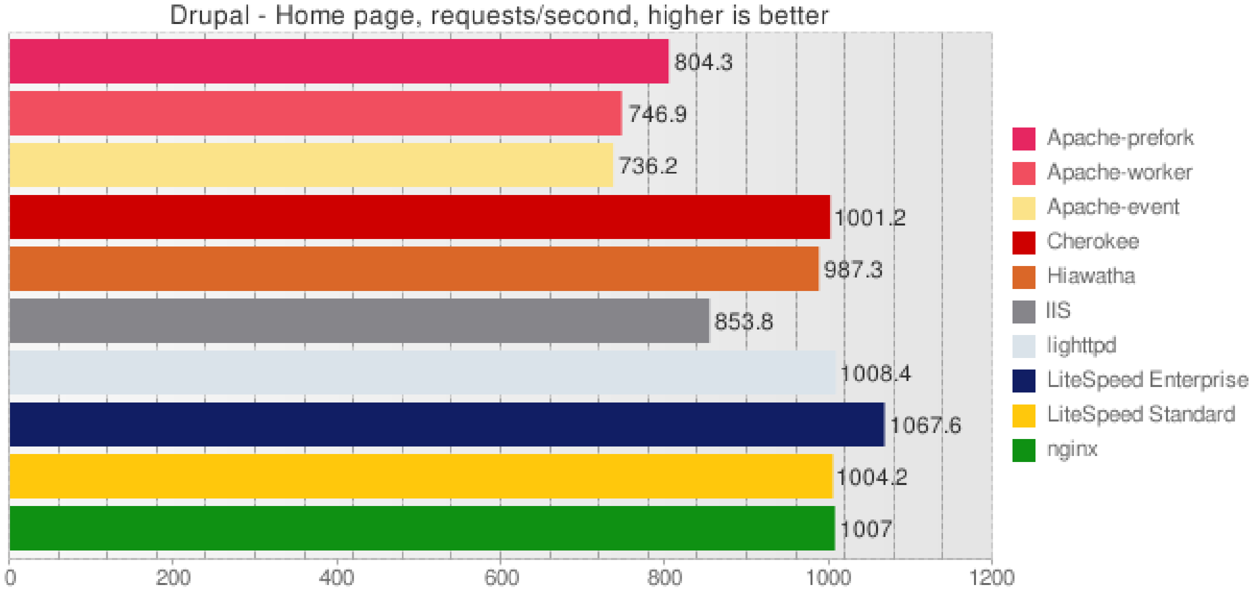

Free Web Hosting test, execution speed in ms, lower is better

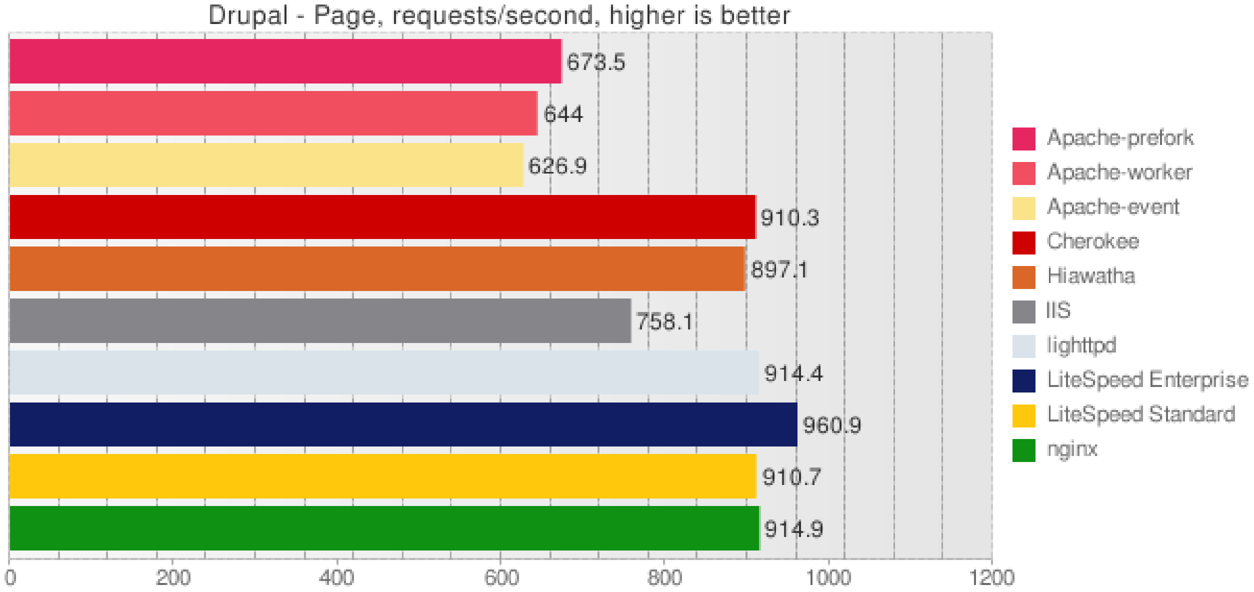

phbench 0.1.1, execution speed in ms, lower is better

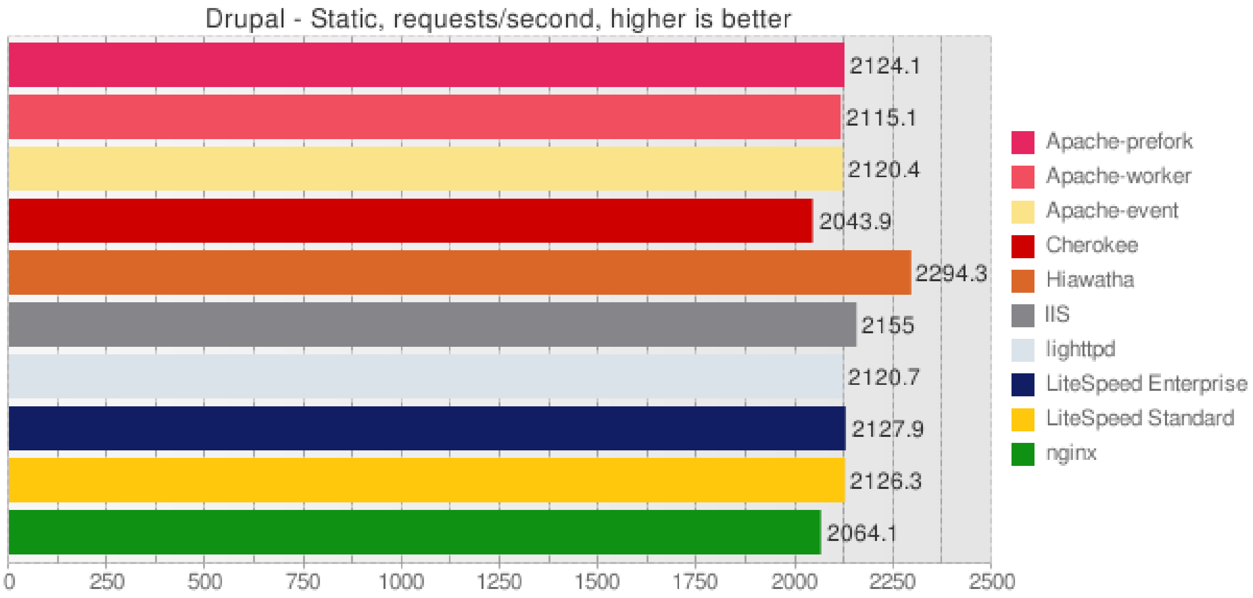

Free Web Hosting stress test, requests/second, higher is better

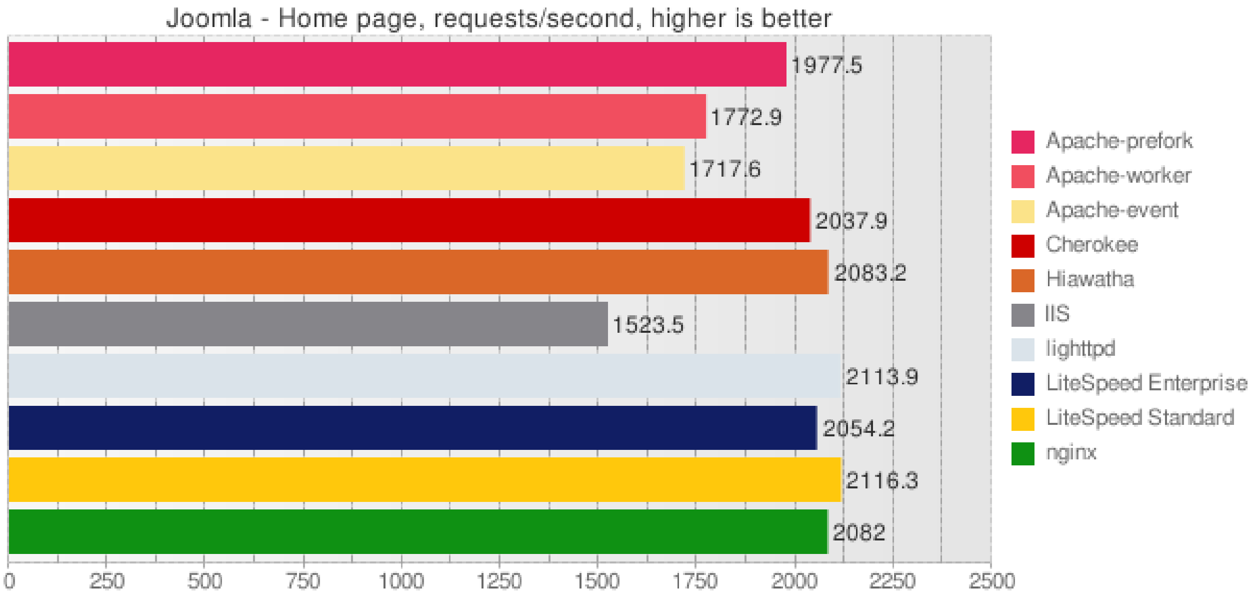

Kohana Framework application stress test, requests/second, higher is better

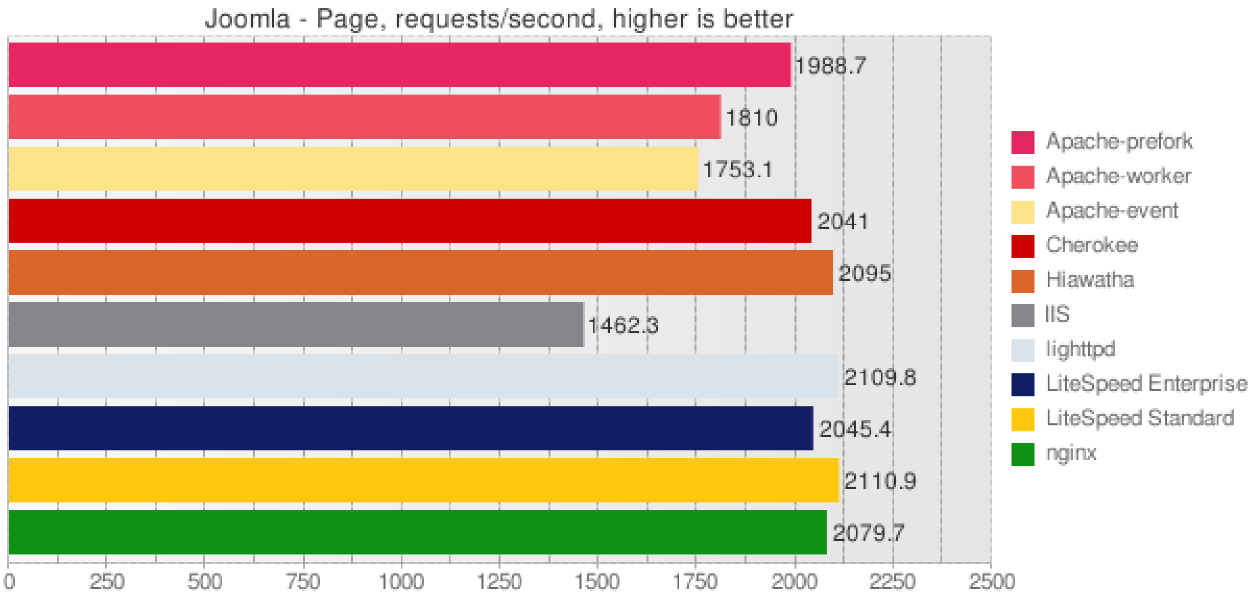
Application Tests

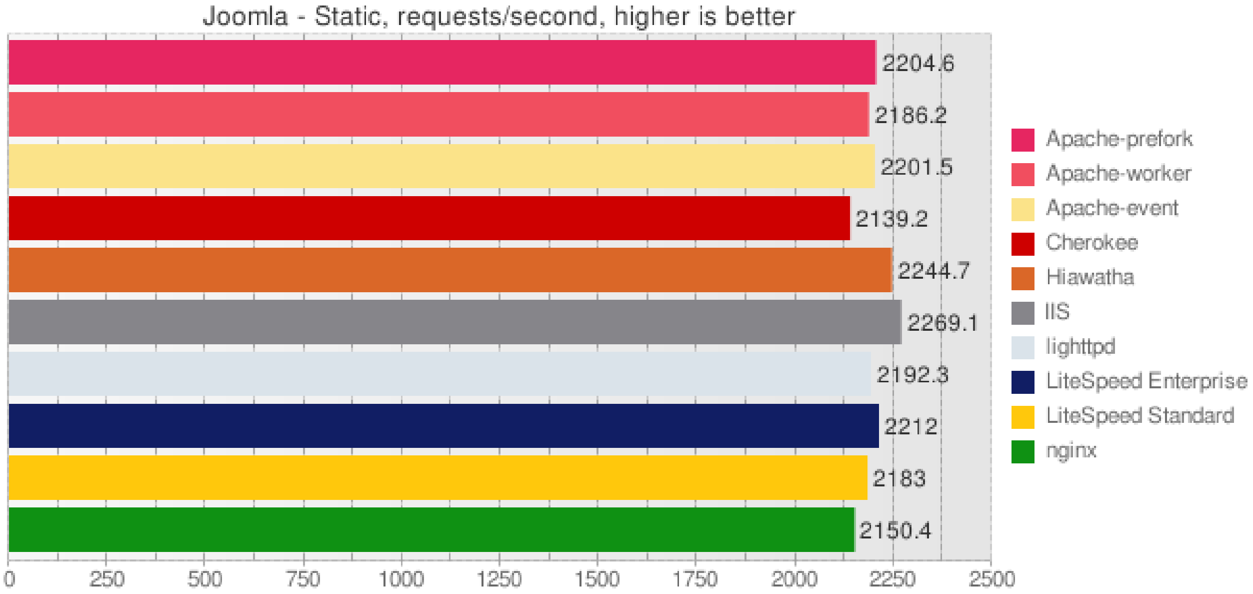


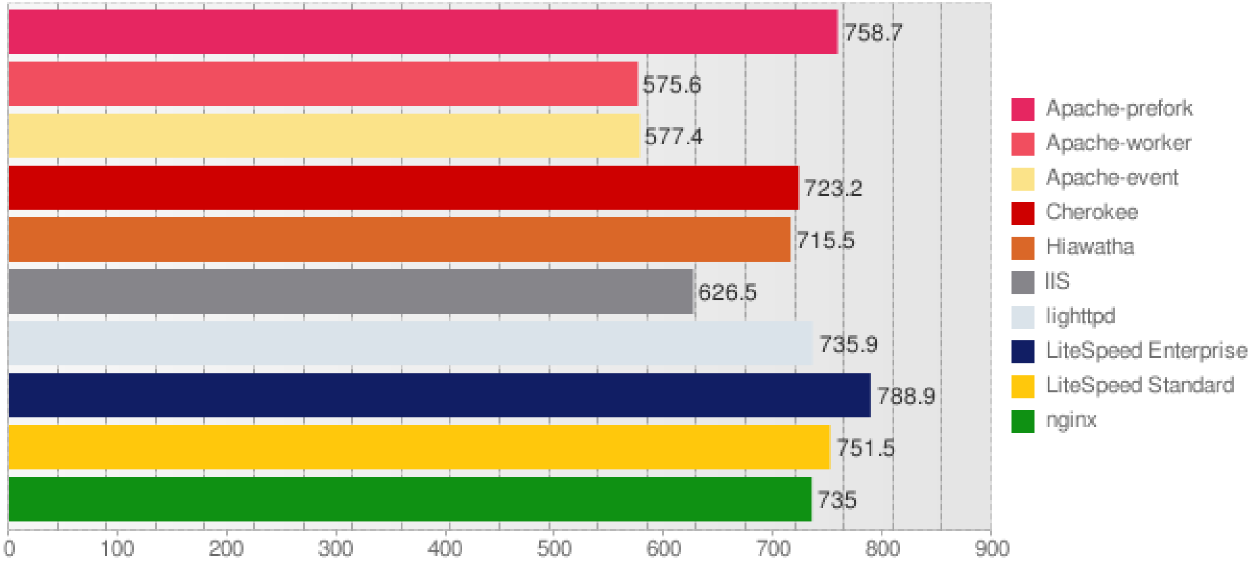



Drupal - Static, requests/second, higher is better

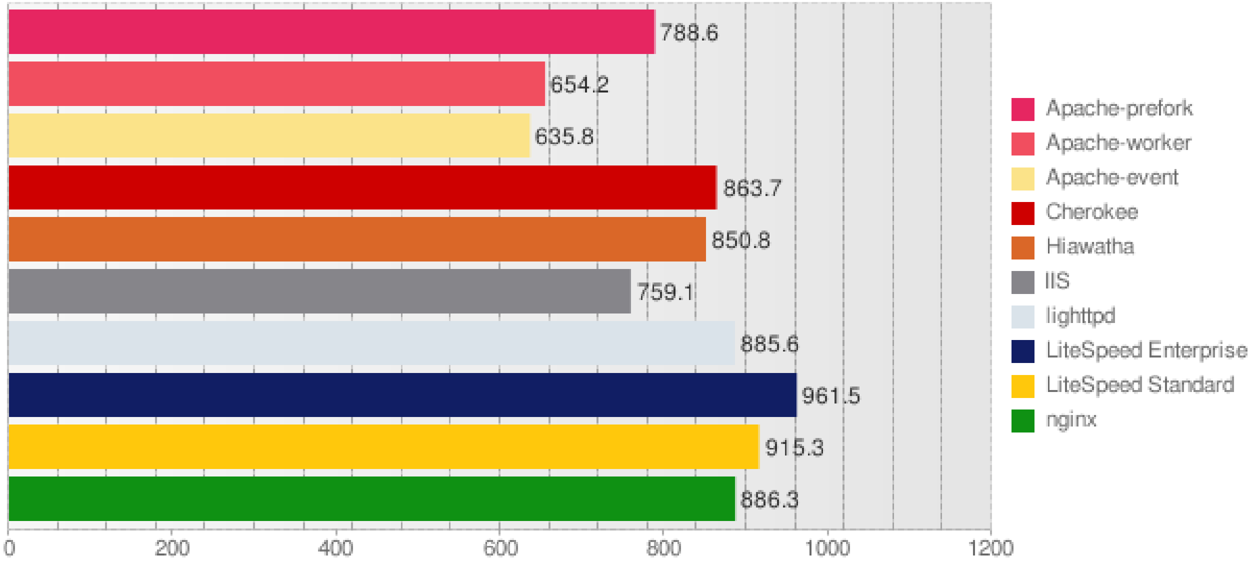

Joomla - Home page, requests/second, higher is better


Joomla - Page, requests/second, higher is better
1988.7
1810
1753.1
2041
2095
1462.3
2109.8
2045.4
2110.9
2079.7
0
250
500
750
1000
1250
1500
1750
2000
2250
2500
Apache-prefork
Apache-worker
Apache-event
Cherokee
Hiawatha
IIS
lighttpd
LiteSpeed Enterprise
LiteSpeed Standard
nginx

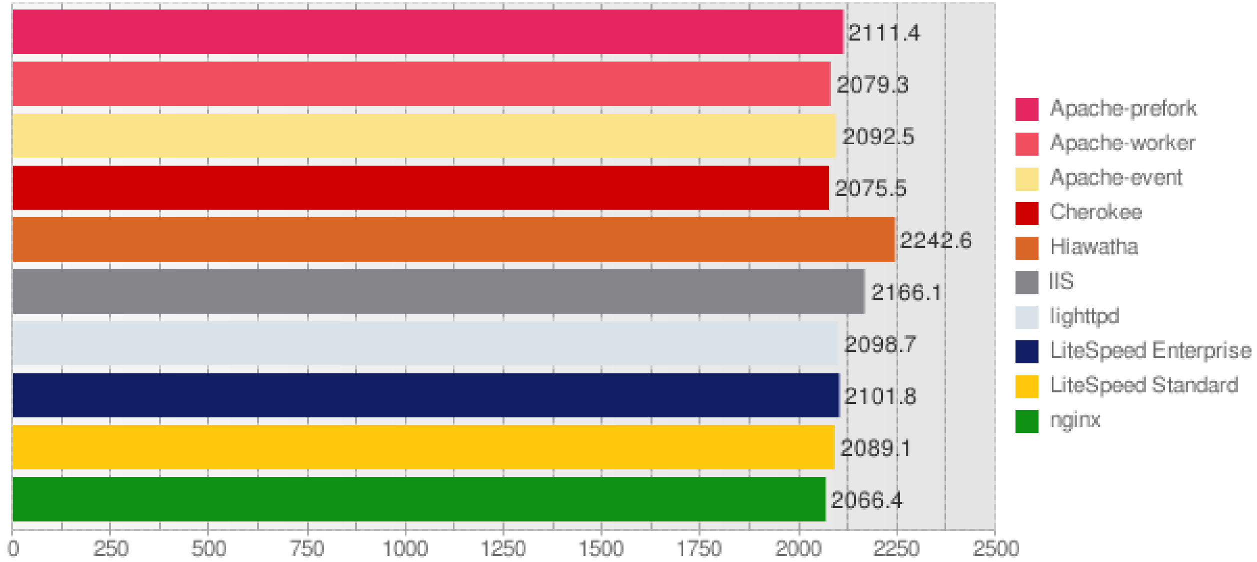

Joomla - Static, requests/second, higher is better
2204.6
2186.2
2201.5
2139.2
2244.7
2269.1
2192.3
2212
2183
2150.4
0
250
500
750
1000
1250
1500
1750
2000
2250
2500
Apache-prefork
Apache-worker
Apache-event
Cherokee
Hiawatha
IIS
lighttpd
LiteSpeed Enterprise
LiteSpeed Standard
nginx



WordPress - Home page, requests/second, higher is better



WordPress - Article, requests/second, higher is better



WordPress - Static, requests/second, higher is better

Document source: PHP_web_serving_study.pdf