COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

PERFORMANCE AND SPEED OPTIMIZATIONS OF WORDPRESS-BASED WEB APPLICATION AND ITS UNDERLYING SERVER STACK

Bachelor's thesis

2015 **Rastislav Lamoš**

COMENIUS UNIVERSITY IN BRATISLAVA
FACULTY OF MATHEMATICS, PHYSICS AND INFORMATICS

PERFORMANCE AND SPEED OPTIMIZATIONS OF WORDPRESS-BASED WEB APPLICATION AND ITS UNDERLYING SERVER STACK

Bachelor's thesis

Study program:	Applied computer science
Supervisor:	Mgr. Kamil Maráz.

Bratislava 2015 **Rastislav Lamoš**

Declaration of authorship

I hereby declare and confirm that this thesis is entirely the result of my own work except where otherwise indicated.

Bratislava, 31. May 2015 ....

Acknowledgement

To be written...

Abstract

In our work, we will be dealing with the performance and speed optimisations of an underlying server and a WordPress-based web application running on it. After a thoughtful study of our work, an ordinary web programmer or administrator will be able to install and configure his web server and refactor the source code of his PHP (WordPress) web application, without having to research what to modify or avoid from other sources, thus saving his time and resources. Product of our work will be a highly optimised server with proper caching and a WordPress application developed to be as efficient as possible.

Keywords: *page loading speed, WordPress, optimisation, PHP, server*

Abstrakt

V našej práci sa budeme zaoberať optimalizáciou výkonu a rýchlosti serveru a web aplikácií založenej na systéme WordPress, ktorá na tomto serveri beží. Po pozornom prečítaní a naštudovaní našej práce si bude bežný web programátor či administrátor vedieť nainštalovať a nakonfigurovať svoj web server a refaktorovať zdrojový kód svojej PHP (WordPress) web aplikácie bez toho, aby musel z iných zdrojov zisťovať čo zmeniť a čomu sa vyvarovať, čím sa ušetrí jeho čas a zdroje. Výsledkom našej práce bude vysoko optimalizovaný server s vhodným caching a WordPress aplikácia vyvinutá tak, aby bola čo najefektívnejšia.

Kľúčové slová: *rýchlosť načítavania stránky, WordPress, optimalizácie, PHP, server*

Glossary

caching test 15

Contents

List of Figures

List of Tables

1. Introduction

1.1 Work Structure

Our work is divided into three main parts.

Firstly, we delve into the problematics, describing what is WordPress and why its performance matters. We present past experiments and their results, pointing out their achievements as well as their deficiencies. We also specify and describe all the necessary tools, software and skills needed to understand and be able to reproduce the work in this part.

Secondly, we present our solutions to the problem(s) with thorough explanations. The author of the work keeps a technical blog [7] where a step by step guide to achieve the results shown in this document is located. We comment and describe only the most important sections of the guide in this document.

Lastly, we recapitulate and select the most efficient and optimized configurations for both server software and WordPress. A command line application (script), which installs all the required software and configures it is also included.

1.2 Problem Definition

Website loading and rendering speed has a critical impact on page abandonment rate among its visitors. According to a Google experiment, web page loading time increased only by **half of a second**, had a 20% drop in its visitor's traffic [5]. On figure 1.1 [16], we can see a chart of page abandonment relative to a web page loading time in seconds. By the time a ten-seconds loading web page is finished rendering to the users, more than 35% of them will close the page, never seeing what is on it.

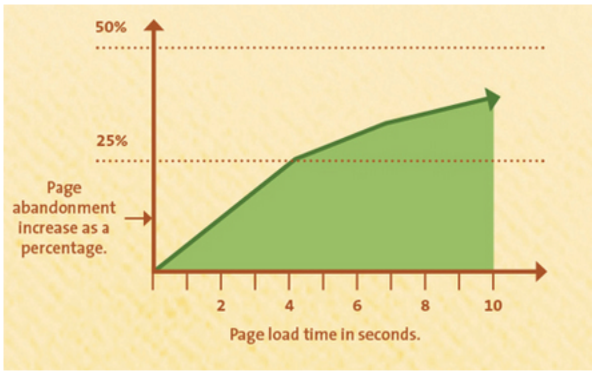
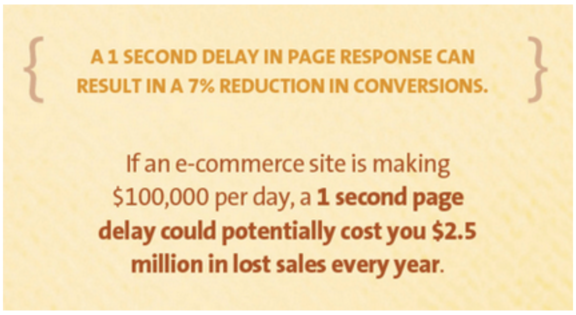


Figure 1.1: Slower page response time results in an increase in page abandonment

If we want to highlight the seriousness of this issue, imagine a situation in which an e-commerce site owner is making $100,000 per day. One second page delay could potentially cost him or her $2.5 million in lost sales every year [16]. See figure 1.2 for more detail.



A 1 SECOND DELAY IN PAGE RESPONSE CAN RESULT IN A 7% REDUCTION IN CONVERSIONS.

If an e-commerce site is making $100,000 per day, a **1 second page delay could potentially cost you $2.5 million in lost sales every year.**

Figure 1.2: What can a one-second page delay cause to your e-commerce site?

What is more, Google incorporated site speed in search rankings in 2010 [1], meaning that the slower a website loading is, the lower rank in Google search results it will receive. The author of this work believes he has shown the reader enough evidence that optimizing the performance and speed of a website is crucial for its success, especially in today's fast-paced world.

1.3 About WordPress

1.3.1 General Introduction

To cite WordPress.org, "WordPress is web software you can use to create a beautiful website or blog."[15] Simply put, WordPress is a very powerful, open source web publishing software, content management system and a web platform for building rich web applications. People and companies are using WordPress for various purposes and reasons, most notably for their blogs, websites, e-commerce solutions and large portals. At the time of writing (January 2015), it is estimated that about 23% [13] of all websites, whose content management system is known, is being run on WordPress. This number is quite astonishing because it means that visiting five random websites, one of them will be a WordPress-powered one.

WordPress, in comparison to other web softwares and frameworks, is a full-featured, stand alone web publishing software and content management system. At its core, it consists of a request-response routing subsystem, classes for managing database and content handling, security features and others. WordPress was initially started as an open source hobby project of Matt Mullenweg in 2003 [4]. Since then, web programmers from all around the world have contributed to it, making it robust, secure and fast. However, as there are several bottlenecks in the system, the author of this work, himself a web developer, decided to analyze and improve them. His findings and results are summarized in this thesis.

1.3.2 WordPress-Powered Website

In order to customize the look and feel of user's instance of the website, there exists a mechanism called the WordPress **Theme**. A WordPress theme is simply a collection of scripts, stylesheets and images which get combined, processed and the generated content is sent back to the client. A user can install any theme which is compatible with his or her WordPress version. There is no central body governing the quality and correctness of a theme. As the themes developers are free to construct them almost arbitrarily, numerous security and performance flaws and issues can occur. What is more, core WordPress developers introduced a handy feature called the **Plugin**. WordPress plugin is a pluggable piece of software which enhances the basic WordPress functionality, thus enabling its users to heavily modify their WordPress-based web applications and sites. Plugins are also open source and without any quality guarantees, thus they are predated with the same problems as the aforementioned themes.

Moreover, if the user has a high-traffic website, his web delivering server is not able to keep up with all the requests resulting in a slow, unresponsive website. The reader might assume that the problems would be solved by increasing plugins quality. While the previous statement is true, it is usually not viable, mainly due to a reason that **work of an experienced web developer is relatively expensive**. In many cases, upgrading the server and/or getting additional ones is the preferred way. In our work, we are concerned with optimizing the server software first and only then examining the best practices, tips and tricks of a plugin or a theme development.

1.4 General Measures and Techniques for Website Speed Improvement

Before we can delve into the actual solutions of our problems, we need to list and describe several general techniques and measures we can use to decrease not only the website loading times, but also the server resources usage and load.

1.4.1 Web-Serving Software Efficiency

Using the most performant and the least resource-hungry web-serving and accompanying software is usually the most effective way of decreasing web page loading speed. Some studies have found that doing even small changes to configuration files can make quite a difference. In our work, we are making comparisons between the two most popular web-serving softwares, namely Apache HTTP and Nginx as well as comparisons between different PHP interpreters. Jump to section **??** for more details and statistics.

1.4.2 Server-Side Caching

Let us divide server-side caching into three main categories:

1. **Caching intermediate PHP code**
2. **Caching the output of PHP interpreter (so called "page cache")**
3. **Static resources (files) caching**
4. **Database caching**

Caching intermediate PHP code

PHP source code is interpreted on each run, thus the PHP interpreter has to read and collect all required files each time a new request is sent to our WordPress-based website. With PHP opcode caching, an intermediate source code is generated on the first run. It is stored temporarily and when a new request comes, instead of going through all the files, PHP loads cached opcode and executes it. We can observe the process from the figure 1.3 [8].

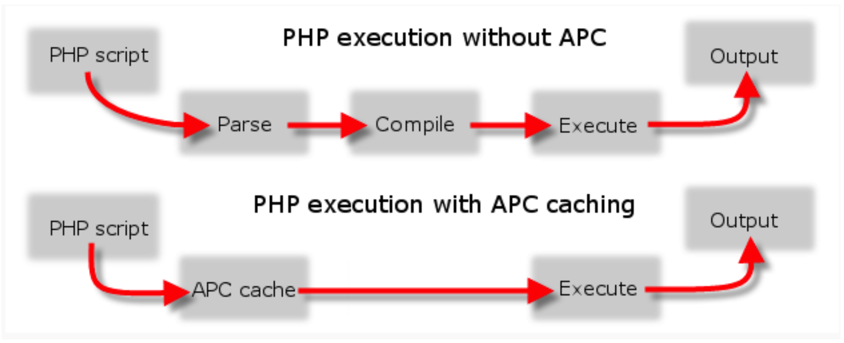


Figure 1.3: PHP execution diagram with and without APC opcode cache.

Caching the output of PHP interpreter (so called "page cache")

While a website based on WordPress is dynamic, in some cases it is possible to store the constructed web page for subsequent usage. This process is usually called *page caching* and it is suitable for websites with static blocks of content. However, problems arise when we have to handle special cases such as when a visitor is logged into our website. In that case we are not able to cache the request because other visitors would see the logged-in visitor's cached content instead of general one.

Static resources (files) caching

If our website consists of a large number of (smaller) files such as JavaScript scripts, CSS stylesheets, images and others, caching these resources is an idea worth mentioning. They are constructed and loaded on the initial request, stored in a local web server cache and retrieved from the cache on subsequent requests.

The largest disadvantage of static file caching is usually a drop in free memory available to different needs of the web server, especially when files are cached in the RAM memory.

Database caching

When multiple requests to our web server trigger the same database queries, it is reasonable to store the retrieved data for later use. When a subsequent request is made, querying into the database is omitted, thus preserving valuable server resources and outputting the resulting web page faster.

1.4.3 Client-Side Caching

What makes client-side caching different from the server-side caching is that data is stored locally in the visitor's browser, not on the server to which the requests are made. [9]

The whole hypertext document with all its resources including JavaScript scripts and CSS stylesheets can get cached in local storage and loaded and executed from it on the consecutive requests. Naturally, the most notable advantage of client-side caching is the fact that the user's browser does not need to download the locally cached resources.

On the other hand, a mechanism handling resource changes has to be implemented on the server-side. If it is not done properly, some of the visitors might see outdated content due to the reason that the visitor's browser has not been instructed to revalidate its cache.

1.4.4 JavaScript and CSS Resources Minification and Combination

At the time of writing (early 2015), a large number of WordPress themes and plugins contain tens or more JavaScript scripts and CSS styles, especially the more professional ones. It is caused by the fact that users like having amazingly-looking, feature-rich websites. The problem with this fact is twofold:

- **There is a limit on the number of concurrent downloads of resources from the same domain. Both Internet Explorer 8 and Google Chrome allow six concurrent downloads, while Firefox eight [17].**
- **Each request carry an overhead of constructing a packet, sending it to the web server and waiting for the reply.**

There are two well-known methods of solving this issue:

1. **Resources combination and/or**
2. **resources minification.**

Resources combination

Resources combination is a process in which resources on the requested web page are collected into (preferably) single file which is then sent back to the visitor's browser. There exist two main downsides of this approach. The first is that collecting the resources consumes additional CPU cycles on the server-side. Another disadvantage happens when the owner of the website modifies source code of any of the grouped resource. The whole group has to be gathered together again, including revalidating browser cache if present.

Resources minification

JavaScript scripts and CSS stylesheets can be minified before being inserted into the response body. Minification is a procedure in which parts of a resource source code are reduced or completely removed, thus reducing its size and length. Advanced minification tools are capable of refactoring the source code in a manner that it becomes even more compact.

1.4.5 Compression of Images, HTML and Other Resources

Image compression

Images, particularly those produced by the JPEG lossy compression mechanism, can be compressed further, thus reducing their size while keeping tolerable quality of picture details. At the time of writing this work, the cost of satisfying web service [18] for image compression is zero.

Hypertext documents and other text-based assets compression

Before the data is sent back in a response to visitor's request, the size of some of them can be reduced further with a process called data compression [3]. Most modern browsers [10] support *GZIP* compression of textual data. One of the downsides of performing this process is that additional CPU cycles on both server and client sides are expended in order to compress and uncompress the data.

1.4.6 Optimizing Images — Image Sprites

Images used for our website's user interface as well as other images can become more optimized for use in the web environment. A mechanism called image spriting [12] is a procedure during which multiple images, sometimes even all of them, get collected and combined into a single larger image — sprite. When a web page is requested, instead of loading tens of user interface icons and images, only this single one is sent back to the user, thus saving additional HTTP requests and bandwidth. When rendering the user interface, icons and images are taken from that single image.

1.4.7 CDN and Resource Distribution

In some cases our website is accessed from many different countries, even continents. If we have web servers located only locally, additional milliseconds start to congregate in these situations. To solve this problem, web administrators usually distribute the website's data across multiple servers positioned in multiple places around the world. CDN services greatly reduce amount of work needed to accomplish the goal by doing it automatically for us. Another quite useful concept is called **load balancing**. It is a method of distributing requests to a website across multiple web servers decreasing the overall load on each machine.

1.5 Previous Similar Studies and Work

In this section, we will be analyzing and discussing similar studies done by other web developers and programmers. Before we look at their results, we need to have a basic understanding of the web serving software they were comparing.

Apache HTTP, also called Apache, is a free and open source world's most widely used (57.9%[14] of all websites whose web server we know) web server software. **Nginx**, released nearly 10 years later than Apache, was designed with a high concurrency, high performance and low memory usage in mind, specifically to solve the C10K problem [6].

Both **PHP** and **HHVM** are PHP script interpreters. HipHop Virtual Machine (HHVM), released by Facebook in 2011, was developed to increase the performance of PHP script execution on the Facebook site. Both of them are open source and free to use.

1.5.1 Using Nginx, Apache, APC and Varnish in Different Scenarios — Garron.me

Guillermo Garron, the author of the work [2], has performed several comparisons between the most popular web-serving and caching softwares. He used a relatively weak web server with highly limited computing power — 512MB RAM, shared CPU and shared disk. However, the result of his experiment is clear — caching the output of the PHP interpreter has a considerable impact on loading times of WordPress-powered web page.

From the figure 1.4, we observe that if the number of simultaneous visitors exceeds ten, response times of the web server start to dramatically decrease in a linear fashion. On the other hand, response times start to worsen only after thirty concurrent visitors, as we can see from the figure 1.5.

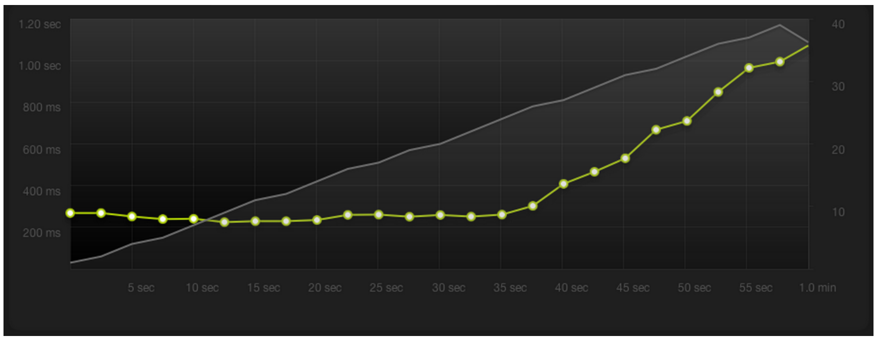


Figure 1.4: Apache HTTP + PHP, no opcode caching

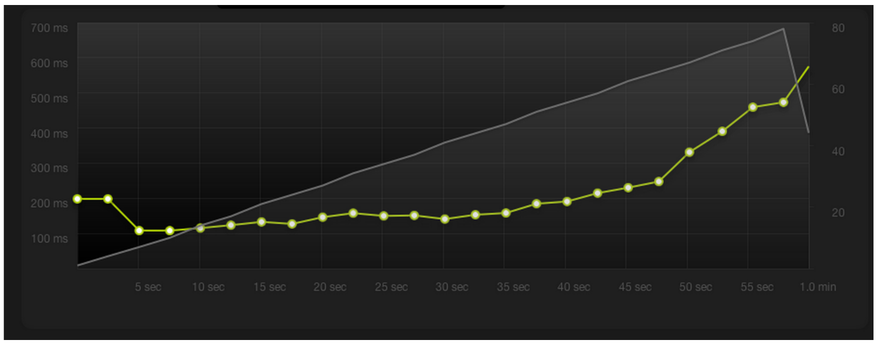


Figure 1.5: Apache HTTP + PHP, APC opcode caching

Although using Nginx instead of Apache HTTP yields additional performance gains, they are less notable. Nginx strengths are demonstrated when a website is composed of many resources. However, Garron used a standard WordPress installation, with no extra plugins or complex custom themes.

1.5.2 WordPress on HHVM vs WordPress on PHP-FPM — WPengine.com

WPengine.com is a company specializing in offering WordPress web hosting services. They have introduced a new hosting plan which differentiates itself from others by using HHVM PHP interpreter instead of the standard PHP. Before releasing the hosting plan, a study in which the performance of HHVM vs PHP was compared was undertaken. The study concluded with a fact that HHVM increased the speed of their servers by 600% [11]. This fact is displayed on figure 1.6.

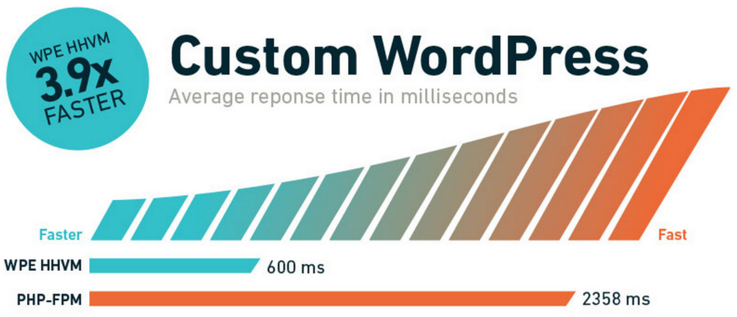


Figure 1.6: WordPress on PHP vs WordPress on HHVM response times

In the study, they found out that HHVM is not 100% stable yet and occasionally stops working. Their solution to this problem was to redirect the incoming requests to a fallback PHP interpreter while HHVM gets restarted and functioning.

1.5.3 WordPress HHVM vs PHP — xyu.io

Xiao Yu, a web developer, benchmarked [19] WordPress running on PHP vs WordPress running on HHVM. His findings show a similar pattern to the findings of WPengine.com in section 1.5.2. Outcome of his experimentation can be observed in table 1.1.

	Response Time	*Ok Responses*	*Errors / Timeouts*
Anon PHP	4.091	8,939	0.94%
Anon HHVM	2.122	18,308	0.00%
Change	48.1%	**2.05X**	
Auth PHP	20.688	457	74.17%
Auth HHVM	14.359	1,242	43.45%
Change	30.6%	**2.72X**	

Table 1.1: WordPress on HHVM vs WordPress on PHP — xyu.io

”In the numbers above anonymous requests represents hits to various pages without a WordPress logged in cookie which are eligible for Batcache caching whereas authorized requests are hits to the same pages with a login cookie thus bypassing page caching.”[19]

2. Configuring testing environment

2.1 Server parameters

2.2 Ansible automation

3. Benchmarking server software

3.1 Apache + mod_php versus Nginx + PHP-FPM

3.2 Nginx + PHP-FPM versus Nginx + HHVM

4. Caching

4.1 Database and objects caching

4.2 Page caching

4.3 Browser caching

5. Client-side performance optimizations

5.1 Measurement tools

5.2 Assets minification and concatenation

5.3 Assets compression

5.4 Content delivery network

6. Source code performance optimizations

6.1 WordPress architecture in brief

6.2 Profiling web application with xhprof

6.3 Offloading code execution to client-side

7. Concluding remarks

8. Future work

8.1 Load balancing

8.2 Better plugin and theme architecture

Bibliography

[1] CUTTS, Matt, *Google incorporating site speed in search rankings*, Feb. 2015, URL: `https://www.mattcutts.com/blog/site-speed/`.

[2] GARRON, Guillermo, *Wordpress performance comparison: using nginx, apache, apc and varnish in different scenarios*, Feb. 2015, URL: `http://www.garron.me/en/linux/apache-vs-nginx-php-fpm-varnish-apc-wordpress-performance.html`.

[3] GRIGORIK, Ilya, *Optimizing encoding and transfer size of text-based assets*, Feb. 2015, URL: `https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer`.

[4] *History « wordpress codex*, Feb. 2015, URL: `http://codex.wordpress.org/History`.

[5] JACOB, Sherice, *Speed is a killer – why decreasing page load time can drastically increase conversions*, Feb. 2015, URL: `https://blog.kissmetrics.com/speed-is-a-killer/`.

[6] KEGEL, *The c10k problem*, 2015, URL: `http://www.kegel.com/c10k.html`.

[7] LAMOS, Rastislav, *Lamosty.com blog — bachelor's thesis*, Feb. 2015, URL: `https://lamosty.com/category/bachelor-thesis/`.

[8] NICHOLSON, Jacob, *Speed up php with apc - alternative php cache*, Feb. 2015, URL: `http://www.inmotionhosting.com/support/website/what-is/speed-up-php-with-apc`.

[9] NOTTINGHAM, Mark, *Caching tutorial for web authors and webmasters*, Feb. 2015, URL: `https://www.mnot.net/cache_docs/`.

[10] OEZI, *Which browsers handle ‘content-encoding: gzip‘ and which of them has any special requirements on encodinq quality?*, Feb. 2015, URL: `http://webmasters.stackexchange.com/questions/22217/which-browsers-handle-content-encoding-gzip-and-which-of-them-has-any-special`.

[11] PUIG, Tomas, *Announcing wp engine’s high availability hhvm platform: mercury – labs alpha*, Feb. 2015, URL: `http://wpengine.com/2014/11/19/hhvm-project-mercury/`.

[12] ROBERTS, Harry, *Front-end performance for web designers and front-end developers*, Feb. 2015, URL: `https://www.mnot.net/cache_docs/`.

[13] *Usage statistics and market share of wordpress for websites*, Jan. 2015, URL: `http://w3techs.com/technologies/details/cm-wordpress/all/all`.

[14] W3TECHS, *Usage statistics and market share of apache for websites*, 2015, URL: `http://w3techs.com/technologies/details/ws-apache/all/all`.

[15] *Wordpress > blog tool, publishing platform, and cms*, Feb. 2015, URL: `https://wordpress.org/`.

[16] WORK, Sean, *How loading time affects your bottom line*, Feb. 2015, URL: `https://blog.kissmetrics.com/loading-time/`.

[17] WORTHAM, Steve, *Get number of concurrent requests by browser*, Feb. 2015, URL: `http://stackoverflow.com/questions/7456325/get-number-of-concurrent-requests-by-browser`.

[18] *Yahoo! smush.it™*, Feb. 2015, URL: `http://www.smushit.com/ysmush.it/`.

[19] YU, Xiao, *Wordpress performance with hhvm*, Feb. 2015, URL: `http://www.xyu.io/2014/09/wordpress-performance-with-hhvm/`.