Performance Gains from Web Performance Optimization
Case Including the Optimization of Webpage Resources in a Comprehensive Way

Juha Vihervaara[1], Pekka Loula[1] and Tommi Tuominen[2]
[1]*Pori Campus, Tampere University of Technology, Pohjoisranta 11, 28100 Pori, Finland*
[2]*Foredata Oy, Kässäläntie 30, 38200 Sastamala, Finland*

Keywords: Website, Performance, Optimization.

Abstract: Web performance optimization tries to minimize the time in which web pages are downloaded and displayed on the web browser. It also means that the sizes of website resources are usually minimized. By optimizing their websites, organizations can verify the quality of response times on their websites. This increases visitor loyalty and user satisfaction. A fast website is also important for search engine optimization. Minimized resources also cut the energy consumption of the Internet. In spite of the importance of optimization, there has not been so much research work to find out how much the comprehensive optimization of a website can reduce load times and the sizes of web resources. This study presents the results related to an optimization work where all the resources of the website were optimized. The results obtained were very significant. The download size of the front page was reduced by a total of about 80 percent and the downloading time about 60 percent. The server can now handle more than three times as much concurrent users as earlier.

1 INTRODUCTION

Web performance optimization (WPO), or website optimization, tries to minimize the time in which web pages are downloaded and displayed on the user's web browser. These are some reasons which make it extremely important that organizations verify the quality of response times on their websites. Galletta (2004) and Fabian (2010) present that the fast download speed of a website increases visitor loyalty and user satisfaction. These studies have also shown that delays have a negative impact to a website. For example, milliseconds of increase in load time can significantly reduce page views. A fast website is also important for search engine optimization. Google has included site speed as part of its search ranking algorithm. It means that the fast site gets a higher ranking in its search.

WPO also means that the sizes of website resources are usually minimized. This leads to less data travelling across the Internet. This cuts the energy consumption of the Internet. So, WPO promotes the Green Internet (Gupta 2003, Bianzino 2011). Minimized resources are also useful for users with slow internet connections and those on mobile devices.

In spite of the importance of WPO, there has not been so much research work to find out how much the comprehensive optimization of a website can reduce load times and the sizes of web resources. There have been few attempts to understand how different aspects of website complexity impact the time to load web pages (Butkiewicz 2011). Instead, the published optimization results often concern only the specific web resource (Abdullah 2010, Ferragina 2010, Qian 2012, Spiesser 2004). This study presents the results related to an optimization work where all the resources of the website will be optimized.

This project represents some kind of an extreme case. In the past, not enough attention has been paid to the optimization of this website. Originally, even inefficient solutions were introduced partly because of the novice software developers (Begel 2008). Now four years later, the same developers will carry out this optimization work. Because these developers are now more experienced, and because WPO methods and tools have evolved since the original work, it is expected that optimization results will be significant.

Vihervaara, J., Loula, P. and Tuominen, T.
Performance Gains from Web Performance Optimization - Case Including the Optimization of Webpage Resources in a Comprehensive Way.
In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST 2016) - Volume 1*, pages 188-193
ISBN: 978-989-758-186-1

2 BACKROUND

Over the years, the web has evolved from simple text content from one server to a complex ecosystem with different types of content from multiple servers under different administrative domains (Butkiewicz 2011). Rendering a single web page involves fetching several objects with varying file types. A typical web page can include HTML, CSS, and image files. JavaScript files can be used to provide client-side functionality, such as functions, plug-ins, applets, and so forth. These files can be called the resources of a web page.

These resources can be optimized in many ways (Hongkiat 2015, Rajeev 2015). This optimization generally involves editing the resources to optimize scripts, HTML or CSS codes for faster loading. The optimization also tries to reduce the number of resources for reducing the number of required HTTP request and response messages. Also client side caching can be used for performance optimization (Ali 2011).

The performance of the website can be evaluated in many ways (Soininen 2012). Performance can be defined as the time within a certain operation is executed. A common meter regarding to the performance of a web page is the time it takes to load a page. Page load speeds have been typically measured as the time it takes for two events to occur on the page. These events are DOMContentLoaded and Load/Onload.

The DOMContentLoaded event fires after the HTML code has been fully retrieved from the server and the complete DOM tree has been created. This event does not necessarily require that any images or style sheets have been loaded. This means that the render tree can now be built. Instead, the Load event fires after the entire page has been fully downloaded, processed, and rendered including all images, stylesheets, and scripts. After the Load event, some resources can still be loaded in an asynchronous way. So, the unofficial term of Totally Loaded can be used for a time unit, when all the resources are totally loaded. Related to the load times, Butkiewicz (2011) says that rather than the total number of bytes fetched to render a website, the number of objects fetched is the most dominant indicator of client-perceived load times.

Performance can also be defined as the capability of executing a number of operations within a unit of time. With websites, this means that how many concurrent users the website can serve. In general, the performance of the system has to be good enough during the peak time to satisfy user needs. To figure out the moment of the top usage, the log files of the web server have to be analysed.

3 OPTIMIZATION WORK

The optimized website is ForeAmmatti. It is a web service which offers information in respect of the labour market of Finland. Data content of the ForeAmmatti information system had been significantly increased during the last few years. The dynamic production of the content in the HTML pages had always been done with the fastest and easiest way because of the very fast development work. The new features and contents had been added with speed. Therefore, the performance of the solutions had not received major consideration. There was a clear need for optimization because most of HTML pages and their resources were enormously big considering both the file sizes and the amount of characters.

After the optimization, some JavaScript files are still loaded in a synchronous way. Otherwise, the optimization work was done almost in a perfect way. So, the optimized result achieved was better than only using the optimization tools offered by browser developers. The next subsections presents the most essential points related to this optimization work.

3.1 Occupation Selectors

The most beneficial modification was the redesign of the occupation selectors. The text content of those is actually static, but the amount of choices is user dependent and the order is editable by a user. The original method was not efficient enough. In the server side, the java code ran a database query for fetching the appropriate occupation list. After that, these list items were put inside the HTML table items. These table items also included JavaScript code for fetching the occupation related information from the database after the user click. A few hundred occupations were typically included to the front page. So, the solution was very simple, but unfortunately inefficient.

In the optimization work, it was given up using HTML tables with the occupation selectors. Using the Ajax concept (Deitel 2012) there is no longer any need to load the whole occupation list to the client side. Ajax is a client-side script that communicates to and from a server/database without the need for a complete page refresh. Based on the needs of a user, it is now possible to update only parts of a web page without the need of reloading the entire

page. Coding of the selectors in a new way reduced the sizes of the front page resources about 30 percent.

3.2 Images

Images were optimized first by using TinyPNG compression software (TinyPNG 2015). TinyPNG uses smart lossy compression techniques to reduce the file size of PNG and JPEG files. By selectively decreasing the number of colours in the image, fewer bytes are required to store the data. The effect is nearly invisible but it makes a large difference in file sizes. In this study, we discovered that the sizes of large PNG-files decreased even 70 percent. The sizes of small files decreased a few dozen percent.

In the second stage, small images were integrated into the part of the CSS file. If an image is integrated into the CSS file, it is retrieved by the same HTTP message as the CSS file without a need for extra HTTP headers. In this study, these image based HTTP request and response messages together included 825 bytes header information. Also, there is no longer any need to open a new TCP connection for image loading. On the other hand, it is needed some more CSS code when an image is integrated into CSS file. Also, the size of a CSS integrated image increases because of the Base64-encoding of this image. We calculated that if an image is bigger than about 3000 bytes, it is clever to keep this image in its own file. For example, the front page of ForeAmmatti had six images. Five of them were small enough to be integrated into the part of the CSS file.

3.3 Crunching of the Text-based Files

An important part of the website build process is the crunching of text-based resources to remove excess characters (Spiesser 2004). This ensures that only the smallest numbers of bytes are transmitted to the browser for parsing. Crunching tools primarily reduce white spaces and comments to ensuring that the resulting code can still be executed without problems. There are several free tools available for crunching with varying compression rations. This optimization project used three of them. Depending on the formatting of the original, these tools reduced file sizes 18-40 percent. The JavaScript files were reduced 38 percent, and the CSS files 18 percent. The XHTML file was reduced 40 percent.

3.4 GZIP Compression

Before the optimization ForeAmmtti web service already loaded many files in a GZIP format from the external servers. As part of the optimization project, we modified the setup parameters of the own server so that some file types will be delivered in the GZIP format. By delivering HTML-, CSS-, and Java Script files in the GZIP format, we could reduce these file sizes from 63 to 76 percent. It is clever to use compression only for text files because binary files, for example images, have already been compressed. It is worth to mention that GZIP compression does not come without drawbacks because processing power is needed for compression and decompression on the server and client side.

3.5 Browser's Cache

The ForeAmmatti web server adds the Expires and Max-age header fields to HTTP response messages. The Expires header field gives the date/time after which the response is considered stale. The Max-age field indicates that the response is to be considered stale after its age is greater than the specified number of seconds. With the help of these header fields, it is possible to advertise long lifetime for resources which are seldom updated. So, browsers can use cached resources immediately without the need for if-modified requests. To prevent the use of modified resources, the ForeAmmatti web service names all modified cacheable resources with the unique file name. Thera was also a lot of Java Script codes inside the XHTML-file. To make possible to cache these Java Script codes, most of them were moved to their own files.

4 OPTIMIZATION RESULTS

In this section it is presented the most significant optimization results. The whole website was optimized but we present here the results related only to the front page. Load Impact-test loaded all the pages of ForeAmmatti.

4.1 Front Page

The sizes of the resources and the load time of the pages were examined by a browser developer tool (Google 2015). Table 1 presents the resources of the front page and their sizes before and after the optimization. The numbers of these resources, before and after optimization, are presented in the first column. It can be seen that the resources of the front page reduced from 947 905 to 190 333 bytes. Most of this decreasing came from the smaller sizes

of the resource, but there were also some user services that were moved away from the front page to the other pages. The number of the front page resources reduced from 29 to 15. Thera are still six images on the front page but five of them reside inside the CSS file. The fonts for CSS are loaded from Google's server. The size of these font files also decreased during the optimization work. So, we can suppose that Google is also doing optimization work.

Table 1: Resources of the front page.

Resource (pieces)	Bytes before	Bytes after
XHTML-file 1 -> 1	420 998	9 047
CSS 2 -> 2	30 162	11 009
Fonts for CSS 2 -> 2	43 400	35 088
JavaScript files 18 -> 9	444 216	134 626
Images 6 -> 1	9 129	563
Total 29 -> 15	947 905	190 333

Table 2 presents different load times of the front page before and after optimization. In this case, the browser and web server were located in the same local area network, but some external resources had to be downloaded over the Internet. The browser cache was switched off. The presented averaged results are based on ten measurements. The standard deviation of these ten measurements is presented inside the brackets. As can be seen, the optimization has downgraded the load times remarkable.

Table 2: Load times of the front page.

Event	Before ms	After ms
DOMContentLoaded	745 (75)	294 (11)
Loaded	806 (119)	303 (10)
Totally Loaded	837 (111)	328 (9)

4.2 Load Impact-test

The Load Impact service (Load Impact 2015) helps determine the capability of handling a certain amount of users on a website. In this study, we used the free version of this test tool which makes possible to test 100 concurrent users. The amount of concurrent users increased evenly from 1 to 100 during the five-minute test. The test generator was situated in USA. So, the test used long-distance connections because the tested website resides in Finland. This kind of long-distance test is typically sensitive to Internet traffic, and therefore, two test cases are never entirely similar.

The ForeAmmatti website was tested before and after the optimization project. The test results are presented in Figures 1 and 2. Figure 1 presents the starting point. Figure 2 presents the situation after the improvements. In these figures, the numbers of concurrent users are presented by the blue colour. The green colour presents delays experienced by the users. The red colour presents the average load time of the front page. Before the optimization, ForeAmmatti was capable of serving only 30 concurrent users without that user delays and page load times increased remarkable. After the optimization work, at least 100 concurrent users can

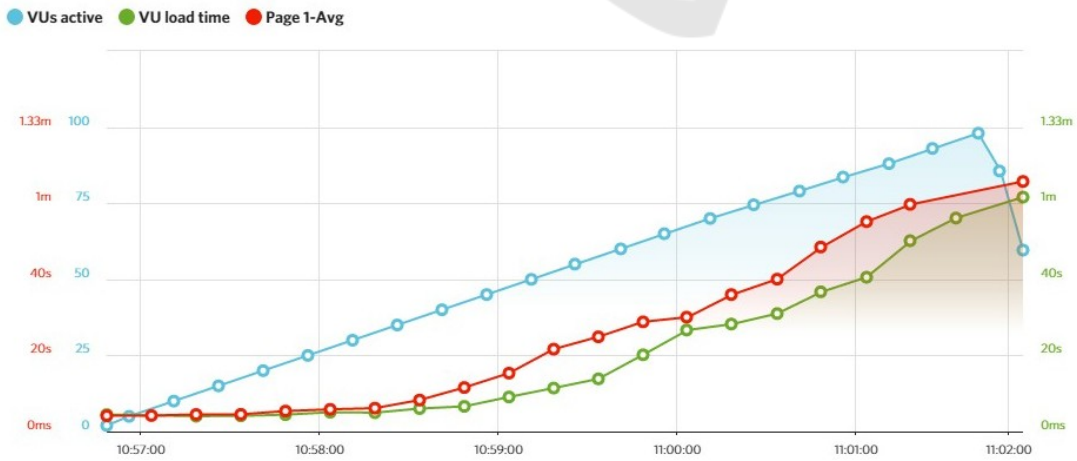

Figure 1: Load Impact-test before the optimization.

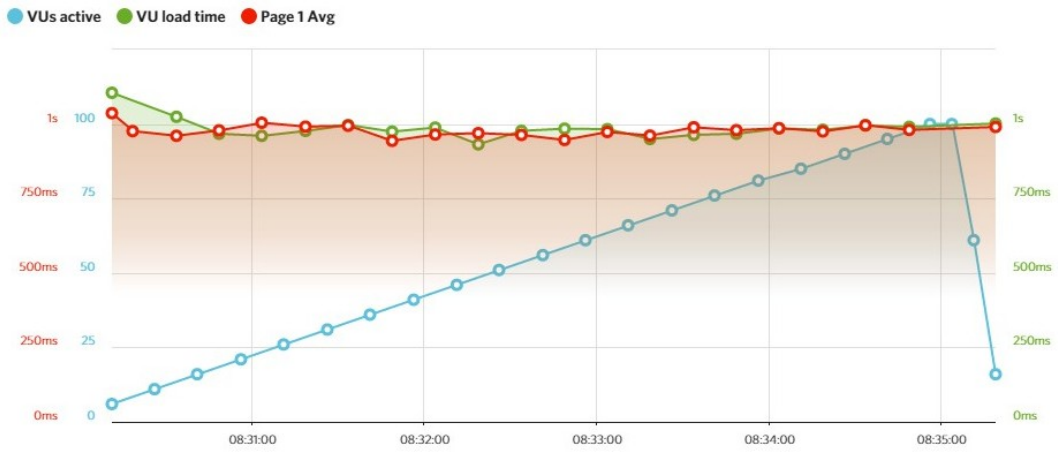

Figure 2: Load Impact-test after the optimization.

be served without problems.

In the test made before the optimization, 206 megabytes were transferred to the test generator. The front page was loaded 206 times so that the minimum loading time was 3.7 seconds. In the test made after the optimization work, 78 megabytes were transferred to the test generator. The front page was loaded 561 times so that the minimum loading time was 0.7 seconds.

5 CONCLUSIONS

This WPO project shows that WPO can have very positive and noticeable effects to web services. The ForeaAmmatti web service is now more efficient and user friendly than before the optimization project. It can now serve three times as much users as before. The load times of resources have been reduced over 50 percent. Thanks to this WPO project, the ForeAmmatti web service can now enjoy the advantages offered by WPO.

However, we have to remember that this project represents some kind of extreme case. In the past, not enough attention had been paid to the optimization of ForeAmmatti. Especially, the occupation and region selectors were originally implemented in a very inefficient way. By recoding these selectors in a clever way, we could reduce the byte amount of the front page resources about 30 percent. In spite of the fact that the starting point of this optimization work was extraordinarily bad, we can present a subjective estimate that WPO can easily increase the performance of a website about 20-30 percent on condition that there has been only a little WPO work before. The performance increase of this magnitude often enhances the overall user experience of the website. But above all, the energy consumption of the Internet reduces because of smaller resource files. We can be environmentally friendly and energy efficient by doing WPO.

It is good to remember that WPO work is a continuous process. There will be a need to reoptimize the ForeAmmatti website when the use of HTTP/2 becomes common. For example, HTTP/2 can concurrently download several resources by using the same TCP connection.

REFERENCES

Abdullah M, Fararjeh A, Amany M, Jabal A. 2010. *Recommendations to Improve Performance of an Enterprise Web-based Application*. The Proceeding of ISWSA 2010 conference. Pages 1-6.

Ali W, Shamsuddin S, Ismail A. 2011. *A survey of Web caching and prefetching*. International Journal of Advances in Soft Computing and Its Applications, Vol. 3, No. 1, March 2011.

Begel A, Simon B. *Struggles of New College Gra-duates in their First Software Development Job*. SIGCSE '08, Proceedings of the 39th SIGCSE technical symposium on Computer science education. Pages 226-230.

Bianzino A, Raju A, Rossi D. 2011. *"Greening the Internet: Measuring Web Power Consumption"*. IT Pro, Published by the IEEE Computer Society, January/February 2011. Pages 48-53.

Butkiewicz M, Madhyastha H, Sekar V. 2011. *Understanding Website Complexity: Measurements, Metrics, and Implications*. IMC'11Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference. Pages 313-328.

Deitel P, Deitel H, Deitel A. 2012. *Internet and World Wide Web: How to Program*. Fifth Edition, Pearson. 955 Pages.

Galletta D, Henry R, McCoy S, Polak P. 2004. *Web Site Delays: How Tolerant are Users?*. Journal of the Association for Information Systems. Pages 1-28.

Google. 2015. *PageSpeed Tools*. Retrieved 10.12.2015 from https://developers.google.com/speed/pagespeed/

Gupta M, Singh S. 2003. *Greening of the Internet*. ACM SIGCOMM, Karlsruhe, Germany.

Fabian B, Goertz F, Kunz S, Muller S, Nitzsche M. 2010. *Privately waiting–a usability analysis of the tor anonymity network*. Sustainable e-Business Management. Pages 63–75.

Ferragina P, Manzini, G. 2010. *On compressing the textual web*. In: Proc. of Third ACM Conference on Web Search and Data Mining (WSDM). Pages 391-400.

Hongkiat. 2015. Ultimate Guide To Web Optimization (Tips & Best Practices). Retrieved 9.12.2015 from http://www.hongkiat.com/blog/ultimate-guide-to-web-optimization-tips-best-practices/

Load Impact. 2015. *On Demand Load Testing for Developers & Testers*. Retrieved 10.11.2015 from https://loadimpact.com/

Qian F, Quah S, Huang J, Erman J, Gerber A, Mao Z, Sen S, Spatscheck O. *Web caching on smartphones: ideal vs reality*. In Proceedings of the 10th international conference on Mobile systems, applications, and services, MobiSys '12. Pages 127–140.

Rajeev B, Bakula K. 2015. *A developer's insights into performance optimizations for mobile web apps*. Advance Computing Conference 2015 (IACC). Pages 671 - 675.

Soininen J. 2012. *Website Performance Evaluation and Estimation in an E-business Environment*. Doctoral dissertation, Tampere University of Technology. 188 Pages. Retrieved 12.11.2015 from http://dspace.cc.tut.fi/dpub/handle/123456789/21230.

Spiesser J, Kitchen, L. 2004. *Optimization of html automatically generated by wysiwyg programs*. WWW 2004, Proceedings of the 13th international conference on World Wide Web. Pages 355–364.

TinyPNG. 2015. *Shrink PNG files*. Retrieved 12.10.2015 from https://tinypng.com/