*Environment. Technology. Resources, Rezekne, Latvia*
*Proceedings of the 10th International Scientific and Practical Conference. Volume III, 87-91*

# Modern approaches to reduce webpage load times

**Aleksejs Grocevs, Nataly Prokofyeva**
*Riga Technical University, Faculty of Computer Science and Information Technology.*
*Address: 1/4 Meza str., Riga, LV-1048, Latvia*

***Abstract.*** **Nowadays, many modern websites offer a variety of information services, providing a dynamically generated content for the end user. However, the more users are trying to obtain such content, the slower becomes its loading time in their browsers. This article explains the mentioned problem in detail, as well as examines the roots of this problem.**

***Keywords:*** **Loading time, performance, web site.**

## I INTRODUCTION

Nowadays, applications deal with many different tasks. With the appearance of the Internet, it became possible to solve these tasks remotely, quickly transferring large amounts of information. End users consider it a favorable way to work with their data, regardless of their geographical location and available forms of communication. These and many other factors led the software development industry to look for a different set of content transfer opportunities via the Internet, using a traditional web user-interface. Today, websites are filled with user-friendly interfaces, animations and other appealing objects that end users may desire. But before all this appears in the user's eyes, it has to go through the server to the user's browser, and whilst the amount of data increases, the time necessary to exit the path, is growing [1]. This paper examines causes of the increase of this time and offers reduction options.

## II INFORMATION GATHERING

Using Firebug extension for Mozilla Firefox browser, developers can view the division of time consumed to process a page request - until the moment of full loading and being displayed in the browser. By exploring the load time of a number of popular Web-sites with this tool, it is possible to see that the load time is mostly spent on data transfer from the server to the browser; for example, in the case of facebook.com all elements of the downloading queue, according to Fig. 1, even after 12 minutes of download, haven't loaded completely.

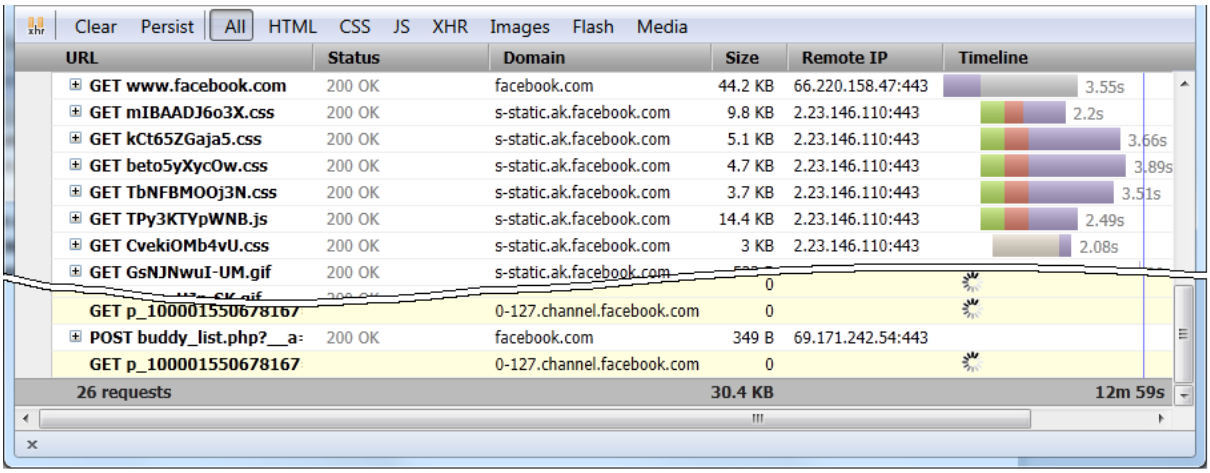


Fig. 1. Facebook.com page element downloading time graph

*ISSN 1691-5402*

*DOI: http://dx.doi.org/10.17770/etr2015vol3.506*

The main page load time distribution is shown in the Fig. 2 separately. It can be observed that most part of the page's loading time is taken up by data transmission. This data from the server can be divided into static HTML text with images and dynamic content, such as JavaScript (JSON) and Flash. Basing on the file list load time, a conclusion can be made, that most part of the data comes from different domains, which is done to increase the count of different data sources. Furthermore, it can be noticed, that even after a long wait some files are not fully loaded, which suggests possible problems with external data providers.

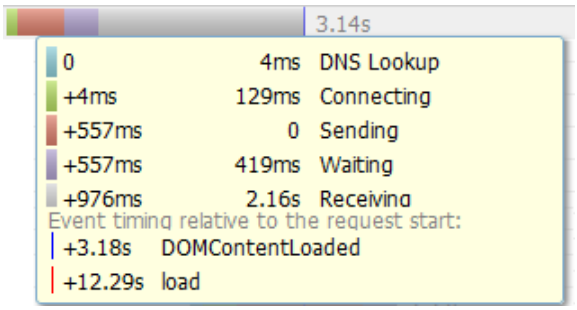


Fig. 2. Facebook.com page loading time distribution

## III MULTIPLE FILE CREATION

One of the factors affecting the load time is multiple CSS file creation, although this content may be transferred as one joined file. Similarly, it permits to process JS-format files with an existing JavaScript code [2]. To impose additional complexity during the processing of an abnormally large file, it is possible to merge these files in the deployment phase, when it is deployed on a web server, for example, using Ant [3] or Maven [4] scenarios. This approach not only speeds up data transfer via the Internet, but also reduces the hard drive usage frequency, without requiring frequent reference to the hard disk file from operating system to read next.

Although usually frequently requested files are already in memory, and the increase of server performance will not be large, the main advantage of this approach is the required acceleration of information transfer, which results not only in the files merging and them being read faster from the media, but it also eliminates unnecessary resources overhead by reducing HTTP header transfer amount. Excessive amount of headers is shown in Fig. 3. The transferred header size (1051 bytes) is twice the size of the data received.

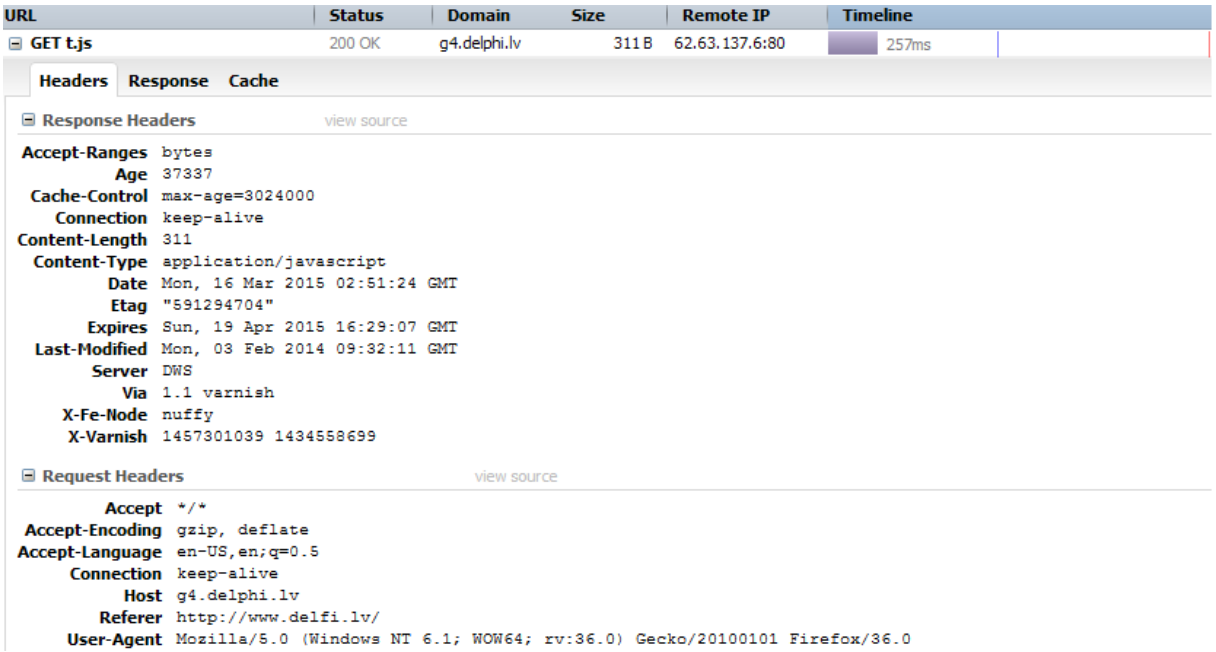


Fig. 3. News site request and response header size comparison

## IV CHOOSING INCORRECT PROTOCOL

In addition to static type, modern websites also provide dynamic content, which varies according to the requests and also to the current status of the users or, similarly, according to the instantaneous requested data modifications. Although the majority of sites send new browser data in plain-text format to the client (or as ready-to-insert HTML-code), the usage of specialized protocols, designed for better use of Internet channels, can reduce the total data transfer time. For example, these protocols could be:

### *A. AMF protocol*

AMF is created by Adobe messaging protocol, structure of which was published in 2007. It is intended for binary data transfer between applications, which are operated on the *Flash* platform and written

on the ActionScript base, all run on a server that can be either Flash Media Server (or any other media-server) or server-side application, written in PHP, Java, Ruby or other language. Before transmission, all data is serialized and all lines of text are encoded in Unicode.

### *B. RTMP*

Real-Time Messaging Protocol – also developed by *Adobe*, allows to send not only video and audio data, but also other information, that makes it possible to build applications without using additional unnecessary resources to maintain service.

### *C. XML*

eXtensible Markup Language, - a simplified SGML (Standard Generalized Markup Language) subset, which is designed for data description and a form of storage, that can be understandable to both computers (easy to read data objects) and humans.

### *D. JSON*

JavaScript Object Notation – JavaScript language object, which can be recognized by the browser's JavaScript engine and treated as a ready-to-use object, without further action. The Fig. 4 shows the received code, which was converted into JavaScript object with key-value pairs.

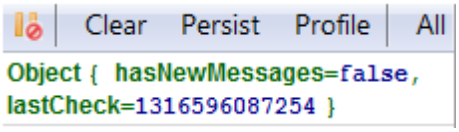


Fig. 4. JSON data evaluation as JavaScript object

### *E. HTTP2*

Second version of HTTP protocol draft has been finalized in the beginning 2015 year and currently is awaiting RFC allocation and standardization finalization [5]. It is too early to estimate implementation and usage benefits, since only latest developer nightly builds of modern browsers supports this protocol, however it must significantly improve data transmission speed, since it is binary protocol as well as AMF and RTMP, and is built upon SPDY protocol extension, including stateful header compression (in HPACK) [6].

## V IMPACT OF NETWORK PERFORMANCE

Although the homepage had been already prepared for viewing in ~3 seconds after making the request, its component loading could affect the parallel connection limit. In the file list it can be seen, that the CSS files are transmitted simultaneously. However, all browsers have a fixed maximum of connections to a particular domain by default and in accordance with W3C recommendations [7], so that *Internet Explorer 9* can be used for four simultaneous connections, *Chrome 12* - for six; *Firefox 7* - for fifteen and *Opera 11* - for sixteen.

This means you can simultaneously download from four to six files, depending on the browser used. The creation and closing of the connection takes a long time, that is why all modern browsers (including web-servers) can keep continuous or Keep-Alive connections, which makes it possible to use a single connection to multiple file transfers. The only requirement is that the server must indicate how many seconds it will allow for maintaining the connection without the actual data transfer. This is done to avoid an overflowing connection pool on a web-server, which is typically around 100 sockets providing client connection. If the first 100 clients connect to the server and continue to do so, it will be impossible to serve other clients because of the pending data requests from the user-server.

As one part of the solution we can suggest to use analysis, which aims to define the average and maximum loading times per element page. Using crawl results, it will be possible to determine the required time needed to apply the server-side Keep-Alive interval [8]. The client connects to the site, obtains all required files and then the server disconnects the client to make room in the pool for the next connection, while the client processes information and displays it to the end user, who will spend some time viewing the obtained information and will most likely not make more requests for some time.

A different approach may give the user an ability to increase the competitor-requested number of files by using other (sub) domains, as shown in Fig. 1 - the JavaScript and CSS files are loaded from a separate *s-static.ak.facebook.com* domain, i.e., they are assigned in addition to the number of connections. Usually, such way of division is implemented to separate the static content and use of the web servers with high static file handling capabilities, such as Lighttpd and Nginx, designed especially to serve the heavy duty sites and to provide minimal configuration/ extensibility options.

## VI UNIQUE DATA USAGE

In order to enable updated data transmission in the background when making a request for information that the user's browser already had, it is possible to use built-in data storage in user's computers. Studies have shown that Adobe Flash plug-in is available on 99% [9] of all computers that have Internet access. It enables data transfer processing to use options provided by technology, one of them being Local Shared Objects [10].

Local Shared Objects (100KB in size by default) is a flash cookies file with a .sol extension, which in itself contains an array that uses the string as a key and any ActionScript language object as a value. The

user can specify a maximum size of Shared Objects, which can be 0kb, 10KB, 100KB, 1MB, 10MB or unlimited. The available amount of developers is relatively high. They keep a constant part of the site, occasionally checking their integrity and, if necessary, replacing any necessary parts.

Of course, this approach has its drawbacks: it is not possible to access another domain's Shared Object, even by using sub-domain's request because it contradicts with Adobe Flash Sandbox Restrictions rules, and, by default, if the Shared Object approaches the size of 100KB, the user will receive a message and a demand for additional space needed for granting permission, which may confuse users who are not accustomed to it. Although this technology allows creating a local caching, it is generally intended for small data storage configurations, rather than actual file-keeping.

Like Adobe Flash, Microsoft Silverlight supports their storage facility known as ISOSTORAGE (isolated local storage), which is separate from the browser's temporary directory and is sized 1MB. The user, just as in Flash, may change its size in steps of 100KB, 1MB, 5MB and 10MB.

Unlike the Adobe Flash technology, Microsoft Silverlight provides access to stored resources directly from JavaScript code without creating additional difficulties existing in Flash <–> JavaScript interaction. Another opportunity that developers may use - ISOSTORAGE format remembers the file system, i.e. it is possible create files and folders, copy, delete, rename and perform other activities that are available with a conventional file system.

## VII WEAK CACHING IMPLEMENTATION

All the files in the example are sent from the Apache Web server that does not use external caching capabilities, which negatively affects the load time. The majority of web sites update their information no more than a few times a day, making it possible to use reverse proxy caching options to speed up data transfer. Conventional proxy server acts as a mediator and usually does not belong to a web page owner. Their task is to maintain some of the traffic that passes through until the next time another user will request the same data to issue from their storage. These proxies tend to be regarded as direct (forward).

Contrariwise, reverse proxy is located close (both in physical and network routing sense) to the web server and mainly serves users through the web server, and not directly.

Reverse proxy can exempt web server from the data compression, and during transition proxy <–> web server does not require small amounts of data transfer. Almost all modern browsers support data compression - they are responsible for sending the Accept-Encoding header in the request. Its value can be GZIP, which means that the browser supports and requests web server capacity-encoded page; or DEFLATE, which means that the browser is not built, or has disabled compression processing. From these two values and the sequence from your options/settings of your web server may be concluded what type of content to transfer in response to the request.

If the compression is available and settings are enabled, the server, before sending the final result, undergoes the GZIP filter that compresses data, particularly the text. This can be observed in data being compressed up to 100 times.

The fact that compressed data is received in headers, symbolized by Content-Encoding and valued as GZIP, which compresses the received requests and responses, can slightly reduce the overall load time.

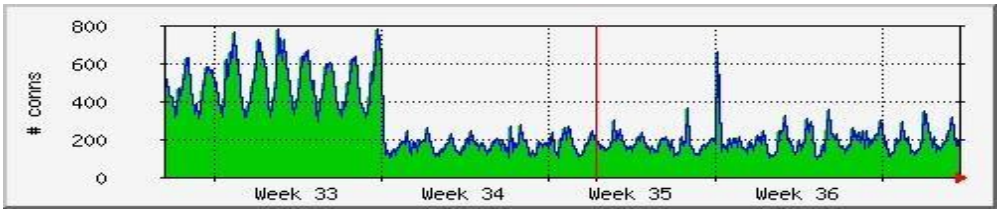


Fig. 5. Reverse proxy implementation performance boost

Another objective that can be attained by a proxy server is the load time balancing that distributes requests among multiple available web servers, depending on their workloads and opportunities to serve new demand [11]. Similarly, the mission can be considered a static page generated in storage and transmission from the cache, rather than a request to the web server where the page will be generated by a script, slowing down the transfer process. In Fig. 3 we can see that in the file, transferred from server, two additional headers appear, which indicates that the server uses reverse proxy usage, Varnish, - it is part of Via, a value of 1.1, used in the HTTP protocol version and in the name of the proxy software X-Varnish. (All HTTP headers that begin with X are not defined in HTTP 1.0 and 1.1 specifications, and are freely available for server-specific information). Its value is the request identifier, by which problems can be found in logs. Another significant reverse proxy server, which can also serve as a direct server, is Squid. It is

used in the Wikipedia Foundation, and the web-server/proxy ratio is 3:1 [12]. Reverse proxy usage enables reducing the requests to the main web server, in certain cases up to four times, as shown in Fig 5.

## VIII CONCLUSIONS

In this article web site load time problem has been examined, including additional hints on how to solve occurring problems. Multiple similar file transmission and extra-header addition were described as server-side influencing factors. Browser's inability to persist already received data and attempts to run unprepared JavaScript code are considered as the main client-side problem. Many CSS and JavaScript file-merging processes that were described significantly affect not only the file transfer speed of the network, but also increases the load- and parse time in the user's browser. As a solution, we offered to combine similar content files into one, just like both the CSS and JavaScript files in the support type, and this combination does not affect the final result.

Furthermore, sending and receiving large headers was also discussed. Since servers usually do not need headers to identify users of the static content files, it is better to remove additional unnecessary headers from all static content either using server side – the same web server software, or the use of programming languages - or JavaScript for removing unneeded request headers.

Another element on which this article is focused: choosing the most appropriate protocol. Many websites in AJAX queries return the JavaScript code, which overfills the browser - these responses take time and increase the total amount of data. It is possible to use JSON or JSONP protocols only for transferring data and information to determine which one from pre-loaded functions to call.

From the set of already implemented improvements, we described the distribution of static files across multiple servers/domains using additional servers and other domain names in order to circumvent the browser restrictions. In addition, the reverse proxy was studied, which was already encountered in examples and also reduced the overall load time.

## IX REFERENCES


[1] S. Souders, High Performance Web Sites: Essential Knowledge for Front-End Engineers, O'Reilly Media, Inc, 2007.

[2] R. Page, Website Optimization: An Hour a Day, John Wiley & Sons, 2012

[3] J. Fuller, *Apache Ant Recipes for Web Developers*, FastPencil, Inc, 2010.

[4] Srirangan, *Apache Maven 3 Cookbook*, Packt Publishing, 2011

[5] *Hypertext Transfer Protocol version 2* [Online]. Available: http://www.rfc-editor.org/internet-drafts/draft-ietf-httpbis-http2-17.txt [Accessed: Mar 15, 2015]

[6] *HPACK - Header Compression for HTTP/2* [Online]. Available: http://http2.github.io/http2-spec/compression.html [Accessed: Mar 15, 2015]

[7] W3C, HTTP/1.1 Connections, [Online]. Available: http://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html [Accessed: Mar 15, 2015]

[8] S. Shivakumar, Architecting High Performing, Scalable and Available Enterprise Web Applications, Morgan Kaufmann, 2014

[9] Adobe Inc, *Adobe - Flash Player Statistics*, [Online]. Available: http://www.adobe.com/products/player_census/flashplayer/ [Accessed: Mar 15, 2015]

[10] Adobe Inc, *Adobe - Adobe Flash Player : What Is a Local Shared Object*, [Online]. Available: http://www.adobe.com/products/flashplayer/articles/lso/ [Accessed: Mar 15, 2015]

[11] H. ElAarag, S. Romano et al, *Web Proxy Cache Replacement Strategies: Simulation, Implementation, and Performance Evaluation*, Springer Science & Business Media, 2012

[12] Possibility Outpost, *High Scalability - High Scalability - Wikimedia architecture* [Online]. Available: http://highscalability.com/wikimedia-architecture [Accessed: Mar 15, 2015]