MIPRO 2014, 26-30 May 2014, Opatija, Croatia

Front end optimization methods and their effect

Eng. Ilian Iliev
Web Software Developer
Sofia, Bulgaria
ilian@i-n-i.org
http://ilian.i-n-i.org

Assoc. Prof. Georgi Petrov Dimitrov, PhD
University of Library Studies and Information Technologies
Sofia, Bulgaria
geo.p.dimitrov@gmail.com

Abstract - With the evolution of the internet the loading speed of the web pages has become a vital factor for their success. A few hundred milliseconds difference in the response time are enough for visitors to choose one page instead of its competitors. This paper describes the increasing demand for faster loading webpages and the need of methods for optimization of the web code in order to increase their speed. It describes the process of loading webpages in browser and lists several methods for front-end performance optimization. The understanding of the loading process allows the correct identification of the parts that can be affected and improved. It also helps for understanding the effect of these methods and the cases when they can be applied. Although the total effect of the optimization depends on the specific webpage the appliance of these rules does not only reduce the page loading speed but it also affects the total bandwidth and load of the web server.

Keywords: front-end, optimization, web, performance, speed, optimization methods

1. INTRODUCTION

In the last few years we are witnessing the increasing usage of internet sites and web based applications. They have become an essential part of our lives and daily schedule. Unnoticed by the regular users together with the increased number of websites there is a huge increase of the number of resources used by these webpages. The increased load on the webservers, forces the hosting companies to add new webservers, to handle it, which leads to increased expenses for the business. The increased number of resources not only adds load to the servers hosting the websites and applications but also to the users. In addition to this the loading speed of the websites has become a major factor determining the user experience and one of the criteria used by search engines for positioning in the search results. The loading time and bandwidth for the websites can be decreased by applying different front-end optimization methods. As a side effect these methods decrease the load of the web servers as a result of the decreased bandwidth and number of established connections.

Section two presented the change of webpage size covering a period of 17 years together with the current state of the web pages. The sections analyzes the effect of the constant growth of the page's size on both the server and client side and marks the basic areas that optimization should target.

Section three starts with description of the webpage loading process followed by presentation of several different methods for front-end optimization of web pages with explanation of their target and effect.

Sections four contains the data of an experiment measuring the webpage size and loading time before and after the appliance of optimization methods. It also shows the difference of the server response time before and after the optimization.

2. HISTORY AND CHANGE OF THE WEBPAGE SIZE

History of the web page size

The evolution of the computer technologies in the last few years resulted in rapid increase of the internet connection speed and also the size of the web pages. According to a research published by websiteoptimization.com in November, 2012 the average size of the webpages has more than tripled for the period 2008 - 2012[1].

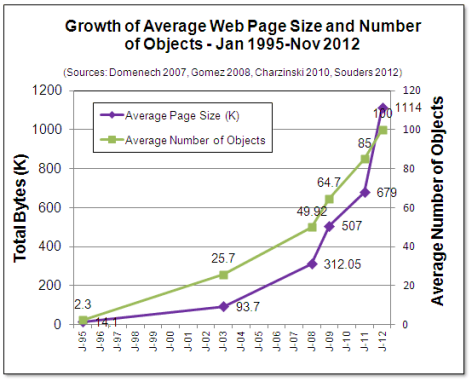

Fig.1 Average webpage size growth 1995 - 2012

As shown on fig.1 together with the growth of the average size there is also a significant growth in the average number of resources per page. From 2.3 up to 100 - this number has increased almost 50 times.

According to HTTP Archive for the period between November 2012 and November 2013 the average webpage size has raised from 1.25MB to 1.7MB, resulting in 40% growth for the last year[2].

Together with the growth of the page size there is also a constant increase of the end user bandwidth. A research made by Jakob Nielsen that covers the period from 1983 to 2013 shows a 50% growth of the connection speed per year[3]. The increase of the connection speed also raises the demand of the customers for faster loading and highly available websites.

The result of page size growth

The growth of the webpages has 2 major effects – on one side it increases the loading time for the end client and on other it increases the load of the server. The second of which is caused by the higher number of resources per page and the increased bandwidth.

For example if the total size of a webpage including its resources is 1.5MB, the total number of concurrent connections that a server can handle is 100 and the average connections speed is 1.5MB/s, which means that for 1 minute the server can serve the page up to 6000 times. This example ignores the fact that modern web browsers can open multiple simultaneous connections to the server to fetch page resources which adds additional load because of the process of establishing and closing connections.

The higher number of concurrent connections and their duration affects directly the total number of clients that the website host can serve. In order to lower the server load and increase the maximum number of users that the server can handle is necessary to either decrease the bandwidth or the number of connections per page.

3. WEBPAGE LOADING PROCESS AND FRONT-END OPTIMIZATION

Webpage loading process

The understanding of the webpage loading process is essential for the understanding of the effect of the front-end optimization methods.

The loading of the website starts by entering an URL in the address bar of the browser or clicking on a link. Then the browser parses the URL and determines the correct protocol, mainly HTTP or HTTPS, the domain, port resource address and request arguments if provided. Once the browser has determined the domain it makes a DNS lookup in order to find the IP address of the server hosting the page. Then a network connection is established between the server and the browser and the browser requests the specified page. The server generates a response and returns it to the browser. The response can be either a static resource or it can be generated by an application.

After the browser has received the response and if it is an HTML document it starts parsing it. In case of incorrect syntax the browser suppresses the error and tries to generate a structure as close as possible to the markup. Then a Document Object Model (DOM) is created. It is a hierarchical structure where every tag and attribute is represented as a separate node. Once the DOM is completely loaded the browser starts to load the additional resources specified in the code – styles, images and scripts.

After the load of each style resource the browser parses it and starts to apply the defined styles to the nodes in the DOM tree. The styles do not amend the DOM tree and as a result the browser can continue loading other resources simultaneously with their processing.

Due to the fact the scripts may cause changes in the DOM tree, during their loading, parsing and execution all other processes are halted. An exception here is the fact that some browsers as Mozilla and Safari continue loading other resources during the execution of the script using separate threads. In this case the DOM manipulations are still placed only in the main thread of the browser. Once the DOM tree is generated and the base resources are loaded and the browser starts rendering the webpage.

The front-end optimization methods primary target the process of page resources loading, their order and the usage of caching techniques.

The effect of front-end optimization

According to Steve Souders' "Performance Golden Rule" average of 80-90% of the user response time, while loading a webpage is spent in the front-end[4]. This time is spent mainly downloading resources and processing them. It is additionally increased by the fact that most browsers pause all other processes while downloading and executing script resources. Based on this fact Souders recommends to the developers to start optimizing the front-end parts of the websites and applications before focusing on back-end optimization[5].

The main goal of the frontend optimization methods is to reduce the loading time by lowering the generated traffic; respectively the time spent downloading resources. Another part of the optimization process is to reduce the number of resources required by the page, again in order to optimize the loading time for the end client. The second main goal is to allow the browser to load the page and its resources progressively and display a rendered page as fast as possible leaving the user with the feeling for quickly loading webpage. Here are described some of these methods and how they affect the loading process and the server performance.

CSS/JavaScript minification – this is the process of removing obsolete characters like blank lines and comments from the code in order to lower its size. The process of minifying JavaScript also includes a code obfuscation that replaces the names of the variables and methods with shorter ones. This process is also known as compression and can combine different scripts or styles into a single resource. As a result not only the size of the resources is lowered but also their number is decreased which leads to lower number of connections opened per page load.

Using Expires and Cache-Control headers – these headers are used to instruct the browsers whether a specific resource can be cached or not. The Cache-Control header was introduced in HTTP/1.1 in order to overload some of the limitations of the Expires header, for example while Expires value is specific date the Cache-Control max-age directive can set a time interval since the request has been served e.g. "X days after the resource has been served". Although the Cache-Control header is actually a replacement of the Expires one, it is recommended to use both of them together in order to ensure compatibility with browsers not supporting HTTP/1.1.

The main benefit of the use of Expires and Cache-Control headers is that once the resource has been cached the browser will make no more requests to fetch it from the server until the resource is considered an expired. This way once the page is loaded and the resources are cached they would not be requested again if the page is reloaded or during navigation inside the website.

Using GZip compression – this method is controversial because on one side it reduces the

bandwidth but on other the process of compressing the resources adds an additional load to the server. However, combining this method with proper caching techniques leads to lower number of requests i.e. lower number of executions of the compression algorithm. This method has good effect on text based resources like HTML documents, styles and scripts but has poor effect on images and other documents. In version 1.1 of the HTTP protocol the capabilities of the Accept-Encoding header are have been extended in order to allow the browsers to send additional data to the web servers regarding the supported compression methods and their preference order.

Using a Content Delivery Network – with this method the task of serving static resources is delegated to different network of servers. The Content Delivery Network (CDN) is a large geographically distributed network of servers aimed to serve content to end-users with high speed and availability. The distribution is made in order to serve the user request from the closest and fastest location. Such networks are widely used in the modern web sites to serve static content, downloadable resources, media and other.

In the CDN networks the servers are split into two groups – edge and origin servers. The end-user communicates with the edge servers and asks them for the needed resources. If the edge server does not possess the required resource it makes a request to an origin server for it. Once the resource is retrieved from the origin server the edge serves it to the user and also stores a copy of it for future uses. This way the end-users can retrieve the static resources of the webpages much faster because of the closer location of the server.

When using a CDN the application server should handle only the request to dynamic pages, e.g. those for which data calculating and preparation is required. In this case, the server hosting the application should not static resources.

Using only external JavaScript and CSS files is another method that can reduce the traffic generated to load the page. If measured directly, loading a page with internal JavaScript and CSS files is faster than using external ones, which is caused by the reduced number of requests.

However this way the end-user cannot benefit from the caching capabilities of the browser because the styles and scripts will be served as part of the HTML code for each page. This way, the traffic generated when navigating inside the website is increased and so is the loading time of the webpages. Vice versa putting all scripts and styles into external resources allows the browser to cache them and this way reduce the traffic and loading time for the page.

All methods presented this far are related to the bandwidth and number of requests made in order to completely load a webpage. The following methods affect the process of progressive loading of the page and the moment when the browser starts rendering the content.

Placing the styles at the top of the page – this way the styles are among the first resources that the browser starts to load after the HTML document has been processed. As a result the declared styles are applied to the DOM tree before the loading and execution of other resources and the browser can start rendering the webpage while waiting for other resources to be loaded. This is the reason why the style tags should be placed as higher as possible in the “head” section of the HTML document.

Placing the styles at the bottom of the page is another method for speeding up the loading of resource and progressive rendering. As described above during the loading and execution of the script resources the browsers halt all other processes. This also affects parallel downloads. Although browsers can download multiple resources simultaneously, most of them stop the downloading process when executing scripts, even if the resources are coming from different hosts. To avoid this blockage the script tags should be placed at the bottom of the page, usually before the closing body-tag.

Removing the duplicate scripts is another essential part of the front-end optimization. Having duplicate scripts has two major drawbacks. First of them is the increased loading time as a result of the repeated parsing and execution of the script. The second drawback is caused by the fact that frequently duplicate scripts lead to unexpected behavior in the front-end and may also lead to errors breaking the whole page.

In modern website quite often some visual elements like images, menus or even sidebars are removed when displaying the page in mobile devices. These changes are focused in two directions – lowering the bandwidth and the loading time due to the lower number of resources to load and also providing better experience for users by better utilization of the smaller screen. However such techniques are usually part of the design and user experience so they are only mentioned in this paper without being described thoroughly.

4. MEASURING THE EFFECT

The total effect of these optimization methods strongly depends on the specifics of the web page – the number of resources that can be cached and that are reused between the different pages of the site, the percent of bouncing users i.e. the ones that visit only a single page and then leave the website and others. However, testing a sample HTML template can show the general effect of the frontend optimization on both the server and the client.

In this case the following optimization methods are applied:

- CSS/JavaScript minification and compression
- Use of Expires and Cache-control headers
- Use of only external styles and scripts
- Placing styles at the top
- Placing scripts at the bottom

As shown on table 1 after the optimization the size of the webpage is reduced by 13.5% for initial load and up to 99.7% while reloading. The cause of the small reduce of the initial size of the pages is the heavy usage of images that represent a significant part of the webpage size and cannot be compressed.

The much lower traffic when reloading the page is due to the heavy use of caching for the resources on the webpage. In this case all resources except the HTML page are cached, and the page itself is served after GZip compression.

TABLE I. FRONT END OPTIMIZATION EFFECT

	First Load		**Reload**	
	Size(KB)	***Loading time(s)***	***Size (KB)***	***Loading time(s)***
Before	790.5	1.82	790.5	1.82
After	683.5	1.28	1.1	1.24
Difference (%)	13,5	29,7	99,7	31,9

The loading time of the page is reduced by 29.7% for the initial load and by 31.9% for the consecutive ones. This decrease shows the effect of the reduced size of the page and mainly the reduced number of requests made for loading external resources. The small difference in the loading time between the initial and the consecutive load results from the fact that the major part of the loading time of the page is spent while processing and executing scripts and not downloading resources.

The response time of the webserver before and after the optimization can be easily measured with the Apache JMeter tool. The test settings used for the test are as follows:

- Test duration: 600 seconds
- Number of threads: 50
- Ramp-up period: 50 seconds

The number of threads means the total number of concurrent users of the webpage and the ramp-up period is the time for starting all threads. The usage of the ramp-up period is for avoiding the extreme load of the webserver at the beginning of the test.

The test case is also set to imitate browser behavior i.e. to use the caching mechanisms used by the browsers and to reuse the connections with “keep-alive”. The results of the test are shown in table 2.

TABLE II. SERVER RESPONSE TIME

	Number of samples	Response time (ms)		
		Average	*Minimal*	*Maximal*
1	10041	33	9	10091
2	10049	0	0	159

Row 1 of the table shows the data for the page before optimization and row 2 after the optimization process. The 0 values for the response time after the optimization are result of using caching techniques for the static resources. This way only the main page is loaded on consecutive loads and the time for its retrieval from the server is close to zero due to the fact that the testing tool and the server reside on the same machine.

In production environments, users are rarely in the same network with the hosting servers, so the time required to download the HTML page will increase, but still it will be much lower if the static resources are cached.

Although for sites and applications in production caching all of the static resources required by the webpages is not always possible the test results still show a valid example of the effect of the optimization. The much lower response time is a strong indicator of the reduced server load and proves the two-side effect of the front end optimization.

5. CONCLUSION

Undoubtedly in the last two decades the web has evolved from a small system for information exchange between scientists to a network connecting billion of users. In the context of this evolution the loading speed of the web pages has become a major factor for the success of websites and applications.

The methods presented in this paper cover the basic and most effective steps for front-end optimization. Their effect is mostly based on the reduced traffic as a result of the usage of compression and resource caching.

As stated above caching all resources is rarely possible but still a major improvement of the bandwidth when navigating between the pages of the website. This, combined with the correct placement of styles and scripts, allows the browser to start rendering the page earlier than for non-optimized pages, giving the user a faster response and progressive loading which improves the user experience and the satisfaction from the usage of the site. In addition to the effect on the browser the server load is reduced due to the lower number of connections and traffic.

The wide usage of mobile devices not only opens additional opportunities for bandwidth and loading speed optimization but also requires it due to the lower connection speed and more expensive bandwidth for mobile traffic.

To summarize these methods improve both the response time of the webpage and the loading speed and also reduce the web server load. Having in mind the fact that the number of internet users is constantly increasing applying these techniques will allow website owners to serve more users without changes of the existing hosting equipment.

The reduced number of requests lowers the load of the server processors so the main load remains on the storage. The reduced expenses for bandwidth and processing power can be invested in faster storage – SSD disks and/or more RAM for even faster resource delivery.

Finally, on the business side this improvement means lowering the expenses for servers and bandwidth, or keeping them the same while increasing the number of users. Even in the case of using a CDN which may add an additional cost to the expenses, the result of increased user satisfaction, respectively conversion rates will diminish the increased expenses.

We are yet to see the new opportunities for optimization that technologies as SPDY[6] and the second version of the HTTP protocol – HTTP 2.0[7] will provide in the area of webpage optimization but the path is clearly set. The effort is focused in the creation of better transport protocols, providing lower latency, better compression and reduced page load time.

REFERENCES

1. Websiteoptimization.com, http://www.websiteoptimization.com/speed/tweak/average-web-page/
2. HTTP Archive : http://httparchive.org/compare.php
3. Nielsen, Jakob, http://www.nngroup.com/articles/law-of-bandwidth/
4. Souders, Steve, High Performance Websites, O'Reilly Media 2007
5. Souders, Steve, The Performance Golden Rule, http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/
6. SPDY: An experimental protocol for a faster web http://dev.chromium.org/spdy/spdy-whitepaper
7. Hypertext Transfer Protocol Bis - https://datatracker.ietf.org/wg/httpbis/charter/