Improved Speed on Intelligent Web Sites

Article · January 2013

CITATIONS
8

READS
605

4 authors, including:

Zsolt Nagy
University Of Nyíregyháza
6 PUBLICATIONS **17** CITATIONS

SEE PROFILE

Improved Speed on Intelligent Web Sites

ZSOLT NAGY
Institute of Mathematics and Computer Science
College of Nyiregyhaza
Nyiregyhaza, Sostoi u. 31/B
HUNGARY
info@nagyzsolt.hu

Abstract: - Intelligent web systems collect, process and serve information and knowledge to users, however the form and the speed of it definitely does matter. Based on statistical reports 57 percent of users abandon web sites if the page does not load in 3 seconds, therefore it is a key question to display information produced by profound AI algorithms in time. This paper investigates the best practises of web system speed optimization, collect and implement the rules and recommendations which can speed up our intelligent web systems. This article tests and measures the top web portals and through a real optimization process presents the efficiency of the discussed technologies.
.

Keywords: - web optimization, high performance web, page speed, intelligent web

1 Introduction

Significant segment of today's research work on the improvement of intelligent web systems, development of recommender systems enjoys focused attention and priority. It is comprehensible as the available information on the web is continuously growing thus only web services that can serve personalized content will remain on feet in the future [1]. Numerous excellent articles discuss about content-based [3], collaborative [2] or knowledge-based [4] filtering, researching ontology [5] [12] and knowledge-base systems is a separate field of science, although it is not enough now. The spreading of mobile devices and the age of Internet of Things requires a device dependent content service and responsive web site [6] development. No matter how brilliant is the mathematical model, how effective is the AI algorithm behind the scenes, if it can not provide and display information on the website in a given time, we loose our visitors. After waiting 3 seconds for loading web pages 57 percent of online consumers will abandon a site. 80 percent of these people will not return and almost half of them go on to tell others about their negative experience (Fig. 1) [7].
Based on Google and Microsoft engineers' experience New York Times expounded that people will visit a Web site less often if it is slower than a close competitor by more than 250 milliseconds [8].

Fig. 1 Online consumer behaviour

Making it worse only 1 second delay in page load time produces 7 percent loss in conversions, 11 percent fewer page views and 16 percent decrease in customer satisfaction [9]. Due to slow loading speeds, online retailers lose an estimated £1.73 billion in global sales each year, according to new research from online customer data platform QuBit

ISBN: 978-960-474-311-7

[10].
On top of all, an average user perceives page load time as being about 15 percent slower than actual page load time. When sharing their experience to others, they will recall that the page was 35 percent slower than it actually was [11].
It is a cardinal problem as all the intelligent web services, recommender systems were born to serve eCommerce, increase visitor number, profit and revenue. Naturally intelligent systems are playing important role in education, research and also in several areas of science. Therefore web pages must be faster, smarter, web developers and performance optimizers have to tie down the page load time under the critical 3 seconds [13]. In addition Google recognized the importance of webpage speed and included a new site speed signal in their search ranking algorithms [14].

2 Optimization techniques

This article has collected several recommendation and rules in order to make web pages faster. Beyond our optimization experience, this paper applies the suggestions of Google Developer (G) [15], Yahoo YSlow (Y) [16], Soulder's great books about High Performance Web Sites (H) [17] and Even Faster Web Sites (E) [18]. We organized, categorized and summarized these rules and chose the most recommended ones to optimize our web portal (Table 1).

	Best practices from			
Best practices	**Y**	**G**	**H**	**E**
Add Expires or Cache-Control Header	x		x	x
Avoid 404s / bad requests	x	x		
Avoid CSS @import	x	x		
Avoid CSS Expressions	x	x	x	x
Avoid document.write		x		
Avoid Empty Image src	x			
Avoid Filters	x			
Avoid, minimize iframes	x			x
Avoid, minimize redirects	x	x	x	x
Combine external CSS and JavaScript		x		
Combine images using CSS sprites	x	x		x
Configure ETags	x		x	x
Defer / split JavaScript payload		x		x
Defer parsing of JavaScript		x		
Develop Smart Event Handlers	x			
Do Not Scale Images in HTML	x	x		
Don’t block the UI thread				x
Enable Gzip compression	x	x	x	x
Flush buffer / document early	x			x
Inline scripts before stylesheet				x
Keep Components Under 25 KB	x			
Leverage browser caching		x		
Leverage proxy caching		x		
Load scripts asynchronously				x
Make Ajax Cacheable	x		x	x
Make favicon.ico Small and Cacheable	x			
Make Fewer HTTP Requests	x		x	x
Make JavaScript and CSS External	x		x	x
Make landing page redirects cacheable		x		
Minify HTML		x		
Minify JavaScript and CSS	x	x	x	x
Minimize DOM Access	x			
Minimize request size		x		
Minimize uncompressed size				x
Optimize images	x	x		x
Pack Components Into a Multipart Document	x			
Parallelize downloads across domains	x	x		x
Postload Components	x			
Prefer asynchronous resources		x		
Preload Components	x			
Put Scripts at Bottom	x	x	x	x
Put Stylesheets at Top	x	x	x	x
Reduce Cookie Size	x			
Reduce DNS Lookups	x	x	x	x
Reduce the Number of DOM Elements	x			
Remove Duplicate Scripts	x		x	x
Remove unused CSS		x		
Serve resources from a consistent URL		x		
Serve static content from a cookieless domain	x	x		
Simplify and use efficient CSS selectors		x		x
Specify a character set		x		
Specify image dimensions		x		
Use a Content Delivery Network (CDN)	x		x	x
Use GET for Ajax Requests	x			
Write efficient JavaScript				x

Table 1: Best Practises

2.1 Analyzer tools

There are several useful collections about free website speed testing tools; one of them is collected by Sixrevisions.com. They collected the top 20 testing tools [19]; we have tried them and selected 5 for our optimization purpose namely: Google's PageSpeed Insights [20], Yahoo's Yslow [21], AOL's WebPageTest [22], GTMetrix [23] and Pingdom [24].

ISBN: 978-960-474-311-7

The two major tools are PageSpeed and Yslow; the others are based on them and on their manufacturer's recommendations discussed in the previous section. WebPageTest and Pingdom are built up from a subset of Google's rules; GTMetrix uses PageSpeed and Yslow best practises. In our research we chose GTMetrix which combines the advices of the two major and additionally gives ergonomic and smart user interface with detailed reporting possibilities.

2.2 Preparation

In order to present a wide overview about the performance states of the most visited websites we measured statistics of the top 5 most visited websites in the following categories: top 5 websites of the world, top 5 of travel category and top 5 of Hungarian travel web sites. Our top selections were made in June 2013 based on Alexa's ranking (Table 2) [25].

By visitors	PageSpeed	Yslow	Average
Top 5 USA sites			
facebook.com	98	97	97,5
google.com	98	94	96
youtube.com	95	86	90,5
yahoo.com	89	77	83
amazon.com	94	82	88
Average	**94,8**	**87,2**	**91**
Top 5 Travel sites			
booking.com	93	82	87,5
tripadvisor.com	94	92	93
xe.com	89	73	81
expedia.com	81	60	70,5
priceline.com	88	78	83
Average	**89**	**77**	**83**
Top 5 Hungarian Travel sites			
itthon.hu*	90	78	84
iranymagyarorszag.hu	83	77	80
limba.com	87	83	85
utisugo.hu	86	69	77,5
szallasinfo.hu	76	72	74
Average	**84,4**	**75,8**	**80,1**

*we have changed the most visited szallas.hu as it was unable to be measured by analysers because of some redirection issues. Itthon.hu is maintaned by the state Hungarian Tourism Plc.

Table 2: Grades of the most visited websites

Our sixth Hungarian travel site is BelfoldiSzallasok.hu [26] - which one is 12th in the Hungarian travel category- will be the object of our optimization research.

Merging categories into one descending list we can identify that our website reached the poor 12th place in the overall table (Table 3).

#	Overall before	PageSpeed	Yslow	Average
1	facebook.com	98	97	97,5
2	google.com	98	94	96
3	tripadvisor.com	94	92	93
4	youtube.com	95	86	90,5
5	amazon.com	94	82	88
6	booking.com	93	82	87,5
7	limba.com	87	83	85
8	itthon.hu	90	78	84
9	yahoo.com	89	77	83
10	priceline.com	88	78	83
11	xe.com	89	73	81
12	*belfoldiszallasok.hu*	*86*	*76*	*81*
13	iranymagyarorszag.hu	83	77	80
14	utisugo.hu	86	69	77,5
15	szallasinfo.hu	76	72	74
16	expedia.com	81	60	70,5

Table 3: Overall grades before optimization

2.3 Suggested tasks

Using GTMetrix we have received and selected the following rules to make our travel portal reaching better performance. We left the original website untouched, made a separate subdirectory instead to apply all the following performance practises.

2.3.1 Specify image dimensions

Specifying a width and height for all images allows for faster rendering by eliminating the need for unnecessary reflows and repaints. When the browser lays out the page, it needs to be able to flow around replaceable elements such as images. It can begin to render a page even before images are downloaded, provided that it knows the dimensions to wrap non-replaceable elements around. If no dimensions are specified in the containing document, or if the dimensions specified do not match those of the actual images, the browser will require a reflow and repaint once the images are downloaded. To prevent reflows, specify the width and height of all images, either in the HTML `<img>` tag, or in CSS.

2.3.2 Leverage browser caching

Web page designs are getting richer and richer, which means more scripts, stylesheets, images, and Flash in the page. Browsers use a cache to reduce the number and size of HTTP requests, making web pages load faster. A web server uses the Expires header in the HTTP response to tell the client how long a component can be cached. Setting an expiry date or a maximum age in the HTTP headers for

ISBN: 978-960-474-311-7

static resources instructs the browser to load previously downloaded resources from local disk rather than over the network. Set Expires to a minimum of one month, and preferably up to one year, in the future. (Google prefer `Expires` over `Cache-Control: max-age` because it is more widely supported.)

Expires headers are most often used with images, but they should be used on all components including scripts, stylesheets, and Flash components.

Based on GTMetrix suggestion we have created an *.htaccess* file with the following content:

```
<IfModule mod_expires.c>
# Enable expirations
ExpiresActive On
# Default directive
ExpiresDefault "access plus 1 month"
# My favicon
ExpiresByType image/x-icon "access plus 1  year"
# Images
ExpiresByType image/gif "access plus 1 month"
ExpiresByType image/png "access plus 1 month"
ExpiresByType image/jpg "access plus 1 month"
ExpiresByType image/jpeg "access plus 1 month"
# CSS
ExpiresByType text/css "access 1 month"
# Javascript
ExpiresByType   application/javascript   "access
plus 1 year"
</IfModule>
```

As it is shown in the first line, this technique requires *mod_expires* module configured and we have to make sure *.htaccess* file is allowed to process by our Apache web server.

2.3.3 Combine images into CSS sprites

Combining images into as few files as possible using CSS sprites reduces the number of delays in downloading other resources, reduces request overhead, and can reduce the total number of bytes downloaded by a web page. Similar to JavaScript and CSS, downloading multiple images incurs additional request-round. A site that contains many images can combine them into fewer output files to reduce latency. Spriting services such as CssSprites [31], SpriteMe [32] or SpritePad [33] can make it easier to build CSS sprites. SpritePad is free, beautiful and an easy-to-use tool; after combining all the images we can easily download both the CSS and the created PNG file.

2.3.4 Minify CSS

Compacting CSS code can save many bytes of data and speed up downloading, parsing, and execution time. Minifying CSS has the same benefits as those for minifying JavaScript: reducing network latency, enhancing compression and faster browser loading and execution. Several tools are freely available to minify JavaScript, including the YUI Compressor [30] or even QTMatrix can offer the minimized version of our CSS file. We can easily save it and rewrite our existing CSS file.

2.3.5 Use a Content Delivery Network

A content delivery network (CDN) is a collection of web servers distributed across multiple locations to deliver content more efficiently to users. Deploying web content across multiple, geographically dispersed servers will make our pages load faster from the user's perspective. The server selected for delivering content to a specific user is typically based on a measure of network proximity. For example, the server with the fewest network hops or the server with the quickest response time is chosen. Generally these are charged services; Techyfuzz [27] collected and offer us 5 free CDN networks. We have tried two of them, however at the final optimization phase we have skipped using CDN. The simple reason is CDNs require modifying DNS records of our web domain and we do not want to meet this requirement at this moment.

2.3.6 Use Cookie-free Domains for Components

When the browser makes a request for a static image and sends cookies together with the request, the server does not have any use for those cookies. So they create unnecessary. We should make sure static components are requested with cookie-free requests. It is advised to create a subdomain and host all static components there. We have created *static.belfoldiszallasok.hu* subdomain to serve all the static images from.

2.3.7 Compress components with Gzip

Compressing resources can reduce the number of bytes sent over the network. Most modern browsers support data compression for HTML, CSS, and JavaScript files. This allows content to be sent over the network in more compact form and can result in a dramatic reduction in download time.

To enable compression, we should configure our web server to set the `Content-Encoding` header to gzip [34] or deflate [35] format for all compressible resources. The *.htaccess* file gives the possibility again to configure our web server for compressing resources, it requires the *mod_deflate* module. Our advice is to use `<IfModule mod_deflate.c>` condition in order to check the existence of a module and avoid Internal Server Error message.

```
<IfModule mod_deflate.c>
#compress text, HTML, JavaScript, CSS, and XML
```

```
   AddOutputFilterByType DEFLATE text/plain
   AddOutputFilterByType DEFLATE text/html
   AddOutputFilterByType DEFLATE text/xml
   AddOutputFilterByType DEFLATE text/css
   AddOutputFilterByType DEFLATE application/xml
   AddOutputFilterByType DEFLATE
application/xhtml+xml
   AddOutputFilterByType DEFLATE
application/rss+xml
   AddOutputFilterByType DEFLATE
application/javascript
   AddOutputFilterByType DEFLATE application/x-
javascript
   </IfModule>
```

Due to the overhead and latency of compression and decompression, we should only gzip files above a certain size threshold; Google Developer suggests a minimum range between 150 and 1000 bytes.

2.3.8 Combine external JavaScript

Combining external scripts into as few files as possible cuts down delays in downloading other resources. Good front-end developers build web applications in modular, reusable components. While partitioning code into modular software components is a good engineering practice, importing modules into an HTML page one at a time can drastically increase page load time. First, for clients with an empty cache, the browser must issue an HTTP request for each resource, and incur the associated round trip times. Secondly, most browsers prevent the rest of the page from being loaded while a JavaScript file is being downloaded and parsed.

2.3.9 Minify JavaScript

Compacting JavaScript code can save many bytes of data and speed up downloading, parsing, and execution time. Minifying or reducing code refers to eliminating unnecessary bytes, such as extra spaces, line breaks, and indentation. Keeping JavaScript code compact has a number of benefits. First, for inline JavaScript and external files that we do not want cached, the smaller file size reduces the network latency incurred each time the page is downloaded. Secondly, minification can further enhance compression of external JavaScript files and of HTML files in which the JavaScript code is inlined. Thirdly, smaller files can be loaded and run more quickly by web browsers.

Several tools are freely available to minify JavaScript, including JSCompress [28], JSMini [29] or YUI Compressor [30]. We can create a build process that uses these tools to minify and rename the development files and save them to a production directory. Google recommends minifying any JavaScript files that are 4096 bytes or larger in size. We should see a benefit for any file that can be reduced by 25 bytes or more (less than this will not result in any appreciable performance gain).

3 Final results

After completing the suggested optimization tasks we identified that total number of HTTP requests decreased almost with 35 percent, the total page size to 32% of the original and as Fig. 3 shows both the PageSpeed and Yslow grades were improved.

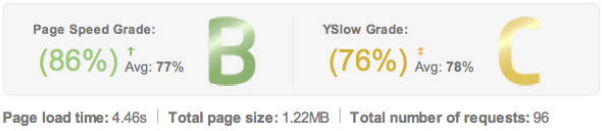

Fig. 2 Before optimization

Fig. 3 After optimization

As a result of this optimization process our website not only qualified as a second best optimized travel database in the Hungarian market but had raised above the average of the top travel sites of the world. It could overtake pages like Yahoo or Priceline; improve its result from the overall 12th position to the 8th place (Table 4).

#	Overall before	PageSpeed	Yslow	Average
1	facebook.com	98	97	97,5
2	google.com	98	94	96
3	tripadvisor.com	94	92	93
4	youtube.com	95	86	90,5
5	amazon.com	94	82	88
6	booking.com	93	82	87,5
7	limba.com	87	83	85
8	***belfoldiszallasok.hu***	***88***	***81***	***84,5***
9	itthon.hu	90	78	84
10	yahoo.com	89	77	83
11	priceline.com	88	78	83
12	xe.com	89	73	81
13	iranymagyarorszag.hu	83	77	80
14	utisugo.hu	86	69	77,5
15	szallasinfo.hu	76	72	74
16	expedia.com	81	60	70,5

Table 4: Grades after optimization

ISBN: 978-960-474-311-7

4 Conclusions

As we can clearly conclude from this research, webpage speed is going to be one of the most important factors in the world of online business. Every second page delay could potentially cost millions in lost sales every year for online companies, improving speed performance tend to become a significant factor in business life. After optimizing browser rendering and network transfer, our next task should be to dig deeper down into client and server-side code (AJAX [36], PHP, ASP) and SQL database optimization. Completing and combining all known issues could result an effective and valuable web application either on desktop or mobile devices.

References

[1] Rossi, G., Schwabe, D., Guimarães, R., *Designing personalized web applications.* In Proceedings of the 10th international conference on World Wide Web, ACM, 2001, pp. 275-284.

[2] X. Su and T. M. Khoshgoftaar, *A Survey of Collaborative Filtering Techniques*. Advances in Artificial Intelligence, vol. 2009, Nr. 4, 2009.

[3] M. J. Pazzani and D. Billsus., *Content-Based Recommendation Systems*. The Adaptive Web, Lecture Notes In Computer Science, Vol. 4321, 2007, pp 325–341

[4] D. Dell'Aglio, I. Celino, D. Cerizza:, *Anatomy of a Semantic Web-enabled Knowledge- based Recommender System*, http://www.larkc.eu, 2010

[5] Gašević, D., Djurić, D., Devedzic, V., *Model driven engineering and ontology development*. Springerverlag Berlin Heidelberg, 2009

[6] Marcotte, E. , *Responsive web design*. A List Apart, 306., 2010, http://alistapart.com/article/responsive-web-design

[7] StrangeLoop Networks, *Website abandonment happens after 3 seconds*, http://www.strangeloopnetworks.com/resources/infographics/web-performance-and-user-expectations/website-abandonment-happens-after-3-seconds/

[8] Lohr, S., *Impatient web users flee slow loading sites*, New York Times, 2012 http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html

[9] Kissmetrics.com, *Loading time*, http://blog.kissmetrics.com/loading-time/

[10] Ecoconsultancy.com, *Slow loading websites cost retailers 1,73 bn in lost sales each year*, http://econsultancy.com/hu/blog/9790-slow-loading-websites-cost-retailers-1-73bn-in-lost-sales-each-year

[11] StrangeLoop Networks, I*nternet users have faulty perceptions of time*, ttp://www.strangeloopnetworks.com/resources/infographics/web-performance-and-user-expectations/internet-users-have-faulty-perceptions-of-time/

[12] Iordan, V; Naaji, A; Cicortas, Al , *Deriving Ontologies Using Multi-agent Systems*, "Wseas Transaction on Computers", Volume 7, Issue 6 (June 2008), ISSN: 1109-2750, 2008, p. 814-826.

[13] StrangeLoop Networks, *Visualizing web performance*, http://www.strangeloopnetworks.com/assets/images/visualizing_web_performance_poster.jpg

[14] Blogspot, *Using site speed in web search ranking*, http://googlewebmastercentral.blogspot.hu/2010/04/using-site-speed-in-web-search-ranking.html

[15] Google Developer, *Make the web faster*, https://developers.google.com/speed/

[16] Yahoo, Exceptional Performance, http://developer.yahoo.com/performance/

[17] Souders, S., *High Performance Web Sites*. O'Reilly Media, 2007

[18] Souders, S., *Even faster web sites: performance best practices for web developers*. O'Reilly Media, 2009

[19] Sixrevisions.com, *Free website speed testing*, http://sixrevisions.com/tools/free-website-speed-testing/

[20] PageSpeed Insights, https://developers.google.com/speed/pagespeed/

[21] Yslow, http://developer.yahoo.com/yslow/

[22] WebPageTest, http://www.webpagetest.org

[23] GTMetrix, http://gtmetrix.com

[24] Pingdom, http://tools.pingdom.com/fpt/

[25] Alexa Top Sites, http://www.alexa.com/topsites

[26] Belfoldiszallasok.hu, http://www.belfoldiszallasok.hu

[27] Techyfuzz.com, http://techyfuzz.com/free-cdn-content-delivery-network-services-website/

[28] JSCompress, http://www.jscompress.com

[29] JSMini, http://www.jsmini.com

[30] YUI Compressor, http://refresh-sf.com/yui/

[31] CSSSprites, http://www.csssprites.com

[32] SpriteMe, http://www.spriteme.org

[33] SpritePad, http://wearekiss.com/spritepad

[34] Gzip, http://www.ietf.org/rfc/rfc1951.txt

[35] Deflate, http://www.ietf.org/rfc/rfc1951.txt

[36] Zs, Nagy, Ajax-Based Data Collection Method for Recommender Systems*, Proceedings of the 16th WSEAS International Conference on Computers*, 2012. pp. 446–451

ISBN: 978-960-474-311-7