IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FACTORS AFFECTING BROWSER PERFORMANCE FOR WEBSITES

Aayushman Sharma[1] , Mrs. Namita Goyal[2]
[1]Student, [2]Assistant Professor
[1]Information Technology,
[1]Maharaja Agrasen Institute of Technology, Rohini, India

Abstract: This paper analyses trends in page loading time which is an important part of any website's user experience. Many times we'll let it slide to accommodate better aesthetic design, new nifty functionality or to add more content to web pages. Unfortunately, website visitors tend to care more about speed than all the bells and whistles we want to add to our websites. Additionally, page loading time is becoming a more important factor when it comes to search engine rankings.

Index Terms **- Container object (CO), External objects (EOs), Websites, html**

I. INTRODUCTION

Web performance from the client point of view is measured as the page load time. This is the lapsed time between the moment a user requests a new page and the moment the page is fully rendered by the browser. Fast web pages render progressively. That is, they display their content incrementally, as it is loaded by the browser. A web page that renders progressively gives the user visual feedback that the page is loading, and gives the user the information they requested as soon as it is available. Google® and Yahoo® both suggest best practices to make web pages render progressively, such as putting style sheets in the document head.

There are several additional best practices that you can apply to optimize progressive rendering for most pages. First, a fast page should render the content that is visible to the user first, and render the off-screen content (that is, the content outside of the current scroll region) later. Second, a fast page might also load and render the lightweight resources such as text before loading and rendering heavyweight resources like images and video. On the other hand, some techniques are known to inhibit progressive rendering. The use of large tables, even for layout, disables progressive rendering in some browsers. Applying style sheets late in the document, even if those style sheets aren't needed for the initial page load, can also prevent progressive rendering.

Everybody just hates those loading time when opening a website. And with Google® using the loading speed of a website as a factor to decide the rank of a website, it is high time for webmasters to get serious about optimizing their websites for speedy access.

Google® recently announced that they consider website speed when determining search engine rankings.

The purpose of the study was to test each web page in a given website so to minimize loading time for that particular website. This paper measures and analyses the Webpages based on their anatomy.

A web page is made up of a **container object** (CO) and **external objects** (EOs). The CO is usually an XHTML file that references EOs such as images, audio, video, and external CSS and JavaScript files. Most non-textual EOs are usually pre-compressed, so you'll only see the benefits of HTTP compression on XHTML and on some CSS and JavaScript files. For more than 60% of web pages, the CO occupies less than 50% of total page size. The average CO takes up about 44% of total page size (Yuan, Li, & Chi 2005 ©).
Therefore, no matter how efficient our XHTML optimization and HTTP compression, the greatest improvement in web page performance that we can expect from XHTML optimization will be less than 50%. For a 300K home page, on average, we'll still

need to download at least 150K of EOs, even after optimizing and compressing our textual data. We can see why it is important to optimize our entire web page, including multimedia, in order to make significant performance gains.

High performance web sites lead to higher visitor engagement, retention and conversions.

Web application performance is affected by many factors, including things like the size of request and response data, the speed of database queries, how many requests the server can queue and how quickly it can service them, and even the efficiency of any client-script libraries we might be using. If performance is critical in an application, or if testing or experience shows that application performance isn't satisfactory, we should follow normal protocol for performance tuning. Measure to determine where performance bottlenecks are occurring, and then address the areas that will have the greatest impact on overall application performance.

II. WEB METRICS: SIZE AND NUMBER OF RESOURCES

Here are some statistics about the size, number of resources and other such metrics of pages on the World Wide Web. These are collected from a sample of several billions of pages that are processed as part of Google's crawl and indexing pipeline. In processing these pages, we not only take into account the main HTML of the page, but discover and process all embedded resources such as images, scripts and style sheets.

Highlights:

- The average web page takes up **320 KB** on the wire.
- Only **two-thirds** of the compressible material on a page is actually compressed.
- In 80% of pages, **10 or more resources** are loaded from a single host.
- The most popular sites could eliminate more than **8 HTTP requests** per page if they combined all scripts on the same host into one and all style sheets on the same host into one.

Statistics about the size and number of resources of Webpages on the World Wide Web.

Metric	Top sites	All sites	Description
Pages	380 million	4.2 billion	Number of sample pages analyzed.
Resources	42.14	43.91	Average number of resources per page.
GETs	42.63	44.56	Average number of GETs per page. Similar to number of resources, but also includes redirects.
Hosts	8.39	7.01	Average number of unique hostnames encountered per page.
Resources Per Host	5.02	6.26	Average number of resources per host (derived from the 'Resources' and 'Hosts' values).
Network Size/KB	312.04	320.24	Average size transferred over the network per page, including HTTP headers. If resources were compressed, this would use the compressed size.
Document Size/KB	477.26	376.67	Average uncompressedsize of a page and its resources, excluding HTTP headers.
Zip able Size/KB	287.51	170.16	Average uncompressed size of the compressible resources on a page, i.e., those with a Content-Type of 'text/*' orequivalent.
Unzipped Size/KB	32.67	57.07	Average size of the compressible resources that were not sent compressed, i.e., the Content-Type was 'text/*', but Content-Encoding did not include 'gzip' or 'deflate'.
Zipped Ratio	89%	66%	Average percentage of compressible bytes that were actually compressed (derived from the 'Zippable' and 'Unzipped' values).
Images	27.58	29.39	Average number of unique images per page.
Image Size/KB	184.73	205.99	Average network size of the images per page.
Scripts	6.75	7.09	Average number of external scripts per page.
Script Size/KB	66.48	57.98	Average network size of the external scripts per page.
Combinable Scripts	4.75	3.75	Average number of requests that could be saved per page if external scripts on the same host were combined.
Stylesheets	4.07	3.22	Average number of external stylesheets per page.
Stylesheet Size/KB	27.17	18.72	Average network size of the external stylesheets per page.
Combinable Stylesheets	3.54	2.02	Average number of requests that could be saved per page if external stylesheets on the same host were combined.
SSL Pages	650	17 million	Number of sample SSL (HTTPS) pages analyzed.

.

III. SOLUTIONS TO IMPROVE WEBSITE PERFORMANCE

The web site performance depends on a variety of factors such as content on your pages, browser, geographic location of access, bandwidth, etc. It is often possible to make the contents of a web page take fewer bytes without changing the appearance or function of the page. Reducing the number of bytes a client has to download makes the page load faster. There are many ways to make websites run faster

A. REDUCING THE FILE SIZE OF HTML DOCUMENTS

A clear way to improve the load time of your site is to decrease the file size of your HTML documents. There are several ways to do this, from rigid compression to acupuncture-like ID and class name changes.

HTML – as opposed to XHTML, even when delivered with the MIME type text/html – allows authors to omit certain tags. According to the HTML 4 DTD, you can omit the following.

</area> </base><body> </body> </br></col>
</colgroup></dd></dt><head></head></hr>
<html></html></img></input>
</link></meta></opti on> </param> <tbody> </tbody></td> </tfoot>
</th></thead></tr>

For example, if you have a list of items marked up as <li> List item , you could instead just write <li> List item. Or instead of a paragraph that you'd usually close with , you could just use My paragraph. This even works with html, head, and body, which are not required in HTML. Omitting optional tags keeps your HTML formally valid, while decreasing your file size and making your code look much leaner. In a typical document, this can mean 5-20 % savings.

B. PROPERLY INCLUDING STYLESHEETS AND SCRIPTS

Typical web pages spend 80-90% of their load time waiting for the network. A powerful technique to reduce the amount time spent blocked on the network is to get rid of patterns that cause some browsers to serialize resource downloads.

1. ***Combine external JavaScript files***

Downloading an external script file is somewhat unique because it causes popular browsers to block subsequent downloads until the script has completed. This is in stark contrast to downloading images which may occur in parallel (up to a limit).

2. ***Include external CSS files before external JavaScript files***

While script files block subsequent downloads, those already in progress will remain downloading. So, if you have an external script and CSS file, always include the CSS file before the script so that they will download in parallel.

3. ***Do not include inline JavaScript between external CSS and other resources***

Inline script tags, even though they don't download anything, will prevent subsequent resources from downloading in parallel with a CSS file. So, if you have an external CSS file be sure not to insert inline script tags between your CSS file and the next downloadable resource.

C. COMPRESS IMAGES

Image files are often created with extra information embedded in the file. For example, JPEG files written by many image programs include the name of the program that wrote them. PNG images can often be made smaller by changing the way the image is encoded. These transformations are lossless. That is, the compressed image looks identical to the uncompressed image, but uses fewer bytes.

D. MINIFY JAVASCRIPT

Removing comments and white space from large JavaScript files can make them substantially smaller, without changing their functionality.

E. REMOVE UNUSED CSS

CSS files contain rules that apply style attributes to elements in a web page. If a rule does not apply to any element in a page, removing it will result in fewer bytes being sent to the client, with no change in the appearance of the web page. However, because external style sheets may be included by more than one page, you must be careful to only remove rules that no page uses.

F. HTTP CACHING

When you set the correct HTTP caching headers, you get a double win because revisits to your web pages load faster and there is less load on your web server.

The cache, which is local copies of resources, works because many resources change infrequently. When a browser can reuse a local copy, it saves the time to set up a connection as well as the time to download. The key to making the cache work effectively is HTTP caching headers, which are sent by the web server to specify how long a resource is valid and when it last changed.

The HTTP protocol gives two ways to define how long a resource is valid: the Expires header and the Cache-Control: max-age header. The Expires header specifies a date after which a resource is invalid. At that point, the browser will ask for the resource again. max-age works much the same way but it specifies how long a resource is after it is downloaded instead of giving a specific date. That is nice because you can configure your web server with a constant value.

G. MINIMIZING BROWSER REFLOW

Reflow is the name of the web browser process for re- calculating the positions and geometries of elements in the document, for the purpose of re-rendering part or all of the document. Because reflow is a user-blocking operation in the browser, it is useful for developers to understand how to improve reflow time and also to understand the effects of various document properties (DOM depth, CSS rule efficiency, different types of style changes) on reflow time. Sometimes reflowing a single element in the document may require reflowing its parent elements and also any elements which follow it.

The best practices cover many of the steps involved in page load time, including resolving DNS names, setting up TCP connections, transmitting HTTP requests, downloading resources, fetching resources from cache, parsing and executing scripts, and rendering objects on the page. Essentially Page Speed evaluates how well your pages either eliminate these steps altogether, parallelize them, and shorten the time they take to complete. The best practices are grouped into six categories that cover different aspects of page load optimization:

i. ***Optimizing caching*** — keeping your application's data and logic *off* the network altogether
ii. ***Minimizing round-trip times*** — reducing the number of serial request-response cycles
iii. ***Minimizing request overhead*** — reducing upload size
iv. ***Minimizing payload size*** — reducing the size of responses, downloads, and cached pages
v. ***Optimizing browser rendering*** — improving the browser's layout of a page Optimizing for mobile — tuning a site for the characteristics of mobile networks and mobile devices

IV. CONCLUSION

Within the last five years, the size of the average web page has more than tripled, and the number of external objects has nearly doubled. While broadband users have experienced faster load times, narrowband users have been left behind. With the average web page sporting more than 50 external objects, object overhead now dominates most web page delays. Minimizing HTTP requests by using CSS sprites, combining JavaScript or CSS files, reducing the number of EOs, and converting graphic effects to CSS while still retaining attractiveness, has become the most important skill set for web performance optimizers. The data appears to suggest that the more popular a web page, the smaller the total file size.

V. REFERENCES

[1] Stave Souders, High performance Websites: Essential Knowledge for Front-End Engineers, O'Reilly Media. September 2007
[2] Bob Wescott, The Every Computer Performance Book.
[3] Microsoft Corporation, Performance Testing Guidance for Web Application.
[4] Andrew B. King, Jakob Nielsen, Speedup Your Site: Website Optimization.
[5] Scoot Ware, Michael Tracy, Louis Slothouber, Robert Barker, Professional Website optimization, 1998.
[6] Patrick Killelea, Web Performance Tunning, 1998.
[7] Daniel A. Menasce, Virgilio A. F. Almeida, Performance: Metrics, Models and Methods.