O3 - A Webpage Preprocessing Tool

Karthik Senthil, Karthik S. Bhat, Nitin Jamadagni, Sudeep Sureshan and Gaurav Prasad
Department of Information Technology, National Institute of Technology Karnataka, Surathkal, India

Keywords: Front-end, Optimization, Load Time, Performance, Optimization Methods, Preprocessing.

Abstract: One of the prime factors for the success of the internet is determined by the time taken to load a web page. Even a difference of a few hundred milliseconds in the response time will largely affect the number of users of a web page to shift from one to the other. So, in the commercial market, providing quick service to the users is of utmost importance in remaining ahead of competitors. In this paper, we mainly address this issue by applying various optimization techniques at the front-end to improve the user experience by reducing the load time of the web pages. Though the overall optimization is purely web page-dependent, the optimization techniques not only reduce the time taken to load the page, but also reduce the load on the server.

1 INTRODUCTION

Web based applications and websites are being visited more and more frequently in the recent times due to the ease of access to the internet. It has reached such an extent that there are many countries where there is an internet connection in every household. This has led to the increased number of web pages altogether, as organizations have created personal web pages as a marketing strategy, social networking sites are on the rise and so on. Website loading is an inherent part of the user experience in using a website.

At the browser level, when a website is requested for, the browser sends a HTTP request to the server and the server in turn sends the response in a HTTP response packet which is then received by the browser. The data is being represented in the form of a Document Object Model (DOM) which can be universally rendered by all browsers. The DOM is a tree structure of the contents of the webpage. Webpages generally consist of HTML, which is the markup, CSS which adds styling to the markup and client-sided JavaScript. During the rendering of the DOM, the browser loads the HTML and its associated CSS together, as they go hand in hand. Scripts are, however, rendered differently. Since scripts have the capacity to modify the DOM structure dynamically, the loading of the DOM is halted until all of them have finished loading. This results in an unnecessary loss of time, as many scripts are irrelevant to the user (like click counter or analytics). This is a drawback that has been widely discussed. In this paper we discuss some ways to improve the DOM loading speed and in essence to improve the loading time of webpages.

In this paper, section two discusses the motivation behind the creation of the tool. Section three discusses the previous work similar/related to this tool. Section five discusses the methodology followed during the implementation of the project. Section six (a), discusses in detail the techniques used to achieve the speed-up of loading of the web pages. Section six (b) shows the results we obtained and compare them to unprocessed websites and thereby shows how optimization effectively improves page-load time.

2 MOTIVATION

Large access frequency and increased user-friendliness of webpages has led to an increase in the amount of resources used by the web pages. The escalated number of page accesses, fuelled by the number of users, has forced the hosting agencies to add extra web servers to handle these requests. This has in turn led to increased expenditure for the organization. This rise in number of resources used by web pages not only multiplies load on the server, but also on the client who is connected to the site affecting the user experience while using the web page. User experience plays a critical role in page ranking, as a better experience leads to better rating for a web page and consequently, a higher page rank.

Steve Souders (Souders, 2012) suggests that 80-90% of the end-user response time is spent on the

Senthil K., S. Bhat K., Jamadagni N., Sureshan S. and Prasad G..
O3 - A Webpage Preprocessing Tool.
DOI: 10.5220/0005438100150020
In *Proceedings of the 11th International Conference on Web Information Systems and Technologies* (WEBIST-2015), pages 15-20
ISBN: 978-989-758-106-9

frontend. This claim has been supported by statistics, among the top 10 ranked websites 76% end-user time has been recorded on the front end,among the top 9990+ ranked websites a staggering 92% of the same and among the top 50000 ranked websites a 87% has been recorded.This seems to suggest that much of the optimization techniques must be concentrated on the front-end of websites.

According to surveys conducted by Akamai (Akamai, 2012), close to half of web users have a time limit expectation of about 2 seconds or less, and tend to abandon a site that hasn't loaded within 3 seconds.79% of the online shoppers who have trouble with their shopping websites tend to not visit the site again to buy products and 44% of them would tell a friend that he had a bad experience with the online shopping website.Seeing these statistics, it is a necessity for online shopping websites to optimize the web pages, which otherwise would lead to a loss of many potential sales.

The page-load time and bandwidth requirement of web pages can be reduced by applying various front-end optimization techniques. These techniques not only optimize the load-time on the client side, but also reduce the load on the web server, by decreasing the bandwidth requirement and the number of connected clients.

3 RELATED WORK

Some related work/tools in webpage optimization are:

1. Google PageSpeed
   PageSpeed (Zhenhua et al., 2012) is a set of tools designed to improve the performance of a website.It helps a developer in identifying high performance oriented practices that can be applied to the website. It also includes optimization tools that can help the developer automate this process.
2. Radware's FastView
   FastView (Radware, 2012) is a web performance optimization(WPO) and acceleration tool which permits better performance of a website or web-based application in real time. It has an automated functionality to reduce the complexity of the front end optimization process.
3. Akamai's Front End Optimization (FEO)
   FEO (Akamai, 2012) is a service offered by Akamai for the clients using their cloud computing service. It allows mobile optimization of web content primarily for smartphone and tablet users. The Akamai FEO engine analyses the target web content and then identifies the possible optimizations for different device capabilities and environments.
4. JCH Optimize
   JCH Optimize (Marshall, 2013) is a plugin of Joomla! that aims at speeding up a website by combining JavaScript and CSS files into a single file,hence minimizing expensive HTTP requests and ultimately optimizing the net download time of a webpage.
5. Iliev, I. et al.'s work (Iliev and Dimitrov, 2014) has also discussed various front end optimization techniques.

In comparison to the above mentioned tools, the O_3 web preprocessing tool performs a comprehensive evaluation and graphically represents the results obtained before and after the optimizations made by the tool.

4 PROBLEM STATEMENT

In the current era of high speed internet we observe that various front-end optimization techniques are prevalent and used, but these are being done manually by the web page designers/developers. This project aims at automating this time-consuming process thereby saving on development time while also implementing optimization techniques for the web page in parallel.

5 METHODOLOGY

The methodology adopted in this project is essentially the development of Python scripts to automate the optimization techniques on a given set of web pages, which we have found to give measurable outcomes.The observations and results obtained by us are specific to WEBrick server version 1.3.1. We have measured the speed-up of the web page's loading time, by using some performance-measuring tools, such as YSlow and ab Tool, as benchmarking tools for quantitative analysis.
We aim to achieve optimization using the following techniques:

1. Placing Scripts at the bottom of the page
2. Removing Duplicate Scripts
3. Using External Scripts
4. JavaScript Minification
5. CSS Minification

6. Using Compression techniques
7. Using CDNs(Content Delivery Networks)
8. Using Cache Techniques

6 IMPLEMENTATION

The following tools/libraries were used during the implementation of the above techniques:

1. BeautifulSoup : BeautifulSoup which is a package of Python, is used for HTML documents parsing. It builds a parse tree for web pages it parses, which can then be used to modify the DOM structure.
2. Requests : Requests is a simple, elegant HTTP library for Python licensed by Apache2.
3. YUICompressor : A Java based compression/minification tool.
4. ab Tool : ab is a tool for benchmarking Apache Hypertext Transfer Protocol (HTTP) server. It simulates any test case by taking number of requests and number of concurrent requests as input.

The optimization was performed using the following techniques:

1. **Placing Javascript at the bottom of the page**
   This technique involves the manipulation of the DOM structure built by the browser, before the rendering of the web page to the user. We find the node which has the script tag and extract it, simultaneously replacing it with an empty tag after its removal. We now traverse through the entire DOM tree again to find the last child node of the body tag and thereafter append the extracted script tag as the last child and complete the DOM structure, which is then finally parsed to display to the user the web page.
   The reason behind this approach is that while rendering a web page, if the browser comes across a script tag, the downloading of web page is paused until the entire script is fetched (as it can affect the DOM structure), thereby increasing the page load time. So, by moving the scripts to the end of the page we can ensure that the page download happens completely before the scripts are fetched, ultimately cutting down a significant portion of page load time.

2. **Moving script contents to an external JavaScript file**
   In our next technique we have removed all the script tags from the HTML page and moved them to a separate external Javascript file. The HTML file is parsed and the content inside script tags are removed and placed in a new file named ”external.js”. Consequently, this external.js is included in the head section of the HTML page. Also, we have ensured that the order of inclusion of other external JavaScript files is maintained.
   This is done to reduce effective HTML document length of the web page, thus reducing the page load time. This also leads to simple DOM tree structure without any script tags.

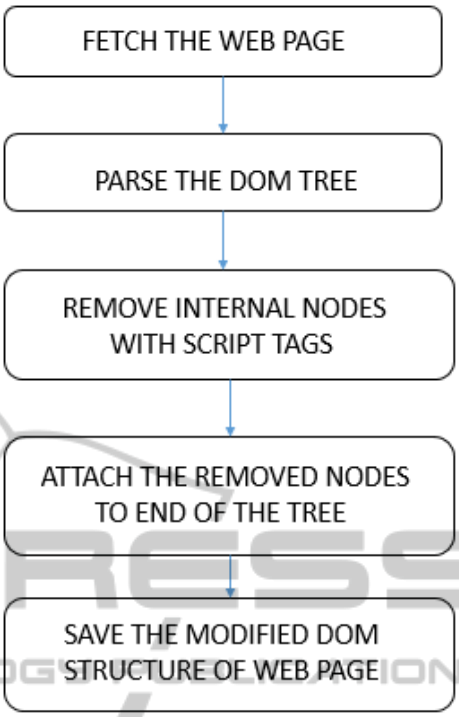

Figure 1: Placing scripts at the bottom of the page.

3. **CSS and JavaScript Minification**
   In our final technique we have made use of the concept of minification. Minification is the removal of all unnecessary characters from source code without changing its functionality. This is done to bring in some significant change in the file size(of the JS or CSS file).

Here we used a JAVA based command-line compression tool, namely YUICompressor. Among all the style-sheets and Javascript files that were fetched for a particular web page, we identified the un-minified files. These were consequently fed to the tool and the minified output was used to replace the unminifed files.

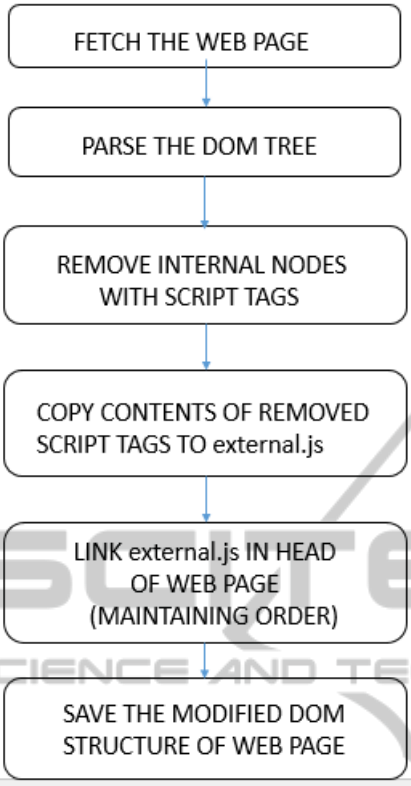

Figure 2: Placing scripts in external file.

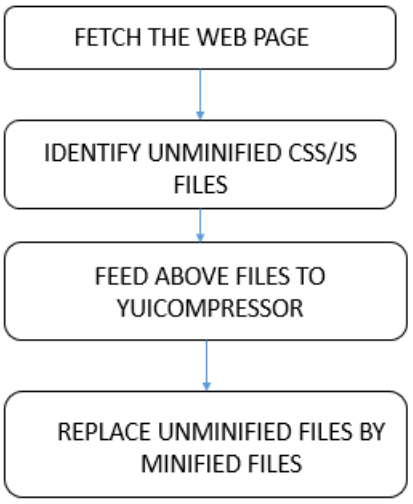

Figure 3: CSS and script minification.

Minification is proven to give an average of 25% efficiency in compression. We observed the same results with our techniques.

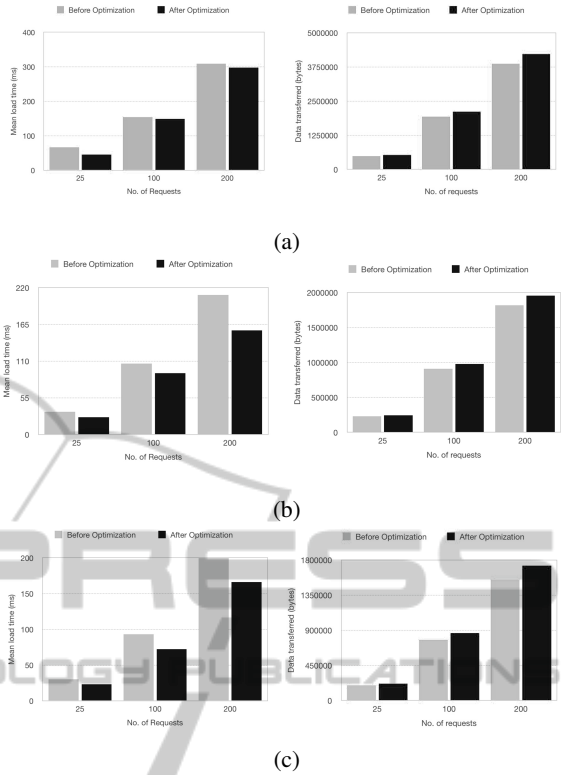

Figure 4: Results obtained **(a)** Technique 1 on google.com. **(b)** Technique 1 with JS on top. **(c)** Technique 1 with JS in the middle.

7 RESULTS OBTAINED AND ANALYSIS

The results obtained on moving the content of the script tags in google.com to the bottom of the page resulted in an improvement of 12.95% in Mean Load Time (MLT) and an increase of 9.3% in Data Transferred (Bytes) as shown in Figure 4(a).
The results obtained on moving the content of the script tags to the bottom of the page with JS on top resulted in an improvement of 12.95% in MLT and an increase of 9.3% in Data Transferred (Bytes) as shown in Figure 4(b).
The results obtained on moving the content of the script tags to the bottom of the page with JS in the middle resulted in an improvement of 20.97% in MLT and an increase of 11.37% in Data Transferred (Bytes) as shown in Figure 4(c).

It can be noted that in all cases of Technique 1, the data transferred has increased after optimization, which is contrary to expected behavior.
The results obtained on moving the content of the script tags to an external JS in goolge.com resulted in an improvement of 28.02% in MLT and an increase of

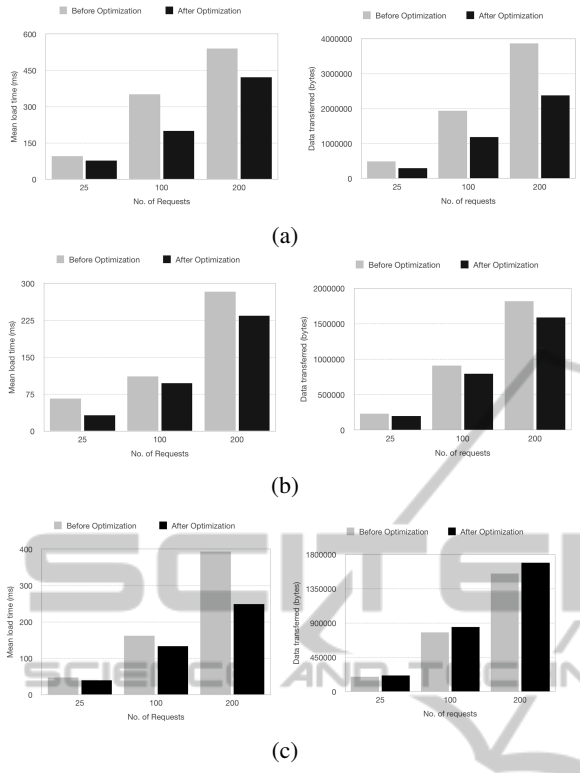

Figure 5: Results obtained **(a)** Technique 2 on google.com. **(b)** Technique 2 with JS on top. **(c)** Technique 2 with JS in the middle.

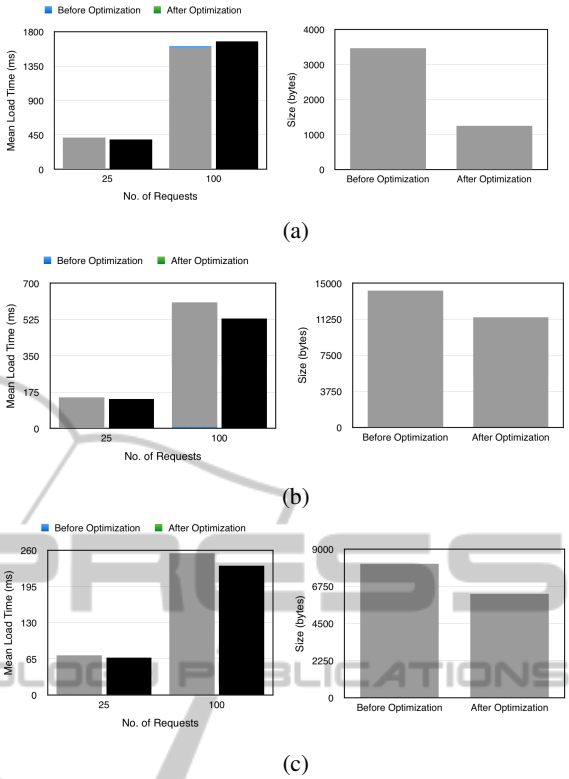

Figure 6: Results obtained **(a)** Technique 3 on yottaa.com. **(b)** Technique 3 on w3schools.com. **(c)** Technique 3 on simplecartjs.com.

9.3% in Data Transferred (Bytes) as shown in Figure 5(a).

The results obtained on moving the content of the script tags to an external JS with JS at the top of the page resulted in an improvement of 27.14% in MLT and an increase of 7.7% in Data Transferred (Bytes) as shown in Figure 5(b).

The results obtained on moving the content of the script tags to an external JS with JS in the middle of the page resulted in an improvement of 23.85% in MLT and an increase of 11.3% in Data Transferred (Bytes)as shown in Figure 5(c).

The results obtained on CSS and JS minification in yottaa.com resulted in an improvement of 5.01% in MLT and an decrease of 64.01% in Size(Bytes) as shown in Figure 6(a). It can be noted from Figure 6(a) that the resulting MLT for 100 requests has increased from the non-optimized version. This is an anomaly to the expected behavior.

The results obtained on CSS and JS minification in w3schools.com resulted in an improvement of 9.495% in MLT and an increase of 19.56% in Data Size(Bytes) as shown in Figure 6(b).

The results obtained on CSS and JS minification in simplecartjs.com resulted in an improvement of 7.32% in MLT and an decrease of 22.45% in Size(Bytes) as shown inFigure 6(c).

Experimental Steps for Implementation and Result Analysis

1. The existing ideas had to be implemented manually, but we have automated three of the optimization techniques through a tool. The tool automates the procedure by changing the DOM structure of the web page to be optimized.
2. We have also done a thorough evaluation of the strategies by:
   (a) Evaluating the techniques for a variety of web pages with different structures.
   (b) Benchmarking with standardized tools like the abTool.
   (c) Testing with real world test cases like:

Table 1: Performance Improvement in Technique 1.

NO. OF REQUESTS	% DECREASE IN MCT
25	27.69
100	11.33
200	13.78
Average	17.6

Table 2: Performance Improvement in Technique 2.

NO. OF REQUESTS	% DECREASE IN MCT
25	28.845
100	31.089
200	25.411
Average	28.448

- Varying number of requests for the page.
- Varying number of concurrent requests for the page.

3. Static and dynamic graphical representations of the benchmarked results, for better comparison and evaluation.
4. Analysis of the size of HTML documents and the related files for further network latency benchmarking.
5. Use of platform independent technologies and easily available/accessible tools for building the optimization tool.
6. All the strategies have been tested on multiple browsers (Ex. Google Chrome, Mozilla Firefox, Apple Safari, Internet Explorer) as well as 'wget' and the results have been confirmed to be consistent.

8 CONCLUSIONS AND FUTURE WORK

The above techniques, when implemented individually, give a slight improvement in performance. However, in conglomeration with the other techniques of optimization, it is sure to give a significant measurable increase in performance in terms of time required to load a webpage.
Other techniques mentioned in the paper need to be implemented, and if possible, a new technique that has not yet been accounted for can be discussed, and the performance optimization can be measured. It is also a significant problem to determine which optimization techniques together generate the best performance optimization. This is because performing *all* optimization techniques may have a performance overhead, so determining the best combination of techniques is necessary. In addition to this, the anomalies obtained in our results needs to be explained, after proper analysis.
An appreciable performance acceleration must be achieved after the implementation of a significant number of front-end optimization techniques.

Table 3: Performance Improvement in Technique 3.

NO. OF REQUESTS	% DECREASE IN MCT
25	5.97
100	1.8
Average	3.885

Table 4: Document Size Analysis - I.

TECHNIQUE	% INCREASE IN DOCUMENT LENGTH
1	9.358
2	9.35

Table 5: Document Size Analysis - II.

TECHNIQUE	% DECREASE IN FILE SIZE
3	26.44

ACKNOWLEDGEMENTS

We would like to thank Mr. Sanket Ingole and Mr. Shridhar Sanshi for their support and valuable inputs towards the improvement and completion of this paper. We would also like to thank Mr. Raghavendra G. S. for his timely help towards the organisation and formatting of this paper.

REFERENCES

Akamai, A. (2012). Akamai's front end optimization. http://www.akamai.com/html/resources/front-end-optimization-feo.html. Accessed on: 14th November 2014.

Iliev, I. and Dimitrov, G. P. (2014). Front end optimization methods and their effect. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, pages 467–473. IEEE.

Marshall, S. (2013). Jch optimize. https://www.jch-optimize.net/documentation/how-the-plugin-works.html. Accessed on: 14th November 2014.

Radware, R. (2012). Radware's fastview. http://www.radware.com/Products/FastView/. Accessed on: 14th November 2014.

Souders, S. (2012). The performance golden rule. http://www.stevesouders.com/blog/2012/02/10/the-performance-golden-rule/. "Accessed on: 14th November 2014".

Zhenhua, C., Crowell, J., Grigorik, L., Kaufman, J., and van der Schaaf, O. (2012). Pagespeed. https://developers.google.com/speed/pagespeed/. Accessed on: 14th November 2014.