Project Report on
Analyzing and Improving Page Load Time of Websites

Anshul Roonwal
SBU ID: 110554783
aroonwal@cs.stonybrook.edu

Aman Raj
SBU ID: 109939872
amraj@cs.stonybrook.edu

Sahil Jain
SBU ID: 110281300
sahjain@cs.stonybrook.edu

1. INTRODUCTION

Internet is getting richer and ricer in terms of content. It's no longer just the text that dominates the websites. In fact, web has evolved from a collection of text and images to a complex system with contents ranging from videos to executable scripts and flash objects. This increased amount of content directly impacts the page load time(PLT), which is the time taken to load the website from the server. PLT has a huge impact on user satisfaction therefore from the point of view of a company, it is very important to try and optimize the page load time as much as possible.

Since these components together make the webpage, downloading them in order to render the website becomes important. It is found that most of the page load time is spent in downloading all the components of a webpage. The key to reduce this Page Load Time is to reduce the number of HTTP request while maintaining the richness of the website. A number of techniques including various versions of HTTP protocol, SPDY have been developed and applied to reduce page load time but in any of these approaches the need to download content from server is not eliminated.

One of the ways to avoid downloading static content from webpages is Caching. The idea of caching is to locally store some or all of these components when website is loaded for the first time, and load the components from local disk instead of server for subsequent requests given the disk contains the latest copy of the component.

2. PROBLEM DESCRIPTION

As the internet continues to grow, the webpages have evolved from mere text and images to a whole new world which include a variety of content like videos and scripts from servers spread all over the world. Time taken to download these pages in order to render them on the browser is called Page load time (PLT). It directly impacts user satisfaction and thus it is very important to try and minimize it as much as possible. We did a study on website complexity and what impacts the loading time of a website. We identified a set of features that affect the page load time of a website both at content and service level. We aim to quantify the features that impacts Page Load time the most and also build a Regression Model to predict the page load time depending on content and service. In this study we have only considered the client side view of website and assumed that the backend server capabilities are the same.

Our aim with this project is also to provide a deeper analysis of effect of DNS or local cache on Page Load Time using WProf[1]. Local cache is when browser stores local copies of website components while DNS cache stores IPs of web servers containing these web pages. The analysis is done using various approaches on both Mobile and Desktop. We will discuss the pros and cons for each type of cache on both mobile and desktop devices.

Besides this we are also trying to predict the average page load time of a website by running a Machine Learning model on set of metrics identified to characterize the complexity of websites at content-level and provide an estimation of the effect of local and DNS cache for the same. This feature vector include value for various features, such as number of different types of objects loaded or number of servers contacted, the day of the week, time of the day and a few more. The analysis is done on Alexa[2] top 200 websites.

3. PRIOR WORK

Aruna Balasubramanian et al. [4] performed an analysis to study the page load time of 350 web pages with a lot variation such as end-host caching, SPDY and the mod page speed server extension. The motivation behind their research lies in the fact that page load time in the webpages is an extremely complex thing and is in fact dependent on a lot of factors, and hence it's difficult to identify the bottlenecks. The authors conclude that computation is a significant factor in PLT that takes up to 35% of the time and the JavaScript plays a significant role in PLT by blocking the HTML parsing.

M Perinotto et al. [5] tried to answer the question that whether users will less wait if proxy cache is implemented in the current network conditions. They took the approach by introducing two caching algorithms. In first the idea is to use cache to keep the documents that take the longest time to download. In second they keep in cache those documents that particularly take longer to download. These new algorithms are compared to the most famous ones - LRU, LFU, and SIZE.

J. Fisher et al. [6] came up with a model to reduce the page load time by using a page mode write cache method. They use an internal memory array for receiving the burst of data sent by microcontroller. The data is initially stored in the SRAM i.e static random access memory, where it's stored sequentially and grouped into pages.

Zhen Wang et al. [7] conducted a study on the mobile browser speed by collecting data from 24 iPhone users continuously over one year. In this study they show the importance of speculative loading which is even more important than caching and pre fetching. Furthermore, they also conclude that the client solutions can improve browser speed by 1.4 second on an average.

Herman Rodriguez et al. [8] came up with a new model that deals with saving the important parameters associated with a web page when saving it in the browser. In this model a user sets a preference parameter that filter web page content to override the block from cache preferences.

4. TOOLS & TECHNOLOGIES USED:

- *Ubuntu 12.04*: Linux operating system (Desktop)
- *Chromium browser*: Chromium is the open-source web browser project. Wprof is created on top of this.
- *Wprof*: WProf is a tool that extracts dependencies of activities during a page load.
- *D3.js & MAMP Server*: For the purpose of creating interactive visuals like Parallel Coordinates.
- *Python*: Scripting language for log analysis, plotting etc.
- *Android Phone*: Nexus 6
- *Android Debug Bridge* [3]: A command line tool that lets us communicate with the connected android mobile device

5. SOLUTION METHODOLOGY

We are working on this analysis of page load time in an iterative fashion. We came up with one approach, analyzed its results, identified the challenges involved and then switched to a better approach by overcoming that challenge. Here are the approaches that we used.

Approach #1

We are working on Alexa top 200 websites. We are using a script that loads the page one after the other. Every page load generates some logs which are saved separately. This gives a correct estimate of what all objects were downloaded, how much time it took for the request to get completed and the page to load. The graph below shows the PLT in seconds with number of iterations that the websites were loaded.

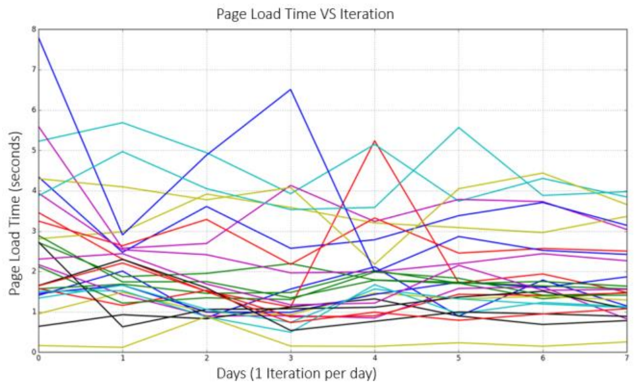

Figure 1

Figure 1 is a graph obtained after running the script of loading the webpages and recording the PLT of individual websites for 8 days. We observed that not all the websites out of these 200 alexa websites were loaded successfully. Thus the results we present are for around 80 websites out of those 200. We found that since the first day there was no local cache available, the browser took more time to load the pages in comparison to the subsequent days. We also noticed that the PLT kept decreasing as the days went by. But this was true only for a few websites. We assert that those websites had lot of static contents in comparison to dynamic content. This is because more and more static content was cached locally. So caching did reduce the PLT but not significantly for majority of them.

Challenges identified with Approach #1

There are a few websites for which the content changes very frequently. There is little static content and a lot of dynamic content on these websites. So every other day, it will be a miss when the local cache is asked for the content because the content might have been changed. In scenarios like these, caching does not help much. Some of such websites that frequently changes its contents are the news websites like Cnn.com, Nytimes.com. This does makes sense as well. Every other hour there will be a news update, a new picture will be posted or even a new video.

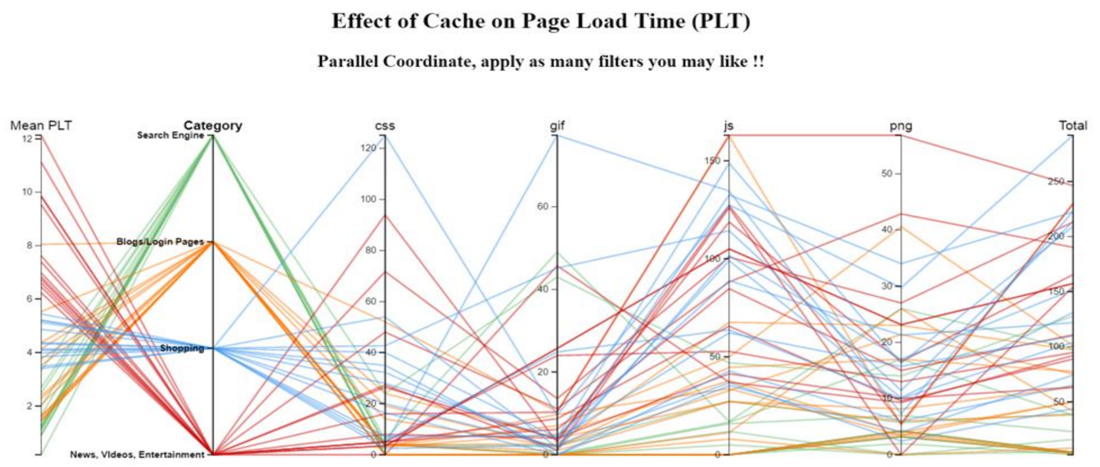

Figure 2

Limitations of Approach #1

The method of loading the pages one after the other doesn't seem effective. This is because the cache size is very limited. Once the cache for first site is saved, by the time the 200th website loads, the old cache will be deleted to accommodate the most recent one. Now in the event when all the 200 website or most of them gets loaded, the chances of the cache of the previous ones being lost is higher and we wouldn't get the proper results. This is why the graph in figure 1 fails to advocate that caching is very effective in reducing page load time.

Approach #2

With this approach, we couldn't address the limitation (addressed with approach 3) but we thought about the above challenge and found if we had control over the server, we could also control the objects that we want to cache locally. Of course, the preferred ones will be the ones that do not change very frequently. However, this was impractical in case of web servers since we do not own them. To analyze the results, we hosted our own apache web server, hosted 40 websites on this apache server. These websites were carefully chosen to show diversity of different objects within each of them. Some had mostly JavaScript objects, other had more images and then there were websites with different ratios of these web objects. We classified these websites into 4 categories (also shown in figure 2) →

1) ***Search Engine***: These websites have a simple landing page like Google, Baidu. These won't be very content heavy.
2) ***Blogs & Login Pages***: These websites will be richer in content than 'Search Engines' but won't have many dynamic content. For instance, box.com, blogspot.com.
3) ***Shopping:*** Websites like Amazon.com has a lot of content right at their landing pages. Lot of images, Java Script and sometimes videos too. These will be definitely heavier than the first two.
4) ***News, Videos, Entertainment:*** This category has websites like cnn.com, and youtube.com. These are some of the richest ones on internet as far as content is concerned. These will have lots and lots of videos and will be dynamic in content.

Around 10 websites from each of these categories were hosted on our local Apache server. But the content we had downloaded were not changing with time since we had them on our local server and not on web server.

Figure 2 above shows an interactive visual made with d3.js and hosted on MAMP server. This form of visualization is called 'Parallel Coordinate' [6]. Every coordinate (Parallel line in the visual) represent a feature of the websites. For instance, the first one represent 'Mean PLT (page load time)', followed by 'Category', CSS objects count, GIF, JS, PNG and total web object counts. A filter can be applied to each parallel coordinate. For instance, one can choose to look only into *Blogs & Login Pages* and *Shopping Websites*, and websites with more than 20 PNGs and less than 100 JS objects. Applying this filter criterion will bring out the Mean PLT of all the websites that comes under that criteria. Not just the Mean PLT but value of each of the feature for those websites satisfying this criterion. With general observation, we found that locally caching the web content significantly reduced the page load time now that the dynamic contents were not there anymore.

Challenge in Approach #2

Since we ignored the use of a real web server by hosting one of our own to study the results, we got drifted from the real world scenario where there is lot of uncertainty as to from where the web content is being served and also change in the content on websites. Also, we didn't address the issue of small cache size which could invalidate the object of first website by the time the 200th website is loaded.

Approach #3

With Approach three we address the limitation of small cache size by making a single change to our approach. Earlier we were loading the websites one after the other for 8 iterations. Now, we are loading the same website 8 times before moving on to load the next website. This scheme helps us solve the small cache problem and help us observe the real effect of cache on PLT. We also narrowed down our analysis of Alexa top 200 from over a range of 8 days to over 8 iterations within a single day. Why are we doing so? We do so in order to have as limited change in the dynamic content of the websites as possible. This will eliminate the effect of frequent change in content of the websites. We have observed that the degree of changes in such frequently changing websites at the granularity of days is higher than at the granularity of hours within a day.

We also realized that we were not using WPROF to the full of its potential. So we involved more functionality of WPROF so that we are not just using the chromium browser to get our logs but are making use of other scripts within WPROF to give us better logs with specific details. These are the steps we followed→

1. Use the pagespeed2.py to getting a single log file from WPROF after loading the webpages. Note, the same website was loaded 8 times before we moved on to load the next.
2. Run slice.pl to split this big log file into individual page load files, where each file represents a page load.
3. Now Execute the analyze.pl followed by json_dag.py to get the structured logs with detailed information.

Wprof being a powerful tool helped us getting these logs with so many important information about each page load. On top of this, we wrote our own python script to extract the information that we needed from those log files in order to get the the important features like download time, computation time, DNS lookup time, number of objects loaded of each type. The graph below shows the effectiveness of cache in reducing the page load time with our new approach.

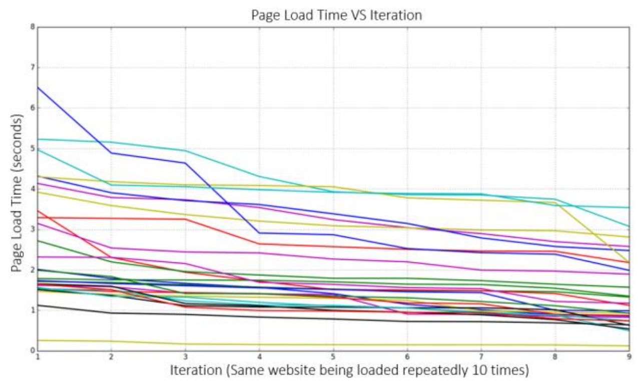

Figure3

Figure 3 shows the reduction in page load time with iterations. For some websites the PLT reduced significantly after the first iteration but for others it remained almost the same after first iteration. Our conclusion from this is that websites with less content could cache almost everything after one iteration making the page load time almost same after first iteration. However, content heavy websites couldn't cache everything in one iteration and thus it took them time to reach the state where almost everything was available in cache.

We carried out the same set of experiments on Android device as well. The ADB (Android debug bridge) was used for debugging. Same steps were followed after collecting the single log file to split it up using slicemobile-old.pl and analyzed using analyse.pl. This is the plot we got.

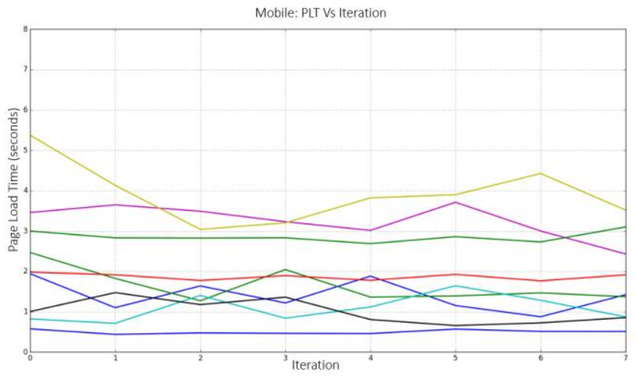

Figure 4

As observed in figure 4, the results that we obtained were not significant. It could be because the cache size of the android device was very small. Therefore, we focused more on desktop browsers and we didn't limit ourselves to just cache but we also predicted the PLT of new websites given the set of features that we discuss further in this paper.

6. Beyond Cache: Machine Learning

With machine learning techniques, we are interested in predicting the page load time for a new website. This requires creating a feature vector. So we extracted all possible features and ran a linear regression model on all of them. Obtained result is depicted in figure 4.

This graph shows that the linear regression model was very effective in predicting the page load time of the websites. We did our analysis with the data that was already present with us to bring out a comparison between the actual page load time with the predicted one

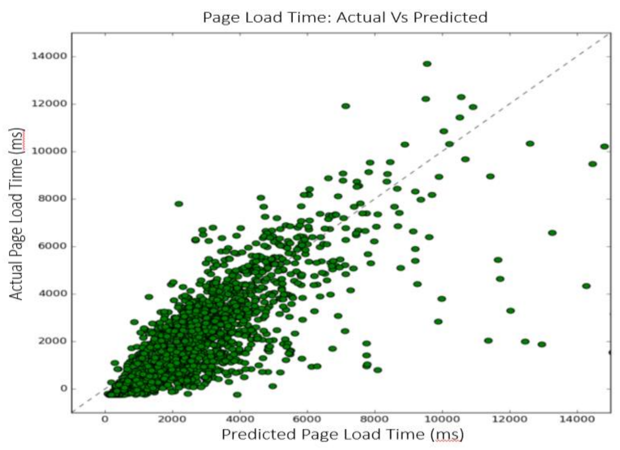

Figure 5

The axis X=Y would have been the idle axis where all the points should have been present in case the actual was same as predicted. However, it is very difficult to predict absolutely accurate values and thus we observe that thought the points are scattered a bit, most of them are around the X=Y. This indicate that our prediction algorithm is effective if not absolutely correct.

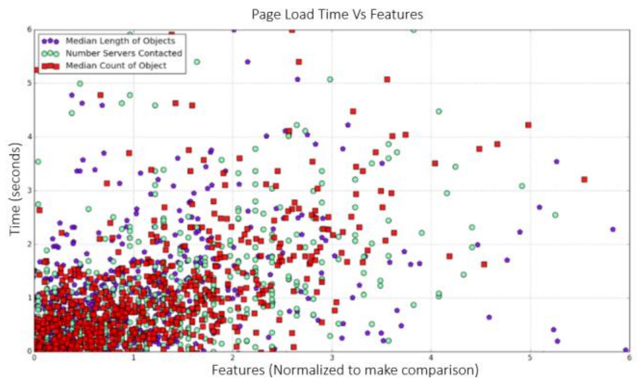

Figure 6

We found that more than the size of objects, it the is total count of objects fetched which has the most impact of Page Load Time, Images being the one with most impact. For this purpose, we plotted a scatter plot between the three attributes,

a) Median length of Objects (*blue*)

b) Number of Servers Contacted (*cyan*)

c) Median count of Objects (*Red*)

Figure 5 clearly shows that Count of Object dominates among the three and affects the page load time the most.

7. EVALUATION & RESULT

The main outcome of our project is that page load time readily decreases due to the presence of cache. With multiple approaches we observed that caching has proved to be very effective. We corrected ourselves as we advanced in the project. For instance, the experiment was carried out in a way that the same website is being loaded for a pre decided fixed number of times (10) before moving on to load the next website instead of loading one after the other. The results with correcting this mistake improved a lot. The graphs representing the page load time with our previous approach and corrected approach have been shared in the solution section earlier in this report.

We also created an interactive visual called Parallel Coordinate made with d3.js and hosted on MAMP server. Every coordinate (Parallel line in the visual) represent a feature of the websites. This works as an interesting filter on features as well. More has been discussed about it in Approach 2 in the solution section earlier.

We also present the overall statistics of the objects downloaded across the websites. Here is the visual to demonstrate this.

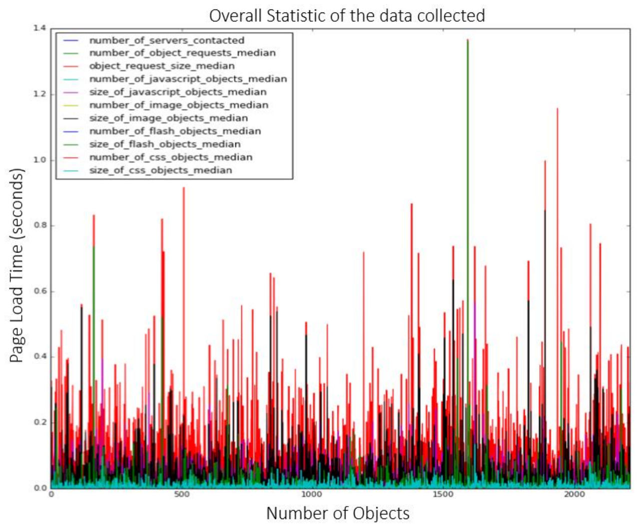

Figure 7

Figure 7 gives us the idea of how the page load time(y axis) depends on the number of objects and of which kind.

Since the number of features were on the higher side, we applied another technique to reduce the features to two dimensional space. After reducing all the features to just two features namely, 'Number of object Requested' and 'Number of Servers Contacted', we visualized them to study which one out of these two is dominating.

Figure 8 as obtained by performing Principle component analysis wasn't much rich in the information that we were interested in. It suggests that both the features were of same importance. However, we have already studied and found that number of servers contacted correlates better with number of servers contacted.

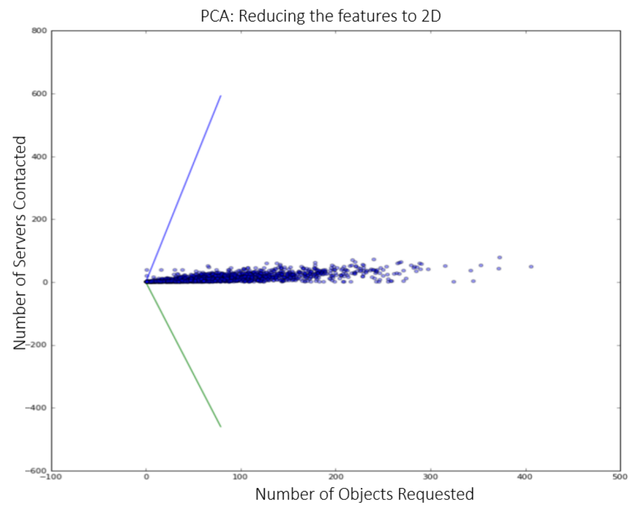

Figure 8

After making the conclusion that caching does help reduction in page load time, we went beyond caching to perform machine learning on the features that we collected through Wprof. We used linear regression technique using various combinations of features to find the list of attributes that impact the Page Load Time. We found that the number of objects fetched is the most dominant indicator of client-perceived load times which alone predicts about 75.27% of the median page load time with Normalized Root Mean Square errors as 0.43. We also found a collection of other attributes like (Number of servers contacted, number of object requests median and number of image objects median) when added to the regression model, it further pushed the Root Mean Square error to 0.37.

- **REFERENCES**

[1] http://wprof.cs.washington.edu

[2] http://www.alexa.com/

[3] http://developer.android.com/tools/help/adb.html

[4] Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall. Demystifying Page Load Performance with WProf in *NSDI 2013.*

[5] Roland P. Wooster, Marc Abrams. Proxy caching that estimates page load delays, *Computer Networks and ISDN Systems Volume 29, Issues 8–13, September 1997, Pages 977–986ys.*

[6] KentD. Hewitt, Samuel E. Alexander, Richard J. Fisher. Serial EEPROM device and associated method for reducing data load time using a page mode write cache.

[7] Zhen Wang, Felix Xiaozhu Lin, Lin Zhong, Mansoor Chishtie. How far can client-only solutions go for mobile browser speed?, *WWW '12 Proceedings of the 21st international conference on World Wide Web, Pages 31-40*

[8] Viktors Berstis, Herman Rodriguez. Blocking saves to web browser cache based on content rating.

[9] Jia Wang, A survey of web caching schemes for the Internet, *Newsletter, ACM SIGCOMM Computer Communication Review Homepage archive ,Volume 29 Issue 5, October 1999 , Pages 36-46.*

[10] Brian D. Davison, A Web Caching Primer, *IEEE Internet Computing (Volume:5 , Issue: 4), Jul/Aug 2001.*

[11] Steve Souders, High-performance web sites, *Communications of the ACM - Surviving the data deluge CACM Homepage archive Volume 51 Issue 12, December 2008 Pages 36-41.*

[12] Li Fan, Pei Cao, Wei Lin, Quinn Jacobson. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance, *SIGMETRICS '99 Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems Pages 178-187.*