Effect of HTTP and DNS cache on web page load time for Mobile and Desktop
CSE 692 - Spring 2015

Ankush Aggarwal
Net ID: anaggarwal, SBU ID: 109748526
Department of Computer Science
Stony Brook University
NY 11790, USA
anaggarwal@cs.stonybrook.edu

Rohan Bharadwaj
Net ID: rbharadwaj, SBU ID: 109758985
Department of Computer Science
Stony Brook University
NY 11790, USA
rbharadwaj@cs.stonybrook.edu

***Abstract*— In this paper, we analyze how HTTP and DNS cache improves page load time on both Desktop and Mobile. We discuss different approaches for analysis and explain benefits and problems with each approach. We also provide a comparative analysis of Desktop and Mobile. Lastly, we list some of the future work which can be pursued in the context of the methods.**

I. INTRODUCTION

80% of the end-user response time is spent on the front-end. Most of web page load time is in downloading all the components of the page such as images, stylesheets, scripts, Flash etc. The key to faster pages is to reduce the number of HTTP requests required to render the web page. This could be achieved by reducing the number of components in a page. One way to do this is to simplify the webpage's design. But most webpages are rich in content so we need a way to build pages with richer content while also achieving fast response times. There are some techniques that can reduce number of HTTP request for content heavy pages. Techniques such as combine files, inline images, CSS sprites reduce HTTP requests require to render a webpage to client[4].

When a browser hits an "empty cache", it has to request all the components to load the page. Figure 1 shows the browser behavior in case of empty cache.

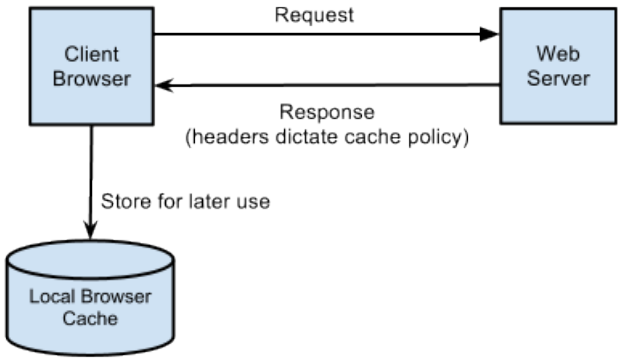

Fig. 1: Browser with empty cache

There may be as many as 30 components in a single web page. A "full cache" means all (or at least most) of the components are found in the disk cache and the corresponding HTTP requests are avoided. Figure 2 represents how browser used local cache available to avoid HTTP requests. The main reason for an empty cache page view is because the user is visiting the page for the first time and the browser has to download all the components to load the page. Once the web page load is completed and the different components are stored in the local cache, only a few components needs to be downloaded for subsequent visits.

This project is intended to analyze the effects of HTTP and DNS cache on Web Page load time. HTTP caching occurs when the browser stores local copies of web resources such as images, scripts, stylesheets for faster retrieval the next time the resource is required. HTTP caching is a universally adopted specification across all modern web browsers, making its implementation in web applications simple[6].

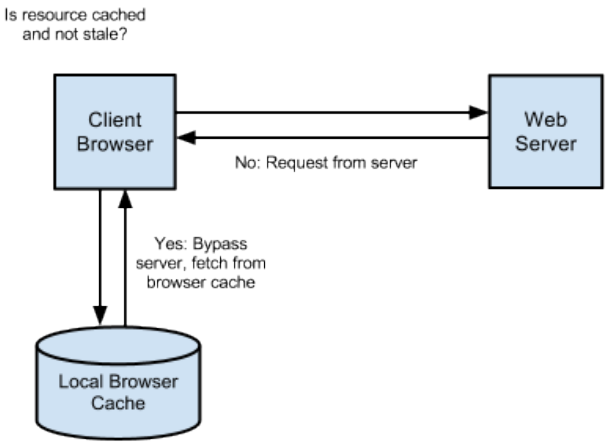

Fig. 2: Browser hits local cache

DNS cache stores the locations (IP addresses) of web servers that contain web pages which user have recently viewed[7]. The analysis is done for Desktop having Ubuntu(Linux Operating system) and Mobile device running Android operating system. The data set used is Alexa's top 200 websites. Also, top 200-400 websites from Alexa are used to get more accurate results as generally top websites are over-optimized so cache doesn't contribute much to their performance. For analysis, Wprof[1] analysis scripts are used on logs generated by Chromium browser.

II. MOTIVATION

Analyzing how HTTP and DNS cache effects page load time is an important task and plays significant role in any organization. Web page load time is a key performance metric that many companies and people are trying to optimize. The motivation behind this project is twofold.

Firstly, to understand different parameters that effect the performance of Web Page load. From the time a user requests a URL in a browser to when a page is finally loaded in the web browser there are lot of things involved like DNS lookup, forming a TCP connection, HTTP request, response, drawing a webpage on to screen.

Secondly, Web page load time is an important aspect for websites. A study from Aberdeen Group [8] shows that a delay of 1-second in a webpage load time results in 11% fewer page views, 16% decrease in customer satisfaction and 7% loss in conversions. Amazon found this to be true, reporting increased revenue of 1% for every 100 milliseconds improvement to their site speed[9]. Moreover, webpage load time also effects whether the user is going to use website or not. A study by Akamai[10] found that 47% of people expect a web page to load in two seconds or less, 40% will abandon a web page if it takes more than three seconds to load and 52% of online shoppers say quick page loads are important for their loyalty to a site. So, we want to study different techniques that effect the Web page load time.

III. TOOLS AND DEVICES

To analyze the effects of HTTP and DNS cache on mobile and desktop systems, different tools are used. Some tools are used for both analysis and some are used specifically for one device. Following is the list of tools and devices that we are using for analysis.

- Chromium browser
- Wprof
- Speed test-cli
- Ubuntu 12.04
- Samsung Galaxy S3
- Android Debug Bridge(ADB)

Chromium Browser: Chromium is the open-source web browser project from which Google Chrome draws its source code. The browsers share the majority of code and features, though there are some minor differences in features and they have different licensing.

Wprof: WProf is a tool that extracts dependencies of activities during a page load. For Web developers, WProf can help identify the bottleneck activities of your Web pages. For browser architects, WProf can relate page load bottlenecks to either Web standards or browser implementation choices. We have used Wprof scripts to get the data related to page load time and have written automated scripts on top of Wprof to run the tests.

Speed test-cli: We used this tool to log the details of the Wi-Fi network like download speed and bandwidth before every run of the experiment. This shows the results of the fastest upload and download burst rates from user location to the location of the test server. It is important to log speed as network speed keeps on

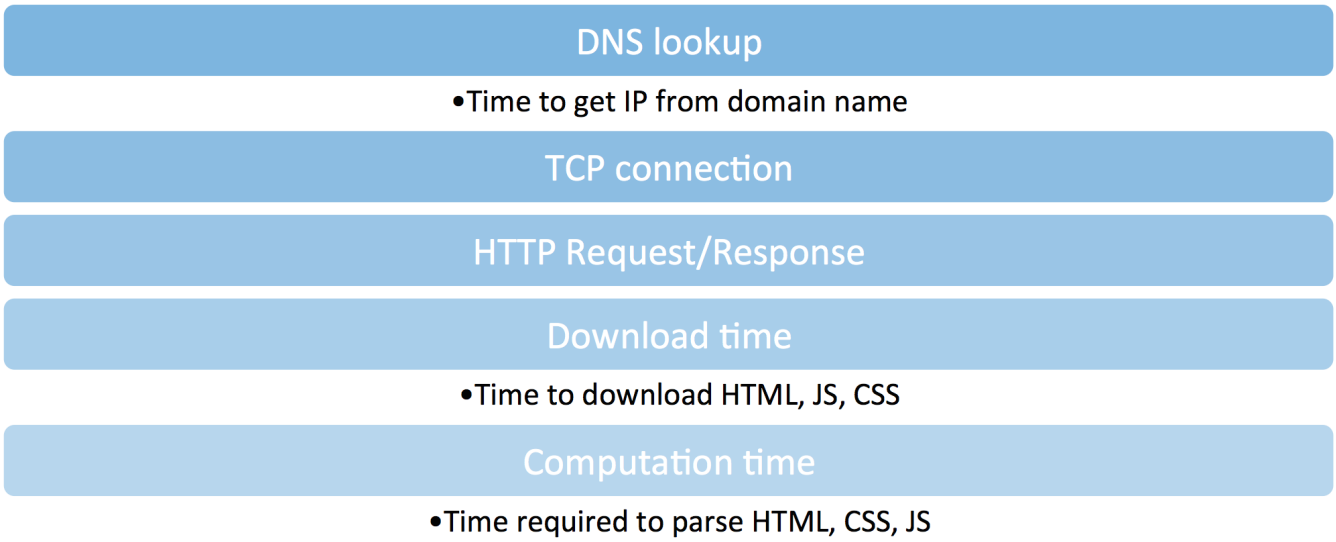

Fig. 3: Component in web page load

changing through out day due to network congestion etc.

Ubuntu 12.04: Ubuntu is a Debian-based Linux operating system used as desktop operating system.

Samsung Galaxy S3: It is a mobile device running android operating system.

Android Debug Bridge: ADB is a versatile command line tool that lets us communicate with the connected android mobile device[3]. The Android logging system provides a mechanism for collecting and viewing system debug output. Logs from various applications and portions of the system are collected in a series of circular buffers, which then can be viewed and filtered by the logcat command[15]. We can run logcat as an adb command to save log in a file generated by mobile device.

IV. COMPONENTS INVOLVED IN WEB PAGE LOAD

From the time a URL is entered in browser to the time the page is displayed on the screen there are lot of events that takes place. These all account for page load time. The major events are shown in Figure 3.

A DNS Lookup is when a device that supports IP asks a DNS server for the IP address associated with a domain name. The DNS Server must ”look up” the IP associated with that domain name. After DNS lookup, the operating system is now in possession of website’s IP address, provides the IP to the Application (browser), which initiates the TCP connection to start loading the page. Client sends the HTTP request to the web server and waits for the server to respond to the request. Web server processes the request, finds the resource, and sends the HTTP response to the Client. At this point, Client loads the content of the response which includes the download of scripts, stylesheets etc. After this, these web resources are computed and page is rendered to user.

In our analysis we consider the DNS lookup time, Download time and Computation time. We have added code to Wprof scripts to get DNS lookup time from the JSON object.

V. DATASET

A. *Desktop*

For Analysis on Desktop, we use 200 websites from Alexa [2]. Initially, we used top 200 websites from Alexa but most of them are optimized and use lot of other techniques to reduce HTTP requests. So, we used Alexa’s top 200 to 400 websites to primarily see the effect of caching on webpage load performance. We categorize the websites as content heavy, JavaScript heavy and normal websites. We run the experiments for 10 times and are able to capture data for around 95 websites. Overall, our analysis for desktop has data from around 95 websites.

S.No	Run#	Date	Observation
1.	Run 1	16 March 2015	The page load time(PLT) was high
2.	Run 2	17 March 2015	There was slight decrease in PLT
3.	Run 3	18 March 2015	Some websites had decreased PLT others it increased
3.	Run 4	20 March 2015	From here we had irregular times and no pattern
5.	Run 5	21 March 2015	-do-
6.	Run 6	19 March 2015	-do-
	. . . . .	. . . . .	. . . . .
15.	Run 15	29 March 2015	-do-

Fig. 4: Desktop Approach I

B. Mobile

For Mobile's Analysis, we used top 200 websites from Alexa[2]. When we run the experiment for 10 times, a very big log file is generated having data from websites loaded multiple times. Upon parsing and analysis of generated data, results are obtained for few websites because of mobile network issues, the page load times out most of the times. To observe the cache effect on page load, we use only 1 website for analysis.

VI. METHODOLOGY FOR DESKTOP

A. Approach I

This approach is straightforward. As we know, when the webpage is visited first time, there is no cache on local disk and upon successive visits to page, browser caches some webpage resources for faster future page loads. We run the automated script to load top 200 websites of Alexa on each day for 15 days. At first, there is no cache in the system due to which browser hits empty cache and requests server for every web component required to render the page and as the days pass, browser accumulates some cache and the page load time should decrease, but we notice that the results are not aligned to our belief. So what is wrong with this approach?

Upon some analysis, we found some problems in this approach. Firstly, the contents of websites change every day so the cache does not help much after some days. For e.g. news websites such as cnn.com changes content too often to display recent news, hence caching doesn't help much here. For such websites, cache of general scripts, stylesheets helps but image cache needs to update which is responsible for high download times.

Figure 4 shows the observation on each day run when experiment is carried for 15 days.

B. Approach II

The problem with first approach is we can not control the content of webpages as we don't have access to remote server. This gives us idea of another approach of having our own web server. In this approach, we developed around 15 websites hosted on a remote Apache web server which could be configured to meet our needs. We could now control server to let client know which objects do we want to cache. To observe the effect of cache on wide variety of website, these 15 websites represents different categories such as image heavy, JavaScript heavy, low content websites. We use Wprof scripts to capture the three units of interest: DNS lookup time, Download time, Computation time. We notice that the cache helps the page load time and page load time decreases as browser hits cache on local disk.

While this approach generates useful results, the obvious problem with this approach is it doesn't capture real world scenarios 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 can not simulate the effect which could be because of CDN. A CDN (Content Delivery Network) is a global cluster of caches that can serve as local caches for static files (objects). If a visitor of a website or a user of an application request certain files (e.g. images, pdfs, JavaScript, CSS files etc.) instead of the hosting server

responding with these objects, the CDN takes care of serving them[11].

C. Approach III

Considering the problems with above approaches, we need an approach that can capture real world scenarios and doesn't effect by content change of websites. We decide to use real websites(not developed by us) so that all the possible scenarios could be captured. And to overcome the problem of website's often content change, we use Alexa's top 200-400 websites and run the experiment in a single day. We run the experiment on same day for 10 times one after the other so that we have the cache and there is not much change in the website content. Overall, we use 200 websites from Alexa and carry out the experiment for 10 run simulating 10 days.

S.No	Run #	Type of cache
1.	Run 1	Cold cache
2.	Run 4	Warm cache
3.	Run 9	Hot cache

Fig. 5: Cache Type

We define 3 different types of cache on the basis of run number(Run #) as shown in Figure 5.

Cold cache: Cold cache is the slowest cache hit possible. This is at the beginning or on the first run when there is no cache, so the page load time will be higher as there is no cache. For our approach, cold cache represents time before first run when chromium browser doesn't have any local cache.

Warm cache: Warm cache is the intermediate stage between the cold and hot cache. Browser may have cache for some websites or might not have any cache stored for some webpages. This is after 4th run when there is some cache stored on disk so that the later request to URL can be served from cache.

Hot cache: Hot cache is the fastest cache hit possible. At this stage, browser has lot of disk local cache for each website and web page load time takes full advantage of cache. In our approach, this is after 9th run, when there is lot of cache stored for website and the page load will be fast and the time will be constant after this time.

An automated script generate the log file for each website for each run using chromium browser. We edit the Wprof analysis code to get DNS lookup time. Then, we use Wprof to analyze these log files and generate JSON files having required data such as download time, computation time, dns time. Upon analysis, we get data for around 95 websites and graphs are generated using this data as shown in next section. We also write some automated scripts to extract DNS lookup time, computation time and download time for each day (we call each run as a day). The next section shows the effect of cache on above properties for different runs.

VII. DESKTOP RESULTS

A. DNS time vs DAY

DNS lookup time is the time it takes for the browser to get the IP address of a domain name. When a URL is entered in a browser it first looks at the local DNS cache if it is hit it directly sends a request otherwise the request is sent to DNS server which is a recursive resolver that has mappings of domain names to IPs.

The graph in Figure 6 shows the DNS lookup time on y-axis which is in milliseconds vs run day (run#) on x-axis. As the number of days increases the DNS lookup time is decreased because of DNS cache. We can observe the time does not decrease linearly as there are other factors like the network speed, bandwidth and the server that serves the requests.

In the analysis, we come across some outliers which take as long as 10 seconds or more time for DNS lookup. We removed these data sets for consistency.

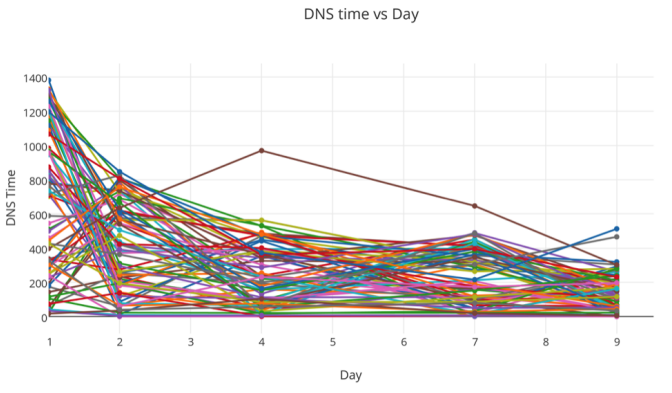

Fig. 6: Desktop DNS time vs DAY

B. COMPUTATION time vs DAY

Computation time is the time taken by the browser to parse HTML, JavaScript and CSS. It also finds out if there are any external requests of resources and is blocked during that time then continued to parse once the requested resources are fetched.

The following graph in Figure 7 shows computation time in milliseconds on y-axis vs number of day on x-axis. We observe that the computation time is decreased as the days are passed because the data is cached locally and is available instantly during parsing rather than waiting to be fetched from server. Also one more interesting observation is that from cold cache to warm cache there is a drastic reduction in computation time after that there is some fluctuations of increase this is because between 1st and 4th run the probability of change in data of website is less so the content is served from local cache but as the time passes there is change in content of websites do the data is requested from the server instead of using local cache which results in increasing trend of computation time.

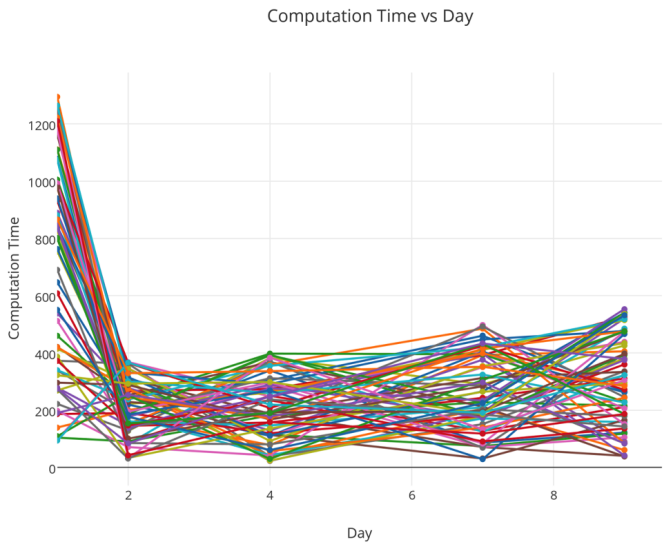

Fig. 7: Desktop COMPUTATION time vs DAY

C. DOWNLOAD time vs DAY

Download time includes time taken to download HTML, JavaScript, Text, CSS, Images and other objects from the server. A progressive download is the transfer of digital media files from a server to a client, typically using the HTTP protocol when initiated from a computer.

The following graph in Figure 8 shows the download time in seconds on x-axis vs. number of day on x-axis. We can observe that the download time decreases as the number of days pass because of the cache stored locally. We can also observe after some time download time reaches to saturation point where it does not decrease further with time. This is because browser can not cache whole website, there are always some components that requires update from server and also there are some components which most servers explicitly restrict browser to cache.

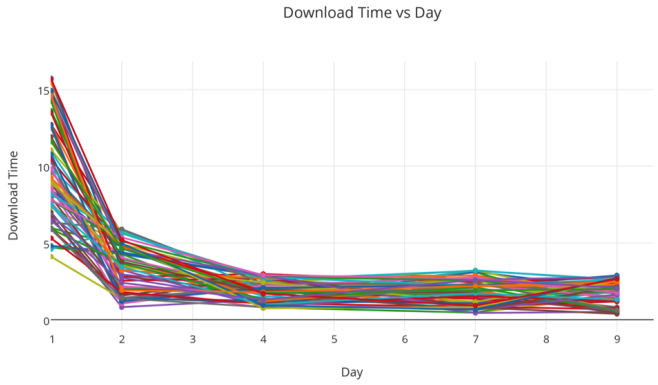

Fig. 8: Desktop DOWNLOAD time vs DAY

VIII. METHODOLOGY FOR MOBILE

A. Approach I

For desktop analysis, after trying different approaches we used a technique to use 200 websites from Alexa and run the experiment for 10 times on same day. So, we use the same approach for analysis in mobile. We use a Samsung Galaxy S3 device running android operating system and install chromium browser application. We use another application that automate the process of loading webpages in browser and generate the log file. We load Alexa's top 200 websites and log the details using ADB [3]. After capturing lot of log data, we run Wprof analysis scripts to get the download, DNS and computation time.

Upon analysis, we find that the results generated are not following the desired pattern. With some research about caching in Mobile devices, we come up with two probable reasons for this discrepancy. First, we find that mobile devices have limited cache (around 2-5MB). This explains the results of analysis, as the cache size is limited, browser overwrites the old cache. Second, 200 websites are too much for mobile analysis, so at

end of 1st run, cache of websites in beginning has been overridden by cache of websites in the end of run. So in next run, no cache is found by browser and it has to request all data from server required for web page rendering.

B. Approach II

From the results of approach I, we realize that it might be meaningful to not use 200 websites for analysis. In this approach, we use a single website and load it 10 times using chromium browser to generate log data. On analysis of log data using Wprof, the result shows that the cache improves the page load time. We try on different websites and do analysis on each website as an separate experiment and anticipated results are obtained. Finally, we use amazon.ca website to generate graphs which mostly show similar behavior as in case of desktop.

IX. MOBILE RESULTS

A. DNS time vs DAY

The basic job of DNS is to turn a user-friendly domain name like "amazon.ca" into an Internet Protocol (IP) address like $\langle 72.21.215.233 \rangle$ that computers use to identify each other on the network. System connect through a domain name server, also called a DNS server or name server, which manages a massive database that maps domain names to IP addresses. DNS lookup time is the time it takes for the browser to get the IP address of a domain name.

The graph in Figure 9 shows the DNS lookup time on y-axis which is in milliseconds vs run day (run#) on x-axis. As the number of days increases the DNS lookup time is decreased because of DNS cache. We can observe the time does not decrease linearly as there are other factors like the network speed, bandwidth and the server that serves the requests.

The graph shows the analysis on amazon.ca website when run on mobile for 10 times.

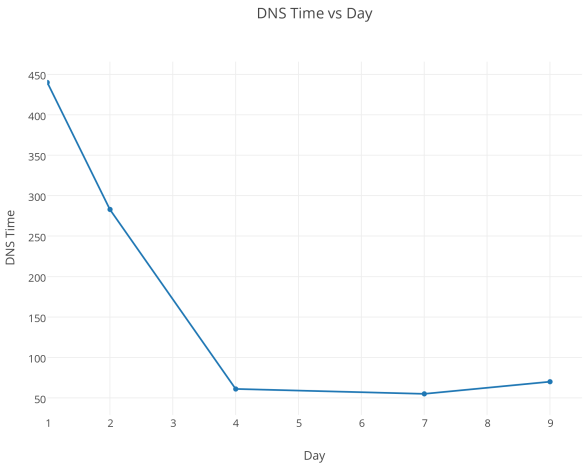

Fig. 9: Moblie DNS time vs DAY

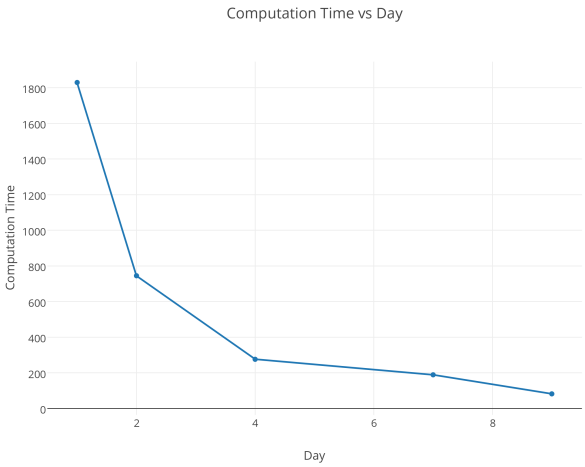

Fig. 10: Mobile COMPUTATION time vs DAY

B. COMPUTATION time vs DAY

When the browser receives HTML, scripts, stylesheets from web server, it parses these components to identify the web resources require to render the requested page by client. This parsing time is known as Computation Time. Wprof uses computation time to check dependency of HTML and JavaScript. If HTML parsing is resumed a long time after the JavaScript is received, Wprof determines that HTML parsing depends on evaluating JavaScript[16]. Computation time is the time taken by the browser to parse HTML, JavaScript and CSS.

The graph in Figure 10 shows computation time in milliseconds on y-axis vs number of day on x-axis. We observe that the computation time is decreased as the days are passed because the data is cached locally and is available instantly during parsing rather than waiting to be fetched from server.

C. DOWNLOAD time vs DAY

After parsing of HTML, scripts etc., browser knows which objects it requires to render the webpage to user. At this point, it request web server for the required objects. Download time includes time taken to download HTML, JavaScript, Text, CSS, Images and other objects from the server. Some web server specify no cache option which indicates browser to not cache data. While in our case, amazon server allows caching and hence web page load improves with time.

The following graph in Figure 11 shows the download time in milliseconds on x-axis vs. number of day on x-axis. We can observe that the download time decreases as the number of days pass because of the cache stored locally.

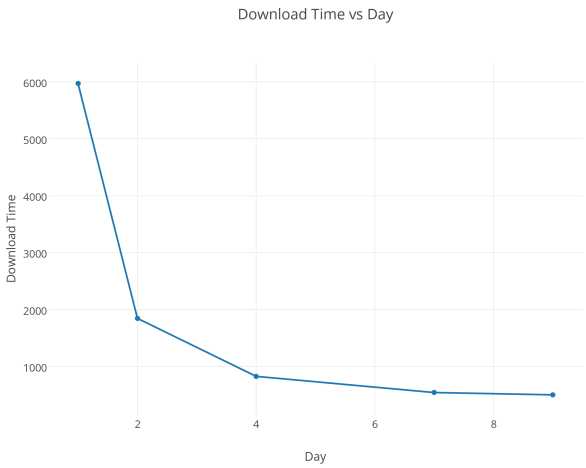

Fig. 11: Mobile DOWNLOAD time vs DAY

X. DESKTOP AND MOBILE COMPARISON

Desktop doesn't have restrictions on cache size whereas in most mobile, cache size is limited.

Some latest mobile with new operating system allows user to change cache size but it is not advisable.

One most research area in mobile is how to increase network speed in mobile whereas high bandwidth speed can be achieved in desktop system.

Generally, desktop operating system such as Ubuntu doesn't interfere with browser caching while in mobiles, each operating system restricts browser cache size.

So analysis on Nexus devices(Android operating system) and iPhone(Apple's operating system) would result in different web page load time for same website because of difference in cache size in system.

XI. CONCLUSIONS

Since browsers spend 80% of the time fetching external components including scripts, stylesheets and images, reducing the number of HTTP requests has the biggest impact on reducing response time. Our observations show that HTTP and DNS cache improves page load time on both desktop and mobile devices. Mobile devices have limited cache and the mobile operating systems have restrictions on the browsers so as to how much caching is allowed by the browser application. Web servers should specify caching of objects like images which do not change regularly but take up lot of time to download. Ironically, we notice many web servers specify no-cache in their response which indicates browser to not cache anything for this webpage.

Mobile devices have limited cache so it is better to have minimal website for mobile rather than using same website for both desktop and mobile. As future work we could explore different methods on server side to specify caching of objects based on type (image/JavaScript) and amount of time to cache and its effects on page load time. We augments the Wprof scripts to get DNS lookup time from the JSON object.

APPENDIX

WProf[1] is a tool that extracts dependencies of activities during a page load. For Web developers, WProf can help identify the bottleneck activities of your Web pages. For browser architects, WProf can relate page load bottlenecks to either Web standards or browser implementation choices.

Web page load time is a key performance metric that many companies and people are trying to optimize. Given its importance, many techniques (e.g., SPDY and mod_pagespeed) have been developed and applied to reduce page load time. However, the performance bottlenecks that limit the page load times are still not well understood. The result is that it is difficult to know whether and when proposed techniques will help or harm page load time. This motivates the researchers

of university of Washington to build WProf that helps uncover the complexity of the page load process underneath.

DNS Lookup[17] is a request to another server requesting the ip address for a certain domain name. For any resource to be downloaded for your page, the browser must "look it up". The browser must do this at least once for each domain your webpage is receiving resources from. As a webpage becomes more feature-rich it often uses more DNS lookups which makes it render much slower. This process can slow down webpage considerably and one of the most important decisions we can make about our page speed is how many DNS lookups are too much.

Speed can be improved by removing or deferring dns calls. Even if we can not remove the things that are making the dns calls, we can still improve the speed of your page by not loading them initially. This is often accomplished by deferring the javascript that is not needed for the initially visible above the fold content of your page.

Browser Caching[14] When we visit a website, the elements on the page we visit are stored on local hard drive in a cache, or temporary storage, so the next time we visit the site, our browser can load the page without having to send another HTTP request to the server. The first time someone comes to our website, they have to download the HTML document, stylesheets, javascript files and images before being able to use your page. That may be as many as 30 components and 2.4 seconds.

Once the page has been loaded and the different components stored in the user's cache, only a few components needs to be downloaded for subsequent visits. In yahoo's test, that was just three components and .9 seconds, which shaved nearly 2 seconds off the load time. It says that 40-60% of daily visitors to a website come in with an empty cache, so it's critical that we make our page fast for these first-time visitors. But we also need to enable caching to shave time off subsequent visits.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Prof. Aruna Balasubramanian for the continuous support of our project, for her patience, motivation, enthusiasm, and immense knowledge. We are also grateful to Javad Nejati, Teaching Assistant, in the Department of Computer Science. We are extremely thankful and indebted to him for sharing expertise, and sincere and valuable guidance and encouragement extended to us. Their guidance helped us in all the time of research and implementing the project. We could not have imagined having a better advisor and mentor for our project.

REFERENCES

[1] http://wprof.cs.washington.edu
[2] http://www.alexa.com/
[3] http://developer.android.com/tools/help/adb.html
[4] https://developer.yahoo.com/performance/rules.html
[5] http://yuiblog.com/blog/2007/01/04/performance-research-part-2/
[6] https://devcenter.heroku.com/articles/increasing-application-performance-with-http-cache-headers
[7] https://documentation.cpanel.net/display/CKB/How+To+Clear+Your+DNS+Cache
[8] http://www.aberdeen.com/Aberdeen-Library/5136/RA-performance-web-application.aspx
[9] http://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt
[10] http://www.akamai.com/html/about/press/releases/2009/press_091409.html
[11] http://www.cloudvps.com/community/knowledge-base/how-does-a-cdn-work/
[12] http://en.wikipedia.org/wiki/Progressive_download
[13] http://students.washington.edu/nettevi/
[14] http://blog.crazyegg.com/2013/12/11/speed-up-your-website/
[15] http://developer.android.com/tools/help/logcat.html
[16] http://wprof.cs.washington.edu/tests/
[17] https://www.feedthebot.com/pagespeed/dns-lookups.html
[18] https://www.udacity.com/course/viewer#!/c-ud884