What's more to Weight: Analysis of HTTP Archives

Tanisha Hegde
Dept. of Computer Sciences
University of Wisconsin-Madison
tghegde@wisc.edu

Madhushree Nijagal
Dept. of Computer Sciences
University of Wisconsin-Madison
mnijagal@wisc.edu

***Abstract*— The weight of a webpage is a factor of all its components like the HTML page, embedded images, hyperlinks and third-party resources. With an increase in the use of CSS, JavaScript and other technologies, there is a noticeable growth in the weight of the web. These components have a computation cost and reflect in computational resources to transmit, process and render in a web browser. This gives rise to the need of studying the weight of the webpage in detail. In this project we use the HTTP Archive database which logs historical data of multiple websites and focus on top technologies that have contributed to the weight of the web from the year 2016 to 2022. We look at mainly Advertising Networks, JavaScript Frameworks, Content Management Systems (CMS), Web Frameworks and Ecommerce related technologies and how they affect a webpage.**

Keywords—HTTP Archive, page weight, compile time, execution time

I. INTRODUCTION

The weight of a web page is important and arguably the most important factor affecting creators, hosting providers, and consumers. The average web page weight has grown by 356% in the past decade due to increased use of images, video, audio, fonts, data collection and processing, and connected services. [1] This can negatively impact user experience, especially for those with slow internet connections. Keeping file payloads smaller can reduce overall energy demands and make payload transmission faster and more efficient. Google now penalizes websites' search rankings for poor Core Web Vitals, including page weight. Examining page weight and finding ways to reduce it can improve website performance and user experience. The technology you use plays an important role in deciding the weight of the page. Using JavaScript Frameworks like React may make your website beautiful but at an expense of execution cost. Realizing the importance of technology, we have analyzed the technologies used from popular categories namely Advertising Networks, JavaScript Frameworks, Content Management Systems (CMS), Web Frameworks and Ecommerce in building a website. We try to answer the question – what technologies are heavy and what contributes most to their weight over the years from 2016 to 2022. We compare this for mobile and desktop websites. We further try to find what metrics do these technologies affect on a web browser. We mainly look at the compile time and execution time of a website for it to load on a browser.

II. RELATED WORK

Many studies have been conducted using data from HTTP Archive to understand the impact of page weight on website performance and user experience. "The Web is Still Too Damn Slow" by Tammy Everts [2], is one that analyzes HTTP Archive data and finds that page weight has increased by 186% since 2010, leading to slower page load times and decreased user engagement. "The Performance Effects of Protocol and Web Page Construction" by Steve Souders, [3] analyzes HTTP Archive data and shows that reducing the number of HTTP requests and optimizing web page construction can significantly improve page load times and user experience. There are studies that talk about how does changing an important contributor like images can reduce the page size. "The Impact of Web Page Image Compression on User Experience" by Ke Zhai et al.,[4] shows that image compression can significantly reduce page weight and improve user experience, but also has trade-offs between image quality and file size. In the paper "Web Performance Optimization: Impact of Front-End Frameworks and Application Architecture" by Yingjun Wu et al., [5] we get to see how using front-end frameworks and optimizing application architecture can help reduce page weight and improve website

performance. Overall, these studies and others highlight the importance of page weight for website performance and user experience, and provide insights into strategies for optimizing page weight to improve website performance.

III. DATABASE

The HTTP Archive is an open source project that tracks how the web is built and provides historical data to show how the web is constantly evolving. It is frequently used for research by the web community, scholars, and industry leaders. There are over 1 million pages tracked on desktop and emulated mobile, and the historical data goes back to 2010. The pages that should be tracked come from the Chrome user Experience Report. These URLs are then passed to WebBasedTest to extract an HAR file. These HAR files are parsed by HTTP Archives to populate tables onto BigQuery. By collecting the data using Google BigQuery, analyzing the data becomes easy because all the storage and indexing is taken care of by GCP. [6]

The HTTP Archive provides summary and detailed information about web page loads, which is stored in various tables in the HTTP Archive. The `summary_pages` table includes details such as page load timings, number and types of requests, and caching information. The `summary_requests` table includes information about each individual object loaded by the page. The HTTP Archive also stores detailed information about each page load in HAR (HTTP Archive) files, which are split into multiple BigQuery tables including `pages`, `requests`, `response_bodies`, and `lighthouse` tables. These tables contain information about URLs, performance data, response headers, and Lighthouse audit results. The tables can be very large, with response bodies and lighthouse tables being the largest. A summary of a few of the tables in HTTP Archive dataset are shown in Table 1.

Tables	Description	Size	History Since
Summary _pages	Summary of all desktop and mobile pages	~1GB	2011
Summary _requests	Summary of all HTTP requests	~125GB	2011
pages	JSON-encoded parent document HAR data	~15GB	2016
requests	JSON-encoded subresource HAR data	~1TB	2016

Table 1: Summary of tables in HTTP Archives

IV. BACKGROUND

Technology has a significant impact on page weight. As technology advances, new features and functionalities become available to web developers, but these often come at the cost of increased page weight. This section elaborates on the top five technology categories that have become popular over the years in creating a webpage.

Advertising Networks: Advertising networks are platforms that connect advertisers with publishers to facilitate the buying and selling of online advertising space. Ad networks aggregate ad inventory from a variety of publishers and offer it to advertisers, allowing them to reach a larger audience than they would be able to on their own. Multiple advertising network platforms provide banner ads while others offer video advertising and templates that make the aesthetics appear good. All these factors contribute to an increase in weight of the webpage.

Content Management Systems (CMS): CMS platforms like WordPress, Drupal, Joomla and commercial platforms like Adobe Experience Manager have become a popular choice when it comes to creating, managing and publishing digital content for websites. As we would see in the next sections, these come with a cost of compilation time and load speed due to the contribution of bytes of the webpage.

Web Frameworks: Web Frameworks have become a standardized way to build web applications. With the availability of pre-built components, libraries, and tools it allows developers to focus on the application logic rather than thinking of the underlying infrastructure. These prebuilt components are the largest contributors to the size of the webpage. The number of HML, CSS and JavaScript files used by these frameworks would be the deciding factor for their contribution to the weight.

E-commerce: E-commerce platforms are user-friendly tools needed to create a professional-looking online store. It also helps the business manage their inventory and process payments securely and efficiently. Examples like WooCommerce, is a plugin of WordPress enables sites built on WordPress to be converted into an online store. Creating an online store requires large number of images, videos and other media files to showcase their products and this would contribute to the bulk of the page.

JavaScript Frameworks: JavaScript Frameworks have been in the talk for their fast and easy development of web applications. By abstracting the low-level implementation details, it allows developers to work on the core functionality. Collections of pre-written code and library makes it one of the most popularly used technologies when building a page. React and AngularJS remain a popular choice when building a web application.

The above-mentioned technologies have certain weight and impact differently when it comes to mobile and desktop versions of a webpage. The later sections quantify the exact contribution of these technologies for bytes HTML, bytes CSS, bytes Image, bytes JavaScript, bytes Font and bytes HTML Doc and show how these bytes effect in the CPU compilation and CPU execution time on the browser.

V. METHODOLOGY

We have done our analysis using Google BigQuery and most of the Visualizations using Tableau Cloud. Big Query uses an SQL like syntax for querying in GoogleSQL for BigQuery. To being our analysis, after narrowing down on the top five technology categories to use, we first queried to get the top five technologies in each category and their bytes for HTML, CSS, Image, JavaScript(JS), Font and HTML Doc for mobile and desktop for the year 2016, 2018, 2020 and 2022. The query logic for the same is shown in Fig. 1. We extracted this data to get the median values rather than average to avoid biased results. The bytes stored were scaled to Kilobytes (Kb) to increase readability. This query required using the `summary_pages` table to get the summary of all the weights and the `technology` table to filter the technologies for each category. The computation for this query increased with the years, starting from 4GB to 200GB for desktop and with mobile showing similar trends.

For the second part of the project, to compare the weights of these technology categories we considered the compile time(`cpuEvaluateScript`) and execution time(`cpuFunctionCall`). The `cpuEvaluateScript` metrics would help measure the time in miliseconds(ms) required by the CPU to parse and compile a specific block of JavaScript code which we would call the compile time on a web page. This would be a good measure as the bytes for JavaScript(JS) have increased over the years. The `cpuFunctionCall` is a performance metrics that would measure the CPU time required to execute specific JavaScript function on a webpage giving us an understanding on the execution time by each of these technologies. To get these metrics from the HTTP Archives dataset we used the `pages` table. The pages table has a payload field that contains HAR file extract. To get the technologies we had to parse the JSON out of the HAR file from the `pages` dataset. From the year 2018, the HAR files contain arrays named `_detected` and `_detected_apps`. An example of these arrays is shown in Fig. 2. To extract list of all technology categories we used `JSON_EXTRACT(payload,"$._detected.<Category_name>")` where `<Category_name>` was among the categories we were interested to query. On using our basic version of the query shown in Fig. 3. We used a regex expression to replace any number, space or special character to combine different versions of the same technology. Further, to extract `cpuFunctionCall` we used the `JSON_EXTRACT(payload,"$['_cpu.FunctionCall']")` and type casted the result values to integer using `CAST()` the same was done for the `cpuEvaluateScript` using `_cpu.EvaluateScript` as the field value. Extracting the metrics along with aggregating it became our outer query and extracting the desired categories became our inner query. The complete structure of our query is found in Fig. 4. On getting all this data we did a few visualizations to understand the trends in our extracted data which are elaborated in the next section. We also created a dashboard in Tableau that allows you to select the year, category and metrics to get a complete picture of all the trends. A preview of the dashboard is shown in Fig. 5.

```
1. Join the `httparchive.technologies.2016*` table with `httparchive.summary_pages*` table
using the `url` field as the join key.
2. Calculate the total number of pages, as well as the total bytes for each of the 10
types of resources (total, html, js, css, img, flash, font, json, other, and html doc)
for each `app` and `category` combination.
3. Filter the results to only include categories that are in the list
'Advertising', 'CMS', 'Ecommerce', 'JavaScript frameworks', 'Web frameworks',
and where the total number of pages is greater than 500.
4. For each category, assign a row number to each record based on the number of pages
in descending order.
5. Return only the top 5 records for each category based on the assigned row number.
```

Fig. 1: Steps to get all metrics from summary_pages table.

```
"_detected": {
    "JavaScript Frameworks": "Moment.js,RequireJS,jQuery,jQuery Migrate,jQuery UI",
    "Ecommerce": "Magento 2",
    "Web Frameworks": "Bootstrap",
    "Web Servers": "Apache",
    "Programming Languages": "PHP"
},
"_detected_apps": {
    "jQuery": "",
    "Bootstrap": "",
    "jQuery UI": "",
    "jQuery Migrate": "",
    "Moment.js": "",
    "Underscore.js": "",
    "Magento": "2",
    "Apache": "",
    "PHP": "",
    "RequireJS": ""
},
```


Fig. 2: _detected and _ detected_apps that can be used for JSON Extract

```
SELECT JSON_EXTRACT(payload,"$._detected.Advertising Networks") adnetworks,
       count(*) freq
FROM `httparchive.pages.`
GROUP BY adnetworks
ORDER BY freq DESC
```

Fig. 3: Generic query to get technologies for Advertising Networks

```
1. Scan `httparchive.pages*` where `_TABLE_SUFFIX` contains `_mobile`
2. Apply projection on `payload` and extract `_detected.Advertising Networks`,
`_cpu.FunctionCall`, and `_cpu.EvaluateScript` fields from the JSON object
3. Apply `REGEXP_REPLACE` function on `_detected.JavaScript Frameworks`
to remove numbers and double quotes from the string
4. Cast `_cpu.FunctionCall` and `_cpu.EvaluateScript` fields to `INT64` data type
5. Apply `GROUP BY` on `adnetworks`
6. For each `adnetworks` group, calculate the `count(*)` of rows,
`APPROX_QUANTILES(cpuEvaluateScript, 100)[SAFE_ORDINAL(50)]` and
`APPROX_QUANTILES(cpuFunctionCall, 100)[SAFE_ORDINAL(50)]` percentile values of
`cpuEvaluateScript` and `cpuFunctionCall` fields respectively
7. Apply `ORDER BY` on `freq` in descending order
8. Project the resulting columns `adnetworks`, `freq`, `cpuEvaluateScript`,
and `cpuFunctionCall`
```

Fig. 4: Steps used to create the final query for mobile

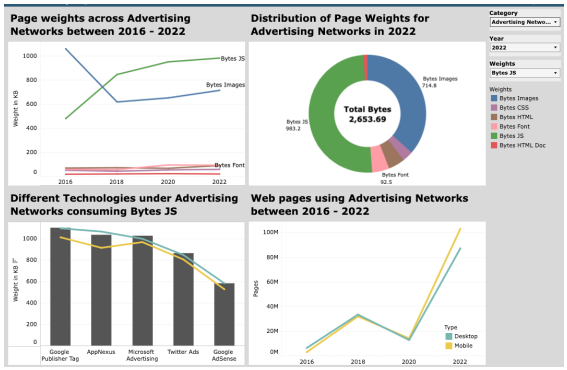

Fig. 5.a: Dashboard giving an overview of page weights

Fig. 5.b: Dashboards used to see compile time and execution time

VI. Results

Each of the technology categories has a different kind of impact on the parse and execution time. We describe the behavior for each of the categories in detail with regards to the metrics' trend for years 2016, 2018, 2020 and 2022 and the effect of these weights on parse and execution time for the year 2018.

A. Advertising Networks

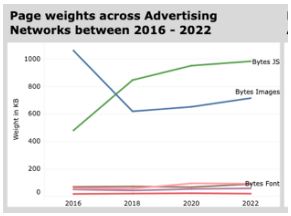

Fig. 6.a. Bytes JS increase by 2022

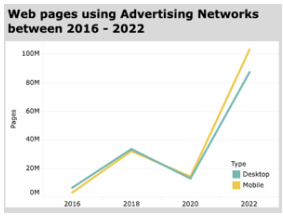

Fig. 6.b: Web pages using Advertising Networks

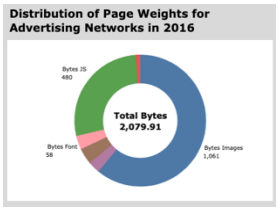

Fig. 7.a: Donut chart for 2016

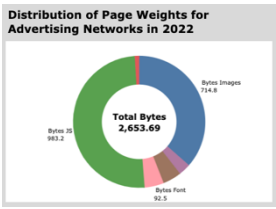

Fig 7.b: Donut chart for 2022

From the years 2016 to 2022 the bytes consumed for JavaScript has increased by 100%. We notice that the usage of Images decreases from the year 2016 to 2018 from ~1000Kb to ~600Kb and increases by only 15% in the year 2022. As seen in Fig.6.b the popularity of advertising networks on mobile webpages increases by 13% in the year 2022. Fig.7.a and Fig.7.b shows that in the year 2016 bytes images contributed around 60% to the weight of a webpage using advertising networks. The second largest contributor – JavaScript (JS) bytes contributed towards 23% of the weight. In the year 2022, we see that JS bytes contribute towards 50% of the weight while Image bytes contribute 36%. As we focus on the year 2018 to see the effect of these weights on the parse and execution time, we notice that Criteo and Prebid are the deciding factors of the time taken in mobile and desktop. Fig.8.a, Fig.8.b and Fig.8.c, show that Prebid being heavy on bytes JS and Criteo being the fourth highest, require ~4s to compile and ~3s to execute, while on desktop this requires ~1s for each.

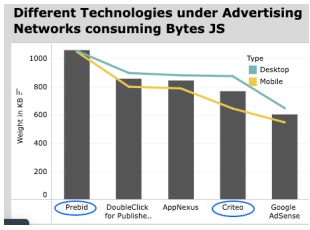

Fig. 8.a: Bytes JS consumed by Prebid and Criteo

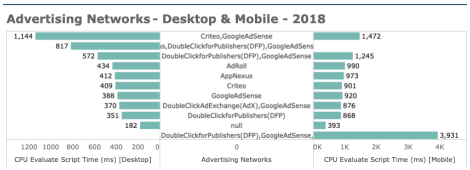

Fig. 8.b: Compile time for Advertising networks

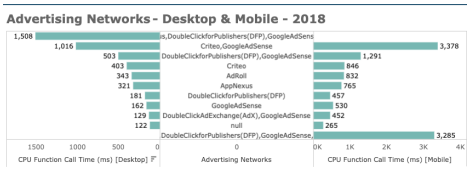

Fig. 8.c: Execute time for Advertising networks

B. Content Management Systems (CMS)

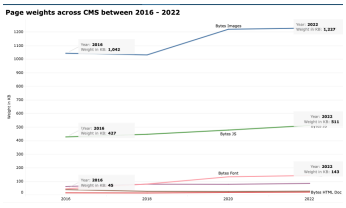

Fig 9.a: Significant increase in bytes Font compared to bytesJS and bytes Image.

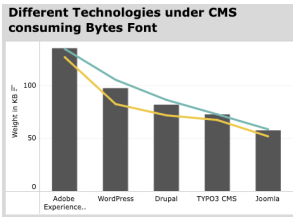

Fig.9.b: Bytes Font consumed by CMS platforms in 2018

In CMS, it is interesting to note that, while the bytes JS and bytes Image increased by ~20% – from 1042kB to 1227kB for bytes Image and from 427kB to 511kB from the year 2016 to 2022, a significant increase is seen in bytes Font. Bytes Font increased from 45Kb to 143Kb resulting in a ~200% increase. This tells us that the contributing factor for CPU processing would be due to fonts along with other metrics. Adobe Experience Manager ranks first for the time required to compile and execution in desktops and in terms of weights it ranks first for bytes Font, and bytes JS. A difference of 38kB is noticed in weight for bytes Font and a difference of 136kB is seen for bytes JS of Adobe Experience Manager when compared to the second heaviest advertising network platform – WordPress. This trend is visible in Fig. 9

C. Web Frameworks

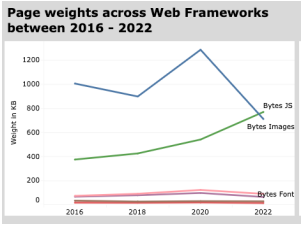

Fig.10: Rise and fall of bytes Image

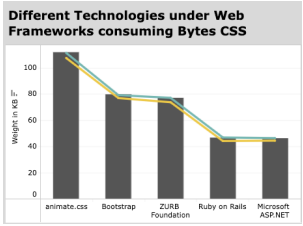

Fig.11: Animate.css highest ranked for bytes CSS consumption

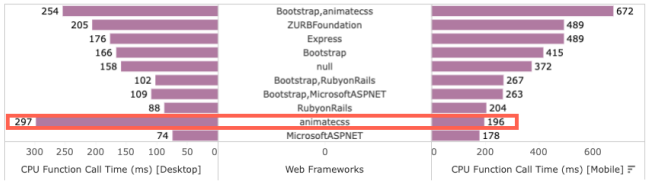

Fig. 12: Animate.css ranks 9th in execution time.

While web Frameworks show a similar trend in the increase of page weight through the years 2016 to 2022 as other categories, we noted a few outliers. We observe that in the year 2020, the weight of images increased by 43% from 2018 but drops down by 45% in the year 2022. This results in bytes JS nearly becoming equal to the weight of bytes Image. Fig. 10 illustrates this trend. Another notable outlier is the time required for parse and execution of the highest web framework when compared to other categories like CMS or Advertising Networks. While these categories take a minimum of ~1s and a maximum of ~3s to compile or execute, the maximum time to compile or execute a web page for web frameworks is ~1s. On referring to Fig.11, we observe that execution time for Animate.css on desktop is ~0.3s but takes less time on mobile – ~0.2s, and ranks 9th among other web frameworks. This outlier deviates from the basic understanding that a high weight in bytes Image, bytes JS and other metrics results in high compile and execution time. As seen in the case where Animate.css ranks the highest for bytes Image, bytes HTML, bytes JS, and bytes CSS.

D. E-commerce

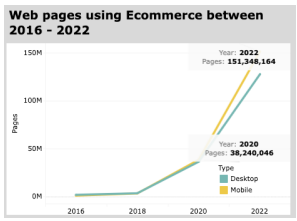

Fig.12.a: Popularity increase of e-commerce platforms in year 2022

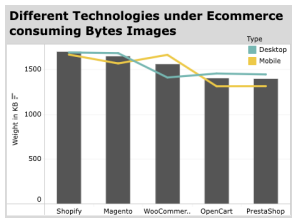

Fig.12.b. bytes Image greater on mobile for WooCommerece

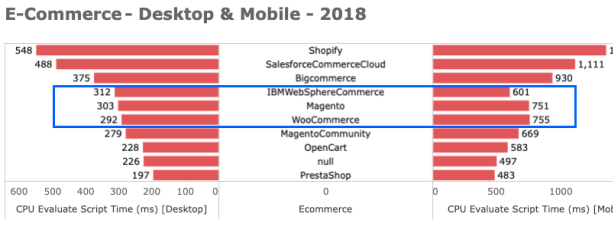

Fig.12.c: IBMSphereCommerece takes less compile time on mobile than desktop

With E-commerce platforms rising in popularity over the years we see that it has gained a sharp popularity after the year 2018 rising to 151 million pages in 2022 from 38 million in 2018. There is an average of 700kB difference between the bytes JS and bytes Image page weight. Looking at Fig.12.a, it can be seen that the use of e-commerce platform for webpages has increased by 100 million for the year 2022. We observe that different e-commerce platforms show similar trends to compile and execute for desktop and mobile with an exception of Magneto and WooCommerce. Fig.12.b shows that the consumption of bytes Image on mobile is more than on a desktop by ~7%. This results in these platforms surpassing IBMSphereCommerce in compile time on mobile.

E. JavaScript Frameworks

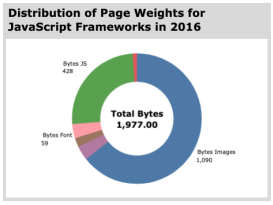

Fig.13.a Bytes Image and Bytes JS having nearly 1:3 ratio in 2016

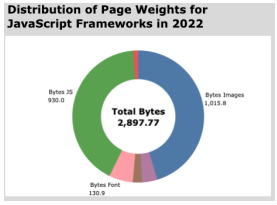

Fig.13.b: Bytes Image and Bytes JS having nearly 1:1 ratio in 2022

JavaScript Frameworks and their impact on page weight has been of interest due to its growing popularity. The contribution of weight bytes JS to the total weight has increased from 25% to nearly 45% each. Fig.13.a and Fig.13.b show that the bytes JS have increased over the years by 80% to the contribution of the weight and reducing the gap between itself and bytes Image. This makes the contribution of bytes Image decrease from 65% to 45% a – 15% decrease. In 2018, libraries like JQuery and its variants like JQuery UI and JQuery Migrate are among the top five contributors to weight. From year 2020, these libraries lose their top five ranking to other libraries like GSAP and Handlebars. In year 2018, we observed that AngularJS is not among the top five frameworks contributing to page weight, but for compile and execution time it ranks first for desktop and is in top five for mobile. There is ~1s difference in the compile time between the highest – React, to the lowest – JQuery, on mobile. This observation is justified by the large difference seen in bytes JS of React compared to JQuery. While JQuery contributes ~400kB of page weight, React contributes ~800Kb. Fig.14a, Fig14.b and Fig.14.c help in understanding the same.

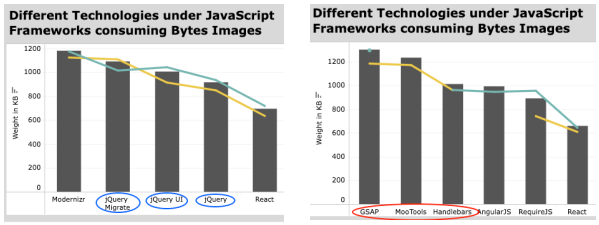

Fig. 14.a: JQuery and its Variants in top five for bytes Image in 2018 lose their spot in 2022 with GSAP, MooTools and Handlebars ranking top three

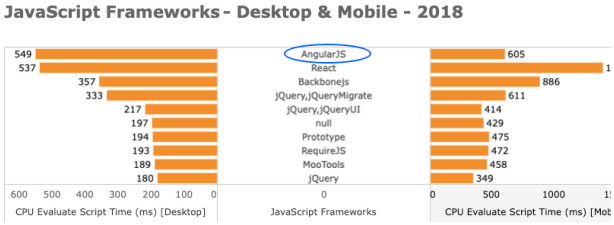

Fig. 14.b: AngularJS ranks first for CPU compile time

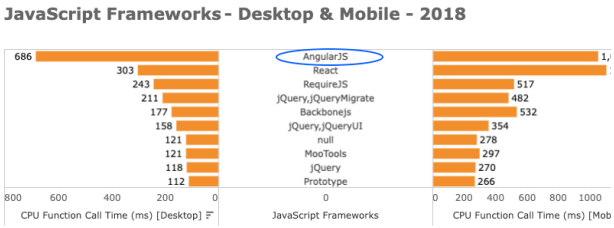

Fig. 14.c: AngularJS ranks first for CPU execution time

VII. CONCLUSION

With this analysis, we conclude that, over the years the weight of a web page has increased due to a large increase in images and JavaScript used while creating websites. The increase of these factors has a direct impact on compile and execution time resulting in slow page load time. Slow page load times can lead to a poor user experience, frustration, and even a higher bounce rate and overall making the website unappealing. This is the main reason why page load speed is one of the factors that search engines consider when ranking websites. Websites with faster loading times tend to rank higher in search results. This gives website creators a clear incentive to keep web pages light. With high end smartphones, high computation power and a 5G network, weight of a webpage may not matter but internet uncertainty and webpage accessibility for all phones should be the desired goal. A heavy webpage also consumes high bandwidth, especially on mobile devices. This can lead to higher data usage and increased cost for users with limited data plans. Using new technologies

for creating websites makes you focus on the problem than implementation but comes with a cost. Developers must be mindful of this tradeoff and find ways to reduce page weight. Common ways to reduce weight are to minimize and compress the code, defer or asynchronously load non-critical resources, and use a content delivery network (CDN).

VIII. Future Work

While looking at optimization techniques to reduce page weight, a common technique is using a content delivery network (CDN). The next step in this analysis would be to find the effect of CDN on page weight. It would be interesting to track a few common websites and their page weight journey through the years. An interesting question whose answer can be found using the HTTP Archives dataset – Which websites have adopted CDN over the years, and how has their page weight changed over time? Does CDN really reduce the weight of the webpage? There are many such interesting questions that can be found using the HTTP Archive dataset.

IX. References

[1] "Part IV Chapter 19: Page Wight" Web Almanac by HTTP Archive. Available online: https://almanac.httparchive.org/en/2019/page-weight

[2] Everts, Tammy. "The Web is Still Too Damn Slow." Smashing Magazine, 26 Feb. 2018, https://www.smashingmagazine.com/2018/02/web-performance-monitoring/.

[3] Souders, Steve. "The Performance Effects of Protocol and Web Page Construction." In Proceedings of the 20th International Conference on World Wide Web, 2001.

[4] Zhai, Ke, et al. "The Impact of Web Page Image Compression on User Experience." Proceedings of the 24th International Conference on World Wide Web. International World Wide Web Conferences Steering Committee, 2015.

[5] Wu, Y., Bao, X., Li, Y., Li, H., & Liu, W. (2019). Web performance optimization: Impact of front-end frameworks and application architecture. Journal of Systems and Software, 154, 119-134.

[6] Mattson, Evan. "Getting Started Accessing the HTTP Archive with BigQuery." HTTP Archive. Accessed May 7, 2023. https://github.com/HTTPArchive/httparchive.org/blob/main/docs/gettingstarted_bigquery.md

[7] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.