Performance Analysis between PWA and Android: DRAFT

Dishant Kaushik

Department of Software Engineering, Rochester Institute of Technology
Rochester, NY, USA
dxk3597r@rit.edu

Abstract— **Making an idea a reality is getting tougher for young startups as more and more platforms are arising. A typical product to be good has to have an Android and an iOS flavor. This not only create a burden for the developers, but also brings inconsistency between apps because of the different implementations thus providing different experience to people with different devices. Progressive web app was designed to have web apps act more like a native web app by providing offline storage, background sync and notifications without having the bloatware of hybrid apps. The user goes the website and he/she gets the app instantly. This proposal aims to research about the performance of progressive web apps and compare it with the same type of native Android apps with terms of memory usage, storage, time to first paint and other performance metrics.**

Keywords— **PWA, Service Workers, Performance, Android, Development.**

I. Introduction

In today's world, an average person uses a mobile phone for getting most of its content. App developers are in demand and a lot of companies are opting for a mobile first development strategy. A major difficulty amongst these companies is to create apps with unified interface for different platforms like Android, iOS, Windows. For an application, they have to design, develop and manage/update three versions of their application. According to [1], Google has recently developed service worker, an API which lets web app behave like a native application. These native behaviors can range from offline functionality, push notifications and even battery sync. This makes development fast, deployment easy and gives a universal interface to all kinds of devices. Many companies are switching to this strategy. This paper [2] has observed that data intensive applications are best for making progressive web applications. This paper [3] also observes that there is no energy impact on the device by using service workers.

The research is an empirical analysis to compare native android apps and their respective progressive web applications in terms of time to first paint and initial processing time when compared with two different types of android phones. (A low end and a high end) on different network status. This research is required to find out whether service worker is any better/ worse than making a native android application and help developers make better architectural decisions.

As we were looking for progressive web applications in the web right now, we realized that many of them are from a well known big companies. They also have an android application and hence they are best to test the difference between the data delivery, the user interface and the performance.

II. Research Objective

There is always a misconception that building native apps is better than making web application when performance, memory and battery usage is a concern. And it was true for the most part until service workers came into play making progressive web apps possible. This research will be a turning point of what people think about what a web app can do. By seeing the performance of progressive web app on a low end phone and comparing it with a native web app will help us prove the mission of service worker. If the performance is comparable, we can say that it is easier for a lot of developers and small teams to make a progressive web app as compared to developing a native application for each platform. If the performance is better, this research can encourage a lot of developers to give progressive web app a try.

III. Research Question(s)/Hypothesis

RQ1. Is PWA faster than native apps in terms of time to first paint ?

RQ2. Does PWA take less memory than native apps.

RQ3. Does PWA perform better or worse on high end phones as compared to native apps?

RQ4. Does PWA perform better or worse on low end phones as compared to native apps?

IV. Literature Review

"Beyond Native Apps: Web Technologies to the Rescue! (Keynote) "[1] talks about how mobile software development teams can follow a number of different development strategies which ranges from native apps and mobile web apps. This resonates with my problem statement because it talks about the recently emerging progressive web app which by the paper is defined by "special kinds of mobile web apps in which progressive enhancement, support for low or no network connectivity, background processing capabilities, push notifications support, and security are the first class characteristics " [1]. One of the research challenges can be the collection, measurement and analysis of performance, reliability and security of Android vs Progressive Web Applications. Also, PWAs are described as "performance boosters, network saver and providers for better user experience."[1] and hence checking these parameters could be a research topic on its own.

The second paper wants to know why mobile browsers performs poorly as compared to desktop operating systems and what are their bottlenecks [4]. Because mobile have lower WiFi chips and Mobile Networks are flaky, I think that that causes bottleneck. By flaky, I mean a connection which are having some trouble getting or processing data with other disturbance. Since the paper was a bit old, (2016) and since processors like the 835 chips are far better in doing even machine learning algorithms, the first bottleneck is mostly not the main one anymore. The second bottleneck can be removed partially by leveraging smart request and response caching[4]. This can at least have the functionality ready and not latest data.

Maybe there are other consideration other than performance and memory that we can compare between Native apps and PWAs. The paper talks about Software Engineering Issues for Mobile Application Development [5]. They include the user experience, non functional requirements like efficient use of resources, robustness, connectivity, scalability [5], The processes, tools and architecture, portability, etc. I realized that most of these issues are a flavor of traditional software engineering where portability is the main concern. It is restricting software to portable environments [5]. From user experience to performance, everything should be considered as an efficient use of space.

In order to find out whether a web based solution to mobile apps is actually required, this paper about an exploratory investigation about hybrid mobile application in the Google Play Store [6]. It was intriguing to know that Amazon, the biggest e-commerce player makes it apps using html views as their user interface in their mobile apps. So even they use web based applications. IBM and Adobe are advocating hybrid mobile apps [6], which are a pseudo native apps built from web technologies. Out of 11,917 application from the Google Play Store, the paper found out that 445 apps are hybrid. On one side, the result shows that they are un-common among the top-500 apps from the Play Store[4]. On the other hand, some hybrid apps are from the top developer [6]. There is another paper which shows why these are not popular yet.

The paper, "End Users' Perception of Hybrid Mobile Apps in the Google Play Store" [2] is a survey paper on how the users think of a hybrid application by analysing the traits and distinctions of hybrid apps from end users perspective by mining 11,917 apps and 3,041,315 reviews [2]. The results show that there are some improvement that can happen by introducing progressive web apps. This can extend the research to checking the behavior of hybrid apps across platforms like iOS and Android.

Looking at the battery consumption on both native and PWAs, the following paper talks about doing it for native. The author created a tool which measure the precise energy used by a mobile browser to render web pages. They use this tool to measure the energy needed to render financial, e-commerce, email, blogging, news and social networking sites. They also provided recommendations on how to design web pages so as to minimize the energy needed to render a page. They also show that how Wikipedia reduced the energy consumption by 30% with changes in the scripts with no change in user interface [7]. By the data they collected, they came up with a few tips on designing energy-efficient web sites. They are:

- Using JPEG for all image formats.
- Use links instead of Javascripts to redirect to another web page.
- A number of static pages can be locally cached and displayed without network access.
- We should position elements using tables instead of css, which takes far more energy to render. [7]

This paper gave a lot of insights about what to look for and what to change in order to reduce the energy consumed by Web Apps.

A paper which has the same goals as our paper was about Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps [3]. It started off by saying that “Mobile Web Apps is a big part of the internet nowadays. However they are still lagging behind native apps in terms of the user experience and PWA can bridge this gap.” [3] Mobile Web Applications is growing a lot since the consumption of digital media has increased by 62%. The browsers are becoming full fledged software platforms which provide APIs for geolocation, accessing the camera and microphone. The goal of this paper is to see the impact of energy efficiency of PWAs. This paper is one of the first empirical investigation involving service workers and progressive web apps which shows that there are no significant impact over the energy consumption [3] of different devices regardless of the network conditions.

The next paper is about “Mining Energy-Greedy API Usage Patterns in Android Apps” [8] where the author talks about which APIs cause more energy than others and which ones are most frequently used in many Android applications. They took 55 apps from different domains and measure the energy consumption of the method calls invoked by them. Some design and implementation practices like MVC or information hiding poses a non-negligible impact on the energy consumption. [8] They say that the developers should choose their architecture wisely while designing the architecture.

To find out more about how to see the energy consumption of mobile devices, the paper “Why are Web Browsers Slow on Smartphones?” [9] examines the internals using WebKit which is an API for Android. One of the things they find out that more than half of the webpages are not optimized for mobile devices [9]. The flow of opening a web page is a is resource loading, HTML parsing, style formatting, scripting and then updating IR which does the painting on the hardware layer. The round trip time of the web cycle can be reduced by having low amounts of requests per site. This can be done by either having a middleware or caching some resources.

V. Detailed Plan to Conduct Research

In this research we will first define the metrics that we are going to use to compare the progressive web app and android app. The first metric will be time to first paint. Time to first paint is time taken by an application between when a user first clicks the application icon and when the user can interact with the app after all of its content has been loaded and functional. This will be calculated in chrome tools for PWAs and Android Debugger tools for Android Apps. We will first try to run this on Facebook Android and Facebook PWA. After this we will run the same task on both the apps to see the battery usage in an hour. We took an hour because we wanted to compare how different application and its type of application takes how much percentage of battery. This can be calculated by the Android Device Management System. It will not be an accurate measure, but it will tell us the approximate difference between the two apps to be compared.

This will determine the energy usage of the app. Then we will take 10 products which have both a PWA and an Android Application and do the same tests. The number is not final because I have not found products which have both an Android application and the PWA. This will be done on two different devices. One of which is a high end device which has the best processor in the market. Another one will be a low end phone which has the slowest CPU performance. As we do not need much RAM for one application, RAM will not be a concern. At last, we will determine the base memory of the app and the data usage for each app and its subsequent PWA. The first paint will be determine by the number of calls made to the server on the first time of opening the app and its subsequent calls. We will also test them on offline mode and on flaky conditions and see the battery usage on each mode. To mimic the flaky condition, we will use the Google Chrome throttle plugin for the network. This will help us to properly evaluate the research. Time to first meaningful paint is a metric which measures the time taken by the application in which the user can interact with the application. This can be a case when the app has loaded all of its components.

This will be measured by the RAIL principle which consists of Response, Animation, Idle and Load. We see the time taken for all these activities to better analyse the performance of the product. The Response time is when the user taps a button, animation is when the user scrolls the page, or sees an animation. Idle time is the main method of the application to work, which is when the user cannot interact with the page, but the main thread should be available enough to handle the next user input. Load time is when the user loads the page and sees the critical path content. This will be for the PWA. For the Android metrics, we can see the time taken to complete onCreate(), onStart(), onResume and layout rendering. We can also see the frame drops for all the animations using the devtools of android for PWA and Android. We still have to find out the phones we are going to try it on and the apps that we are going to use.

VI. Detailed Plan to Evaluate Research

With the results we can answer our research questions. Let us first reiterate the research questions.

RQ1. Is PWA faster than native apps in terms of time to first paint ?

Analysing the RAIL data from the web app and the android app will help us analyse performance for different scenarios. These scenarios could be similar calls made by the PWA and Android. We are choosing the RAIL principle because there are many performance metrics which can tell us about the performance, but sometimes they might not matter much. For example, an application might perform certain functions faster than the other, but to the end user, it might look just the same.

RQ2. Does PWA take less memory than native apps.

Analysing the data taken by the progressive web app and android apps can help us figure out which apps takes more memory. We can also analyze the data after the app is installed and data when the app is just installed to find out the usage of disk space of the base app. This can help us a lot to see what the final product can look like when you develop an app using either of the technology.

RQ3. Does PWA perform better or worse on high end phones as compared to native apps?

Seeing the first time to paint and the time to load certain resources under normal and flaky internet connection can tell us about which platform process the data faster, or which of the, uses less bandwidth to perform the same API call.

RQ4. Does PWA perform better or worse on low end phones as compared to native apps?

The RQ4 is all the previous research questions done on two different devices. As app developers, there is always the area that an app can cover. The table below shows the distribution of people using different android versions.

Version	Cumulative Distribution
4.0 ICS	99.2
4.1 Jelly Bean	91.4
4.3 Jelly Bean	96
4.4 KitKat	90
5.0 Lollypop	71.3
5.1. Lollipop	62.6
6.0 Marshmallow	39.3
7.0 Nougat	8.1
7.1 Nougat	1.5

As we can see, that we have to balance the usage of our app with the features we add. This can be different for a web app because it does not interfere with the Android API levels and uses the HTTP standard. This is very important to find out because in the end the companies want as many people as possible to use their application.

VII. References

[1] I. Malavolta, "Beyond native apps: web technologies to the rescue! (keynote)," in Proceedings of the 1st International Workshop on Mobile Development - Mobile! 2016, 2015, pp. 1–2.

[2] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, "End Users' Perception of Hybrid Mobile Apps in the Google Play Store," in 2015 IEEE International Conference on Mobile Services, pp. 25–32.

[3] I. Malavolta, G. Procaccianti, P. Noorland, and P. Vukmirovic, "Assessing the Impact of Service Workers on the Energy Efficiency of Progressive Web Apps," in 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), pp. 35–45.

[4] J. Nejati and A. Balasubramanian, "An In-depth Study of Mobile Browser Performance," in Proceedings of the 25th International Conference on World Wide Web - WWW '16, 2015, pp. 1305–1315.

[5] A. I. Wasserman, "Software engineering issues for mobile application development," in Proceedings of the FSE/SDP workshop on Future of software engineering research - FoSER '10, 2009, p. 397.

[6] I. Malavolta, S. Ruberto, T. Soru, and V. Terragni, "Hybrid Mobile Apps in the Google Play Store: An Exploratory Investigation," in 2015 2nd ACM International Conference on Mobile Software Engineering and Systems, pp. 56–59.

[7] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who Killed My Battery?: Analyzing Mobile Browser Energy Consumption," in Proceedings of the 21st International Conference on World Wide Web, 2011, pp. 41–50.

[8] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, R. Oliveto, M. Di Penta, and D. Poshyvanyk, "Mining energy-greedy API usage patterns in Android apps: an empirical study," in Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014, 2013, pp. 2–11.

[9] Z. Wang, F. X. Lin, L. Zhong, and M. Chishtie, "Why are web browsers slow on smartphones?," in Proceedings of the 12th Workshop on Mobile Computing Systems and Applications - HotMobile '11, 2010, p. 91.