International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249 – 8958 (Online), Volume-9 Issue-5, June 2020

Performance Assessment of Caching Techniques in PWAs

Alankrit Gupta, Ritika Jain, Uma Tomer

Abstract: Mobile software development is an emerging technology. The aim of working on this technology is to make it user-friendly and improve the user experience. This paper focuses on the advancing technology - Progressive Web Apps (PWAs). These apps combine the experience of both native and web applications. Progressive web apps are cross-platform developed which means that the app should function both on Android and iOS platform. While web/hybrid/native apps are costly to build, PWAs are way cheaper to build. These apps provide far better user experience than the conventional native/web/hybrid mobile applications. The service worker is the foundation of a PWA. Service Worker enables caching of assets and controls the network traffic. Manifest file lets the app to be installed on the user's device. Different caching techniques are discussed in the paper and their performance has been monitored. The performance of the Progressive Web App is analyzed using Blazemeter as Remarkable growth has been seen in the performance of several business platforms after the implementation of progressive web apps. This paper assesses: (1). The difference in features of Native/Mobile Web/ Hybrid Web Mobile with PWAs, (2). Performance Analysis of the caching techniques in PWAs.

Keywords: — Progressive Web App, Service Worker, Manifest, Caching, Performance Analysis, Cross- platform

I. INTRODUCTION

Progressive web apps are software applications that are developed so that the main features of an application can work with or without an internet connection [1]. These apps do not need to be installed either from the Google play store on Android or from the Apple App Store for iOS. Due to its Cross - platform approach, the feature of adding it to the home screen turns out to be very useful. The word "progressive" means that the application is functional even if the user has an unsupported browser. Background sync with the server allows the PWA to function even in the absence of internet connection. They have numerous advantages over native, hybrid and web apps. These apps provide far better user experience than conventional mobile or web applications. Services like push notifications and app like icons increase the activity of the user and help the user to get acquainted with the app in a short period of time. So, these apps act as an envelope that covers both the native applications as well as web applications.

Revised Manuscript Received on June 15, 2020.
* Correspondence Author
Alankrit Gupta*, Student, Department of Computer Science & Engineering, Akhilesh Das Gupta Institute of Technology and Management, New Delhi, India. E-mail: alankritgupta091099@gmail.com
Ritika Jain, Student, Department of Electrical & Electronics Engineering, Akhilesh Das Gupta Institute of Technology and Management, New Delhi, India. E-mail: jainritika181999@gmail.com
Uma Tomer, Assistant Professor, Department of Computer Science & Engineering, Akhilesh Das Gupta Institute of Technology and Management, New Delhi, India. E-mail: uma.tomer@gmail.com

II. COMPARATIVE STUDY OF NATIVE/WEB/HYBRID APPLICATIONS

A. Native Applications

These applications are platform-dependent which means that the codes are specifically written for different platforms like Android, iOS etc [2]. One of the challenges faced by these applications is fragmentation. Fragmentation is the event that occurs when some mobile devices use older versions of OS while others use newer versions.[3]Native apps add on to the local storage of the device and require more time to access with areas having bandwidth 2G and less than that or 3G network. Usually, Java is used to build native apps for Android and Swift or Objective-C for iOS. The cost of building a native app is more than hybrid apps. Native apps must be compatible with their respective operating systems which provide security and device compatibility.

B. Mobile Web Applications

These apps are developed using HTML, CSS and JavaScript and require web browsers to function. These are very similar to the native applications but are executed in a web browser [2]. They look like native apps but really are websites. They occupy less device memory and necessarily require an internet connection for their functionality.

It is not essential to install these apps but can be used on a web browser. Web apps condense the website content to improvise functionality.

C. Web-Based Hybrid Mobile Applications

These apps are partial native and partial web apps. They acquire the attributes of both native and web applications and are more efficient. These are developed using web technology and can be distributed on any mobile platform. The frameworks like Kotlin, React Native etc are used to design these applications support major platforms. The native wrapper is used to collect the best from both native and web apps [5]. Hybrid apps are easier to scale which means that it is easier to launch the app on another platform.

D. Progressive Web Applications (PWAs)

These apps provide user the experience of both a website and a mobile application. It can be installed and has an icon added to the home screen of mobile device. It has two main components which are responsible for its activity, service worker and app manifest file.

Service worker is responsible for caching the pages upon the installation of the app. Manifest file is responsible for executing the features like push notifications. Working in the offline mode outstand them from the native and other web applications. These apps need to be installed once and work even when they lose network connection or slow network. They are available on secure origins served through TLS, using the HTTPS protocol.

Table I represents the feature comparison of PWA with Native/Web/Hybrid Applications.

Table I. Feature Comparison of PWA with Native/Web/Hybrid Applications.

Feature	*Native*	*Mobile Web*	*Hybrid Web-Mobile*	*PWA*
Cross-Platform	X	✓	✓	✓
Installable	✓	X	✓	✓
Offline-Capabities	✓	X	✓	✓
Performance	Fast	Depends on Network	Relatively slow	Fast
Size	Large	Very Small	Large	Very Small

III. MANIFEST FILE

It is referred to as the file which is required so that the application is identified on a mobile device. The information regarding the application is held by the JSON (JavaScript Object Notation) file [5]. This file tells the browser how the application will behave when installed on a device [6]. It also contains the icon of the application to be installed along with splash screen settings. It gives the user a better experience and provides quick access. The user can modify the appearance, theme, and the other regions they expect [7]. Fig. 1. Shows the example of a manifest file.

```
{
  "short_name": "Demo App",
  "name": "Demo PWA",
  "icons": [
   {
    "src": "favicon.ico",
    "sizes": "64x64 32x32 24x24 16x16",
    "type": "image/x-icon"
   },
   {
    "src": "logo192.png",
    "type": "image/png",
    "sizes": "192x192"
   },
   {
    "src": "logo512.png",
    "type": "image/png",
    "sizes": "512x512"
   }
  ],
  "start_url": ".",
  "display": "standalone",
  "orientation": "portrait",
  "theme_color": "#2A2A72",
  "background_color": "#ccc",
  "prefer_related_applications":false
}
```

Fig. 1. Example of Manifest File

IV. SERVICE WORKER

Service Worker is a very important building block of a progressive web app. It is basically a JavaScript file that is executed separately from the main browser thread, caching or retrieving resources from the cache and intercepting network requests. It is responsible for loading the app on multiple devices, controlling push notifications and other assets like handling network traffic. Fig. 2. Shows the working of a Service Worker in the offline mode.

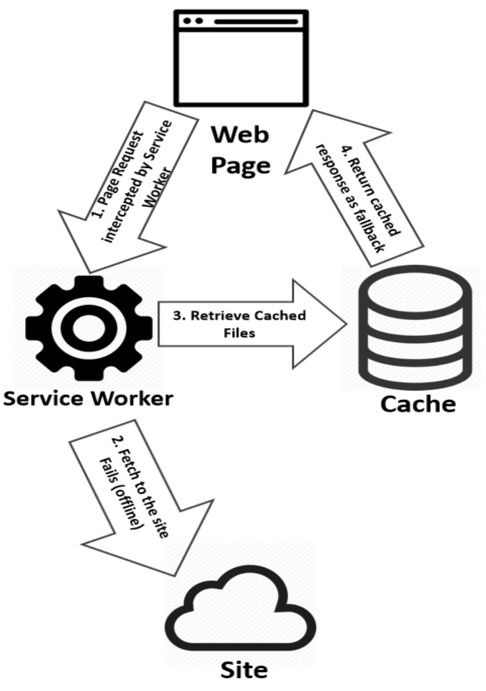
Web Page
1. Page Request intercepted by Service Worker
4. Return cached response as fallback
3. Retrieve Cached Files
Service Worker
Cache
2. Fetch to the site Fails (offline)
Site

Fig. 2. Working of Service Worker when Offline

Normal Web apps need a stable internet connection to work properly. When network connection fails, these apps fail to work and that's when service worker comes in to picture in case of PWAs [6]. The cached assets are loaded, making them work offline.Service worker adds dynamism to an app and increases its performance by helping to display and dropping the right content and background programs [3].

A. Working of Service Worker

Since most of the groundwork of PWA is done by the service worker, it becomes crucial to check the compatibility of the service worker with the web browser first. An application without a service worker cannot be called an application, it is merely a website. It is a JavaScript file which is executed separately from UI. Service worker is a property of the navigator object of the window. It has to be registered first. Registration can be done on every page load as this process doesn't start from the beginning every time. If service worker is new or updated, the installation process will be initiated. The mechanism of a service worker accommodates three major events:

1) **Install**: This is the very first event which is triggered when an app is visited on any device. Its main function is to load the service worker so that the assets of the website can be cached.

2) **Activate:** This event is triggered to modify (update/remove) the previous data which was cached. This event is the ideal location for removing cached static content if it has a newer version.

3) Fetch: This event is triggered every time a response is fetched from the server. It is triggered multiple times. Network traffic can be tracked and managed from this event.

B. Types of Caching

Caching techniques that can be employed with the service worker to have an ideal experience are:

1) Precache: The content files and assets are cached all at once when the 'install' event is triggered i.e. while the service worker is being installed.

Fig. 3. gives the code to be added in install event.

```
// Open cache
caches.open(cacheName) //cacheName: Name of Cache
    .then(cache => {//adding all the assets to cache
        cache.addAll(cacheAssets);
        //cacheAssets: Array of all the static Assets
    })
```

Fig. 3. Code to be added in install event

The developer has to manually enter the path of all the assets and files which is a tiring and cumbersome task. All the assets are mentioned in the array and are cached by the service worker at once.

2) Dynamic caching: The assets or content files are cached in the "fetch" event. All the assets from the response which are requested by the browser are cached here all at once.

Fig. 4. gives the code to be added in fetch event.

```
// Make copy of response
const resClone = res.clone();
// Open cache
caches.open(cacheName)//cacheName:Name of Cache
 .then(cache => { //returning resClone to Initial Request
   cache.put(e.request, resClone);
   //e.request: Initial Request
});
```

Fig. 4. Code to be added in Fetch event

Dynamic caching turns out to be very useful in those cases where there are a lot of assets to be cached and adding path manually is not practically possible.

V. PERFORMANCE ANALYSIS

Since Caching plays a very important role in PWA and different caching techniques have been discussed above, a test was conducted on a 2-page static website using Blazemeter [8] to check performance of website performing different caching techniques. Table II represents the cached assets of the website.

Table II. Cached Assets of the website

Name	*Content-Type*
/about.html	text/html; charset=UTF-8
/css/style.css	text/css; charset=UTF-8
/index.html	text/html; charset=UTF-8
/js/main.js	application/javascript; charset=UTF-8

A. Precaching

Test Configuration 1
Server Location: US East (Virginia, Google)
Time: Feb 01, 2020, 12:01:05 PM
Table III represents results and observations of Precaching Technique.
Fig. 5. shows the response time of Precaching.

Table III. Test 1 (Precache)

	Max Users	*Duration (min)*	*Avg. Response (ms)*	*Avg. Throughput (No. of hits/ sec)*
Precache	20	20	0.93	2065.43

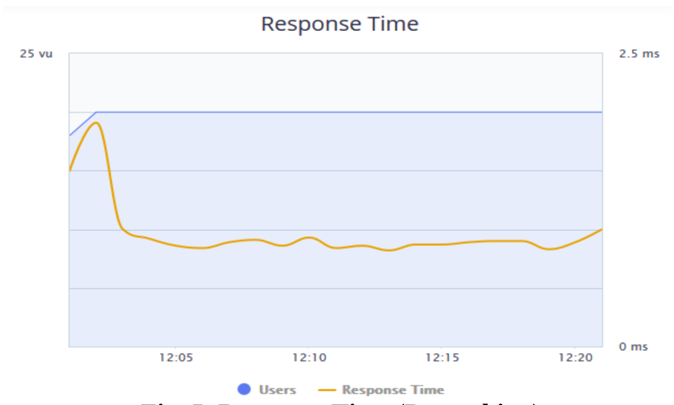

Fig. 5. Response Time (Precaching)

B. Dynamic Caching

Test Configuration 2
Server Location: US East (Virginia, Google)
Time: Feb 01, 2020, 7:45:40 PM
Table IV represents results and observations of Dynamic Caching Technique.
Fig. 6. shows the response time of Dynamic Caching.

Table IV. Test 2 (Dynamic Cache)

	Max Users	*Duration (min)*	*Avg. Response (ms)*	*Avg. Throughput (No. of hits/ sec)*
Dynamic cache	20	20	0.96	1971.73

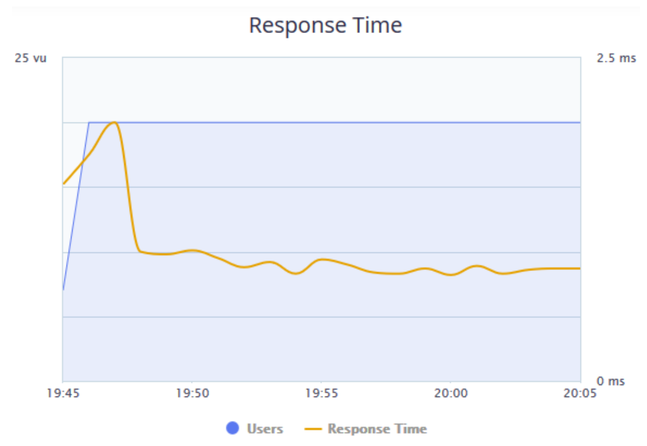

Fig. 6. Response Time (Dynamic Caching)

After analyzing the results of the tests performed, it is observed that the response time for PWA turns out to be almost same in both the cases. The response time may change depending on the size of the PWA. Be it precaching or dynamic caching similar graphs are obtained in both the cases. So, it is upto the developer, what approach he wants to use. But dynamic caching is highly recommended as it reduces the pain of adding routes of all the assets manually and is a practically more feasible for larger websites.

VI. CASE STUDY: TWITTER LITE

The mobile website of Twitter had many issues which twitter wanted to overcome. Slow mobile connectivity and limited space on the mobile devices were few of the problems. This ended up in a reduction in the number of visitors on Twitter's website.

Twitter was looking for features such as instant loading, lower data consumption, and large accessibility. Hence, they switched to PWA.

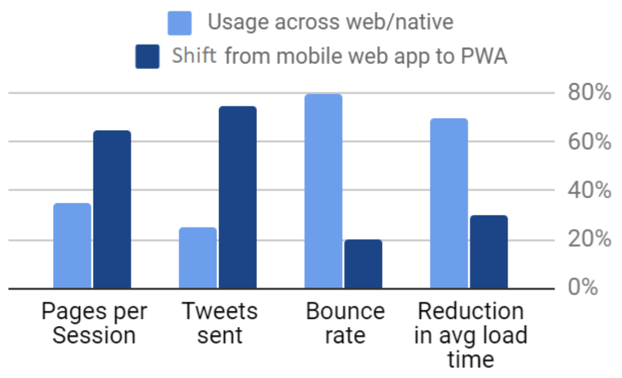

Fig. 7. Growth of Twitter after PWA

The results are as follows: [9]

1) The bounce rate reduced from 80% to 20%
2) Rise in pages per session increased to 65%
3) Number of tweets send increased to 75%
4) Reduction in the average loading time.

VII. FUTURE SCOPE

Progressive web apps are the ultimate future of mobile software development. It is a huge upgrade in the responsive web apps. Google uses an acronym- FIRE – Fast, Integrated, Reliable, and Engaging to explain why PWAs are so effective [10]. With the latest browsing technologies coming, these apps combine the accessibility of the web with mobile apps. Currently the features of a progressive web app are supported by various degrees of web browsers like Microsoft Edge, Google Chrome, Mozilla Firefox, Apple Safari but more web browsers may support the features in future.

VIII. CONCLUSION

It is concluded through this paper that Progressive Web Apps are eventually the best technology and the future of web development to improvise the user experience and overcome the problems faced in conventional native web applications. The working of the Service Worker is discussed to improve the caching process. Performance Analysis has been done on different caching techniques of a PWA, pointing out their advantages and disadvantages.

REFERENCES

1. A. B. Hansen, T. A. Majchrzak, and T. M Grønli, **(2018)** "Progressive Web Apps for the Unified Development of Mobile Applications". International Conference on Web Information Systems and Technologies. WEBIST 2017: Web Information Systems and Technologies. Lecture Notes in Business Information Processing, vol 322. Springer, Cham. pp 64-86.
2. I. Malavolta, "Beyond Native Apps: Web Technologies to the Rescue!
3. (Keynote)." ACM Mobile!'16, Amsterdam, Netherlands, **(2016)** October pp. 5–6.
4. R. S Mishra, "Progressive WEBAPP : Review". International Research Journal of Engineering and Technology (IRJET) Volume: 03 Issue: 06 June **(2016)**.
5. A. Gambhir and G. Raj, "Analysis of Cache in Service Worker and Performance Scoring of Progressive Web Application"/2018 International Conference on Advances in Computing and Communication Engineering (ICACCE-2018) Paris, France 22-23 June **(2018)**.
6. S. S. Timalsina, "Progressive Web Application with React JS". Supervisor(s): Lasse Haverinen , Spring-Autumn **(2019)**
7. "The Web App Manifest| Tools for Web Developers" [Online] Available: https://developers.google.com/web/fundamentals/web-app-manifest
8. Y. M. Tashtoush, A. Alsmadi, A. Al-Abdi, N. Ababneh, O. Almousa "An Analysis of Android Web App Manifest". Proceedings of the 23rd Conference of Open Innovations Association FRUCT. **(2018)** November, Article No.: 78, pp. 556–559
9. "Blaze Meter Online URL Testing and Analysis Tool." [Online]. Available: www.blazemeter.com.
10. "Twitter Lite PWA Significantly Increases Engagement and Reduces Data Usage | Tools for Web Developers" [Online]. Available: https://developers.google.com/web/showcase/2017/twitter
11. J. Lockhorn, J. Rzutkiewicz "Why ProgressiveWeb Apps are the future of Mobile Web". Available: https://ymedialabs.com/progressive-web-apps
12. R. Nunkesser, "Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development". Proceedings of the 5th International Conference on Mobile Software Engineering and Systems, **(2018)** May, pp. 214–218
13. S. S. Tandel 1, A. Jamadar, "Impact of Progressive Web Apps on Web App Development". International Journal of Innovative Research in Science, Engineering and Technology, Vol. 7, Issue 9, **(2018)** September.
14. V. Sharma, R. Verma, V. Pathak, M. Paliwal, P. Jain "Progressive Web App (PWA) - One Stop Solution for All Application Development Across All Platforms". **(2019).**

AUTHORS PROFILE

Mr. Alankrit Gupta, currently studying in Third Year, pursuing B.Tech CSE from Akhilesh Das Gupta Institute of Technology & Management, GGSIPU. He is the Vice-Chairperson and Treasurer of IEEE ADGITM. Also being a part of IEEE Delhi Section, he is actively involved in its activities. He is a full stack developer and has an experience of 2 years in this field along with excellent academic performance.

Ritika Jain, currently studying in Third year, pursuing B.Tech EEE from Akhilesh Das Gupta Institute of Technology & Management, GGSIPU. She is the Vice-Chaiperson of IEEE ADGTIM and also a part of IEEE Delhi Section with active participation in its various activities. She is currently working in the field of Machine Learning and Data Science along with excellent academic performance.

Uma Tomer received the MCA degree form IGNOU in 2003. She is currently pursuing the Ph.D. degree in Computer Application from Manav Rachna International Institute of Research and Studies, Faridabad, India. She is currently working as Assistant Professor in the Department of Computer Science & Engineering at Akhilesh Das Gupta Institute of Technology and Management, GGSIPU. She is teaching since 2004 and actively supervised many students at bachelor's level. Her area of research is Software Quality with IoT and Web Application.