ISSN:2229-6107

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY

E-mail :
editor.ijpast@gmail.com
editor@ijpast.in

www.ijpast.in

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 9,Issuse 2.May 2019

Evaluation of the effectiveness of creating mobile apps across a variety of platforms

Dr. R. Rambabu , Mr. P S S K Sarma, Mrs. A. Josh Mary

Abstract:

Developers find it difficult to determine which platform to prioritize because each mobile operating system has its own standards, programming languages, and distribution methods. Nevertheless, several web-based applications have been reported to suffer significant performance drops when using these technologies; in response, web-based multiplatform development tools follow the "create once, deploy everywhere" principle and can be distributed across multiple platforms. This article presents the results of a study that looked at the effectiveness of mobile web applications powered by Android that were created with the PhoneGap framework. We also provide the results of an experiment that measured execution time to define the performance over

Keywords: Task Duration; Performance; PhoneGap; Android; Mobile

Introduction

Advances in mobile systems have made it possible for portable terminals to transform from basic communicators to potent computing instruments. Modern mobile phones are so efficient, so accessible, and so powerful that they can achieve things that were before unthinkable.effectiveness, as well as other options [1]. The foundation of smartphones has always been robust operating systems that resemble a PC-like modular program structure and make it simple for consumers to install and uninstall apps. Every device has a different operating system (OS), and each OS has its own set of standards, languages, tools, and channels for downloading and purchasing apps. Programmers are faced with a dilemma since each platform has several customers. Software developers may need to incorporate a larger user base into their business plans as theyThe utilization of multiplatform development tools that follow the "create once, deploy everywhere" philosophy is one efficient method to address this problem. These tools include Sencha Touch, Appellatory, PhoneGap, and others. These assets leverage cross-platform technologies like HTML, CSS, and JavaScript to control the functionality of the mobile device using a suite of application programming interfaces (APIs). API.In studies that predict a good increase of web browser use as execution environment, mobile target-agnostic development has been taken into consideration [2, 3, 4, 5].Development-focused surveys and case studies have demonstrated that tools still have constraints that prevent them from offering a comprehensive cross-platform solution, even if mobile apps may be easily generated for many platforms [6, 7, 8].The main issues are the differences.

Professor & HOD, Assistant Professor[1,2]

Department of Computer Science & Engineering,

Rajamahendri Institute of Engineering & Technology, Rajamahendravaram.

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST
ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

Many believe that switching to web development slows things down considerably, yet there are no studies that quantify the amount that performance drops as a result of the shift to the web to substantiate this claim.This article attempts to shed light on important performance issues raised by web-based multiplatform development tools for mobile applications using the results of a research into the performance of mobile web apps created using PhoneGap and deployed on the Android operating system. We describe an experiment that measured performance in terms of execution time and compared the effort required to operate a web application against its native version. The remainder of the document is organized as follows: The development tool that was chosen is introduced in the following sections.

Assessment of performance

Metrics including execution time, memory use, and energy consumption are a few that frequently yield insightful results when assessing performance [9]. The primary emphasis of our research is application execution time as a proxy for overhead.The impact on the app's user experience and its ability to communicate with the device's hardware and operating system is evident.The time it takes for a routine to finish cannot be determined only by sampling that time. Ensuring fairness and implementing suitable protocols for data interpretation are essential when comparing two machines, languages, or approaches. We developed a suite of software algorithms that leverage various mobile device hardware and software resources to comprehend the influence of web technologies on mobile application performance.

We took these actions and included them into two different applications: one that was created using a web-based environment and the other that made use of a mobile OS's built-in development capabilities. In this experimental context, we may objectively compare and evaluate the two methodologies. PhoneGap was chosen as the development tool, while Android OS was chosen as the target platform because of its openness, adaptability, and accessibility. We compared the two programs' execution times after running them in an experimental setting in order to conclude the investigation.

Framework for PhoneGap

The PhoneGap [11] framework, which is now part of Apache Cordova, is a component of the Apache Incubator. Using PhoneGap, you can create your logic layer using HTML5 and JavaScript, and utilize your smartphone's web browser as an abstraction layer.the HTML and CSS display layer. This foundation can be readily transferred to different web browsers, just like it can with desktop PCs. Unfortunately, this implies that JavaScript cannot fully utilize the capabilities of the mobile device, such as manipulating hardware components, because script-based apps can only operate inside the web browser's runtime.Developers may easily manage low-level components and telephony with PhoneGap's native engine and APIs.

These APIs are made available to the browser by the PhoneGap JavaScript engine, after which JavaScript can utilize them. Because the logic layer will utilize the appropriate interfaces and extensions to access additional resources via methods, this frees up developers to focus on creating websites. Please see Figure 1. Because a web browser and logic layer may function on any operating system, this approach is ideal for developing cross-platform apps.

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

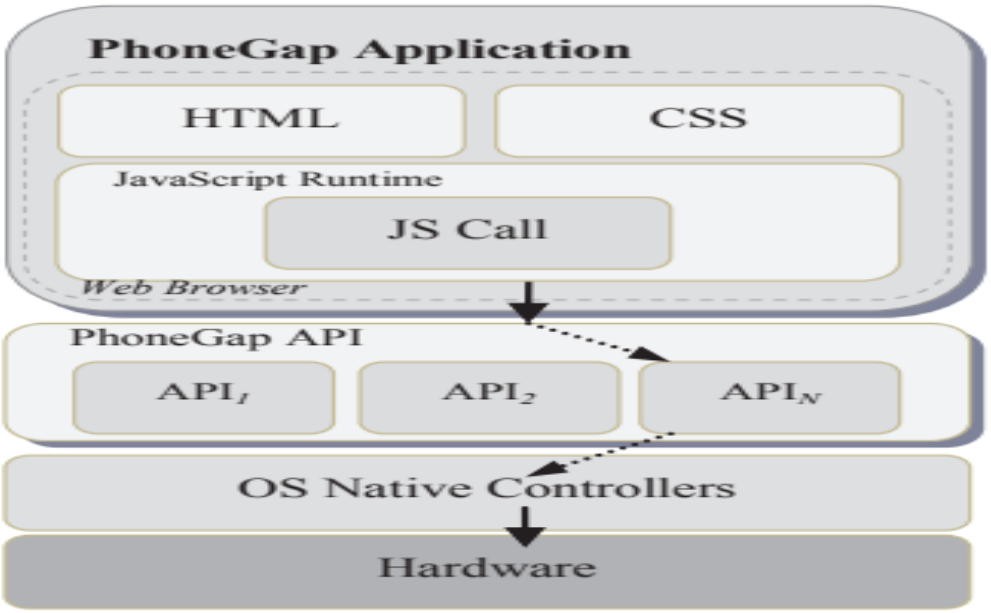


Fig. 1. PhoneGap application architecture

As of version 1.3.0, PhoneGap is compatible with all the main mobile OSes (such as Android, iOS, RIM, Windows Mobile, etc.), yet it does not provide complete control over the device's capabilities in a few of these platforms. [11].

Evaluation of native and web app runtimesExperimental apparatus

We evaluated two Android apps, one written in JavaScript and one in standard Java, on an actual mobile handset. Apps can use the mobile device's native API to access their own private set of subroutines. The program keeps track of how long a sine-tilted operation lasts. To achieve this outcome, we included code that snaps a picture at two distinct times: t0, which occurs right before the function is run, and t1, which occurs right after the function is finished and we receive a successful response. (t2).

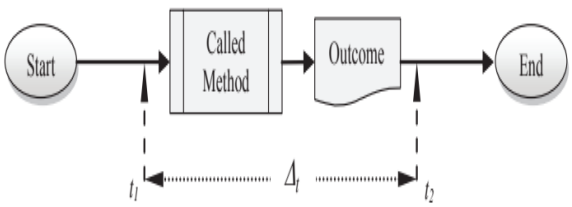


Fig 2. Operational definition of the measured job

The operational definition determines the job boundaries, as seen in Figure 2. Thus, we guarantee to track each function's execution time from the time it is activated until it is completely deactivated. actionable response. The total time it took to execute is equal to the difference ('t') between the two dates.

Technical Approach

When comparing the performance of two computers using the same metric, we followed the advice in [10] to help us evaluate and interpret the data. Before the geometric mean can be calculated, data must be standardized to a "known machine" in order to accurately depict relative system performance.It is advisable to use this geometric mean and average the normalized data when comparing relative performance. Java has become more reputable as a programming language since Android comes with native support for it. One way to normalize time samples in Java and JavaScript is to divide them by the Java value.

App for mobile devices

The user can initiate a certain procedure by tapping buttons on the mobile app's graphical user interface. The user experience for the two mobile applications was intended to be the same. according to Figure 3. During each operation, access to a hardware or software resource is confirmed by a response or other piece of information. The program tracks and reports how long it takes to do the task.In order to do this thorough examination of the mobile terminal, we considered several sources:Playback of sound alerts, activation of vibrators, and accelerometer access are a few instances of x-factors. Hardware

INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

availability.Internet access: utilize it to learn how to connect to the web, use GPS to find destinations, etc.A few examples

Since the JavaScript timer could only manage milliseconds, the choice was predicated on the fact that Java can accomplish time samples down to the nanosecond. Every procedure was run a thousand times to ensure statistical dependability.The test device was an Android OS 2.2 HTC Nexus One smartphone. To ensure the repeatability and reproducibility of our results, we repeated the tests on an HTC Magic smartphone; the results were not included in the final report, but were saved for future reference.to write this essay.

Data analysis

Figure 1 is a summary of the results. The data distribution is shown by reporting the mean and standard deviation values in milliseconds. We just examined the data in relative time units to do performance study after normalizing all time samples for the Java application.as well as the geometric means of such figures. Due to machine knowledge, the geometric mean of Java tasks is always 1. If a JavaScript job is statistically more efficient than the known machine at doing the same task, the geometric mean will show a number less than 1, and if it is statistically less efficient, it will show a value greater than 1.

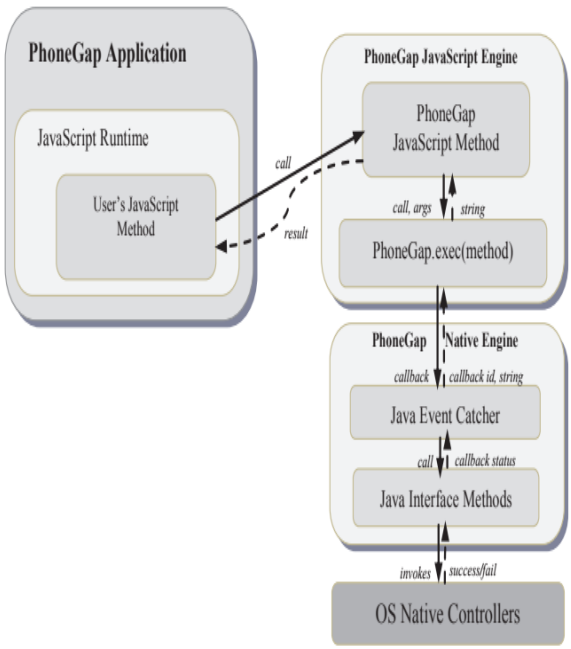
Table 1: Time required to run an Android native app vs. a PhoneGap web app

Measured Job	Arithmetic Mean (milliseconds)		Standard Deviation		Geometric Mean (relative)	
	Native App	Web App	Native App	Web App	Native App	Web App
Access to accelerometer	0.7136	2.0021	0.9984	3.0025	1.0000	2.5974
Launch sound notification	18.4835	26.7481	13.3665	47.5036	1.0000	0.6534
Trigger vibrator	1.5134	3.2222	1.2234	4.1248	1.0000	2.2593
Request data from GPS	2.1881	809.2352	6.7244	12.5523	1.0000	528.9298
Request network information	1.1015	1.01419	1.2052	0.6096	1.0000	1.1044
Write a file	4.7146	7.9221	9.2085	6.4558	1.0000	3.3657
Read a file	13.3036	255.7381	13.8829	74.1943	1.0000	29.9005
Retrieve data from contact list	95.8686	1841.4689	13.8747	491.5454	1.0000	18.7518

Table 1 reveals that there was only one procedure where the web app was on par with or even better than the native app. This was the process of beginning a sound notification, which took 35% less time. while it comes to anything else, the performance drops anywhere from barely perceptible (like 10% slower while getting network data) to very noticeable (like when using the GPS sensor).

Subject for debate

To gain an understanding of the situation, we examined the resource calls' code-level structures in each version. We discovered that Java usually utilizes native methods to access the given resource directly, but JavaScript can only access it indirectly by utilizing a sequence of calls that includes one or more callbacks. As a result, it takes longer to get in touch with the method, go through the callback process, and respond to the first requester. There is a noticeable increase in execution time when an API requires a complicated call sequence to be used.PhoneGap is designed so that a foreground executive method called PhoneGap takes as an argument a user-space JavaScript function. The sentence has two objectives.



INTERNATIONAL JOURNAL OF PURE AND APPLIED SCIENCE & TECHNOLOGY
IJPAST

ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

Fig 4. PhoneGap's method call flow path.

When dealing with resources that need intricate execution trees to access them, this design could become costly because of the resource's typical response time and the overhead involved in tracing the method via the call back tree and displaying the outcome. Alternative WebFor resource-specific functionality in the browser's display, apps built using frameworks that employ this architecture will experience the same performance overhead.The experimental setup has shown that there is no performance loss on certain frequently used functionalities, even if the execution time for web apps has increased. Importantly, in some instances. These findings corroborate the assertion in [12] that commercial applications or those with light code loads are better suited to web-based mobile apps. (For example, while doing tasks that heavily use hardware, such as producing 3D graphics).

Last thoughts

Discussing the pros and negatives of web-based mobile development requires taking into account a wide range of viewpoints. Now that platform-specific effort is unnecessary, savearl platforms may utilize a single application.software development and distribution processes. When it comes to using device-specific features or interfacing with other software resources, the present level of development tools is severely lacking. Additionally, reports show that web-based mobile apps have terrible speed, which negatively impacts the user experience.To demonstrate how much more time is required to do the same task when employing web-based programming as opposed to native, platform-specific capabilities, we examined the performance of web-based mobile applications built using PhoneGap and the Android operating system. Using the phone's hardware and software, we ran experiments and collected data to determine when execution time begins to increase and under what circumstances this happens.We found that in seven of eight tests using machine benchmarks, the web-based version performed worse than the native one. We tracked it down to the web-based solution's execution slowdown caused by repeatedly calling methods with a callback and then waiting for their response. The more complicated the execution tree, the longer it takes to retrieve the resource that was asked for and answer to the requesting process. There will supposedly be a performance hit, but it'll be too little to matter for most business applications.Before committing to a multiplatform framework, developers should weigh a number of factors, including the likelihood of worse performance as compared to native programs. The reasons and extent of a benchmark software's performance degradation are illuminated by this research. Since the execution time viewpoint is the primary focus of this article, more work is needed to evaluate other performance analysis criteria. (Think: data collected from customer satisfaction surveys, battery life, memory utilization, etc.).When people love using a mobile app, it becomes a success. Developers working within the web-based paradigm should be cognizant of pertinent performance concerns, work toward improved design and coding processes, and increase access to multiplatform development tools in order to provide a truly cross-platform, cohesive user experience.

Works Cited

[1] Succi, Corral, and Sillitti are the authors. ensuring that strategies for creating mobile apps are capable of meeting mission-critical requirements. The MOBICASE Workshop on Software Engineering for Mobile Application Development, 2011, articles 9–11. [2] based on studies conducted by Salminen, Cavallari, Mikkonen, and Anttonen. Binary software is fading as end-user software migrates to the web. The Ninth International Conference on Computing for Creation, Connection, and Collaboration, Proceedings, 2011, pages 17–23.
Cavallari and Mikkonen are the three. The 10-year battle between apps and the open web. The results of the 2nd Annual Workshop on Software Engineering for Mobile Application Development, which took place in tandem with MOBICASE 2011, cover pages 22–26. Cavallari and Mikkonen [4] look at the internet as a forum for

[5] Sillitti A, Succi G, Ramella P, Garibo A, and Corral L. shift from a platform-specific to a web-based multiplatform approach to developing mobile apps. ACM Symposium on New Ideas in Programming and Reflections on Software, 2011 (ONWARD! 2011, pages 181–183). The source of this data is Bloom et al. (2006). Review of runtime environments for mobile devices Write once, use wherever.

Abstract: Geisler, Zelazny, Christmann, and Hagenhoff, editors. 2008 Third International Conference on Grid and Pervasive Computing Workshops, pp 132-137. studies on the effectiveness and usability of different distribution mechanisms for mobile apps. Pages 210–218 of the proceedings of the tenth annual International Conference on Mobile Business (ICMB), 2011.

[8] Duarte C. and Afonso AP. A case study from JIL on designing once and implementing globally. Mobilize, Volume 8, Issue 4, Pages 641-644 2011: Mobile Web Information Systems, Eighth International Conference.
[9] Germanic A. La valuationdiescalculator is the

ISSN 2229-6107 www.ijpast.in
Vol 8,Issuse 1.Jan 2018

Italian term for assessing computer hardware. The lecture notes for the computer platform engineering seminar. 2011 Italian university; University of Genoa, School of Engineering.
[10] Fleming and Joseph Wallace. Truthful statistical summarization: how not to deceive with data. 1986; Vol. 29, no. 3, pp. 218-221. In: The ACM Communications.