Assessment of the efficacy of developing mobile applications on many platforms

ParnandiSrinuVasaRao Assistant. Professor, Dr.ShaikMeeravali Associate Professor,

Vudi SriLakshmi Assistant. Professor

Abstract:

Each mobile operating system has its own standards, languages for programming, and distribution methods, and these in turn affect the mobile devices itself. Because of this, developers have a tough time deciding which platform to prioritize.However, there have been complaints of web-based applications experiencing significant performance decreases when employing these technologies. To address this, web-based multiplatform development tools adhere to the "create once, deploy everywhere" principle and may be delivered on several platforms. This article details the findings of a study that examined the efficiency of Android-powered mobile web applications developed using the PhoneGap framework. We provide the findings of an experiment that measured execution time to define the performance overhead of a web app compared to identical native software.

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

Introduction

Mobile system advancements have allowed handheld terminals to evolve from simple communicators to powerful computing tools. Modern mobile phones are capable of doing things that were previously unimaginable due to their processing power, accessibility, and efficiency.efficacy, and al'choice beyond that [1]. Strong operating systems that mimic a PC-like modular app structure have been the backbone of smartphones from the start, letting users effortlessly install and remove apps. Operating systems vary from one device to another, and with each OS comes a unique collection of standards, languages, tools, and distribution channels for app purchases and downloads. Since there are several clients for each platform, this variety poses a challenge for programmers. Software producers may need to include wider user bases in their business strategies as developing for only one platform would leave out a large chunk of prospective consumers. The process of developing a unique software product for every platform may be time-consuming and expensive due to the need to repeat substantial portions of the software life cycle for every published application.

One effective way to handle this issue is by using multiplatform development tools that operate under the principle of "create once, deploy everywhere." Examples of such tools include PhoneGap, Appellatory, Sencha Touch, etc. These assets use a suite of application programming interfaces (APIs) to manage the functionality of the mobile device, using cross-platform technologies such as HTML, CSS, and JavaScript. (API).Mobile target-agnostic development has been considered in works that anticipate a good expansion of web browser use as execution environment [2, 3, 4, 5].Despite the fact that mobile apps may be quickly produced for many platforms, development-oriented surveys and case studies have shown that tools still have limitations that prohibit them from providing a complete cross-platform solution [6, 7, 8].The biggest problems are the variations in user experience, the limitations in accessing hardware operations, and the challenges in integrating the software with native components. A lot of people think that moving to web development slows things down significantly, yet there haven't been any studies that measure how much performance decreases because of the move to the web to support this assertion.Using the findings of an investigation of the performance of mobile web apps produced using PhoneGap and deployed on the Android operating system, this article aims to shed light on key performance concerns brought up by web-based multiplatform development tools for mobile software. We detail an experiment that contrasted the effort needed to run a web app against its native version and quantified performance in terms of execution time. The rest of the paper is structured as follows: The next sections provide an introduction of the selected development tool, discuss performance analysis in Section 3, describe our experimental setup and the data we obtained in Section 4, and conclude with some conclusions and future ideas in Section 5.

Assessment of performance

Execution time, memory use, and battery consumption are a few metrics that often provide useful findings when measuring performance [9]. Our study mainly focuses on application execution time as an indication of overhead.It plainly affects the app's user experience and the app's interaction with the device's OS and hardware.It

is not possible to evaluate the time it takes for a routine to complete only by sampling that time. Adopting appropriate procedures for data interpretation and creating an environment that encourages fairness are crucial when comparing two machines, languages, or methodologies. We built a set of software algorithms that use different hardware and software resources of mobile devices to understand how web technologies impact the performance of a mobile app. We followed these steps and integrated them into two separate applications, one developed in a web-based environment and the other utilizing the native programming tools of a mobile OS. We can compare and assess the two strategies objectively using this experimental setting. Due to its accessibility, versatility, and openness, Android OS was selected as the target platform, while PhoneGap was selected as the development tool. To wrap up the study, we compared the two programs' execution times after executing them in an experimental environment.

Framework for PhoneGap

Now known as Apache Cordova, the PhoneGap [11] framework for building mobile applications is a member of the Apache Incubator. With PhoneGap, you can build your logic layer in JavaScript and HTML5 and then use the web browser on your smartphone as a layer of abstraction.the presentation layer of HTML and CSS. Just as in desktop computers, this foundation may be easily moved to various web browsers. Unfortunately, this means that script-based applications can only be executed inside the runtime of a web browser, which means that JavaScript can't make full advantage of the mobile device's capabilities, including controlling hardware components.The native engine and APIs that come with PhoneGap make it easy for developers to handle low-level components and telephony. The PhoneGap JavaScript engine makes these APIs accessible to the browser, and then JavaScript may make use of them. This frees up developers to concentrate on building websites, since the logic layer will use the right interfaces and extensions to access other resources via methods. (Refer to Figure 1). This design is perfect for building cross-platform applications since any operating system can run a web browser and a logic layer inside of it.

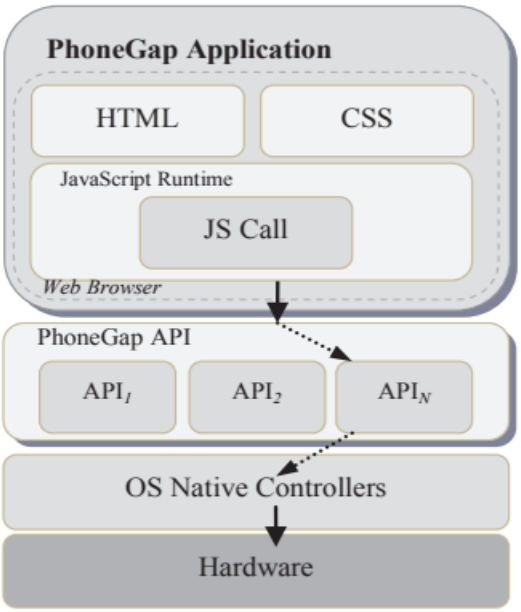

Fig. 1. PhoneGap application architecture

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

On a real mobile device, we tested two Android apps: one made with JavaScript and one with regular Java. Apps may access their own private set of subroutines using the native API of the mobile device. The software records the duration of a sinetilted process. We obtained this result by inserting code that takes a snapshot at two specific times: t0, just before the function is executed, and t1, immediately after the function is completed and we get a successful answer. (t2).

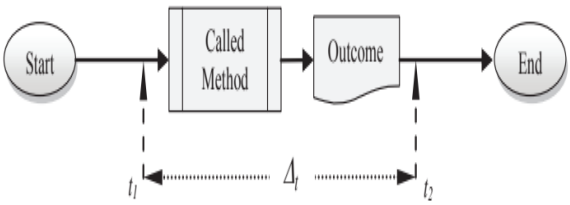

Fig 2. Operational definition of the measured job

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

We used the recommendations in [10] to guide our data processing and interpretation as we compared the performance of two computers using the same metric. Data must be standardized to a "known machine" before the geometric mean can be computed in order to provide an accurate representation of relative system performance.When comparing relative performance, it is best to average the normalized data and utilize this geometric mean. Java gained credibility as a reliable computer language because Android natively supports it. Java and JavaScript time samples may be normalized by dividing them by the Java value.

App for mobile devices

The graphical user interface of the mobile app has buttons that the user may tap to start a certain process. The two mobile applications were designed to provide the same user experience. Based on Figure 3. A response or other piece of information is received as confirmation of access to a hardware or software resource during each operation. The application logs and reports the amount of time it takes to complete the assignment.For this comprehensive analysis of the mobile terminal, we took into account a wide variety of resources:A few examples of x-factors include access to the accelerometer, sound alert playback, and vibrator activation. Access to hardware.Internet access: use it to seek up locations using GPS, learn how to connect to the web, etc.Some instances of data access include creating new files, reading existing ones, and retrieving material from providers.The item is accessed when the button is pressed by invoking the specified method. after our operational requirements, the software records the time immediately before each method call and the time immediately after its successful completion and saves both timestamps to a file. We chose the method that provides the value of the most precise system timer to ensure that the time samples are as accurate as feasible. The decision was based on the millisecond because, whereas Java can achieve time samples down to the nanosecond, the JavaScript timer could only handle milliseconds. To guarantee statistical reliability, each process was executed a thousand times.An HTC Nexus One smartphone running Android OS 2.2 wasused as the test bed. We also ran the same tests on an HTC Magic smartphone to check that our findings were repeatable and reproducible; these data were not included in the final report but were instead kept on file for future use.to author this paper.

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

We compared the code-level structures of the resource calls in each version to get a sense of what's going on. We found that Java typically makes use of native methods to directly access the specified resource, whereas JavaScript is only allowed to do it indirectly through a flow of code that uses at least one call back. The time it takes to contact the method, follow the call back path, and provide the answer to the original requestor is increased as a result. A significant increase in execution time occurs when using an API that necessitates a complex call sequence.A user-space JavaScript function is supplied as a parameter to a foreground executive method named PhoneGap, according to PhoneGap's design. The sentence aims to do two things: first, it calls a JavaScript function called prompt(); second, it passes the required arguments via a JSON string; and third, it fires an event that PhoneGap's native engine can catch. The native layer of PhoneGap interprets the JavaScript method and its arguments before sending the request to the relevant API or controller. Refer to Figure 4.The execution of the JavaScript function is sent to a call back provided on the application's web page while the real result, a prompt dialog, is displayed. After the request has been processed and the parameters have been checked, the original JavaScript caller will get a string with the result from the native Java method.

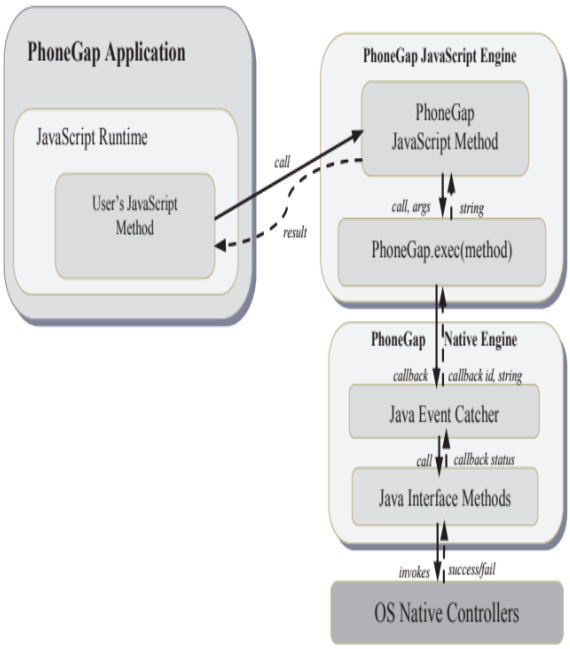

Fig 4. PhoneGap's method call flow path.

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] Authors: Corral, Sillitti, and Succi. Making sure that methods for developing mobile applications can handle mission-critical needs. Articles 9–11 from the 2011 edition of the MOBICASE Workshop on Software Engineering for Mobile Application Development.

[2] Based on research by Cavallari, Mikkonen, Anttonen, and Salminen. Software for end users is moving to the web, which means binary software is dying. Pages 17–23 of the 2011 Proceedings of the Ninth International Conference on Computing for Creation, Connection, and Collaboration.

Three, Mikkonen and Cavallari. Apps vs. open web: The decade-long conflict. The 2nd Annual Workshop on Software Engineering for Mobile Application Development, held in conjunction with MOBICASE 2011, with proceedings spanning pages 22–26, 2011.

In their work, Cavallari and Mikkonen [4] examine... The web as a platform for applications: The story goes on. Pages 170-174, 2011 Proceedings of the 37th EUROMICRO Conference on Software Engineering and Advanced Applications.

[5] Ramella P, Garibo A, Succi G, Sillitti A, and Corral L. Changes in mobile software development approach from platform-specific to web-based multiplatform. Pages 181–183, 2011 ACM Symposium on New Ideas in Programming and Reflections on Software (ONWARD! 2011).

This information is sourced from Bloom et al. (2006). A review of mobile runtime environments - Write once, execute anywhere.

Paper presented at the 2008 Third International Conference on Grid and Pervasive Computing Workshops, edited by Geisler, Zelazny, Christmann, and Hagenhoff, pages 132-137. Research on the efficacy and user-friendliness of various mobile software distribution strategies. The tenth annual International Conference on Mobile Business (ICMB) proceedings, 2011, pages 210–218.

[8] Afonso AP and Duarte C. A JIL case study on developing once and deploying everywhere. Volume 8, Issue 4, Pages 641-644, Mobilize 2011: The Eighth International Conference on Mobile Web Information Systems.

[9] Germanic A. La valuationdiescalculator, the Italian name for computer equipment evaluation The seminar's lecture notes on computer platform engineering. Italian university, 2011; school of engineering, University of Genoa.

[10] Fleming and Wallace, Joseph. Truthful statistical summarization: how not to mislead with data. Vol. 29, no. 3, pp. 218-221. 1986. In: Communications of the ACM.