International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

Performance evaluation of cross-platform mobile app development

KUKKUNOORI SHIVARAM, GOGULA GOPINATH , KATURU VIVEK KUMAR REDDY
MATHANGI SAMRAJ
SUPERVISOR , K VIJAY KUMAR
Associate Professor
ANURAG ENGINEERING COLLEGE
AUTONOMOUS
(Affiliated to JNTU-Hyderabad,Aprroved by AICTE-New Delhi)
ANANTHAGIRI (V) (M), SURYAPETA (D), TELANGANA-508206

Abstract:

Different mobile devices use different operating systems, each of which has its own set of standards, programming languages, and distribution channels. This creates a difficult choice for developers in terms of which platform to focus on first.Web-based multiplatform development tools provide a remedy by adhering to the "develop once, deploy everywhere “technologies that may be deployed on a variety of platforms, however, there have been reports that web-based apps experience severe performance drops when using these technologies. In this article, we describe the results of an investigation on the performance of mobile web apps built using PhoneGap and the Android operating system. Define the performance overhead found in a web app in comparison to an equivalent native software, we present the results of an experiment that focused on measuring execution time.

Keywords: Mobile; Multiplatform; PhoneGap; Android; Performance; Execution Time

Introduction

With the development of mobile systems, handheld terminals have progressed from basic communicators to sophisticated computing tools. Cell phones now have the processing power, availability, and efficiency to carry out previously impossible tasks.efficiency, and a l opt more besides [1]. Smartphones, since their inception, have been powered by robust operating systems that use a PC-like modular app framework, allowing users to easily add and delete programs. Different devices run on different operating systems, each of which has its own set of standards, programming languages, development tools, and even distribution marketplaces where users can buy and download apps. This diversity is a problem for program mummers since there are many clients for every one platform. Developing just for one platform would exclude a huge portion of potential customers, hence it may be necessary for software developers to aim for broader user bases as part of business plans. Meanwhile, producing a separate software product for each platform necessitates going through large chunks of the software life cycle many times for each published application, which may become tedious and costly over time.

Multiplatform development tools (like PhoneGap, Appellatory, Sencha Touch, etc.) that provide a solution under the idea of "develop once, deploy everywhere" are one effective method of addressing this problem. These resources make use of cross-platform technologies including HTML, CSS, and JavaScript to control the mobile device's features via a collection of APIs. (API).Works that predict a positive expansion on the usage of the web browser as execution environment [2, 3, 4, 5] have discussed target-agnostic development on mobile devices.Although mobile applications can be efficiently built for more than one platform, tools still present drawbacks that prevent them from offering an integral cross-platform solution, as highlighted by development-oriented surveys and case studies [6, 7, 8].Limitations in accessing hardware functions, difficulties integrating the program with native components, and differences in user experience stand out as the most significant flaws. It has also been said that switching to web development causes a major slowdown, however, there have not been any studies done to back up this claim by quantifying howmuch performance drops due to switching to the web.With the hope of shedding light on crucial performance issues brought up using web-based multiplatform development tools for mobile software, this paper presents the results of an analysis of the performance of mobile web applications built with PhoneGap and deployed on the Android operating system. We describe an experiment that measured performance in terms of execution time and characterized the extra effort required by a web program compared to its native counterpart. Here's how the remainder of the paper is laid out: In Section 2, we explain performance analysis; in Section 3, we provide an overview of the chosen development tool; in Section 4, we detail our experimental setup and the resulting data; and finally, in Section 5, we provide some findings and suggestions for the future.

Performance analysis

Sevearl criteria often produce helpful results for performance measurement [9]; they include execution time, memory utilization, and battery consumption. As an indicator of overhead, application execution time is the primary focus of our research.It has an obvious impact on how an app feels to use or how an app interacts with the device's hardware or software.Evaluating the time, it takes a routine to execute is not something that can be done by taking random samples of that time. When comparing two machines, languages, or approaches, it is important to adopt acceptable processes for data interpretation and create an atmosphere that promotes fairness [10]. In order to learn how web technologies, affect a mobile app's performance, we developed a series of software routines that make use of various mobile device hardware and software resources. We then incorporated these procedures into two parallel apps, one built using a web-based development environment and the other with the programming tools inherent to a mobile operating system. This experimental setup will allow us to compare and evaluate the two methods in an objective manner. PhoneGap was chosen as the development tool, and Android OS was chosen as the target platform, because of the accessibility, adaptability, and openness they provide. We finished the project by running the two programs in an experimental setting and comparing their execution times.

PhoneGap framework

A mobile application development framework, PhoneGap [11] is now part of the Apache Incubator as Apache Cordova. The idea behind PhoneGap is to construct the logic layer in JavaScript and HTML5, using the device's web browser as an intermediary level of abstraction.HTML and CSS's display layer. This framework can be readily transferred to other web browsers, much as it is in desktop computing. However, this only enables script-based apps to be performed inside a web browser's runtime, limiting JavaScript's ability to fully use the capabilities of the mobile device (such as handling hardware components).PhoneGap's native engine and accompanying set of APIs allow developers to easily manage telephony and other low-level components. After being exposed to the browser via the PhoneGap JavaScript engine, these APIs may be used by JavaScript. Therefore, developers may focus just on web development; the logic layer will rely on the appropriate interfaces and extensions to get access to additional resources through methods. (Figure 1). The fact that any OS can launch a web browser and execute a logic layer inside it makes this architecture ideal for developing cross-platform apps.

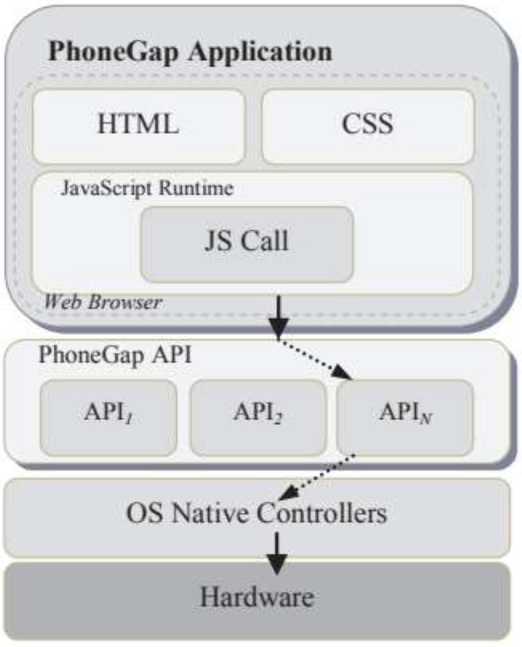


Fig. 1. PhoneGap application architecture

As of version 1.3.0, PhoneGap supports all the major mobile operating systems (e.g., Android, iOS, RIM, Windows Mobile, etc.), although in some of them it does not provide full management of device'sfunctionality [11].

Analysis of execution time on web and native applicationsExperimental setup

Two Android apps, one written in JavaScript and one in standard Java, were developed and evaluated on a genuine mobile device. Each app may use the mobile device's built-in API to access its own set of subroutines. When a process sinetilted, the program logs the amount of time spent on the task. We were able to retrieve this value by instrumenting the code to capture a snapshot of time t0 (just before running the function) and t1 (just after receiving a successful response of its completion). (t2).

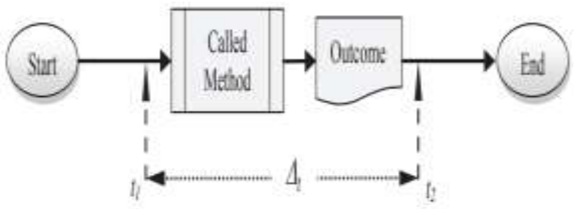


Fig 2. Operational definition of the measured job

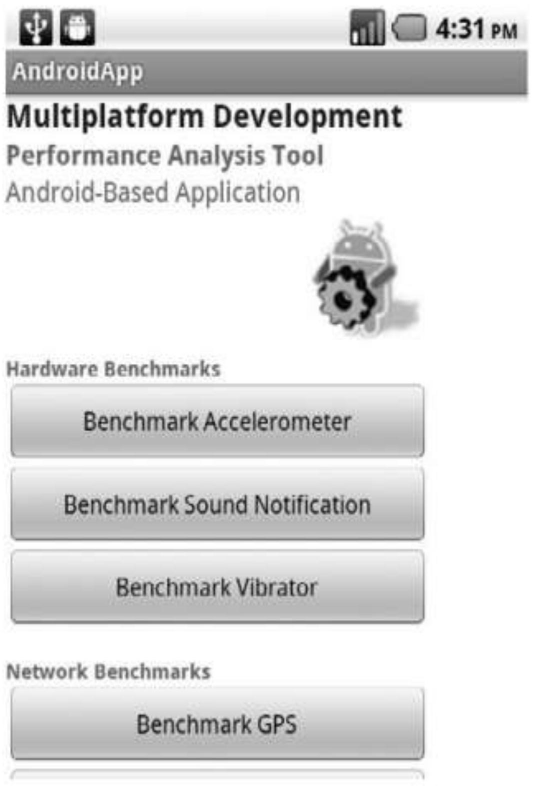
The task boundaries are shown in Figure 2 and are determined by the operational specification. In this way, we promise to account for every second that goes into using each function, from the moment it is ACTivoted until it is fully shut down. practical reaction. The delta between the two samples of time ('t') represents the final execution time.

Methodology

Assess the efficiency of two machines using the same measure, we based our analysis and interpretation of the data on the suggestions made in [10]. To accurately portray relative system performance, first the data must be normalized to a "known machine;" then the geometric mean must be calculated.The normalized data should be averaged, and then this geometric mean may be used to make comparisons of relative performance. Due to Android's built-in support for the Java programming language, Java was established as a trusted machine. By dividing the time samples obtained by Java and JavaScript by the Java value, normalized values may be determined.

Mobile application

The mobile app has a graphical user interface with buttons the user may press to initiate a certain procedure. The user experience on both mobile apps was designed to be identical. (Figure 3). The purpose of each procedure is to make aaccessing a hardware or software resource and receiving a response or other piece of information as confirmation of such access. The time it took to do the task is recorded and reported by the program.We considered many types of resources in order to do a thorough study on the mobile terminal:Access to the accelerometer, the ability to play a sound alert, and the activation of the vibrator are all examples of x Hardware access.Access to the Internet: look up GPS coordinates, find out how to go online, etcFile creation, file reading, and content provider retrieval are all examples of data access.When the button is pushed, the designated method is invoked to access the item. The program takes a snapshot of the time just before each method is called and again after it has finished successfully, saving both snapshots to a file in accordance with our operational criteria. We opted for the approach that gives us the value of the most accurate system timer so that we can get the time samples as accurately as possible. While Java can get time samples down to the nanosecond, the most the JavaScript timer can do is the millisecond, so that is what wassettled on. Each procedure was performed a total of a thousand times to ensure statistical reliability.



International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

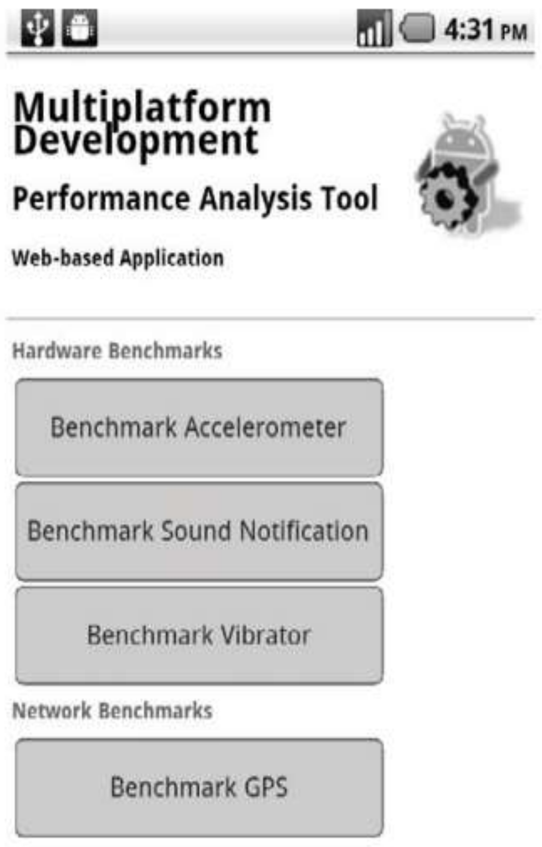


Fig 3. (a) Android native application; (b) PhoneGap web application

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

Table 1 summarizes the findings. Mean and standard deviation values in milliseconds are reported to show how the data are distributed. After normalizing all time samples about the Java program, we just looked at the data in relative time units to conduct performance analysis.and the geometric averages of those values. The geometric mean of Java jobs is fixed at 1 because the machine is known. The geometric mean will display a number less than 1 for each JavaScript job if it is statistically more efficient than the identical work completed by the known machine, and a value more than one if it is statistically less efficient.

Table 1. Comparison of execution time between Android native application and PhoneGap web application

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 shows that the web app only performed as well as or better than the native app in one routine (such as starting a sound notification, 35% quicker). For anything else, performance suffers in a range from negligible (e.g., 10% slower while obtaining network information) to severe.significant (e.g., accessing the GPS sensor).

Discussion

To get a feel for what's going on here, we looked at the inner structure of the resource call at code level for each version and found that Java usually uses native methods to directly access the specified resource, while JavaScript is only permitted to do so by following an indirect path.the flow of code that uses at least one call back. This adds extra time for the method to be called, for the call back route to be followed, and for the original requestor to get the response. When using an API that requires a complicated call sequence, execution time increases dramatically.According to PhoneGap's architecture, a JavaScript method defined in user space is passed as a parameter to a foreground executive method called PhoneGap. Exec (), which in turn invokes a JavaScript function (i.e.,prompt ()) with two goals in mind: to fire an event that can be caught by PhoneGap's native engine and to pass the necessary parameters via a JSON string. The JavaScript method and its parameters arecapoured by PhoneGap's native layer, which then forwards the call to the appropriate controller or API. (See Figure 4).While the actual result (a prompt dialog) is shown, a call back specified on the application's web page is informed that the JavaScript function was executed. Once the arguments have been validated and the request has been executed, the native Java method will return a string containing the result to the original JavaScript caller.

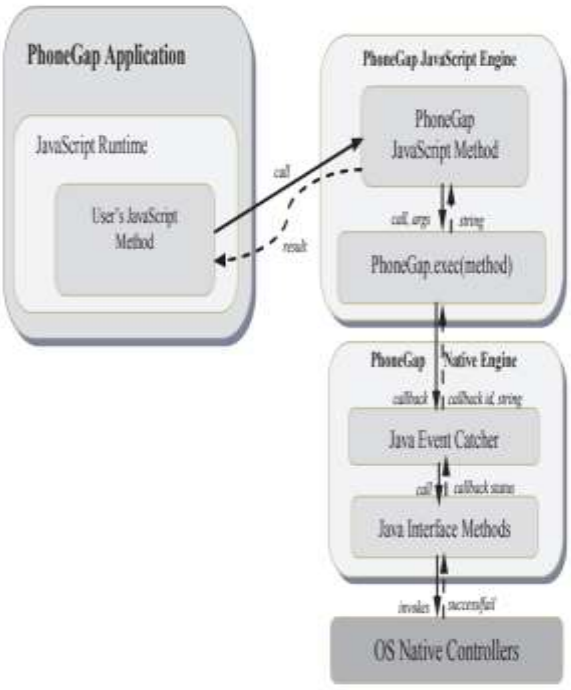


Fig 4. PhoneGap's method call flow path.

Due to the resource's normal response time and the overhead incurred by tracing the method via the call back tree and presenting the result, this architecture may become expensive when interacting with resources that need complex execution trees to reach them. Variant OnlineThe same performance cost will be seen by applications developed using frameworks that use this architecture to provide resource-specific functionality in the browser's view.Despite an increase in execution time for web applications, the experimental configuration has shown that the performance penalty on a number of commonly used features is minimal. (And in somecases, crucially). These results agree with the claim made in [12], which states that web-based mobile apps are more appropriate for commercial applications or those that do not heavily use resource-hungry code. (Like rendering 3D graphics or performing other heavy hardware-consuming operations).

Conclusions

There are maNy different perspectives to consider while discussing the advantages and downsides of web-based mobile development. Since platform-specific work is no longer required, a single program may now be used on savearl platforms.the procedures involved in creating and distributing software. The current state of development tools has limitations, particularly when it comes to using device-specific capabilities or interacting with external software resources. Reports also demonstrate that the user experience suffers due to severe performance losses in web-based mobile applications.We investigated the speed of web-based mobile apps utilizing PhoneGap and the Android OS to show how much longer it takes to complete the same work when using web-based programming instead of native, platform-specific capabilities. We utilized the phone's hardware and software to do tests and gather data that allowed us to pinpoint the point at which execution time starts to rise and the conditions under which this occurred.According to the results of our machine benchmarks, the web-based version is slower than the native one in seven out of eight tests. We determined that this was because the web-based solution slowed down execution by invoking methods with at least one call back and waiting for its answer. The time it takes to get the requested resource and respond to the asking process grows in proportion to the complexity of the execution tree involved. It is said that there will be a drop in performance, but that it will be negligible for most commercial uses.A variety of issues, such as the possibility of lower performance compared to that of a native program, must be considered before settling on a multiplatform framework for development. This study sheds light on what causes and to what degree a benchmark software degrades in performance. More work is required to assess additional performance analysis parameters, since this paper focuses on examining performance from the execution time perspective. (Such as memory consumption, battery usage, user experience surveys, etc.).The success of a mobile app is highly reliant on the satisfaction of its users. Web-based paradigm developers should be aware of relevant performance issues, strive toward better design and coding methodologies, and expand multiplatform development tools to produce a genuinely cross-platform, unified user experience.

References

[1] Corral L, Sillitti A, Succi G. Preparing mobile softwaredevelopment processes to meet mission-critical requirements. Proceedings of the 2nd Annual Workshop on Software Engineering for Mobile Application Development, in connection with MOBICASE 2011, pp. 9-11. 2011.

[2] Cavallari A, Mikkonen T, Anttonen M, Salminen M. The death of binary software: End user software to the web. Proceedings of the 9th International Conference on Creating, Connecting andCollaborating through Computing, pp.17 -23. 2011.

[3] Mikkonen T, Cavallari A. Apps vs. open web: The battle of the decade. Proceedings of the 2nd Annual Workshop on

Software Engineering for Mobile Application Development, in connection with MOBICASE 2011, pp. 22-26. 2011.

[4] Cavallari A, Mikkonen T. The web as an application platform: The saga continues. Proceedings of the 37th EUROMICRO

Conference on Software Engineering and Advanced Applications, pp. 170-174. 2011.

[5] Corral L, Sillitti A, Succi G, Garibo A, Ramella P. Evolution of mobile software development from platform-Specific to

web-Based multiplatform paradigm. ACM Symposium on New Ideas in Programming and Reflections on Software, (ONWARD!

2011), pp. 181-183. 2011.

[6] Bloom S, Book M, Gruhn V, Hauschka R, Köhler A. Write once, run anywhere – A survey of mobile runtime environments.

Proceedings of the 3rd Int. Conference on Grid and Pervasive Computing Workshops, pp. 132-137. 2008.

[7] Geisler S, Zelazny M, Christmann S, Hagenhoff S. Empirical analysis of usage and acceptance of software distribution methods on mobile devices. Proceedings of the 10th International Conference on Mobile Business (ICMB), pp. 210-218. 2011.

[8] Duarte C, Afonso AP. Developing once, deploying everywhere: A case study using JIL. Proceedings of the 8th International Conference on Mobile Web Information Systems Mobilise 2011, pp. 641-644. 2011.

[9] Frisian A. Evaluation of computer machinery (In Italian: La valuationdiescalculator) Lecture notes of the Seminar on computer platforms engineering. University of Genoa, Faculty of Engineering, Italy, 2011.

[10] Fleming P, Wallace J. How not to lie with statistics: The correct way to summarize benchmark results. Communications of the ACM. vol. 29, no. 3, pp. 218-221. 1986.