Performance Evaluation of Native and Hybrid Android Application

AJAYI, Olusola Olajide & OMOTAYO, Ayokunle Abiodun
Department of Computer Science
Adekunle Ajasin University
Akungba-Akoko, Ondo State, Nigeria
olusola.ajayi@aaua.edu.ng , fredrickbdn@gmail.com
Phone: +2347056433798, +234706 713 9018

ABSTRACT

Android has become most popular and powerful embedded OS. Nowadays, it is used in other electronic items other than mobile phones like TV, Camera, etc. The purpose of this study is to find out the difference in performance between the different methods for developing applications due to an increasing market for platform independent applications. In this, work-in-progress research, we present our current findings concerning performance efficiency in cross-platform and native mobile applications (apps) and how they can contribute to a general benchmarking approach. At first, several test cases for evaluating performance of mobile applications are described with which two applications where built to implement a mathematical calculation for both native and hybrid respectively. This is used as benchmark because of the recursive nature and memory usage of both applications for CPU and memory usage. Then, the performance efficiency of native and hybrid apps is compared on a mobile device. The results show that hybrid applications still suffer performance issues in comparison to native apps. The performance deviations and reasons for them are discussed and evaluated. It is concluded that the performance of mobile applications is crucial to user experience and satisfaction.

Keywords: Android, OS, Platform, Performance, Benchmarking, Native, Hybrid, CPU, Mobile, Deviation, Evaluation

1. BACKGROUND TO THE STUDY

Mobile communication is so integrated into our lives that many people feel uncomfortable without a cell phone. Once upon a time, the most popular functions of phones were calling and sending texts. A smart phone is a multifunctional device that not only communicates, but helps to learn, earn, and have fun. This is made possible by the development of mobile applications.

Mobile applications date back to the end of the twentieth century. Typically, they were small arcade games, ring tone editors, calculators, calendars, and so forth. The beginning of the new millennium saw a rapid market evolution of mobile content and applications.

Development for the mobile application market has drastically increased in size and magnitude therefore the requirements for developing applications has changed along with the market (Dan, 2015). According to Andersson et al (*2015),* Hybrid application is one of the three main development paradigms along with native development and HTML5. Hybrid applications featured a single code-base, bridging the different platforms as opposed to the native applications. HTML5 shares the platform independence with hybrid applications; however it lacks the ability to communicate with the low level Application Programming Interface (API). The three development paradigms differ in performance, development resources and user interface. Due to performance limitations of mobile devices the performance is a major factor for the selection of development method. The technology for developing mobile applications is evolving rapidly and that makes the performance to evolve along with it.

2. STATEMENT OF PROBLEM

No doubt, research has been tremendously done on mobile platform performance, preciously on android due to high market share of Android devices at the time of the study and the availability of developer tools. 90% of work done centered on the area of performance (connectivity, data storage etc.) with the consideration of 4.1 (Jelly Bean) as the version of android used. After much research on the existing work, statistical based approach where used for the performance metrics.
Ghada et al (2015) in their research reveals that new metrics can be proposed and empirical result analysis can be used in the mobile application performance metrics. However, application used on the existing work are not tested in terms of CPU usage and app load time, and the maximum version of the android platform used was 4.1 (JellyBean) as early stated. This study will therefore carry out its own evaluation with a higher version of the platform (e.g. Lollipop, KitKat). KitKat will be considered for the implementation of the application to be used.

3. OBJECTIVE

The specific objectives of the research are as follows: To explore the limitation of the mobile android development frameworks (i.e. hybrid and native); to express and analyze the App load time, memory usage and CPU performance of android platforms; to analysis the hybrid app efficiency performance as an important factor for software application quality.

4. METHODOLOGY

This research first classifies general approaches to cross-platform development of mobile applications. We then analyze and compare existing cross-platform solutions based on Web technologies like HTML, CSS, and JavaScript. As these differ in their general architecture and their capabilities, it is not obvious which to prefer. We will outline criteria that are important when making a decision as well as evaluate the popular approaches mobile Web apps, PhoneGap and Titanium Mobile according to these criteria.

The first concepts that we defined in the evaluation process are the system under test (SUT) and the component under study (CUS). The SUT in this case is the display, in the mobile device, of web pages hosted by a local server. This, SUT includes the device, the host, the webserver, etc. The CUS is an Android implementation of an internet browser that receives an address in the web, searches the page on the server and displays it on the mobile device screen. The CUS is the focus of the evaluation.

The following aspects were used for the performance assessment of the application: CPU time, memory footprint, battery usage, communication demand, and the total execution time of the application for displaying web pages (hybrid). Data related to each aspect would be collected during the execution of the application in the target platforms.

We try to reduce the influence of external factors in the measured data by controlling as much as possible the experimental setup. Still, some variables are beyond control. For instance, one cannot control embedded Android processes such as garbage collector or other internal processes of the operating system.

5. RELATED LITERATURES

Dalmasso et al (2014): "comparison and evaluation of cross platform mobile application development tools". They measure and evaluate the tools needed for hybrid application**.** The article provides several criteria other than just portability and performance, for example user experience, development cost and ease of updating. The cross-platform tools studied in the article are PhoneGap, PhoneGap & JQuery mobile, PhoneGap & Sencha Touch 2.0 and Titanium. The section about performance evaluation in the article features two subcategories being memory usage, CPU usage.

Methodology: Statistical-based approach.

Solution: Conclusions are that cross-platform development tools have lower costs and quicker time to market at the cost of user experience. Out of the evaluated platforms PhoneGap uses the least amount resources but has a very simple user experience.

Limitation: The empirical study was limited to evaluation of hybrid application only, leaving the other approaches e.g. Native, Web. Also the system does not give an analysis of user interface.

Dan et al (2015): "evaluates the performance Study of Hybrid Mobile Applications Compared to Native Applications using android platform, and titanium framework". They eevaluate the performance difference between native and hybrid applications when accessing the device native hardware through the low level API. This study evaluates the performance and ability to access the device low-level API in An-droid and Titanium, in context of this study performance will be defined by execution time, disk storage space and memory usage.

Methodology: Statistical-based approach (The general benchmark was created and tested by using a Prime number program to evaluate the general performance difference between Android and Titanium. A prime number program was created to find the all prime number within 100 000 numbers. The bench-mark was created identically between Android and Titanium to get as accurate data as possible.).

Solution: From the results listed below it is clear that Titanium has an advantage over Android in terms of execution time in the prime number benchmark, the reason for this advantage is that the functions used from the math library in the prime number benchmark are more efficient in Titanium than Android, this was concluded by breaking the prime number benchmark down into smaller pieces e.g. loop and math functions benchmarks.

Limitation: simple test applications were developed to evaluate the performance of low-level API functions and general performance.

Ghada et al (2013): "evaluates the performance Study of Hybrid Mobile Applications Compared to Native Applications using android platform, and titanium framework***".*** Statistical-based approach (created and tested by using a Prime number program to evaluate the general performance difference. From the results listed below it is clear that Titanium has an advantage over Android in terms of execution time in the prime number benchmark. Only seven simple test applications were developed to evaluate the performance of low-level API functions and general performance

6. RESEARCH FRAMEWORK AND DESIGN

Research framework is a collection of things to ask and things to observe in particular contexts, along with contextually appropriate techniques for doing so. It also includes processes for integrating research/data from other practices areas as well as specific methodologies for making meaning of the raw research.

We now give a thorough testing to performance of android mobile application (native and hybrid application) and hence came up with the framework shown below:

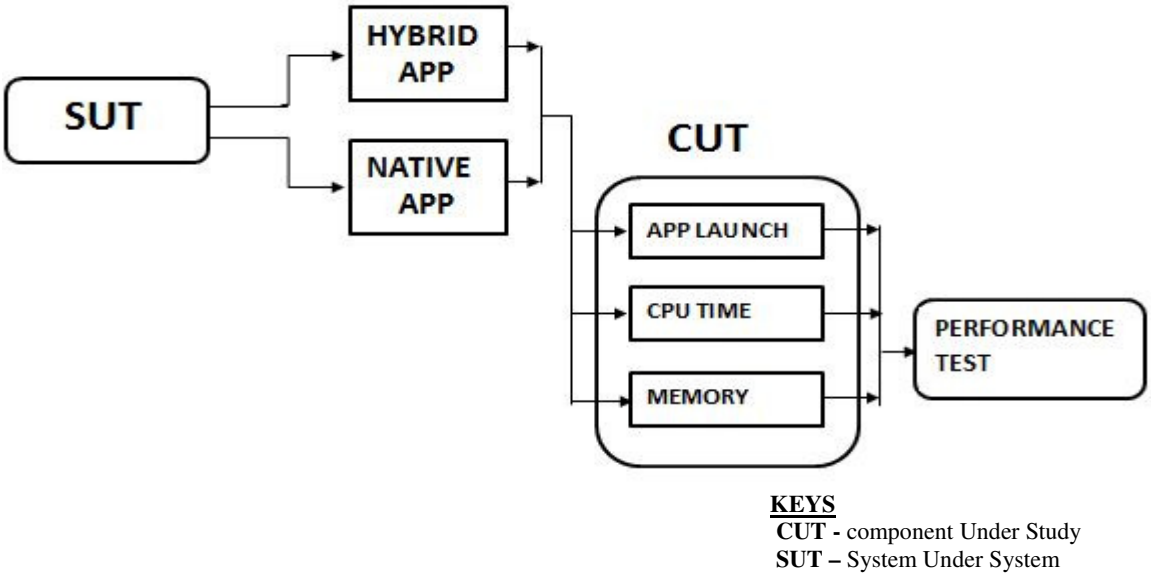

Fig. 1.0 Contextual representation of the proposed model for performance testing

This model shows the general flow chart. A flow which depicts the process involved in implementing and testing the actual component needed.

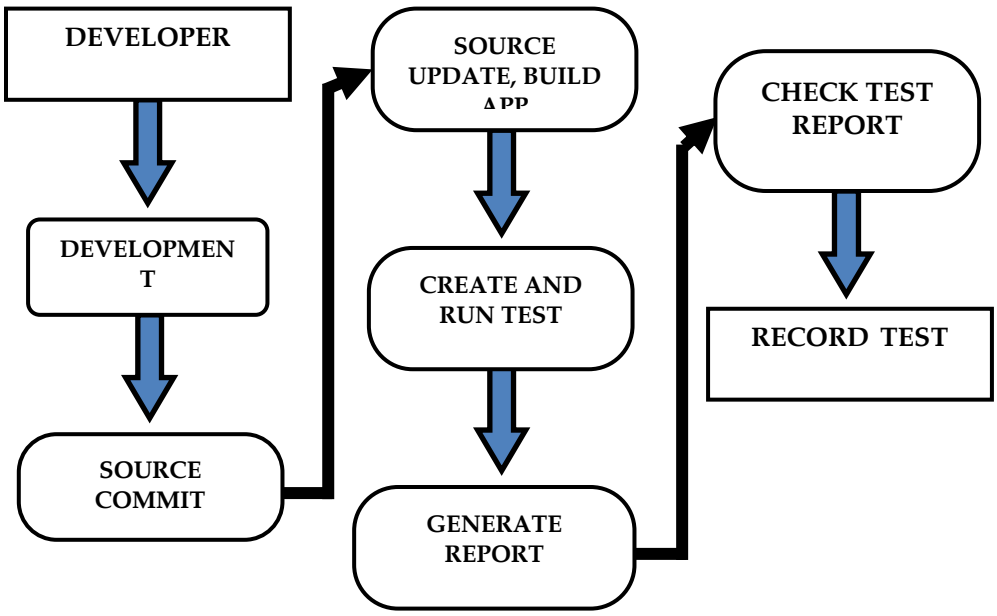

Fig. 2.0 Process flow of the android mobile performance testing

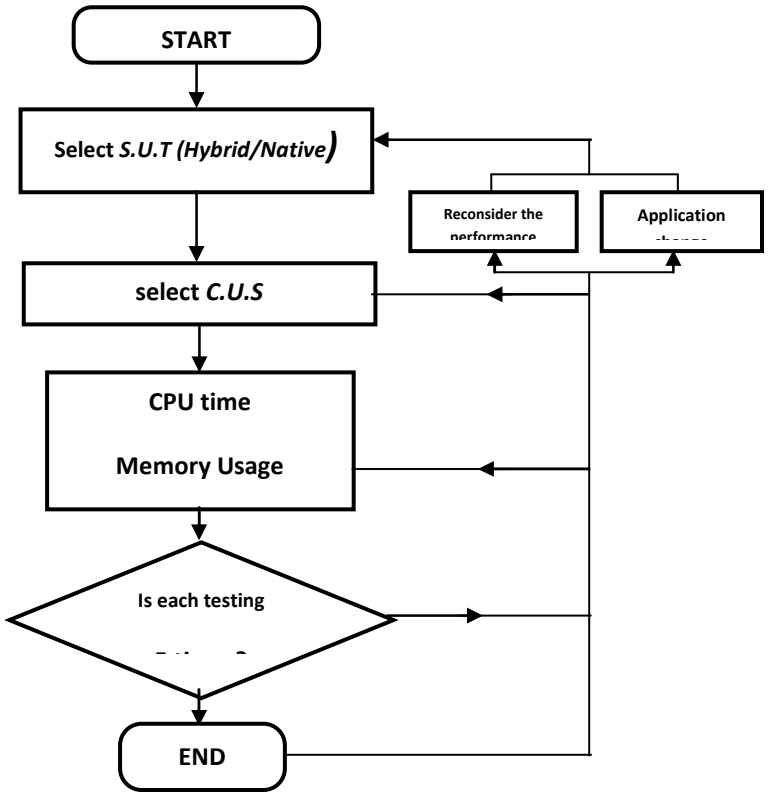

Fig. 3.0 General flow chart for the proposed performance testing of hybrid and native android app

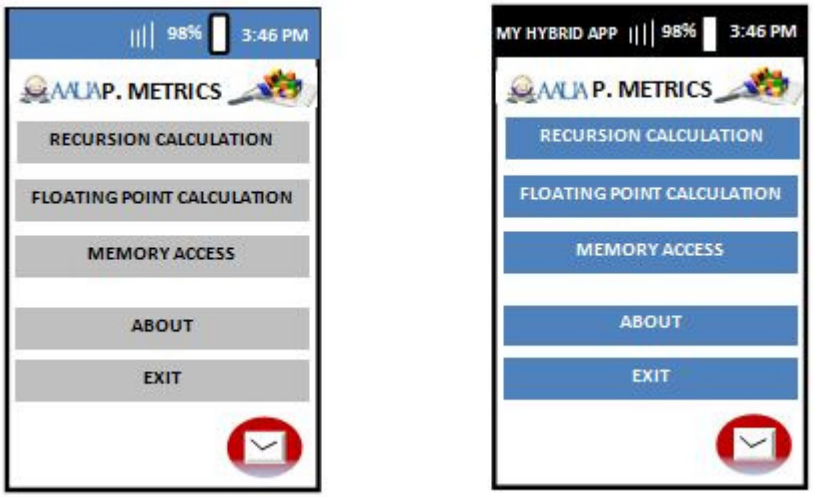

Fig. 4.0 Prototype Screen for first interface in both Native and hybrid app

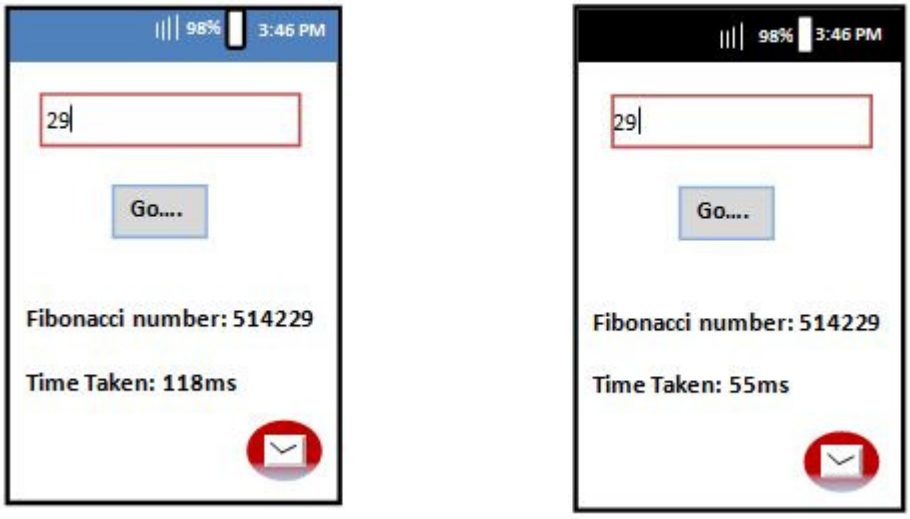

Fig. 5.0 Prototype Screen for Fibonacci number Application using native and hybrid approach as CPU benchmark

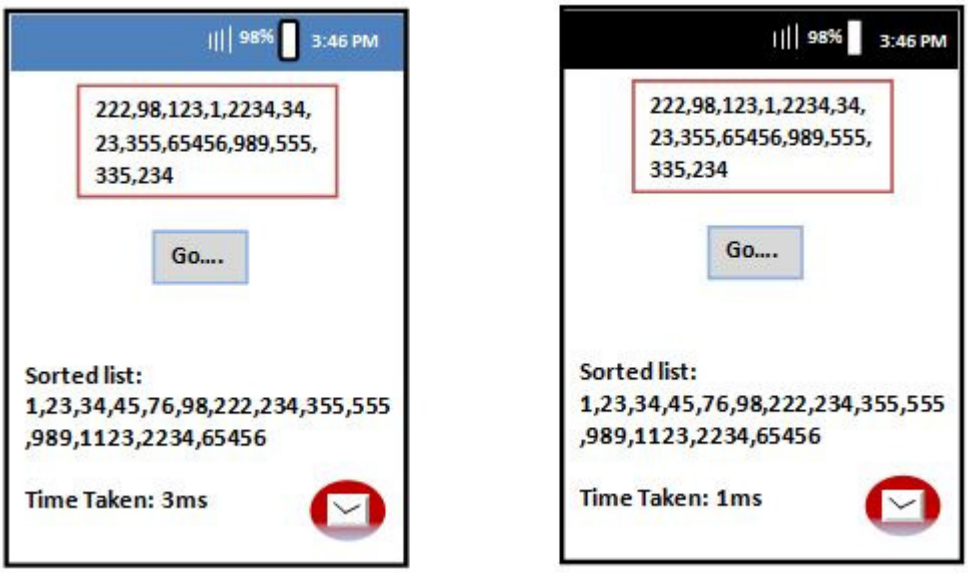

Fig. 6.0 Prototype Screen for Merge sort Application using native and hybrid approach as Memory usage benchmark

6.1 DATA PRESENTATION

Table 1.0 Data for Response time performance measurement on both app

System type	Test 1	Test 2	Test 3	Test 4	Test 5	Average Time
Native	201	200	200	200	200	**200.2**
Hybrid	2000.12	2001.34	2000.10	2002.42	2000.27	**2000.85**

Table 2.0 Data for CPU performance measurement on both app

Time / Value	SUT	Test 1	Test 2	Test 3	Test 4	Test 5	Average Time
50	**Native**	1	1	1	0	1	**0.8**
	Hybrid	78.526	60.132	50.330	46.181	40.015	**55.0**
100	**Native**	1	1	1	1	1	**1.0**
	Hybrid	59.118	58.478	44.073	43.283	42.962	**49.6**
300	**Native**	3	2	2	2	2	**2.2**
	Hybrid	66.608	58.420	54.136	50.308	50.112	**55.9**
400	**Native**	4	4	3	3	3	**3.4**
	Hybrid	54.646	51.044	49.212	47.993	41.972	**49.0**
500	**Native**	5	4	3	3	2	**3.4**
	Hybrid	84.142	54.314	45.744	41.809	39.319	**53.1**
1000	**Native**	6	6	5	5	3	**5.0**
	Hybrid	61.674	60.880	58.964	45.211	41.285	**53.6**
1500	**Native**	7	6	5	4	3	**5.0**
	Hybrid	86.256	62.378	52.227	50.887	49.747	**60.3**

Table 3.0 Data for Memory Access performance measurement on native and hybrid app

Time / Value	SUT	Test 1	Test 2	Test 3	Test 4	Test 5	Average Time
100	**Native**	0	0	0	1	0	**0.2**
	Hybrid	20.23	19.10	18.01	18.48	16.87	**18.538**
150	**Native**	2	1	1	1	1	**1.2**
	Hybrid	37.34	36.11	35.29	34.23	33.34	**35.262**
200	**Native**	5	6	7	6	5	**5.8**
	Hybrid	47.12	46.21	44.43	44.25	43.16	**45.034**
250	**Native**	26	29	28	27	26	**27.2**
	Hybrid	56.20	54.10	54.29	53.40	52.13	**54.024**
300	**Native**	49	49	48	47	46	**47.8**
	Hybrid	75.48	76.45	74.23	74.32	73.67	**74.83**
400	**Native**	62	65	66	65	64	**64.4**
	Hybrid	108.36	108.34	107.23	104.25	102.82	**106.2**
500	**Native**	117	116	106	107	102	**109.6**
	Hybrid	209.10	206.26	207.18	205.47	204.46	**206.494**

Table 4.0 Data for CPU performance measurement on both app

Time / Value	SUT	Test 1	Test 2	Test 3	Test 4	Test 5	Average Time
10	**Native**	0	0	0	1	0	**0.2**
	Hybrid	3.80	3.02	3.60	3.04	1.87	**3.1**
15	**Native**	1	1	1	1	1	**1.0**
	Hybrid	4.65	4.71	4.52	4.30	2.86	**4.2**
20	**Native**	1	1	1	1	2	**1.2**
	Hybrid	8.73	8.18	8.13	7.98	7.91	**8.2**
25	**Native**	14	7	7	6	6	**8.0**
	Hybrid	43.34	38.72	38.70	38.43	23.48	**36.5**
30	**Native**	67	58	58	57	54	**58.8**
	Hybrid	284.03	278.47	270.70	260.40	242.31	**267.2**
35	**Native**	586	580	580	577	578	**580.2**
	Hybrid	2643.61	2635.98	2608.58	2551.37	2533.77	**2594.7**

6.2 EVALUATION RESULT

6.2.1 Response time

Results: The outcome of the first test case reveals that, while the native application is nearly immediately loaded, the hybrid counterpart is significantly slower by a factor of around 20 (see table 1.0). With a startup time of more than two seconds

The hybrid app shows a remarkable delay, which is noticeable when run. In an app, which contains real content, this additional loading time may negatively influence a user's satisfaction. When analyzing the start-up process further, it becomes clear that the native shell, which wraps hybrid apps, takes up a majority of the time span followed by the UI toolkit's loading time. During this time, the internal Cordova server is started, which refers JavaScript calls to their native counterparts.

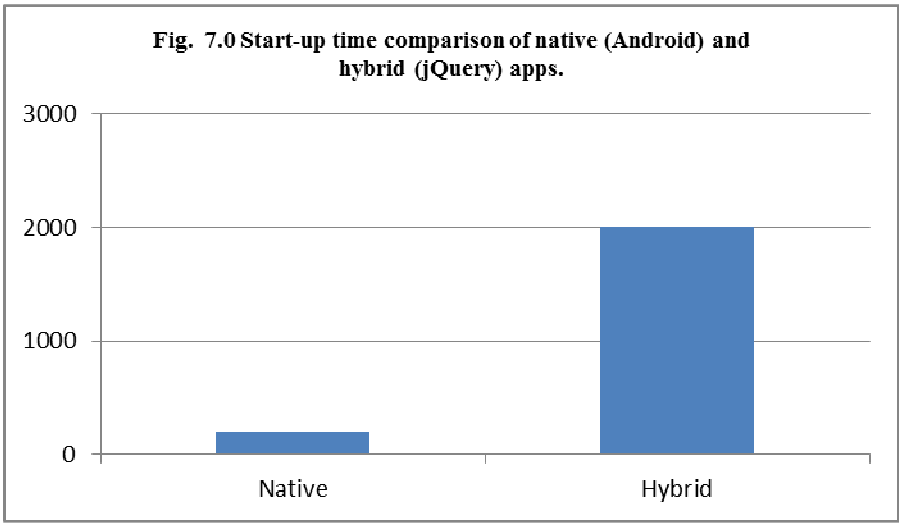

Fig. 7.0 Start-up time comparison of native (Android) and hybrid (jQuery) apps.

6.2.2 Integer Calculation

Results: A Graph showing timed performance in milliseconds the implementation of the Armstrong Number calculation test.

An integer calculation test performed on hybrid and native shows that the java implementation is about 6 times faster than the hybrid implementation, which could work as an indication for this particular test, namely that Native implementation is faster on Android. However, since the test device (Tecno 7c) has a dual-core Cortex-A7 CPU, it is likely to believe that there are significant performance differences between any other test devices that might be used, with Tecno 7c being much faster, since it uses dedicated hardware for calculation. This means that Native implementation will probably be faster than hybrid, at least on Tecno 7c.

Armstrong Integer is used as a benchmark for the CPU time performance measurement due to the loop nature required to get the right output, where this loop only depends on how fast the App can relate with the processor of the device used to make the process as quick as possible.

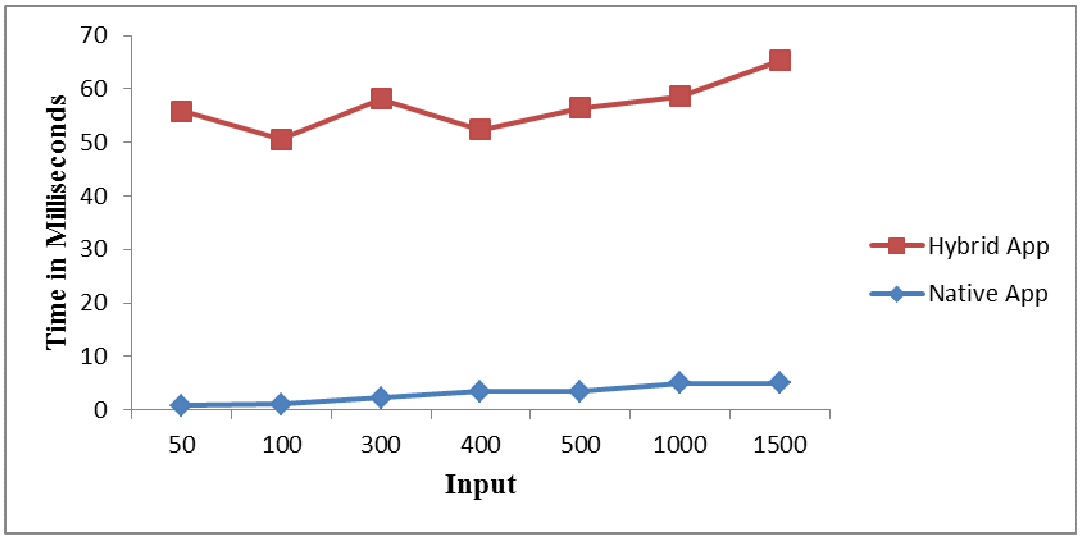

Fig. 8.0: A graph showing timed performance in milliseconds for each Platform and implementation of the Armstrong Calculation test. The plots are estimated from 7 measured data of different input parameters.

6.2.3 Recursive

Results: A graph showing timed performance in milliseconds the implementation of the Fibonacci number calculation test.

This test will produce lots of method calls. Informal sources state that method calls in Java are basically free, where one of the arguments consists of smart compilers that will inline automatically. Inline boosts performance by in lining the code instead of making a method call. In recursion however, this is not a possibility due to the constant method calls that results in an immense amount of code to inline. Native is still expected to hold strong against hybrid.

The results are presented in the figures below.

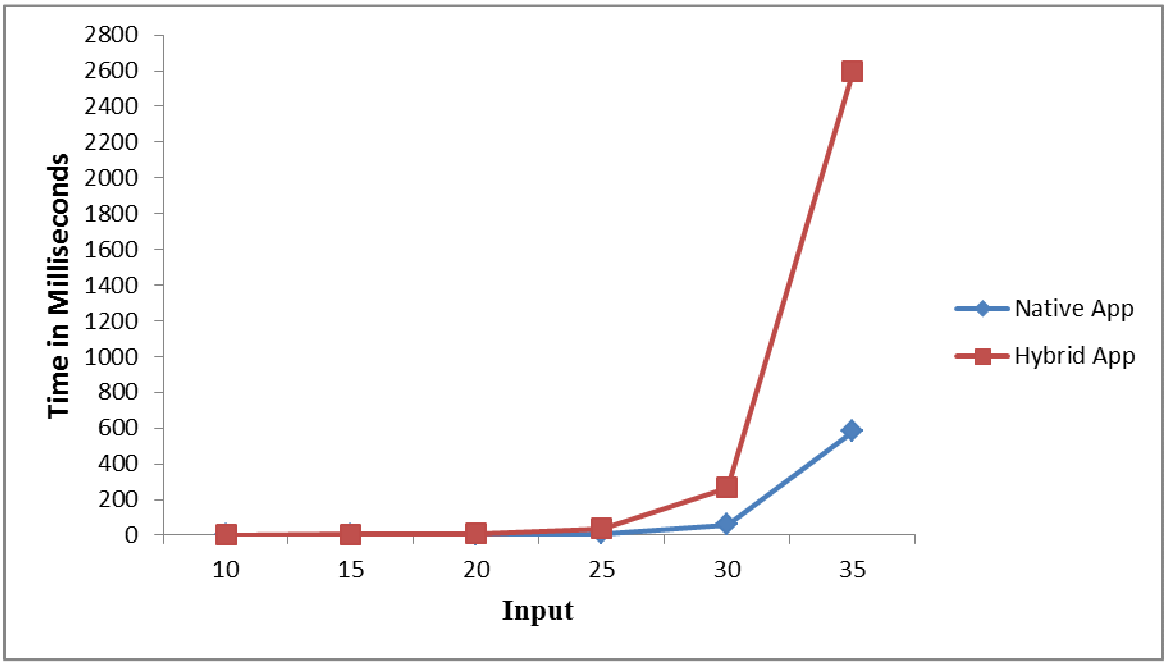

Fig. 9.0: A graph showing timed performance in milliseconds for each platform and implementation of the Recursion test. The plots are estimated from 7 measured data of different input parameters.

6.2.4 Memory Access

Results: A graph showing timed performance in milliseconds the implementation of the Memory Access test.

We implemented a module in simple application to check whether there was something overshadowing the changes arising from performance evaluation based on memory usage. We evaluate this by a Quicksort implementation using no specific Android library. Resulting measurements for memory usage can be seen in ***Fig. 10***. The graph shows the variation of the data in sequential executions. One can observe a very large variation in the measurement between the two applications executions of the Quicksort app.

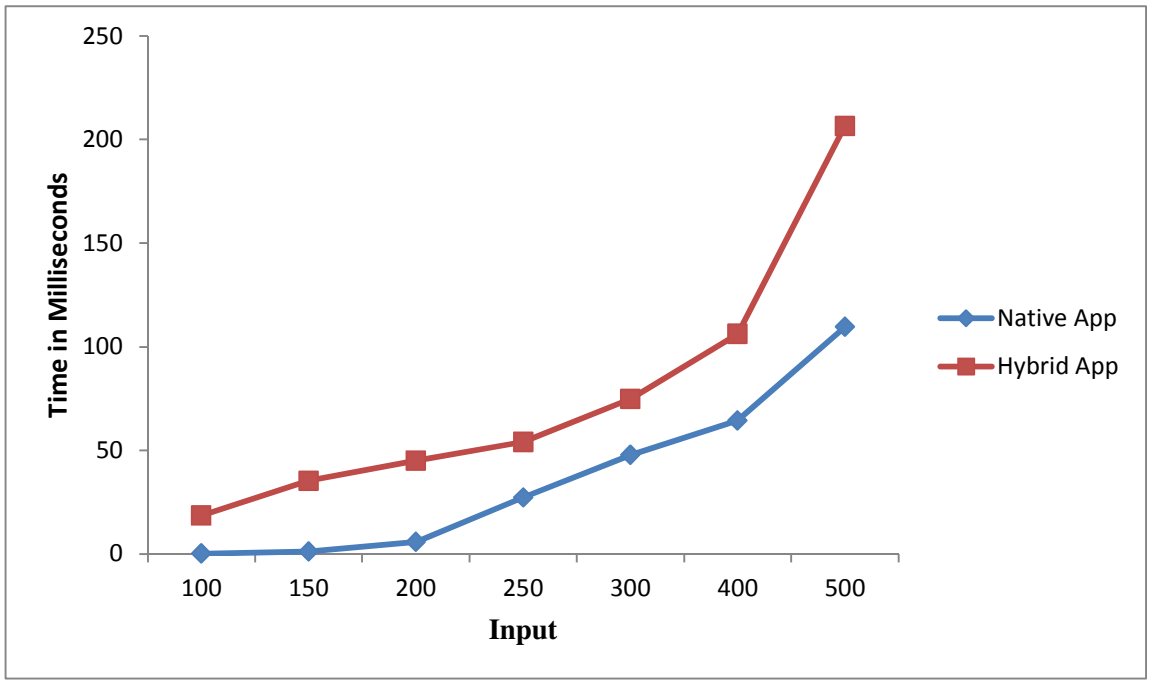

Fig. 10.0: A graph showing timed performance in milliseconds for each Platform and implementation of the Memory Access test. The plots are estimated from 7 measured data of different input parameters.

Memory Access is a benchmark where the difference between JQuery and native code is expected to be the most. Sources indicate that memory access in hybrid is one of its bottlenecks partly due to bound checks. Benchmarks show results of huge difference in performance. A performance expectation of the native implementation is therefore high compared to the hybrid implementation.

7. CONCLUSION

The goal of this research is to evaluate the performance of android hybrid and native mobile applications and to explore the limitation of the mobile android development frameworks. The research was conducted by selecting the SUT and CUS (System under study and Component under study). We then perform a computation and record the time taken, which is later visualized with a graph.In Integer calculation there is slight difference between the performances of the platform, both the app showed no much different in results. In Integer calculation not much difference was seen in the time of Native implementation and Hybrid implementation. Using recursion showed remarkable differences between the performance of native and hybrid.

As the input grows the difference increases on both the platform. Merge sort results showed not much difference in the native and hybrid implementation Hybrid apps were analyzed in terms of performance efficiency, which is an important factor for the software quality of apps. In all conducted tests, native apps were superior to hybrid apps. Since performance is considered crucial for user experience, low performance is likely to influence a user's satisfaction and rating of the app.

8. RECOMMENDATION

Despite this, companies should focus on their clients who expect a satisfying performance, which is more likely to be achieved with the native approach. Some cases cannot yet be covered sufficiently in terms of responsiveness using hybrid approaches. Although web technologies and hybrid frameworks are progressing steadily, native development prevails, at least for consumer-facing apps. While many papers have already covered performance efficiency of hybrid mobile apps, there is still no clear statement of which approach to choose for a certain project.

8.1 FUTURE WORK

Is of no doubt the world is driven towards a mobile view, so there is need for us to thoroughly test our app for better performance. This research as at when conducted was done based on 3 components (CPU usage, Memory usage and App load time), we will recommend the addition of battery consumption to the following component.

This research was conducted based on android 5.0 (Lollipop) platform and device with 1GB RAM, we will also suggest an upgraded device architecture (e.g. with android 6.0 - Machshallow) with internal storage of 2GB for better evaluation and conclusion. This evaluation was strictly based on performance metrics only, but work can still be done by evaluating Functional/UI Testing, and Interruption Testing. Since conclusion as being drawn from this research work that hybrid is slower compare to native application, much work can still be done to evaluate the reason in the delay of hybrid application to further improve on this framework.

REFERENCES

1. Adobe PhoneGap (2013a). PhoneGap Documentation Overview. [online] Available at: <http://docs.phonegap.com/en/2.9.0/guide_overview_index.m d.html#Overview> [Accessed 04 January 2016].

2. Charland, A. and LeRoux, B. (2011). Mobile Application Development: Web vs. Native. In *Communications of the ACM,V*ol. 54, 5, pp. 49-53. DOI= http://dx.doi.org/10.1145/1941487.1941504.

3. Masi, G. Cantone, M. Mastrofini, G. Calavaro, and P. Subiaco (2012). *Mobile apps development: A framework fortechnology decision making;* in *Proceedings of International Conference on Mobile Computing, Applications, and Services.*, ser. MobiCASE'4, pp. 64–79.

4. Lim, C. Min, Y.I Eom (2013). "Enhancing Application Performance by Memory Partitioning in AndroidPlatforms,"IEEE International Conference on Consumer Electronics (ICCE), 2013.

5. Heitkötter, H. Hanschke, S., and Majchrzak, T. (2012). Comparing Cross-Platform Development Approaches For MobileApplications. *Lecture Notes in Business Information Processing*, vol. 140, pp. 120-138.

6. H.J Yoon (2012). *A Study on the Performance of Android Platform*, International Journal on Computer Science and Engineering (IJCSE), Vol. No. 4.

7. IBM Software. (2014). *Require technical skills for mobile application development.* Retrieved from http://www.computerworld.com.au/whitepaper/371126/

8. Kan, S. (2002). *Metrics and models in software quality engineering* (2nd ed.). Addison-Wesley. Schiller, J. (2000). *Mobile communications.* Addison-Wesley.
9. N. Lee, S. Lim (2011). "A Whole Layer Performance Analysis Method for Android Platforms," IEEE, 2011.
10. Peter Skogsberg (2013). *Quantitative indicators of a successful mobile application*; Retrieved from http://th4.ilovetranslator.com/rrS-ZXXNfD=d/