Android: Resource Consumption in Native and Web Applications

Christoffer Cortes and Adam Krauser

Contact Information:

Authors:
Adam Krauser       Adam.Krauser90@gmail.com
Christoffer Cortes Christoffer.Cortes@gmail.com

University advisor:
Kari Rönnkö
Nina Dzamashvili Fogelström

School of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona
Sweden

Internet: www.bth.se/com
Phone: +46 455 38 50 00
Fax: +46 455 38 50 57
Abstract

There is an ongoing debate by people in the industry whether to make native or web applications. These discussions mostly surround issues about development costs, user experience and capabilities. Another aspect of this debate is the fact that mobile devices have varying hardware specifications which is another factor to consider when making this decision.

What we want to shed some light on is how performance is affected on the device when using these two different approaches of application development. The use of CPU/RAM and Energy is our primary concern and in our experiment we measure these values on two similar applications where one uses Nested Layouts and the other a WebView.

The experiment was made on three different devices with varying specifications. What we found was that Web applications have a bigger impact on overall performance and because of this use more battery. While the debate certainly won't come to a close with results they are conclusive when it comes to the topic of performance and will be of value to developers who are concerned about it.
## Innehåll

Android: Resource Consumption in Native and Web Applications ............................................. 1
Introduction ........................................................................................................................................ 4
  Technology ....................................................................................................................................... 4
    Android ......................................................................................................................................... 4
  HTML/CSS/JavaScript ..................................................................................................................... 5
  UI rendering in Android .................................................................................................................. 5
Literature Study .................................................................................................................................... 7
  Method ........................................................................................................................................... 7
    Research Questions ...................................................................................................................... 7
    Literature Design .......................................................................................................................... 7
  Result ............................................................................................................................................. 9
  Search Results ............................................................................................................................... 9
  Answers ......................................................................................................................................... 10
Empirical Study ................................................................................................................................... 13
  Method .......................................................................................................................................... 13
    Experiment Design ...................................................................................................................... 13
    Preparations ............................................................................................................................... 13
    Application ................................................................................................................................. 15
    Execution ..................................................................................................................................... 18
  Result ........................................................................................................................................... 19
    Samsung Galaxy S2 .................................................................................................................... 19
    Samsung Galaxy 3 ...................................................................................................................... 21
    AllWinner A10 ............................................................................................................................ 23
Discussion and Conclusion ............................................................................................................... 26
Future Work ........................................................................................................................................ 27
References ......................................................................................................................................... 28
Introduction
Because of their small size, mobile devices cannot have as powerful components as Laptops and PCs. This means that the devices will have significantly less resources available. Memory management and alleviating the load on the CPU is something that is always on the minds of the developers of Android.

There is a big debate on the web whether it is best to develop applications using a platforms native UI framework (here referred to as Native Layout) or using WebView which allows developers to display web documents as a UI [18][19][20][21]. Most of these discussions are about development costs, user experience and capabilities. We could not find any academic or non-academic tests that showed that either Native Layouts or WebView has the better performance. This we see as an opportunity to produce new knowledge.

According to the Android Developer site, the hierarchy of the layouts in a native application has an impact on performance depending on how well it is built; using many nested layouts can increase the time to measure, layout and draw the components[7]. On the other hand, the Google Developer site points out that if an application is rendered through WebView, no nested layouts are necessary. In that case, the amount of resources consumed will vary depending on how the HTML document along with its CSS and JavaScript is written[8]. Since these two approaches are handled differently, it is reasonable to assume that there might be a in the amount of resources consumed when using Native Layouts and WebView.

Our goal was to find if there is a significant difference in the amount of resources consumed when using Native Layouts and WebView.

We created two applications, one using Native Layouts and the other using a WebView which had the same functionality. We ran a use case on both applications and collected data about how much CPU, RAM and power it had consumed during each operation. This was done on three different devices.

What we found out is that WebView consumes significantly more resources than Native Layouts. The WebView used between 2,44 to 6,2 times more CPU (depending on device), 1,8 to 7 times more RAM (depending on device), 2,5 to 10 times more power (depending on device) than Native Layouts.

These results might be used by developers concerned about performance in their applications.

Technology

Android
The Android OS is developed and maintained by Google. Though not originally conceived by Google it was bought by them in 2005 from Android Inc. It’s a smart phone and tablet operating system that is mainly intended for touchscreens. In earlier releases hardware buttons were an option but as time has moved on these functionalities have been migrated to the actual display surface. It’s developed as an Open Source OS and phone makers can modify and use the OS liberally.
The latest version is 4.2 “Jellybean” and changes are constantly being made. The downside of this is that source code of older applications can become deprecated. The adoption of newer versions is up to the manufacturers and many either take a long time to update or simply don’t support upgrades on certain models. This is the reason why a large portion of devices still run 2.3 “Gingerbread”[1].

The Android market share reached 70% on the 4th quarter 2012 according to IDC, with roughly 159 million devices shipped[2] and recently Larry Page, CEO of Google Inc., claimed that 750 million devices have been activated globally[3]. These are impressive figures that put Android at the top of the smart phone market.

Android is based on the Linux kernel with code written in C. Although the OS is open source users are not given full access as root. The applications themselves are written in Java and Google provides a complete developers package for Eclipse. To be able to run this Java code on Android, applications are compiled and converted to run through Dalvik VM. Given that applications are written in Java, developers have access to plenty of built in libraries.

When applications are in a paused state the OS will try to minimize the impact of the program running on the battery. It does this by putting applications in a suspended state and if the system is in a low memory situation it will kill applications to conserve energy. There are official tips and tools that can be used by anyone willing to take the right steps to improve performance and minimize impact on battery life. The best practice guidelines are available on the developers page and do not only concern performance but also things such as user experience and security. There is only so much they can do though and it is up to every developer to optimize their own products.

**HTML/CSS/JavaScript**
When writing a web page, three languages are used HTML, CSS and JavaScript. HTML is used for the structuring of the content of the page while CSS manages the presentation. JavaScript allows the webpage to become dynamic; information and elements can change without having to reload the webpage.

**UI rendering in Android**
Android applications use layouts to structure their content and are defined in an XML file similar to the HTML. They can contain components as well as other layouts, which could lead to a very complex hierarchy of nested layouts[4].

The Android OS allows developers to display web pages either in an internet browser app in their applications using a specialized layout called WebView. WebView works like an embedded browser and handles the connection to the website server and the translating of HTML and CSS[5]. It is also possible to bind the JavaScript code to the Android code, allowing the developer to use the native android UI visuals such as Dialog and Toast. It is also possible to handle links in the WebView as well as navigating backwards and forwards[6].
Literature Study

Method

Research Questions

We had three important questions that needed to be answered before any kind of testing could be done. The first and the second questions concerns how performance is affected by the different UI design methods and how they can be improved according to the suggested best practices. This is something that we wanted to find as much information about as possible since it is central to our comparison. The value in this was so that our applications could be built in such a way that we did not see performance loss because of badly structured code. This we felt was of great importance if we were going to conduct a fair comparison. The third question is directly related to the experiment where we needed to gather data on resource consumption for our applications. This of course needed to be done with reliable numbers which made this question essential.

1. How does Native Layout design affect performance and what are some of the suggested practices that increase performance?
2. How does HTML/CSS/JavaScript design affect performance and what are some of the suggested practices that increase performance?
3. How can resources used by an application be measured correctly?

Once we answered these questions we could move on to the empirical study to answer our main question.

- Is there a difference in the amount of resources consumed when using Native Layouts or WebView?

Literature Design

When searching for papers/journals etc. we mainly used Engineering Village and Summon BTH. Both of these sites use a meta crawler and find information from several sources. When deciding if a paper was relevant or not, we read the abstract of papers whose title seemed relevant and if the abstract was interesting to us then we continued on to reading the introduction and conclusion. If at that point we found that the paper is still interesting to our work then we read the whole paper to make sure it was what we are looking for.

The search string used to find information on the first questions is as follows:

(\text{Android AND Layout AND (Performance OR CPU OR RAM OR Batter* OR Power OR Energy OR "Best practice" OR Design)})

It encapsulates a fair amount of keywords and synonyms. This string gave a handful of results in Engineering Village and about 316 on Summon BTH where the search was limited to Journal Articles and Conference Proceedings. Another source was Androids official developer site.

For the second question we used the following string:
There aren’t any synonyms for “best practice” so we simply searched for design as an alternative in order to broaden the search hits. About 9000 hits were found in total.

On our third and last question we used this string:

```
Android AND (Battery OR Power OR Energy OR CPU OR RAM OR Layout) AND (Measure OR Consumption)
```

As well as academic sources we searched the web for tools and advice from Androids and Google’s developer sites.
**Result**

**Search Results**
We had a hard time actually finding something that was directly relevant to our topic. Many of the results we got touched the subject of energy/battery but did not go our specific issue. This gave us motivation to carry on with our study but at the same time leaves us starving for relevant academic sources.

The results we got in Engineering Village and BTH Summon only gave us a handful of results that were of some relevance but turned out to give us little information that we needed. Already when looking at page 2 the relevance to our first question seemed to thin out. This isn’t such a big issue though since we suspected from the start that this search might not be so fruitful. The Android developer site on the other hand gave a great deal of information on how layouts should be structured.

We found one paper that had done testing on HTML elements and CSS on mobile devices, and it provided a bit of good information. Google developer has a section called ‘Make the Web faster’ that gives plenty of tips on designing web sites. These developer sites were our main source of information regarding this question.

We managed to find a couple papers about energy measurement which were of good use to us. Sadly we could not find any general or directly related information to our topic when it comes to CPU or RAM measurement. After looking around on the web and the developer site we found a tool that visualizes time spent on specific activities by the CPU. It’s an official debugging program called Traceview. When it comes to RAM measurement we found an add-on for Eclipse called Memory Analyzer (MAT). There are other programs as well that we looked at. PowerTutor and Little Eye were two of these tools that are able to measure energy consumption and the latter can also measure CPU and RAM performance. The important thing was to get the correct measurements.

Overall we found that there wasn’t a lot of information regarding our specific issue which bolsters our position. While we might not be able to make many direct references our study has a clear purpose and will provide useful results.
Answers

**Question 1: How does Native Layout design affect performance and what are some of the suggested practices that increase performance?**

The Android developers site gave several tips on how to optimize layout design. The Android SDK includes a tool called the Hierarchy Viewer which allows the developer to optimize and debug the UI. The Hierarchy Viewer visualizes the layout hierarchy of a running application and takes measurements of how long it takes to measure, layout and draw the components of a UI. Using this tool will help in identifying areas that are more performance intensive[9].

The developer site recommends developers to avoid complex hierarchies created by many nested layouts. It advises developers to try and flatten the hierarchy of their layouts by using Relative Layouts that provide more control over its children components than Linear layouts[7].

The Android Development Tools for Eclipse contain a Lint tool that helps the developer by analyzing their code and identifying errors and performance traps, among them are layout performance problems[10].

**Question 2: How does HTML/CSS/JavaScript design affect performance and what are some of the suggested practices that increase performance?**

The greater part of advice given regarding web performance is about optimizing code in a way that helps the website render on slower internet connections. These advices are irrelevant to our study since our application will be hosted locally. What we are concerned with is what increases performance and reduces energy consumption.

The Google developer site gives several tips to reduce the workload on the browser[8]. These are the ones that are relevant to us:

1. Efficient CSS selectors.
   a. Avoid universal selectors.
   b. Rules should be as specific as possible, using class and id is better than tags.
   c. Descendant and inherited selector should be avoided because they are inefficient..
   d. Remove redundancies such as overly qualified selectors.
2. Image size should be specified. If image size is not present, then the browser cannot layout the other elements around the image and will have to repaint the document once it knows the dimensions of the image.

Sampson et al. made a study showing several pitfalls in HTML and CSS[11]. They found that using the CSS opacity controls slow down the rendering of translucent elements on mobile devices. They also found that using Background fills on an Android device used a significant amount of energy. The <SPAN> element was also found to negatively affect performance and energy.
Question 3: How can resources used by an application be measured correctly?

A study was made by Shye et al. that showed that not only does a lot of energy go into keeping the device idle but the biggest two energy hungry components are the screen and the CPU. The idle energy consumption is not relevant to our study and the screen energy is not something that we will be able to affect unless we regulate the brightness. They determined that 16.3% of the energy consumption came from the screen just being turned on and 19.2% from the brightness setting. Assuming that we will not be fiddling with the brightness between experiments there is not much else to do than to deduct the consumption from the screen. The goal is after all only to measure the consumption of our specific applications[12].

Kapetanakis and Panagiotakis have also conducted an energy study and they explain how they went about getting the data. They mention that when conducting their tests they deactivated any function that can consume energy like for example WI-FI, etc. and that they reverted devices back to factory settings in order to prevent unnecessary applications from running in the background and bog down the results[13].

When it comes to energy consumption on HTML pages the rendering of the elements and transmission is what uses energy. Thiagarajan et al. have made a study around the subject of webpage energy consumption. While there experiment method might not directly apply to Android WebViews they came up with interesting data. They took a set of sites and measured how much different web components used in energy. While this differs between sites depending on how much JavaScript is used or perhaps how if images are positioned with HTML tags or with CSS etc., the general trend was that CSS and JavaScripts were the most energy consuming components. This will be taken into consideration. The way they set up the test is well documented and we will use their findings as a comparison to ours[14].

To measure energy consumption by specific applications appears to be a complex thing. Zhang et al. made a study around the PowerBooster and PowerTutor tools. Most interesting is the PowerTutor tool which can give the user accurate power estimations on running applications. They concluded that in 10-second intervals it averages at 0.8% with at most 2.5% error rate[15]. These are fairly accurate results.

We also found a tool called Little Eye that is capable of measuring Power/CPU/RAM.
Empirical Study

Method

Experiment Design
Two applications are built, they have the same functionality and as similar presentation as possible. The first is an application that uses Native Layouts and the second uses a WebView to display an HTML document. One use case would be performed on the applications and the following measurements would be taken:

- CPU usage
- Ram usage
- Power consumption

The data is then compiled and analyzed to see if there is a difference.

When looking at CPU consumption we decided to go for the average CPU load over a five second period. The reason for this is that we want to catch the entire load of a task and the longest CPU load of any single task across all platforms was five seconds. That way we could keep a common unit on all tests.

The RAM is measured in KB and the power is measured as the total amount of mW used by the application for every task.

Preparations

Devices
We had three distinctly different devices available for this experiment.

- Samsung Galaxy S2 (Smartphone, 2011, high-end), running Android 4.2 (Jelly Bean)
- Samsung Galaxy 3 (Smartphone, 2010, low-end), running Android 2.2 (Froyo)
- AllWinner A10 custom (tablet, 2011, low-end), running Android 4.0 (Ice Cream Sandwich)

Both the Samsung phones were retail bought but the AllWinner A10 is a custom version with a different case and it lacks a battery for portable use. The lack of a battery did however not affect our power measurements and the values could still be read. All three devices were connected to a computer with a micro USB cord during the tests.

Measuring Tool
After doing some searching we decided to use Little Eye[16]. It’s a tool that can monitor CPU/RAM/Energy activity on any user application on android phones. It also records the entire use case and you can easily see what you were doing at the moment of a CPU spike for example. We were a bit worried about the accuracy of the numbers so we decided to verify them by checking the values on the phone through ADB shell[17]. Memory readings were accurate as well as Energy consumption and CPU measurements were more accurate than the readings we had been making previously through bash scripts. The other option, PowerTutor,
showed only the total power used while Little Eye also gave us the power consumption for each timeframe. This was important because we wanted to get the power consumption for each elapsed second. Another big advantage of Little Eye was that it also measured CPU and RAM consumption as we mentioned before.
**Application**

The application that the experiment was performed on is a simple recipe application called Strawberry Treat. The recipes are divided into different packages and it is possible to see information about the recipe such as a description, ingredient count, instructions and nutrition values as well as an image.

When the application opens up, a splash screen is shown while the application loads all the data about the recipes from an SQLite database and puts it into a data model so that it can be easily retrieved by the user interface with minimal effort. Images are the only thing that is not loaded into memory, instead a file path is stored in the data model. Once the data is loaded, the application switches from the splash-screen to another activity/page containing a ListView/table where all the recipe packages are listed. Clicking on a recipe package makes the application switch to a new activity/page containing a ListView/table listing all the recipes in that package. When a recipe is selected a new activity/page is presented showing an accordion with 5 different tabs containing the following:

- ImageView/img tag that contains an image of the meal.
- TextView/div that contains a description of the meal.
- ListView/table that contains a list of the ingredients as well as buttons to change the number of servings.
- TextView/div that contains instructions on how to make the meal.
- TableLayout/table that contains nutrition facts about the meal.

The following mockups describe our application in pictures and should provide anyone wanting to recreate this experiment with a good picture of how it was formed:
**Recipe List**

- Apple Pie
- Lava Cake
- Peach Pie
- Strawberry Pie
- Banana Crumb Muffins

**Recipe Image**

Apple Pie

**Description**

Using fresh apples, a homemade pie crust recipe, and the perfect blend of spices; this is the apple pie that is true delight in every forkful.

**Ingredients**

- 1
  - Crusts
- 3
  - Apples, sliced & peeled
- 2
  - White sugar

**Instructions**

- Recipe Ingredients
Application Architecture

Both applications use the same logic to load and store the data, this means that the only difference between the applications is the GUI, this makes the comparison between their respective performances fair since only the parts which we want to measure are different. To help the comparison be more fair we tried to do as little in the GUI as possible.

For the web application we used JQuery which was needed to use an accordion from JQuery UI. The library we used was trimmed down to just the essentials to reduce impact on the application performance. While this might still have some impact on the performance it is necessary for us to use it to make the accordion.
Execution

The use case presented was used during the measurements:

Measurements were taken at all points where the user made a selection. From CPU and Power measurements, we measured from the point that the user made a selection until the application was done with its tasks; while for the RAM measurements, we measured the RAM consumption after the application was done with each of its tasks.
Result

Samsung Galaxy S2

Galaxy S2 - CPU measurements

Galaxy S2 - RAM measurements
The result is very much conclusive. The Native application is far more optimized consistently takes CPU for all tasks. When accounting for all tasks in Native there is a total load of 6.8% and 37.2% for the WebView. That is a 547% increase in load when using the WebView.

This of course leads the to the power consumption being greater. The Native and WebView applications use 131 mW and 1313 mW respectively.

The RAM in both applications doesn’t move that much and most of it is taken at the point of entry. The Native application saw an increase of 1.6Mb or a 7.4% increase from start to finish while the WebView saw an increase of 2.9Mb and 8.7%.
Samsung Galaxy 3

**Galaxy 3 - CPU measurements**

<table>
<thead>
<tr>
<th></th>
<th>Native</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0.00%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Package</td>
<td>5.60%</td>
<td>19.00%</td>
</tr>
<tr>
<td>Recipe</td>
<td>6.40%</td>
<td>22.60%</td>
</tr>
<tr>
<td>Image</td>
<td>4.80%</td>
<td>11.20%</td>
</tr>
<tr>
<td>Description</td>
<td>5.20%</td>
<td>11.60%</td>
</tr>
<tr>
<td>Ingrident</td>
<td>9.00%</td>
<td>12.80%</td>
</tr>
<tr>
<td>Instruction</td>
<td>9.80%</td>
<td>13.00%</td>
</tr>
<tr>
<td>Nutrition</td>
<td>3.80%</td>
<td>13.80%</td>
</tr>
</tbody>
</table>

**Galaxy 3 - RAM measurements**

<table>
<thead>
<tr>
<th></th>
<th>Native</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>10768</td>
<td>16927</td>
</tr>
<tr>
<td>Package</td>
<td>10904</td>
<td>18352</td>
</tr>
<tr>
<td>Recipe</td>
<td>11004</td>
<td>20196</td>
</tr>
<tr>
<td>Image</td>
<td>11380</td>
<td>20736</td>
</tr>
<tr>
<td>Description</td>
<td>11388</td>
<td>20740</td>
</tr>
<tr>
<td>Ingrident</td>
<td>11348</td>
<td>20824</td>
</tr>
<tr>
<td>Instruction</td>
<td>11364</td>
<td>20952</td>
</tr>
<tr>
<td>Nutrition</td>
<td>11368</td>
<td>21104</td>
</tr>
</tbody>
</table>

**Galaxy 3 - Power measurement**

<table>
<thead>
<tr>
<th></th>
<th>Native</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Package</td>
<td>155</td>
<td>513</td>
</tr>
<tr>
<td>Recipe</td>
<td>103</td>
<td>694</td>
</tr>
<tr>
<td>Image</td>
<td>135</td>
<td>370</td>
</tr>
<tr>
<td>Description</td>
<td>85</td>
<td>327</td>
</tr>
<tr>
<td>Ingrident</td>
<td>272</td>
<td>374</td>
</tr>
<tr>
<td>Instruction</td>
<td>283</td>
<td>365</td>
</tr>
<tr>
<td>Nutrition</td>
<td>97</td>
<td>381</td>
</tr>
</tbody>
</table>
The results show a similar trend to that of Galaxy S2. The gap in CPU load is smaller though still quite pronounced. The reason for this smaller gap could be that the Galaxy 3 is a low-end device and has a tougher time processing the tasks. The Native application has a total load of 42.6% and the WebView shows 104%. This is a 244% increase and a more modest difference in relation to the Galaxy S2.

The energy consumption is for Native and WebView is 1190 mW and 3024 mW respectively. It’s harsh to see that even the Native application uses 908% more energy consumption on this smaller phone compared to the Galaxy S2.

When it comes to RAM this phone had a lower base consumption at the point of entry with almost half that of Galaxy S3 on both applications. The Native application saw an increase of 0.6Mb or a 5.6% increase from start to finish while the WebView saw an increase of 4.2Mb and 24.7%. The WebView was harder on this phone and the relative increase in memory consumption was far higher than on Galaxy S2.
This tablet was the last device that we made the test on and just like on the other devices it shows how the Native application is easier on resources than the WebView one.

The CPU gap is once again relatively big. The Native application has a total load of 16.8% and the WebView has 104% across all tasks. That is a 619% increase in load and is the largest relative difference in CPU load on our three devices.

The energy consumption for Native and WebView is 506 mW and 2628 mW respectively.

The RAM consumption places itself in between the two other devices in entry point. The Native application saw an increase of 1.3Mb or a 7.8% increase from start to finish while the WebView saw an increase of 4.2Mb and 16.7% respectively.
Discussion and Conclusion

There is very much a difference between a Native and a WebView application. We could see some difference between the devices but the general conclusion is that the Native application puts a lot less pressure on the CPU, uses less RAM and in turn consumes less energy.

On all three devices the CPU load for all tasks together was considerably higher on WebView.

- Samsung Galaxy S2 - CPU load: 547% higher on WebView
- Samsung Galaxy 3 - CPU load: 244% higher on WebView
- AllWinner A10 - CPU load: 619% higher on WebView

It appears that JavaScript and CSS isn’t very kind to the CPU. Thiagarajan et al. did a research on this and found out that JavaScript heavy web pages took more energy than those that were lighter and with less complex scripts[14]. Their tests were only done between web pages though while ours are a comparison between Native and WebView.

In the RAM department we can see that the applications take a sizeable amount of memory at the start and then grow ever so slightly.

- Samsung Galaxy S2 - Memory increase - Native: 1,6Mb (7,4%), Web: 2,9Mb (8,7%)
- Samsung Galaxy 3 - Memory increase - Native: 0,6Mb (5,6%), Web: 4,2Mb (24,7%)
- AllWinner A10 - Memory increase - Native: 1,3Mb (7,8%), Web: 4,2Mb (16,7%)

The Native application increased about 5-8% during the entire use case and has a rather relative increase on memory consumption. A reason to why the Native application is increasing by a more steady percentage could be that images are not scaled in Android if you make images of different resolutions available. This is what we did and it is possible that it contributed to the lower load. The WebView on the other hand has a less flexible increase of memory consumption. Galaxy 3 and A10 has the same amount of memory consumption even though they have very different hardware and resolution. The WebView unlike the Native application only uses one set of assets and needs to scale images to fit the screen. No assets are dynamic on the WebView and because of this needs to load the same resources regardless of the device. This we believe is the reason for this inflexible consumption.

Power consumption is also larger on the WebView and this is directly related to the CPU.

- Samsung Galaxy S2 - Power consumption - Native: 131 mW, Web: 1313 mW
- Samsung Galaxy 3 - Power consumption - Native: 1190 mW, Web: 3024 mW
- AllWinner A10 - Power consumption - Native: 506 mW, Web: 2628 mW
Since we do not measure data transfer nor the display that leaves us with the remaining hardware. It’s been clear to this point that the WebView has a heavier load on the CPU and consumes more RAM so it’s no surprise that it also uses more power.

The impact of JQuery on our application does not seem to affect the results in a significant way. The first two operations of our use case do not use JQuery while the later ones do. In all cases the Native application performed better than the Web application.

**Future Work**

What we measured here is the resource consumption by the application to present a certain activity or page. This is the resources it takes to retrieve data, make calculations and draw the UI elements. What could be measured is how much it resources it takes to simply draw UI elements in a Layout or in a WebView. One might also be able to create an application that does not need to use JQuery and get more accurate measurements.

This paper might be improved by testing the applications on more devices. The result might be strengthened if several devices from different manufacturers and different qualities gave the same pattern.
References


