MATAC: Mobile Application Translation Analysis and Concerns

Ethan Holder
Department of Computer Science
Virginia Tech
eholder0@vt.edu

1 Response to Reviewers

I would first like to thank all of the anonymous reviewers for their time and effort to provide such thorough critiques of my proposal. While I would like to address each and every criticism, this medium does not provide the space necessary to individually answer all of them. However, I found that the following themes ran through each review:

1. There needs to be more background and related work discussed upfront, especially about what application translation and energy profiling approaches already exist in the state of the art as well as what the existing tradeoffs actually entail.

2. There needs to be a clearer explanation of how to evaluate the work stated in this proposal in addition to a more defined separation of what work will and will not be performed here.

3. Finally, how I am going to accomplish all of this work needs to be explained in more detail upfront.

Based on these themes, I can say unequivocally that all of those items have been addressed in the final version of this proposal. I had originally intended not to present much background in the introduction, simply because the very next section is to be a lengthy background section that details exactly what related and orthogonal work already exists in the state of the art. However, in addition to including that large section, I've added a brief subsection into the introduction so as to cover more details about the PowerPack approach as well as various translation efforts and the tradeoffs each present.

Beyond that change, I have separated out some of the previous solution subsection. The timeline for paper submissions there originally was intended to touch on evaluation methodology, but it was cut for space. Since the abstract now flows on with the remainder of the paper (without a page break), this extra space allows me to restore those details. The new evaluation subsection should address any previous shortcomings.

Finally, I've added more details to the solution subsection to show exactly how I will perform this research. The impact subsection also has some additions in this light. I've also performed a thorough proofread myself along with an outside reviewer so as to reduce any potential spelling or grammatical errors, as the group review mentioned this problem briefly.

As for the rest of the feedback received, I apologize that I cannot address it formally here, but I appreciate it all nonetheless. Once again, please accept my humblest thanks to all the anonymous reviewers. I hope the changes implemented here in the final revision have addressed all of your concerns and more.

MATAC: Mobile Application Translation Analysis and Concerns

Ethan Holder
Department of Computer Science
Virginia Tech
eholder0@vt.edu

December 15, 2014

Abstract

Modern mobile applications (apps) must run across a plurality of platforms in order to maximize their user base and ultimately, revenue. While app developers currently have a wealth of resources available to facilitate translating an app from one platform to another, there is no single best approach that maximizes performance and aesthetic appeal. Given that developers utilizing these translation solutions are seeking to maximize their user base while minimizing their input time, they need their applications to perform correctly and efficiently after translation. This proposed technique alleviates those concerns by applying the extended PowerPack approach to profile translated apps for energy consumption and performance. It will also leverage existing approaches for comparing aesthetic appeals. This work will provide a novel solution to the mobile app development market, saving developers and users time and resources.

2 Introduction

Mobile applications (apps) running on tablets and smartphones present unique business opportunities in today's world. Smartphone sales alone are expected to surpass 1 billion units in 2014 according to a recent Gartner report [Riv14], while the number of apps available in the Google Play Store [Goo14] and Apple App store [App14] have each already reached over 1 million [Bra14] [Per14], and the number of registered developers for iOS alone are over 9 million. All of this data points to the booming business of apps. More and more developers are creating apps everyday, and these developers need to reach the masses of users out there across platforms in order to maximize their reach. The following subsections will detail what problems these developers face 2.1, why those problems are important 2.2, what the state of the art looks like 2.3, how this proposal will overcome those problems 2.4, how to evaluate this solution 2.5, who will benefit from this work and how much 2.6, and finally what to expect from the rest of this paper 2.7.

2.1 Problem Statement

Mobile translation, also known as cross-platform development, schemes have sought to answer developers' basic needs for far reaching exposure while minimizing their burden for some time. Direct translation, intermediate language creation, and ecosystem emulation have all been implemented previously as improved solutions over simple manual porting. While these solutions are promising in that they all solve the problem of porting an application from one platform to another, each has its own drawbacks. Ideally, after translation these apps need to run at comparable speeds and comparable energy efficiencies across platforms with the native aesthetics specific to each given platform (or at worst, have a unique aesthetic that is not standard to any platform but is reasonably consistent across them all). The current state-of-the-art approaches each do not address all of these concerns on any one system. Furthermore, the plurality of features and tradeoffs present between systems does not make the choice of which to use clear for developers in a given situation. App developers need a clear definition of these tradeoffs to better understand which to use where as well as a system that seeks to maximize each dimension, rather than focusing on one over another.

2.2 Problem Merit

Given the sheer volume of app developers as stated above and that those developers need to maximize their user base to increase their revenue, this problem clearly has far reaching effects. Beyond maximizing their user base, developers also need to minimize their time spent on translating apps so as to allow them more time for value-adding activities, such as implementing new features, fixing troublesome bugs, or enhancing user experience. In this light developers need their apps to perform correctly and efficiently after translation, or else they incur further burdens later in fixing the translated versions.

Obviously, these problems are not trivially solvable in today's market of smartphone ecosystems. However, the need for improvement is dire. Without a proper, robust translation system of some kind, developers are forced to expend significant amounts of time and/or money to address this issue themselves. Even with today's translation solutions, a key dimension of developer concerns is largely neglected due to the lack of energy profiling on mobile devices to date. This work will address each of those problems.

2.3 Background

Previous work has given us many means of developing apps across platforms. Direct translation and cross compilation are the most obvious improvements over native development schemes [PA13, ZTX$^+$10]. However, these approaches still require in depth knowledge of one native platform and are not capable of handling all API's (Application Programming Interfaces). Intermediate language translation shows further progress by allowing apps to be written in a simpler, intermediate language, such as HTML and Javascript in PhoneGap[AGL10, GP12], before being translated or compiled out to other platforms. This allows for more options in the initial development and increased API coverage, but removes the native aesthetics of direct translation. Ecosystem emulation combines these approaches by utilizing an intermediate language and directly translating interactions back and forth during use [HSDT13, Hol13]. This allows for increased coverage over direct translation and purely native aesthetics, but it can incur high latencies during use due to communicating with an emulator, where neither previous approach requires this.

Obviously, each approach has its benefits and drawbacks, however the debate between them currently does not revolve around energy efficiency, but largely on latency and aesthetics [CL11, HHM12]. While latency and aesthetics are no doubt important, it is unwise not to consider the energy impact of one platform over another. Thus, these tradeoffs do not present the whole story for which platform is truly better than another. A detailed analysis of energy consumption is required to thus show more conclusively which approach is more beneficial.

This research will rely heavily on the PowerPack approach of profiling energy consumption. PowerPack relies on capturing the voltage flowing through each power rail to various hardware components while an application is running. By showing exactly what call in code generates what change in energy, it determines how much energy each component drains for each call. It was previously implemented on larger scale systems rather than mobiles [GFS+10], but has since seen more studies on mobile devices [CAC11]. In my own work with Dr. Cameron's SCAPE lab, we have already implemented PowerPack on the Asus TF300T tablet with interesting results [Asu14]. Research by other groups has proved that profiling mobile energy is not only possible but can yield many novel results [FS99, QWG+11, PHZ12, PHZ+11]. I seek to build on these results and expand PowerPack to perform my own energy profiling and analysis.

2.4 Solution

The solution presented herein is twofold:

1. Profile the energy consumption of cross-platform applications under different translation systems in order to gauge what tradeoffs are being made for portability.
2. Combine energy consumption measurements with existing tradeoffs in aesthetics and performance in order to yield a novel approach to application translation.

This solution will firstly attempt to address the energy consumption concerns that remain largely unstudied within the mobile translation field. Developers obviously need their translated apps to perform correctly, but to do so efficiently means leveraging the underlying hardware correctly. By profiling apps built with PhoneGap, with other translation systems, and with native code and then comparing the results, this work will address how well translated apps are able to perform versus their native counterparts.

Secondly, this solution shall provide a new approach to translation that seeks to maximize each tradeoff previously discussed (performance, aesthetics, and energy efficiency) along with others currently explored in the art. The goal here is to create one system that can effectively leverage the benefits of direct translation, intermediate language creation, and ecosystem emulation together while eliminating their pitfalls. One example could be an expansion to an existing system, such as PhoneGap or Cloud Twin, that replaces calls to energy inefficient code with more efficient equivalents while maintaining the existing translation approach.

2.5 Evaluation

In short, this solution will be delivered in parts through several conference or journal papers. I aim to present at least three papers to top conferences in the software engineering and systems spaces. The first will be submitted within the next year and roughly cover energy consumption on translated apps, leveraging past work on the PowerPack system. The second will be an intermediate paper showing preliminary results with a new translation system based on the energy consumption work. It should be submitted within the next two and a half years. The third paper will combine the intermediate results along with performance, aesthetic, and other tradeoffs to form an overall system. It should be submitted within the next four years. Other papers may come out of this research if more in depth results can be uncovered at intermediate stages. However I predict at least these three will highlight the majority of the work. More information about the timeline for this project can be found in the timeline section later 7.

The first deliverable for this proposal will be the paper on energy consumption on translated apps. It should cover the expansion of PowerPack (or perhaps an entirely new system) such that it can effectively profile apps running on mobile devices. The second deliverable shall be the new translation system in basic form. This may come in the form of an expansion to PhoneGap or other current systems such that the energy consumption is improved by at least **10%**. The third and final deliverable will be the finished translation system that incorporates aesthetic and performance improvements along with energy consumption. This final system should demonstrate at least **20%** increase in energy efficiency as well as notable improvements in aesthetics and latency. However, energy efficiency is the primary concern here, so specific measures of improvement elsewhere are not presented.

2.6 Impact

The biggest impact of this work will be that applications can be automatically translated without placing extraordinary burdens (more so than hand written applications) on the translated platform. This means that translated applications will perform comparably well to their hand written or manually translated counterparts on the same system. The translator will provide the largest impact, but the work on energy consumption analysis should also help pave the way for future research in that space. This can in turn benefit other translation platforms which can adapt and take this work into account.

At the top, most obvious level, cross platform application developers can benefit from this work, because they need their applications to reach a wide audience quickly (multiple platforms) and outperform their competition (maintain native aesthetics, run comparably quick, and utilize power at comparable efficiency) in order to make the most revenue. Even developers that haven't previously created apps on multiple platforms can benefit from this work if the development system created is easy enough to trivialize the process for even a novice.

More importantly however, all mobile users can benefit from this work, because it enables more users to utilize their favorite applications across a variety of platforms with comparable overall performance. No longer will that one application people like on iOS be restricted to Android or Windows Phone users (or vice versa), or suffer exorbitant issues of lag or battery consumption.

2.7 Outline

The remainder of this proposal is structured as follows: Related Work covers foundational and orthogonal work from past and present 3; Research Questions and Goals outlines what this study aims to accomplish 4; Research Methodology dives deep into how this study will achieve its goals 5; Evaluation Methodology describes how this work shall be evaluated for completion and meeting its goals 6; Timeline outlines what intermediate deadlines I will use for signaling progress 7; and finally, Research Qualifications briefly explains why I am qualified to undertake this study and successfully complete it 8.

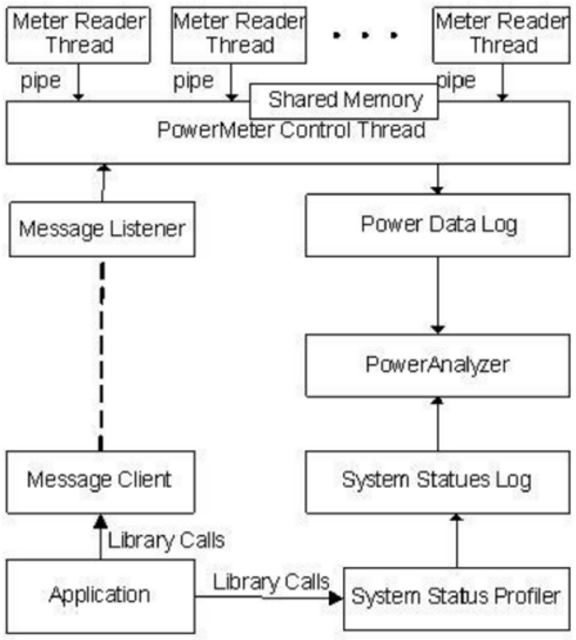

Figure 1: Software architecture design of PowerPack from [GFS+10].

3 Related Work

In the next subsections I discuss related research to this project. First, a summary of past studies in both energy profiling and app development is given in 3.1. Next, I mention some ongoing work that this proposal shall leverage in 3.2. Finally, related but orthogonal work is presented to show other interesting studies that aim for similar goals but leverage differing approaches in 3.3.

3.1 Past Studies

3.1.1 Profiling

Previous work in Dr. Cameron's SCAPE lab pioneered the PowerPack approach [GFS+10] for energy profiling, although not for mobile device profiling specifically. PowerPack examines the amount of energy flowing through the power lines of the system (cables from the power supply on desktop and battery rails on mobile devices) during interaction by externally tracking software calls throughout [1]. By associating the difference in energy consumption from one unit of time to another with the corresponding software calls that executed in that span, the authors are able to show exactly which calls trigger increases or decreases in power consumption and which components are consuming it. Figure 1 shows a design diagram for PowerPack from [GFS+10]. This approach and its extensions [SGFC09, LCS+14] form the basis for this proposal.

Other work by Dr. Cameron's group showed some basic profiling of Android devices already and provided a hierarchy of suggestions for future tooling to minimize energy consumption [CAC11].

[1] More information about iterations of PowerPack can be found at the SCAPE lab page: `http://scape.cs.vt.edu/software/powerpack-3-0/`

This work was a basic proof of concept in this lab for carry out profiling on an Android device with significant results. Based on their information, further software tuning can be performed at the application or operating system level to improve scheduling and thus increase idle time.

Several outside works have performed energy profiling on mobile devices in some sense already. Both Pathak et al. works [PHZ+11, PHZ12] demonstrate the *eprof* system of energy profiling. In utilizing *eprof* along with their energy bundles mechanism, the authors are able to refactor the source code of well-known target applications to save up to 65% in energy consumption. Both of these works were performed under Dr. Y. Charlie Hu at Purdue University.

Beyond the SCAPE lab and Dr. Hu's group, the Qian et al. paper (under Dr. Oliver Spatscheck) [QWG+11] demonstrates another means of profiling energy consumption, this time based on layers of interaction rather than calls down to hardware layers. The authors consider hardware components, low-level systems calls, and high-level source code calls in profiling where energy is consumed in a given app. The authors profiled several well known applications, such as Pandora, Fox News, and BBC news among others, and were able to identify problem areas then infer a reasoning for what causes those areas to exhibit such high energy consumption.

3.1.2 Cross-Platform Development

By mining the API's of various platforms, direct translation and cross compilation efforts have shown promise in porting apps between platforms [PA13, ZTX+10]. Yet, each of these approaches cannot fully translate an app between all major platforms. Some degree of manual rewrite is still required. However, these approaches are able to retain native aesthetics on each platform (for example, normal Android buttons are translated to normal iOS buttons and vice-versa, rather than looking like those belonging to another platform).

PhoneGap has seen great success in the area of app translation and is of special focus in this work [AGL10, GP12]. It allows developers to create an web app in HTML and Javascript then simply upload the related source files to the PhoneGap website for compilation on a number of platforms. PhoneGap thus keeps up with the required API's of many platforms, including Android, iOS, Windows Phone, Blackberry, Firefox OS, and Ubuntu, so developers only have to know how to develop for web apps [Pho14].

Past work on translating user interfaces (UI's) alone has yielded interesting work [ST11, HSDT13, Hol13] on emulation. By only translating the relevant UI, the amount of API coverage shrinks dramatically which means that translation approaches become feasible. However, in order to properly emulate execution, one must run the original version of an app on an emulator and connect to it from the new translated app. This incurs network overhead and increases overall latency. Cloud Twin has shown that it can perform this without overall latency exceeding the one second barrier necessary for responsive UI's [Mil68], however even just one second may still be too long.

3.1.3 Context Sensing

Additionally, there has been past work on mobile context sensing that aimed to reduce energy consumption of hardware sensors. Researchers primarily achieved this by applying intelligent algorithms and layers of abstractions to infer some user context rather than directly determining it through a plurality of sensors [NCG14, Nat12]. These approaches primarily work by defining a hierarchy of checks and keeping track of previous exclusive states to limit the amount of sensor data needed to make a determination. For example, when attempting to determine whether the

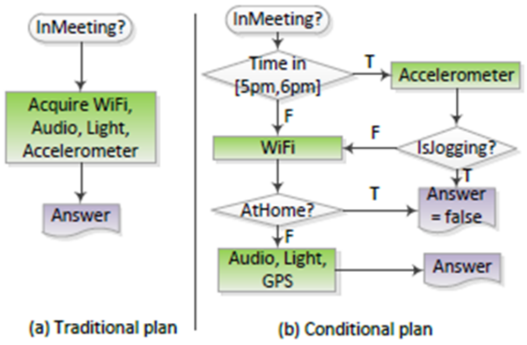

Figure 2: Context sensing plan from [Nat12].

user is in a meeting, instead of checking and combining the microphone audio, the Wi-Fi signal, the accelerometer movements, and the gps location, these systems would first check if the user is in any state that excludes him or her from being in a meeting (such as jogging or not being at work). Determining this other state may require some sensor data itself, but if it only requires checking the accelerometer to say that the user is definitely jogging (and thus definitely not in a meeting), then it saves energy overall by not utilizing all the other sensors. Figure 2 shows an example diagram of this methodology from [Nat12].

3.2 Current Work

As stated previously, a collaborative work with Dr. Cameron's SCAPE lab is already underway to profile energy consumption on the Asus TF300T tablet [Asu14] running Android "Jelly Bean" [And14]. Initial results from that research are shown in figures 3, 4, and 5 where PowerPack was able to profile energy consumption of two power rails while running the AnTuTu benchmark on the device via both battery and outlet power. This result as well as energy measures from popular apps across development platforms shows promise for this research.

3.3 Related Orthogonal Work

Tangential context sensing works by Dr. Hu's group have leveraged energy profiling approaches in order to detect energy-based bugs in smartphones, such as the "no sleep" bug that can keep a device's hardware components awake indefinitely thus draining energy [PJHM12, JPHM13]. Other research [VJLA12] has also attacked the "no sleep" bug by profiling and examining the Android WakeLock API. Vekris et al. was able to verify 145 out of 328 apps did not have the bug, while the remainder were either confirmed to have it or present a warning.

Previous research in my own group has looked into cloud offloading as a means of simultaneously increasing performance and energy efficiency while retaining overall aesthetic appeal [TK14, Kwo13]. The concept of offloading some form of mobile execution to servers has seen great success, especially in reducing energy in papers such as [CIM$^+$11]. These works were not originally intended to be used with app translation, but the basic concept was expanded upon even in Cloud Twin [HSDT13, Hol13] and serves as an avenue for further research.

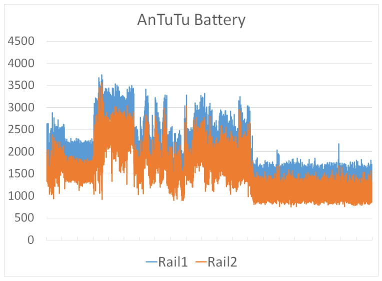

Figure 3: Millivolts for both power rails during AnTuTu under battery power.

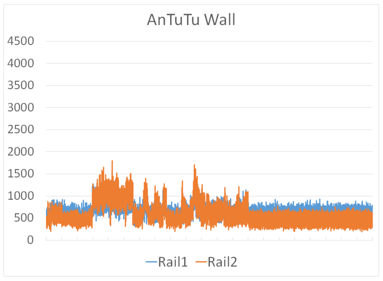

Figure 4: Millivolts for both power rails during AnTuTu under outlet power.

Test	Battery	Outlet
Overall	15983	15909
Multitask	2822	2530
Runtime	996	993
RAM Operation	1524	1527
RAM Speed	628	650
4x CPU Integer	1385	1384
4x CPU Float	1554	1546
1x CPU Integer	883	883
1x CPU Float	976	983
2D Graphics	682	716
3D Graphics	3716	3819
Storage IO	407	443
Database IO	410	435

Figure 5: AnTuTu score for both rails during AnTuTu under battery and wall power.

4 Research Questions and Goals

The goal of this research project is simply **to improve the landscape of cross-platform development approaches by considering the impact of energy efficiency on app performance**. As stated above, the solution proposed here is thus to perform the following:

1. Profile the energy consumption of cross-platform applications under different translation systems in order to gauge what tradeoffs are being made for portability.
2. Combine energy consumption measurements with existing tradeoffs in aesthetics and performance in order to yield a novel approach to application translation.

Based on the solution, the following key research questions must be answered in the proposed research:

1. What method of development is currently the most energy efficient?
2. What low- and high-energy calls exist on which platforms?
3. Which high-energy calls can be effectively replaced by more efficient versions in existing development approaches?
4. Can more energy efficiency be generated without a subsequent loss in performance or aesthetic appeal?
5. Can existing development approaches be reimplemented with this consideration in mind?

These questions will not be answered in this proposal, but instead they will serve as guiding questions for this research. In publishing the results of this research, these questions will hopefully appear again to be answered. The goal stated above will pervade this proposal as well as subsequent research publications.

5 Research Methodology

The following research methodology outlines exactly how each research question above shall be answered. The initial study 5.1 will yield more information about the translation approaches being studied and how they are implemented. Energy profiling 5.2 will then systematically examine the energy profiles of various apps and translation approaches. These two in combination will outline how to answer the first three research questions. Finally, system implementation 5.3 will unveil how to answer the remaining two research questions.

5.1 Initial Study

The first step in this research will be to perform a thorough survey of the current landscape of cross-platform development approaches. This proposal presents a great deal of work in the state of the art, but there is even more out there than can be covered here. An in-depth investigation of all the approaches presented here as well as those presented at recent top conferences will yield more information about exactly how each is currently implemented. This survey is necessary in order to determine what calls each platform uses to implement various features. One would expect direct translation to yield similar calls as those developed natively. However, those based on a persistent internet connection to communicate with an emulator [HSDT13, Hol13] or those rendered in a web page rather than using the default structure [GP12, AGL10] will undoubtedly present different calls to access various hardware components.

The next step for investigating these approaches will be to implement each on a variety of apps. The best comparison between apps would be those developed with equivalent functionality on each platform. Most apps developed in the wild will not meet this qualification since developers do not have the time to separately develop the same app multiple times. Thus, to start the best comparison may be between similarly functioning "classes" of apps developed on separate platforms, such as games, online streaming services, and social media platforms. Over time, it may be possible to implement some test apps on separate platforms in order to generate a fairer comparison.

Because of its popularity, especial focus will be placed on examining PhoneGap's approach to cross-platform development [GP12, AGL10]. Running apps built for PhoneGap means relying heavily on the Javascript engine for each phone which should demonstrate significantly different execution from that of native apps. Improving this case will likely yield the most novel and interesting results of all the available development approaches.

5.2 Energy Profiling

After analyzing these apps to understand what exactly each does during execution, the next step is to determine how each performs in terms of energy efficiency. By profiling apps from the aforementioned "classes" down to a fine-grained, component-by-component level with PowerPack, I will easily be able to see how much power each individual hardware component draws. For instance, one would expect that games would exhibit high energy consumption from the screen, whereas streaming services would have nearly as much energy consumption from the screen but comparatively more energy consumption by the network radio (either Wi-Fi, 3G, or 4G).

Combining the results of this profiling with the initial knowledge gained from surveying the development systems, I can then answer some key research questions about the efficiency of each development system as well as what low- and high-energy calls are utilized. By comparing the development systems, it will become clear which approaches are able to utilize lower energy calls and which could be modified. Furthermore, by utilizing some existing work on context sensing on mobile devices, I can even implement these improved sensing algorithms to tilt the energy to performance tradeoff in favor of more energy efficiency [NCG14, Nat12].

5.3 System Implementation

The last step of this research is to implement an improved cross-platform development system by utilizing everything I've learned from the work above. To implement this, the first step will be to improve an existing system before moving on to creating an entirely new one. The most important feature of any improved system will have to be the energy efficiency 5.3.1, however performance 5.3.2 and aesthetics 5.3.3 should not drastically suffer in spite of this.

5.3.1 Energy Efficiency Calibration

To first improve an existing system, say PhoneGap for instance, I must remove the high-energy hardware component calls generated by the development platform and replace them with low-energy variants of them. This could mean something as simple as reducing the amount of time a given component is kept awake while gathering data (such as reducing the network tail time), but it could also be as complex as defining a specific hierarchy or algorithm for when to actually turn on GPS sensors rather than formulate location information from less expensive context. At this stage, I aim to improve such a system's energy efficiency by at least **10%** in the average case if not more. This modest result should be enough to show progress in development efforts while not setting the bar extremely high. Later improvements should achieve even more benefit, as much as **20%** overall in the average case. In this situation average case would be determined from the typical runtime of a plurality of apps. More information about this can be found in the Evaluation Methodology 6.

5.3.2 Performance Calibration

Beyond simply improving the energy efficiency of a development platform, I also need to retain the overall performance of the apps running on it. That is to say that development of apps themselves should be relatively easy compared to the existing approach, but also that these apps should not suffer exorbitant latencies during runtime. One negative example of this could be that an app incurs too much latency to be responsive after reducing the network tail time. This could easily happen with games that are heavily dependent upon network communication. Thus, special care must be taken such that any improvements in energy efficiency are not outweighed by a subsequent reduction in performance. Careful tuning of changes to the platform will be necessary to ensure this. I suspect that the improved development platform will undergo many iterations to perfect this and will apply some logical abstraction to decide at runtime which is best.

5.3.3 Aesthetics

In addition to retaining performance, the aesthetic of apps developed for multiple platforms should at least be retained if not improved. Users typically prefer the native look and feel of apps on a given platform over generic cross-platform elements, so if at all possible, I will implement an improvement in this direction [MC12]. One means of performing this on PhoneGap for instance would be to dynamically adjust how an app is rendered at runtime so that it is rendered with native UI instead of HTML. To do this would require parsing the HTML UI tree at runtime and translating those components to their native version, similar to the work in Cloud Twin [HSDT13, Hol13]. While this endeavor could constitute a research proposal in and of itself, an attempt in this direction is made here so as to improve the overall development platform if PhoneGap is to be pursued further.

At the extreme case, I must retain the original aesthetic of whatever platform is modified or strive for native aesthetics when creating an entirely new one. While some past research has shown that altering the UI to reduce energy is in fact possible [LTH14], I will not explore this endeavor. The primary concern here is to improve energy efficiency, but not at the expense of performance or aesthetics. If the UI can be improved along the way, then further research may be included, but no tradeoff to explicitly reduce the user experience will be made.

6 Evaluation Methodology

To evaluate the research proposed here, I will employ a variety of metrics. First, I will examine how many apps can be profiled using PowerPack. For the initial research paper deliverable solely about profiling apps, there must be at least 3 different development approaches considered. These will likely include purely native development and PhoneGap development along with another approach. From each approach, at least 10 different apps will be profiled unless more are required to show some statistical significance. These apps will form several "classes" as mentioned before. Likely choices for these classes lie in different principle use cases, such as mobile gaming, media streaming, social networking, location determination, and light generation. One could easily foresee comparing a native media streaming service, such as Pandora [Pan14], to one developed using cross-platform approaches, such as Phonostar [pG14] on PhoneGap.

The next metric is the improvement of energy efficiency in the average case. The average case will here mean apps running during normal use. A user study will likely be employed here to ensure a broad range of use cases are covered without bias. By profiling the energy consumption during these use cases, I shall then show that the improved apps are at least **10%** more energy efficient during these average cases than non-improved counterparts which were previously measured. Similarly, for the final system deliverable, another user study will perform the same experiment, but show that the final system demonstrates at least **20%** improvement in energy efficiency.

To determine whether performance has been impacted, some simple benchmarks will be employed to measure the latency of interaction with each app. This interaction latency should encompass not only changes to the underlying development system but also changes to how context is obtained (if any changes are in fact made). Ideally this latency measurement should remain in the same range as previous systems. Any more than **10%** increase in latency will be considered unacceptable unless further changes are made to improve performance otherwise.

Similarly, aesthetic changes should be kept to a minimum. A user study comparing the look and feel of the more energy efficient apps versus their unimproved counterparts should reveal some sense of preferences. By correlating this with past data about what aesthetics are preferred from existing systems, there should be little change. No specific number is attached to this measurement since obviously user opinions can vary widely over time as to which app looks or feels better. However, if this user study shows statistically anomalous results compared to those in the past, further research may be necessary to examine why that is and whether it should or should not hinder this energy efficiency improvement.

7 Timeline

The overall timeline for completing this research is over the next four years. By the end of that time period, an improved system for cross-platform development should be fully implemented (based on the requirements outlined above) and several papers should be presented. Intermediate deadlines will focus on steps towards completing that final deliverable.

7.1 Initial Study

The initial study will consume roughly 6 to 12 months work within this group (possibly including other graduate students). It will encompass performing an in-depth survey of development approaches as well as getting PowerPack up and running on several real world mobile systems for performing the energy profiling. The uncertainty in this time frame lines with instrumenting PowerPack. Simply implementing it has not proved overly difficult, but there may need to be changes to the methodology on mobile devices to capture enough component energy data to formulate a reliable power profile.

7.2 Implementation

The implementation of the improved cross-platform development system will consume the remaining 36 to 42 months. I anticipate spending the first year simply improving on an existing system, such as PhoneGap, based on the insights gained from the initial study. After that year, I will begin developing a new system while continuing improvement of the existing one. This will allow time to test if a better framework can actually be created from scratch or if it is instead more beneficial to build on the existing approach. After roughly 6 months of development on both systems, it should be clear where the most promise lies. Further improvements will thus be made on whichever system seems most beneficial at the time. This should span the remainder of the research or until more funding is procured.

7.3 Paper Deadlines

As mentioned previously, it is my hope that several research papers will come out of this work. The first will be submitted within the next year and roughly cover energy consumption on translated apps, leveraging past work on the PowerPack system. The second will be an intermediate paper showing preliminary results with a new translation system based on the energy consumption work. It should be submitted within the next two and a half years. The third paper will combine the intermediate results along with performance, aesthetic, and other tradeoffs to form an overall system. It should be submitted within the next four years. Other papers may come out of this research if more in depth results can be uncovered at intermediate stages. However, I predict at least these three will highlight the majority of the work.

8 Research Qualifications

As a student at Virginia Tech, I have worked with Dr. Tilevich's Software Innovations laboratory for 3 years now as well as with Dr. Cameron's SCAPE laboratory for the past year. While working with Dr. Tilevich, we have had two previous works on cross-platform development and translation accepted to top conferences in software engineering [HSDT13, Hol13]. Dr. Tilevich will additionally be overseeing this work with me to ensure its successful completion. He has previously completed several NSF sponsored projects with numerous publications in top conferences such as SPLASH/OOPSLA, ICSE, ECOOP, and ICSM. Dr. Cameron will be collaborating on this project as well, given his interest in the expansion of PowerPack. Dr. Cameron and I have previously collaborated on several projects currently in submission to top systems conferences including SOSP and OSDI. Both Dr. Tilevich and Dr. Cameron are associate professors of computer science at Virginia Tech, having over 200 publications and over 2000 citations between them according to Google Scholar [GS14]. While this is the first proposal on which I have lead authorship, each of my collaborators' track record on these projects is above reproach and goes to show what the NSF can expect from this group.

References

[AGL10] Sarah Allen, Vidal Graupera, and Lee Lundrigan. Phonegap. In *Pro Smartphone Cross-Platform Development*, pages 131–152. Apress, 2010.

[And14] Android. Jelly bean. `http://developer.android.com/about/versions/jelly-bean.html`, 12 2014. Accessed: 2014-12-10.

[App14] Apple. itunes preview app store. `https://itunes.apple.com/us/genre/ios/id36?mt=8`, 12 2014. Accessed: 2014-12-10.

[Asu14] Asus. Asus transformer pad (tf300t). `http://www.asus.com/us/Tablets_Mobile/ASUS_Transformer_Pad_TF300T/`, 12 2014. Accessed: 2014-12-10.

[Bra14] App Brain. Number of android applications. `http://www.appbrain.com/stats/number-of-android-apps`, 11 2014. Accessed: 2014-11-18.

[CAC11] Hung-Ching Chang, A.R. Agrawal, and K.W. Cameron. Energy-aware computing for android platforms. In *Energy Aware Computing (ICEAC), 2011 International Conference on*, pages 1–4, Nov 2011.

[CIM$^+$11] Byung-Gon Chun, Sunghwan Ihm, Petros Maniatis, Mayur Naik, and Ashwin Patti. Clonecloud: Elastic execution between mobile device and cloud. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 301–314, New York, NY, USA, 2011. ACM.

[CL11] Andre Charland and Brian Leroux. Mobile application development: Web vs. native. *Commun. ACM*, 54(5):49–53, May 2011.

[FS99] Jason Flinn and M. Satyanarayanan. Powerscope: A tool for profiling the energy usage of mobile applications. In *Proceedings of the Second IEEE Workshop on Mobile*

Computer Systems and Applications, WMCSA '99, pages 2–, Washington, DC, USA, 1999. IEEE Computer Society.

[GFS+10] Rong Ge, Xizhou Feng, Shuaiwen Song, Hung-Ching Chang, Dong Li, and K.W. Cameron. Powerpack: Energy profiling and analysis of high-performance systems and applications. *Parallel and Distributed Systems, IEEE Transactions on*, 21(5):658–671, May 2010.

[Goo14] Google. Google play store. `https://play.google.com/store?hl=en`, 12 2014. Accessed: 2014-12-10.

[GP12] Rohit Ghatol and Yogesh Patel. *Beginning PhoneGap: Mobile Web Framework for JavaScript and HTML5.* Apress, Berkely, CA, USA, 1st edition, 2012.

[GS14] Google scholar. `http://scholar.google.com/`, December 2014.

[HHM12] Henning Heitkötter, Sebastian Hanschke, and Tim A. Majchrzak. Comparing cross-platform development approaches for mobile applications. In *WEBIST*, pages 299–311, 2012.

[Hol13] Ethan Holder. Cloud twin: Interactive cross-platform replay for mobile applications. In *Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, &#38; Applications: Software for Humanity*, SPLASH '13, pages 119–120, New York, NY, USA, 2013. ACM.

[HSDT13] Ethan Holder, Eeshan Shah, Mohammed Davoodi, and Eli Tilevich. Cloud twin: Native execution of android applications on the windows phone. In *ASE*, pages 598–603. IEEE, 2013.

[JPHM13] Abhilash Jindal, Abhinav Pathak, Y. Charlie Hu, and Samuel Midkiff. On death, taxes, and sleep disorder bugs in smartphones. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, HotPower '13, pages 1:1–1:5, New York, NY, USA, 2013. ACM.

[Kwo13] Young-Woo Kwon. Orchestrating mobile application execution for performance and energy efficiency. In *Proceedings of the 2013 Companion Publication for Conference on Systems, Programming, &#38; Applications: Software for Humanity*, SPLASH '13, pages 125–126, New York, NY, USA, 2013. ACM.

[LCS+14] Bo Li, Hung-Ching Chang, Shuaiwen Song, Chun-Yi Su, Timmy Meyer, John Mooring, and Kirk W. Cameron. The power-performance tradeoffs of the intel xeon phi on hpc applications. In *Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops*, IPDPSW '14, pages 1448–1456, Washington, DC, USA, 2014. IEEE Computer Society.

[LTH14] Ding Li, Angelica Huyen Tran, and William G. J. Halfond. Making web applications more energy efficient for oled smartphones. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 527–538, New York, NY, USA, 2014. ACM.

[MC12] Tom Melamed and Ben Clayton. A comparative evaluation of html5 as a pervasive media platform. In *Proceedings of the First International ICST Conference, MobiCASE*, 10 2012.

[Mil68] Robert B. Miller. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I*, AFIPS '68 (Fall, part I), pages 267–277, New York, NY, USA, 1968. ACM.

[Nat12] Suman Nath. Ace: Exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[NCG14] Nima Nikzad, Octav Chipara, and William G. Griswold. Ape: An annotation language and middleware for energy-efficient mobile application development. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 515–526, New York, NY, USA, 2014. ACM.

[PA13] Arno Puder and Oren Antebi. Cross-compiling android applications to ios and windows phone 7. *Mobile Networks and Applications*, 18(1):3–21, 2013.

[Pan14] Pandora. Pandora internet radio. `https://play.google.com/store/apps/details?id=com.pandora.android&hl=en`, 11 2014. Accessed: 2014-12-10.

[Per14] Sarah Perez. itunes app store now has 1.2 million apps, has seen 75 billion downloads to date. `http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/`, 06 2014. Accessed: 2014-11-18.

[pG14] phonostar GmbH. phonostar radio-app. `https://play.google.com/store/apps/details?id=de.phonostar.player`, 10 2014. Accessed: 2014-12-10.

[Pho14] PhoneGap. Platform support. `http://docs.phonegap.com/en/4.0.0/guide_support_index.md.html#Platform%20Support`, 12 2014. Accessed: 2014-12-10.

[PHZ$^+$11] Abhinav Pathak, Y. Charlie Hu, Ming Zhang, Paramvir Bahl, and Yi-Min Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the Sixth Conference on Computer Systems*, EuroSys '11, pages 153–168, New York, NY, USA, 2011. ACM.

[PHZ12] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[PJHM12] Abhinav Pathak, Abhilash Jindal, Y. Charlie Hu, and Samuel P. Midkiff. What is keeping my phone awake?: Characterizing and detecting no-sleep energy bugs in smartphone apps. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 267–280, New York, NY, USA, 2012. ACM.

[QWG+11] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling resource usage for mobile applications: A cross-layer approach. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 321–334, New York, NY, USA, 2011. ACM.

[Riv14] Janessa Rivera. Gartner says annual smartphone sales surpassed sales of feature phones for the first time in 2013. `http://www.gartner.com/newsroom/id/2665715`, 02 2014. Accessed: 2014-05-09.

[SGFC09] Shuaiwen Song, Rong Ge, Xizhou Feng, and Kirk W. Cameron. Energy profiling and analysis of the hpc challenge benchmarks. *Int. J. High Perform. Comput. Appl.*, 23(3):265–276, August 2009.

[ST11] Eeshan Shah and Eli Tilevich. Reverse-engineering user interfaces to facilitate porting to and across mobile devices and platforms. In *Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE!'11, AOOPES'11, NEAT'11, &#38; VMIL'11*, SPLASH '11 Workshops, pages 255–260, New York, NY, USA, 2011. ACM.

[TK14] Eli Tilevich and Young-Woo Kwon. Cloud-based execution to improve mobile application energy efficiency. *Computer*, 47(1):75–77, January 2014.

[VJLA12] Panagiotis Vekris, Ranjit Jhala, Sorin Lerner, and Yuvraj Agarwal. Towards verifying android apps for the absence of no-sleep energy bugs. In *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems*, Berkeley, CA, 2012. USENIX.

[ZTX+10] Hao Zhong, Suresh Thummalapenta, Tao Xie, Lu Zhang, and Qing Wang. Mining api mapping for language migration. In *Proceedings of the 32Nd ACM/IEEE International Conference on Software Engineering - Volume 1*, ICSE '10, pages 195–204, New York, NY, USA, 2010. ACM.