Characterizing Energy-Aware Software Projects: Are They Different?

Shaiful Alam Chowdhury
Department of Computing Science
University of Alberta
Edmonton, Canada
shaiful@ualberta.ca

Abram Hindle
Department of Computing Science
University of Alberta
Edmonton, Canada
abram.hindle@ualberta.ca

ABSTRACT

The improvement in battery technology for battery-driven devices is insignificant compared to their computing ability. In spite of the overwhelming advances in processing ability, adoption of sophisticated applications is hindered by the fear of shorter battery life. This is one of the several reasons software developers are becoming conscious of writing energy efficient code. Research has been conducted to model software energy consumption, to reduce energy drains, and to understand developers expertise on energy efficiency. In this paper, however, we investigate the nature of energy-aware software projects. We observed that projects concerned with energy issues are larger and more popular than the projects that do not address energy consumption. Energy related code changes are larger than others (e.g., bug fixes). In addition, our initial results suggest that energy efficiency is mostly addressed on certain platforms and applications.

1. INTRODUCTION

"The world has changed." Nowadays people move and play with pocket computers (i.e., smartphones and tablets), with more power than the traditional desktop computers of yore. Smartphones in particular, built primarily for voice communication, are now adopted for daily life use cases: people use their smart phones as calculators, as to-do lists, as game consoles, and of course for accessing the Internet. This is because of the overwhelming improvements in smartphones computing capability.

Ironically, despite such improvements, users reported longer battery life as the most desired smartphone feature [6]: the insignificant improvement in battery technology compared with their computing counterparts causes reduced battery life. Energy consumption, in addition to the mobile devices, has also become a subject of concern for large data centers, and was reported as one of the pivotal issues Google faced when scaling their operations [1]. Last but not least, the devastating effect of energy consumption is climate change, as most of the electricity is produced by burning fossil fuels [1].

Although the hardware components are responsible for draining energy, efficient software systems can play a vital role on how the components can be accessed and used to minimize the energy consumption. Subsequently, energy awareness is now observed among software developers [6], and the awareness has increased linearly over the last few years [7].

Previous research has analyzed energy related code and questions from open source software projects and programming Q&A forums [6, 7, 5]. The main objectives of these research were to investigate what programmers do for software energy optimization, and how often programmers are successful with their attempts. Voltage scaling, reducing number of messages and update periods, imposing idle periods, and selecting efficient data structures have been found as the most frequently applied energy optimization techniques. In spite of their urgency, however, developers are not confident that their code changes will really reduce energy consumption [6].

In this paper, we analyze commit messages similar to the previous work [6, 3]. However, we investigate a different research issue: are energy-aware software projects different in nature than others? In general, we answer the following four different questions that jointly address the big question.

RQ1. Is energy efficiency considered only by large and popular software projects? To answer this, we collected different statistics of the selected projects: number of contributors, number of commits, number of forks etc.

RQ2. What are the most frequently used programming languages in energy-aware projects? This also indirectly answers the types of projects that address energy efficiency the most.

RQ3. Which developers address energy efficiency the most? Are there few energy experts? And are the energy related changes in source code longer than other changes (e.g., code change related to bug fix)?

RQ4. When addressing energy efficiency, do the developers express differing sentiment than other issues like bug fix?

Our results, for **RQ1-RQ3**, suggest that energy-aware software projects are significantly different than projects that do not consider energy efficiency. No noticeable difference, however, was observed in sentiment analysis. For that matter, we found that traditional corpora of sentiwords are not suitable for understanding the sentiments of programmers' commit messages—although such a corpus was previously used for GitHub sentiment analysis [3].

MSR'16, May 14-15, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2901739.2903494

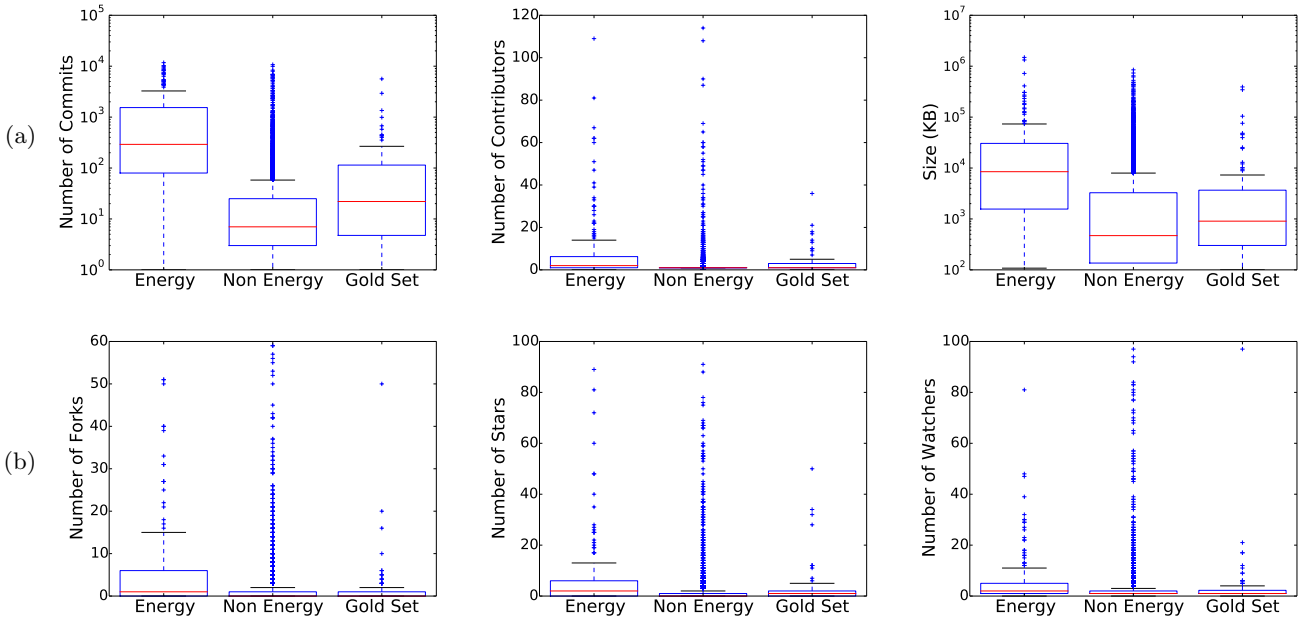

Figure 1: Energy projects VS Non-energy projects: larger (Figure (a)) and more popular (Figure (b)) projects are more likely to address energy issues.

2. DATA COLLECTION

Boa [2]—a framework specifically designed to analyze large software repositories—was used for our data collection. We collected all the available commit messages with their project IDs from GitHub as provided by Boa. This way we were able to collect 282,779 GitHub projects. In order to separate out energy related commit messages, we used keywords used in previous studies [6, 7, 5]. We also included missing keywords like "*energy drain.**", "*energy leak.**", "*tail energy.**" etc. Initially, we ended up with 522 energy commits from GitHub. A project with at least one energy commit was labeled as an energy project in contrast to the non-energy projects where no energy related commit message was found. The problem with GitHub energy projects was that we observed lots of exactly similar energy commits across different projects—indicating that many of the energy projects were forked or cloned. Unfortunately, using the GitHub API we found only three forked energy projects in our list; we found many cases where a project had been cloned and then uploaded to GitHub without using GitHub's *fork* directly. As a result, we wrote a script that reports projects with very similar commit messages, and very similar group of contributors. Duplicated projects were then deleted from our dataset after manual inspection. Finally, the dataset consists of 223 unique GitHub energy projects with 400 energy commits. Similarly, 65,935 projects were collected from SourceForge. We found only 182 energy commits from all of our SourceForge projects. These commit messages were included with the GitHub ones to answer the **RQ4**.

In addition to Boa, the GitHub API was used to collect information like number of forks, number of watchers of all the GitHub energy and selected non-energy projects. In order to answer how different energy related projects are than others, we need information about non-energy projects as well. We selected 5000 projects randomly from our labeled non-energy projects; collecting information for all the available GitHub projects is problematic due to the rate limit imposed by the GitHub API. Previous research [4], however, has found that a GitHub repository is not necessarily a GitHub project. In fact, from our quick inspection we found lots of student projects, and tutorials (e.g., how to implement heap sort in Java) from our 5000 selected non-energy projects. This might lead to an unfair comparison—all of the energy projects in our dataset were true software projects, as confirmed by manual inspection. As a result we created a goldset of 100 true non-energy software projects without any bias towards their statistics that we use in this paper. After randomly selecting a project from our dataset of 5000 projects, we manually inspected if the project is an actual software project [4]. If yes, we included the project in our goldset. After trying about 180 projects, we were able to create our goldset of 100 desired projects.

3. RESULT ANALYSIS

RQ1. Are energy-aware projects different in size and popularity? In order to evaluate how distinct the energy projects are compared to the others, we consider two different metrics: size and popularity. The total numbers of contributors, commits, and the actual size (in KB) are used as the indicators of a project's size. Likewise, the total numbers of forks, stars and watchers are used to measure popularity. Figure 1(a) and 1(b), without outliers, illustrate the difference in size and popularity respectively. The non-energy projects of our goldset, as expected, are more popular and larger than the non-filtered non-energy projects. However, we observe very distinct characteristics for the energy projects. In general, Figure 1 evidently suggests that large and popular software projects have higher probability to incorporate energy efficiency issues. A project with more contributors is more likely to have developers with energy

consumption expertise.

RQ2. Are energy-aware projects biased to a group of programming languages? We ranked all the programming languages used in our energy projects. Most of the projects, however, have more than one programming language. As a result, we calculated the percentage of projects that used a specific programming language. Likewise, the top 40 programming languages across all the GitHub projects as provided by Boa were collected. Figure 2, with the top 10 most frequent programming language in the collected energy-aware projects, clearly suggests that energy-aware software projects are biased to the programming languages that are most frequently used for hand-held and embedded device applications. Although Java is the 5^{th} most frequently used programming language across all the GitHub projects, it is the most frequently used language among the energy-aware software projects. In fact, 95% of the energy-aware software projects have Java as one of the used languages. This is intuitive, as Android is the most famous platform for smartphones and tablets,[1] and Java is the primary language used for Android applications. Similarly, Objective-C (for iOS platforms) is more frequently used in energy projects. C and C++ also have ascended ranks in energy projects compared to all other Boa provided GitHub projects. This might be because of their frequent use in embedded systems.[2] Interestingly, D—purportedly designed for efficiency and improved programmers' productivity[3]—also belongs to the list of top 20 languages used in our energy-aware projects, which is not even within the top 30 languages used in all the collected GitHub projects.

Lack of Web/server-side programming languages, however, is noticeable in Figure 2. For example, the rank of JavaScript is 5^{th}, whereas in general ranking this is the most frequently used programming language. PHP, the 6^{th} most frequently used programming language in all the collected GitHub projects, does not make a position in our list of the most frequently used languages in energy-aware projects. For the *Go* programming language, a language mostly used for server-side programming, we observed similar pattern; although *Go* belongs to the top 30 among all the GitHub projects as provided by Boa, it is not even within the top 40 languages of our energy-aware projects list. We conclude that developers are mostly concerned about energy efficiency in case of mobile and embedded systems. This is unfortunate to some extent, as server energy efficiency has been found to be crucial for CO_2 emissions [1].

RQ3. Who deals with energy issues in a project? Are there developers who only deal with energy issues? In order to answer this question, we calculated the percentile ranks of all the contributors in our energy projects. A 20^{th} percentile rank of contributor x means 80% of the authors in a project committed equal to or less number of commits than x. For all of our energy projects, we also collected bug fix contributors through filtering commit messages using "*bugfix*", "*bug fix*", and "*fixed bug*" keywords. Figure 3 illustrates the cumulative distribution function of all the contributors in different roles: all commits, energy commits, and bug fix commits. Results suggest that energy issues are addressed by the authors who in general

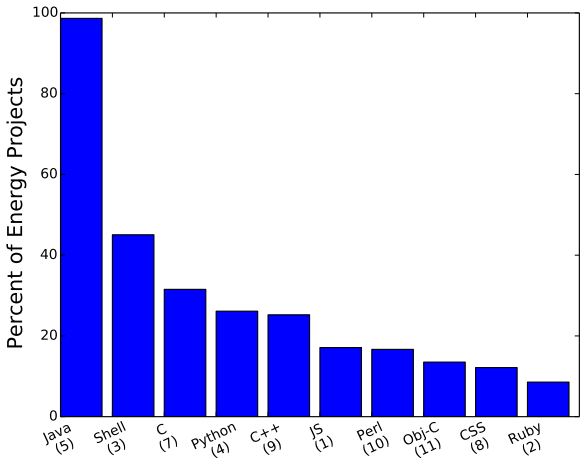

Figure 2: Distribution of programming languages: Java is the most frequently used language in energy-aware software projects.

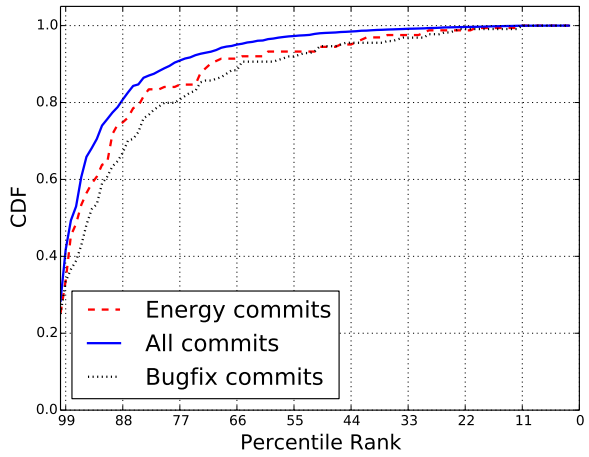

Figure 3: Cumulative distribution of energy commits among the contributors: most contributing authors deal more with energy issues compared to bug fix issues.

contributes the most in a project. In contrast, bug fix committers are more distributed. We conclude that most of the software projects do not have developers that only deal with energy issues. Developers who are more connected to a project—i.e., make more contributions than others—are more likely to care about and address energy issues.

Do energy related issues require more changes in source code? We captured the changes made in the source code (number of additions + number of deletions) for all of our collected energy and bug fix commits. Figure 4, excluding the outliers, suggests that energy related code changes are most of the time significantly larger than the bug fix related changes. This is unsurprising as energy is relatively a new concept in software development, and the developers are yet to master this art. In fact, we observed lots of energy commits where a significantly large previous code has been deleted as the objective was not achieve, leading the developers to try very different ideas. However, the number of energy commits with only a few line of code changes is

[1]http://techland.time.com/2013/04/16/ios-vs-android/
[2]http://www.eetimes.com/author.asp?doc_id=1323907
[3]http://dlang.org/

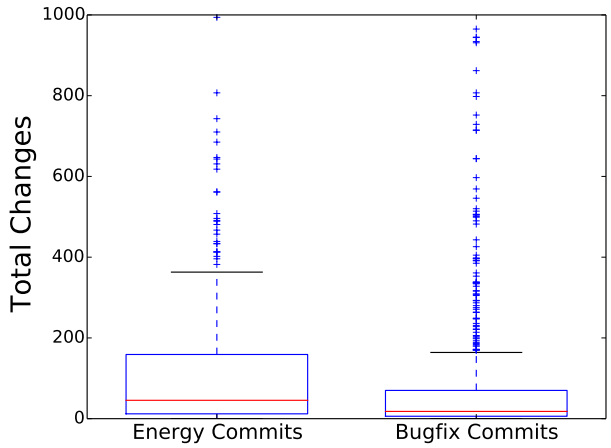

Figure 4: Distribution of number of changes in the source code: energy related changes are larger than bug fix changes.

noticeable; we found 73 energy commits with less than 10 lines of changes in the source code. To indulge our curiosity, we manually investigated some of these committed changes. Encouragingly, we observed that some of the developers are aware that a little tweaking can improve energy efficiency significantly. For example, one of the developers disabled automatic screen brightness, and was confident that this little change will reduce energy drain. A lot of commits dealt with changing a parameter value to optimize energy such as GPS-timeout, number of logs per event etc. Some others dealt with efficient release of CPU and Wi-Fi locks, requiring very little change in the source code.

These observations imply that achieving energy efficiency does not necessarily require sophisticated code changes, but rather demands knowledge on how different hardware components operate. In fact, we found developers' discussions on significant energy reduction through small changes in source code; tweaking on a CPU related conditional variable claimed to reduce energy consumption by 90% in an audio player project.[4]

RQ4. Are software developers more frustrated while dealing with energy efficiency? To answer this question, we did sentiment analysis on the energy related commit messages compared to bug fix and all other commit messages. The first impediment for such analysis is that most of the commit messages are very short. We adopted SentiStrength because of its specialization with short messages and for its inclusion of the most frequently used emoticons. Moreover, SentiStrength was used for analyzing sentiment of GitHub commit messages in an earlier study [3]. Initially, we observed that energy commit messages are more positive in contrast to the bug fix commit messages. Unfortunately, this was because of the selected keywords used in our commit filtering. Most of the energy commit messages have the word "*save*" whereas "*bug*" is the most frequently used word among the bug fix commits. SentiStrength has a +1 score for "*save*" and −1 for "*bug*"—thus leading to an unfair comparison. After deleting these two words from our dataset, no difference was observed. In fact, the sentiment score distributions for both energy and bug fix commit messages were found very similar to all other commit messages. After searching for "hesitating" words in the commit messages [6], however, we found 6% hit for energy commits and 3% hit for the bug fix commits—i.e., less confidence with energy issues than bug fix issues.

We conclude that a manually curated corpus is required to analyze developers sentiment. For example, SentiStrength can not differentiate between "*It fixed the bug*" and "*It did not fix the bug*". In both cases, it returns a negative score (−1) although they carry very different sentiment.

[4]http://stackoverflow.com/questions/29310100/why-does-a-conditional-variable-fix-our-power-consumption

4. THREATS TO VALIDITY

This work is based on dataset provided by Boa, which might bias to certain languages. The number of captured energy-aware projects is very small compared to the enormous size of GitHub, thus can produce skewed observations. This is a common problem in mining software energy issues [6, 7]. We made assumptions, such as types of projects from the used programming languages, without actual verification; this, however, demands manual inspection.

5. CONCLUSION

We have observed that energy-aware projects tend to be large, mature, and popular. The developers who create energy commits tend to be the top developers of the project and may have different rank distributions than those who contribute to bug fixes. Many of the energy commits belong to languages and projects that target mobile/embedded platforms. More code changes in energy commits compared to bug fix commits suggest that energy concerns are often cross-cutting that may affect many modules.

Acknowledgment

Shaiful Chowdhury is grateful to the Alberta Innovates - Technology Futures (AITF) to support his PhD research. Abram Hindle is supported by an NSERC Discovery Grant.

6. REFERENCES

[1] S. Chowdhury, S. Varun, and A. Hindle. Client-side Energy Efficiency of HTTP/2 for Web and Mobile App Developers. In *SANER '16 (to appear)*, Osaka, Japan, March 2016.

[2] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *ICSE 2013*, pages 422–431, May 2013.

[3] E. Guzman, D. Azocar, and Y. Li. Sentiment analysis of commit comments in GitHub: An empirical study. In *MSR 2014*, pages 352–355, 2014.

[4] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian. The promises and perils of mining GitHub. In *MSR 2014*, pages 92–101.

[5] H. Malik, P. Zhao, and M. Godfrey. Going green: An exploratory analysis of energy-related questions. In *MSR 2015*, pages 418–421, 2015.

[6] I. Moura, G. Pinto, F. Ebert, and F. Castor. Mining Energy-Aware Commits. In *MSR 2015*, Florence, Italy, May 2015.

[7] G. Pinto, F. Castor, and Y. D. Liu. Mining Questions About Software Energy Consumption. In *MSR 2014*, pages 22–31, 2014.