Mining Questions about Software Energy Consumption

Gustavo Pinto[†‡], Fernando Castor[†], Yu David Liu[‡]

[†]Federal University of Pernambuco
Recife, PE, Brazil
{ghlp, castor}@cin.ufpe.br

[‡]SUNY Binghamton
Binghamton, NY 13902, USA
davidL@cs.binghamton.edu

ABSTRACT

A growing number of software solutions have been proposed to address application-level energy consumption problems in the last few years. However, little is known about how much software developers are concerned about energy consumption, what aspects of energy consumption they consider important, and what solutions they have in mind for improving energy efficiency. In this paper we present the first empirical study on understanding the views of application programmers on software energy consumption problems. Using StackOverflow as our primary data source, we analyze a carefully curated sample of more than 300 questions and 550 answers from more than 800 users. With this data, we observed a number of interesting findings. Our study shows that practitioners are aware of the energy consumption problems: the questions they ask are not only *diverse* – we found 5 main themes of questions – but also often more *interesting* and *challenging* when compared to the control question set. Even though energy consumption-related questions are popular when considering a number of different popularity measures, the same cannot be said about the quality of their answers. In addition, we observed that some of these answers are often flawed or vague. We contrast the advice provided by these answers with the state-of-the-art research on energy consumption. Our summary of software energy consumption problems may help researchers focus on what matters the most to software developers and end users.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Documentation, Human Factors

Keywords

Software Energy Consumption, Practitioners, Q&A

MSR '14, May 31 – June 1, 2014, Hyderabad, India

1. INTRODUCTION

Nowadays, thanks to the rapid proliferation of mobile phones, tablets, and unwired devices in general, energy efficiency is becoming a key software design consideration where the energy consumption is closely related to battery lifetime. It is also of increasing interest in the non-mobile arena, such as data centers and desktop environments. Energy-efficient solutions are highly sought after across the compute stack, with more established results through innovations in hardware/architecture [2, 14, 28], operating systems [10, 19, 24], and runtime systems [8, 25, 31]. In recent years, there is a growing interest in studying energy consumption from higher layers of the compute stack and most of these studies focus on application software [13, 15, 18, 23, 26, 29]. These approaches complement prior hardware/OS-centric solutions, so that improvements at the hardware/OS level are not cancelled out at the application level, *e.g.,* due to misuses of language/library/application features.

We believe a critical dimension to further improve energy efficiency of software systems is to understand *how software developers think*. The needs of developers and the challenges they face may help energy-efficiency researchers stay focused on the real-world problems. The collective wisdom shared by developers may serve as a practical guide for future energy-aware and energy-efficient software development. The conceptually incorrect views they hold may inspire educators to develop more state-of-the-art curricula.

The goal of this work is to obtain a deeper understanding of (i) whether application programmers are interested in software energy consumption, and, if so, (ii) how they are dealing with energy consumption issues. Specifically, the questions we are trying to answer are:

RQ1 What are the distinctive characteristics of energy-related questions?

RQ2 What are the most common energy-related problems faced by software developers?

RQ3 According to developers, what are the main causes for software energy consumption?

RQ4 What solutions do developers employ or recommend to save energy?

Our study is based on data from StackOverflow, a collaborative development questions and answers (Q&A) website. As one of the most popular forums in the software development world, StackOverflow is often used for software engineering studies [20, 30, 32]. It contains over 2

million users, 5.5 million questions and 10 million answers. StackOverflow data can be easily accessed through an open backup[1]. In this paper, we are looking for questions related to "energy efficiency", "energy consumption", and related terms. We found a total of 325 questions and 558 answers from more than 800 software developers. We employ a thematic analysis [9] to examine the data and identify recurring topics in the questions and associated answers. The raw data is available for replication and verification[2]. The main findings of this study are the following:

- Energy-related questions have distinct characteristics relative to the average StackOverflow questions. On the average, they have 2.6 times more answers, are marked as favorites 3.89 times more often, have 68% more views, 10% more "up-votes", and 11% more comments. Yet, the answers to them are *less* frequently "up-voted." Albeit *interesting*, we believe these questions are also more *challenging* for the software development community.

- We identify 5 main themes regarding energy consumption questions, namely: *Measurement*, *General Knowledge*, *Code Design*, *Context Specific* and *Noise*. Questions that pertain to the *Code Design* theme receive more attention.

- Energy consumption questions sustain a near-linear growth in the last 5 years, with contributors from 9 geographic regions. This suggests that energy consumption is an *emerging* topic.

- We identify 7 major causes for energy consumption problems according to developers. Factors such as unnecessary resource usage, background activities, and excessive synchronization are considered to be the common causes of energy consumption problems.

- We summarize 8 common solutions suggested by developers to improve energy efficiency. We discuss their perceived effectiveness, compare them with the state of the art of software energy consumption research, and highlight some of the mistakes made by StackOverflow users.

2. RESEARCH METHODOLOGY

In this section we describe our data collection procedure and the qualitative research approach we employ.

Our data collection follows a mixed-method approach, collecting both quantitative and qualitative data. Using the StackOverflow dump, we extracted questions, answers, tags, and other metadata. The data reported in this paper are based on questions that were asked between July 31, 2008 and September 06, 2013. We filter through more than 5 million questions, 10 million answers and 2 million users.

StackOverflow allows users to register, post questions, and answer posted questions. Since users are registered, one could track the questions/answers on a per-user basis. For each posted question, a user can include a **title** and textual description of the problem in the **body**. The user can also include code snippets. Code snippets are often separated from regular text. **Tags** are used to organize questions. Users have to attach at least one tag and can attach up to five tags when asking a question. For each question, multiple answers can be given by different users. The original user who ask the question can then either post a comment or indicate one of the answers as correct. Other people can also rate whether they like either the questions and/or the answers. The community itself is responsible for assuring the quality of the questions and answers: if a question or answer is considered relevant, thorough, or correct, users "up-vote" it; if not, they "down-vote" it. Users who posted those up-voted questions and answers receive "reputation points". That is, building reputation within the community is a key motivator for contributing to StackOverflow.

We follow a two-phase approach to filter out the questions related to energy consumption. We first load all data in a relational database. Then we query the table with the following terms: *energy consum*, *energy efficien*, *energy sav*, *save energy*, *power consum*, *power efficien*, *power sa*, *save power*. The character '*' in each term works as a wildcard: the query will select questions that match at least one of these keywords, regardless of the beginning or the end of the content. The searched keywords are matched against the subject, body or tags associated with each question. After this automatic process, we found 615 questions and 1.197 answers.

Next, we manually eliminated the false positive questions. For instance, a false positive question is when a programmer has an application that shows the energy consumption levels for a given scenario, and the programmer wants to add a feature to this application[3]. After this extraction phase, a total of 325 questions and 558 answers were selected. We refer this group throughout the paper as our **base group**. The amount of data we extracted comprises 123,075 words.

Once the data is collected, we extract reliable information through a thematic analysis approach [9]. Thematic analysis involves examining, identifying and recording patterns (or "themes") within data. Themes are patterns across data sets important to the description of a phenomenon and are associated with a specific research question. Themes are identified by bringing together components or fragments of ideas or experiences, which are often meaningless when analyzed in isolation. These themes become the categories for analysis. Thematic analysis is performed through the process of coding in six phases to create established, meaningful patterns. These phases are: familiarizing with the data (see Section 2, on data collection phase), generating initial codes, searching for themes among codes, reviewing themes, defining and naming themes (see Sections 3.2, 3.3 and 3.4), and producing the final report.

Due to the large amount of data extracted from our review process, we established 5 as a threshold for the minimum number of repetitions of a pattern required for it to be considered a theme [27]. This heuristic is based on the assumption that if more users are supporting a specific theme then it is likely to be stronger and more useful.

3. EMPIRICAL EVALUATION

In this section we describe our results grouped by research questions.

[1]http://blog.stackoverflow.com/category/cc-wiki-dump/

[2]Our data is available at: http://bit.ly/MSR2014

[3]www.stackoverflow.com/questions/413227

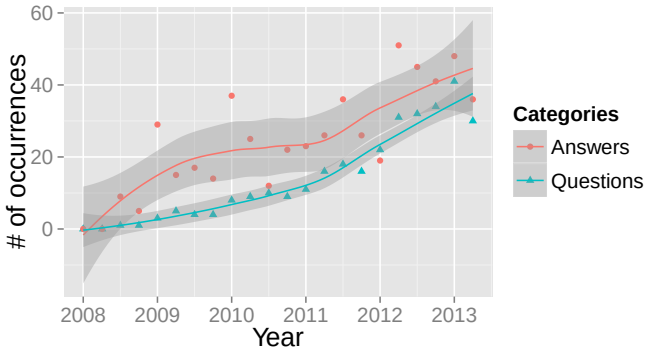

Figure 1: Questions and answers, from the base group, per year.

3.1 RQ1: What are the distinctive characteristics of energy-related questions?

To answer our first research question, we analyzed several quantitative aspects regarding our base group of questions.

First, we have analyzed whether the interest in energy consumption is increasing or decreasing over the years. To that end, Figure 1 shows the distribution between questions and answers created during the analyzed period.

The figure presents a chart indicating what trend is suggested by the data. The shaded areas surrounding the lines indicate the confidence interval calculated by smoothing. Each point in the figure represents quarterly data. We can observe a couple of interesting findings from this figure. First, it suggests an overall growth in the number of questions throughout the years. We observed a great increase in the number of questions until the first quarter of 2012, where the number of questions had increased 100% when compared to the same period of 2011. The number of questions kept increasing until it reached the highest part of the graph, located on the first quarter of 2013. At this point, we observed an increase of 183% in the number of questions when compared to the same period of 2012. The last point of the graph was not provided because it does not cover the entire quarter: the data is available until September 6, 2013. Second, the figure shows a great number of answers followed by a large standard deviation (overall SD: 13.89 for answers and 12.36 for questions). We observed that most of questions have less than 2 answers (3rd quartile: 2, median: 1, mean: 1.71), and only 15 questions (4% of them) have received more than 5 answers. However, the number of answers regarding the latter group is about 25% of the overall number of answers.

Second, on StackOverflow, the user who asks a question can mark at most one answer per question as **accepted**. We use this feature to define the *success* of a question as follows: A **successful** question has an accepted answer, an **ordinary** question has answers, but none of them was accepted, and an **unsuccessful** question has no answer. Following these definitions, Table 1 shows the number of questions comparing our base group of questions with the overall questions in StackOverflow (excluding the base group).

As we can see in the Table 1, the majority of the questions have answers – 85.85% and 90.80% for our base group of questions, and the remaining group of questions on StackOverflow, respectively. However, only part of them are successful. For the base group of questions, more than a half of these questions have successful answers. On the other hand, about half of them are successful for the remaining questions on StackOverflow. Notwithstanding, the base group also has the highest percentage of unsuccessful answers. As we will discuss later (Section 3.2), some categories of questions are more unsuccessful than others.

We also observed that there are no obvious "energy consumption experts". Only 1 user has answered 3 different questions, and 27 users (4.83% of the total) have answered 2 different questions. Also, no user asked more than one energy consumption-related question. Still, we observed that the questions in the base group come from a greatly diverse group of developers. They are from 9 geographic regions: North America, Central America, South America; Europe; Africa; West Asia, Central Asia, East Asia; and Australia/New Zealand. The average age, reputation and views per user page are, respectively, 28.58, 1,798 and 218.1.

We have also compared the popularity of energy consumption questions with the average StackOverflow questions. To measure if a given question is popular, we derived the following formula – which we believe provides a summarized view that aggregates a number of different popularity measures:

$$\mathbb{P} = \mathbb{S} + \mathbb{A} + \mathbb{C} + \mathbb{F} + \mathbb{V}$$

where $\mathbb{S}$ is the score of the question. The user can "up-vote" a question if she thinks that the question is good enough, or "down-vote" it otherwise. The score is the result of this votation process. The variables $\mathbb{A}$, $\mathbb{C}$, and $\mathbb{F}$ are, respectively, the number of answers, comments, and favorizations per question. A *favorite* question is one that developers want to know more about. By indicating a question as favorite, a developer receives notifications whenever that questions is updated (modified, answered, etc). The $\mathbb{V}$ variable corresponds to the number of views. We normalize each variable to avoid distortions caused by the very large absolute values. Taking the $\mathbb{V}$ variable as an example, it is normalized by the average of the number of views of the overall StackOverflow questions (which is 380.06), as follows: $\mathbb{V} = questionsViews \big/ stackOverflowViews$. The other variables follow the same rule. We chose this normalization approach because it can give us a fair way to compare the two groups of questions. Table 2 shows the normalized results of each variable of our base group of questions.

Since the value for each metric for the StackOverflow group is normalized to be 1 (then its $\mathbb{P}$ value is 5) we can observe that questions from our base group are two times more popular when compared to the average questions on StackOverflow. It not means, however, that energy consumption is the most popular topic on StackOverflow. In particular, only the $\mathbb{S}$ and the $\mathbb{C}$ variables have similar values. For the $\mathbb{S}$ variable, we observed that 3.69% (12 out of

Table 1: The success status of questions on StackOverflow.

Source	Successful	Ordinary	Unsuccessful
Base Group	45.85%	40.00%	14.15%
StackOverflow	39.99%	50.81%	9.20%

Table 2: The popularity of questions in each group of data. We removed outliers from the histograms, but not from the numerical results.

	Base Group	Median	Standard Dev.	Histogram
$\mathbb{S}$	1.10	1.00	4.73	
$\mathbb{A}$	2.67	1.00	2.35	
$\mathbb{C}$	1.11	1.00	2.46	
$\mathbb{F}$	3.89	0.00	1.79	
$\mathbb{V}$	1.68	1.00	3.37	
$\mathbb{P}$	10.45	–	–	–

325) of the questions have negative scores. Analyzing those questions and the associated comments, we have found that they were negative because they are (i) poorly elaborated[4], (ii) already answered[5] or (iii) impossible to answer[6]. The values for $\mathbb{A}$ and $\mathbb{F}$ present the most significant differences.

Table 3 shows the results for $\mathbb{S}$ and $\mathbb{C}$ variables, but now regarding the answers of those questions. We did not compute the other variables because answers do not have such information. The popularity of answers is calculated as $\mathbb{PA} = \mathbb{S} + \mathbb{C}$. Here, comments and scores for the answers are similar although the results for the base group were lower. Although it is easy to create and favorite (and then, follow) such questions, as we will discuss later, providing reliable answers to them is not straightforward.

Table 3: The popularity of answers from each group. We removed outliers from the histograms, but not from the numerical results

	Base Group	Median	Standard Dev.	Histogram
$\mathbb{S}$	0.96	1.00	4.52	
$\mathbb{C}$	0.86	0.00	1.80	
$\mathbb{PA}$	1.82	–	–	–

Finally, we investigated how tags are used in StackOverflow. A StackOverflow user employs tags to categorize a question and group it with other similar questions. We found a total of 970 tags, but only 360 different keywords were used to tag questions. Each energy consumption question has between one and five tags (median: 3, mean: 2.99, 3rd quartile: 4, SD: 1.28). The five most common tags are: (i) "android" with 172 occurrences, (ii) "power-management" with 33 occurrences, (iii) "Java" with 28 occurrences, (iv) "iOS" with 24 occurrences, and (v) "iPhone" with 24 occurrences. This result indicates that energy consumption questions are strongly related to mobile development – 26.28% of the tags are mobile-related. In particular, the Android platform has 17.78% of overall tag usage. This is probably due to its strong adoption, and the ease of developing Android applications. Another indicator is the Java programming language appears to be of the most often used tags. Since Android applications are written in Java, developers are also interested in saving energy in this particular language.

RQ1 Summary: 85% of the questions from the base group have answers and 45% of them are answered successfully. The average number of answers per question is 2.6 times greater than the average StackOverflow question. Questions from the base group have 1.68 times more views and favorited 4 times more often than the average of StackOverflow questions. Questions and answers are increasing year by year, with peaks of 180% of increment in one year. About 1/4 of the energy consumption-related questions pertain to mobile development.

3.2 RQ2: What are the most common energy-related problems faced by software developers?

To further understand the characteristics of software energy consumption-related questions on StackOverflow, we analyzed the titles and bodies of the questions from the base group in order to identify what recurring categories these questions define. The categories are our themes (Section 2). We identified a total of 5 themes.

Measurements. Questions about measurement tools or techniques, e.g., "*I want to measure the energy consumption of my own application (which I can modify) [...] on Windows CE 5.0 and Windows Mobile 5/6. Is there some kind of API for this?*"[7].

General Knowledge. Questions that do not have a concrete use case, e.g., "*Can a code optimized for least MCPS be guaranteed to have least power consumption as well?*"[8].

Code Design. Questions about programming techniques that can help in saving energy, e.g., "*Are there any s/w high level design considerations [...] to make the code as power efficient as possible?*"[9].

Context-specific. The authors of such questions need to provide more details in order for other users to better understand the problem, and to facilitate replication, e.g., "*I want to prevent the monitor from going to sleep. [...] What call do I make?*"[10].

Noise. These questions are not directly associated with an energy consumption issue. Usually, the user first wants to improve one aspect of her application and, as a secondary goal, also improve energy consumption, e.g., "*What are the good features of a processor should have which help in carrying out multimedia(Video/Image)?. [...] PS: It has to be low power as it is for portable applications.*"[11].

Table 4 shows the distribution of questions and answers per theme as well as the number of successful, ordinary and unsuccessful questions in each category. The table also shows the value of each normalized popularity variable. We use **boldface** to highlight the highest value for each case.

From this table we can make some interesting observations. First, the number of questions and answers per category can vary greatly. For instance, *Code Design* is the category with the smallest number of questions (36 questions;

[4]www.stackoverflow.com/questions/4946600
[5]www.stackoverflow.com/questions/9187303
[6]www.stackoverflow.com/questions/11011260
[7]www.stackoverflow.com/questions/724349
[8]www.stackoverflow.com/questions/506452
[9]www.stackoverflow.com/questions/506452
[10]www.stackoverflow.com/questions/1003394
[11]www.stackoverflow.com/questions/3625568

Table 4: Common energy-consumption themes with their status and popularity. Q means questions, A means answers. All the values for variables $\mathbb{S}$, $\mathbb{A}$, $\mathbb{C}$, $\mathbb{F}$ and $\mathbb{V}$ are normalized.

Themes	Q	A	A/Q	Successful	Ordinary	Unsuccessful	$\mathbb{S}$	$\mathbb{A}$	$\mathbb{C}$	$\mathbb{F}$	$\mathbb{V}$	$\mathbb{P}$
Measurements	59	97	1.64	32.22%	47.45%	20.33%	1.12	2.56	1.04	4.42	**1.97**	11.11
Context Specific	83	110	1.33	42.18%	**50.60%**	7.22%	0.64	2.06	0.81	2.26	1.89	7.66
Code Design	36	133	**3.69**	**72.22%**	25.01%	2.77%	**2.80**	**5.76**	**1.57**	**10.21**	1.63	**21.97**
G. Knowledge	40	84	2.10	50.00%	40.00%	10.00%	1.87	3.28	1.32	4.73	1.58	12.78
Noise	**107**	**134**	1.25	45.65%	33.00%	**21.35%**	0.76	1.95	1.19	2.94	1.24	8.08

113 answers) and *Noise* is the one with the greatest number (107 questions; 134 answers). On the one hand, questions that lie in the *Code Design* category are usually focused on how programmer decisions may improve energy consumption. We believe that this category has a low number of questions because energy consumption, as an emerging concern, has not yet firmly established itself in modern software engineering practices outside of specific domains such as embedded systems. On the other hand, the high number of questions in *Noise* category shows that, although energy consumption is not the primary software requirement, programmers usually refer to it as a desirable requirement. Usually, programmers are interested in improving energy consumption's closest relative: performance. Another point regarding the *Code Design* category is that it has the highest number of answers per question (A/Q ratio: 3.96), which suggests that developers are strongly interested in this kind of questions. However, we found 3 outlier questions with a high number of answers. Together, these three questions have 43% of the overall answers in this category. Analyzing those questions, we have found that they have a high number of answers because they are (i) challenging[12][13] and (ii) polemic[14]. Despite the low number of questions, we believe that the *Code Design* category is the most important one for future energy-aware software development. Thus, we dedicated Section 3.4 to describe this category in greater detail.

Second, we observed that the success rate can vary significantly. For instance, *Code Design* is the category that has more accepted answers (72.22%), and *Measurements* is the worst (32.22%). Analyzing the answers to those questions, we observed that, for the *Code Design* category, an accepted answer usually provides a number suggestions, and one of them may work for the programmer who asked the question. In contrast, energy consumption measurement is not always straightforward due of the lack of specialized tools. Programmers often want to measure energy consumption at different levels of granularity. We observed a number of granularities, including hardware devices (USB, Camera, GPS), network sensors (wifi, 3G, bluetooth), operating systems, kernel, virtual machine, process, thread, and application (line, method, and whole program). Although it is easy to create such questions, answering them is not. For most of these granularities, there is no standard mainstream hardware device, tool or technique available. In fact, only recent studies propose methods to estimate energy consumption at the source line of code level [11, 16]. At the same time, StackOverflow users are clearly interested in "Measurement" questions. It is the category with the highest number of views per question (1.97 times the average StackOverflow question).

Third, the rate for unanswered questions (the % of unsuccessful questions) is often less than 20%, with a small variation (SD: 8.18). Only for the *Measurement* and *Noise* categories, the rate for unanswered questions is greater than 20%. We have explained why the *Measurement* has a high percentage of unsuccessful questions. Since questions in the *Noise* category are not mainly about energy consumption, the answers are not as well. Thus, the low answer rate is not directly related to energy consumption issues. Finally, the popularity (Section 3.1) of the questions in each theme varies from 7.66 to 21.97 (SD: 5.79). Again, the *Code Design* category has the highest values for score, answers, comments, and favorites (variables $\mathbb{S}$, $\mathbb{A}$, $\mathbb{C}$ and $\mathbb{F}$, respectively). In fact, the value of the $\mathbb{F}$ variable for *Code Design* is more than twice higher than the value of $\mathbb{F}$ for the *General Knowledge* category, which has the second highest value. This result might suggest that, although a small number of users are asking these questions, they are more interesting than any other kind of energy consumption questions.

RQ2 Summary: We identify 5 main categories of energy consumption questions: Measurement, Context-Specific, Code Design, General Knowledge, and Noise. The percentage of questions without an answer is often less than 20%. Despite the low number of questions, the "Code Design" category has the highest popularity value, as well as the highest success rate and the lowest rate for unanswered questions. On the other hand, "Measurement" questions have the lowest percentage of successful questions, even though they are the most frequently viewed questions.

3.3 RQ3: According to developers, what are the main causes for software energy consumption?

Analyzing the answers from the base group, we identified seven common themes regarding the sources of energy consumption.

Unnecessary resource usage. When one resource is not being used, the programmer should power it down (or put it to sleep). We found 49 occurrences of this theme in our data. A resource running at full power but sitting unused can represent a powerful source of energy drain. Indeed, a user suggested, "*to have a background application that monitors device usage, identifies unused/idle resources,*

[12] www.stackoverflow.com/questions/61882
[13] www.stackoverflow.com/questions/422539
[14] www.stackoverflow.com/questions/1318851

and acts appropriately"[15]. However, this suggestion might consume extra energy by polling the device, and by creating new background activities.

Faulty GPS behaviour. StackOverflow users (42 occurrences) suggest that application programmers should take special care when dealing with GPS features. As a user pointed out "*[..] Make sure that the GPS is only on for the bare minimum of time. Of course, when there are bugs that are introduced that keep the GPS turned on too long they go to the top of the list to get fixed*"[16]. Another user also suggested to "*avoid using fine grain location where a coarse location would do (GPS vs cellular location)*"[17]. In fact, GPS sensors are known for their high impact on energy consumption [35].

Background activities. Some users (40 occurrences) highlight unnecessary background activities, such as performing HTTP connections or changing some device features, as another important source of energy consumption problems. When combined, such activities might represent an important energy drain. The application programmer could avoid such background activities by debugging her application, or by providing features to turn on/off such activities. As a StackOverflow user has pointed, "*Inefficient background activity has a dramatic impact on system performance, power consumption, responsiveness, and memory footprint*"[18].

Excessive synchronization. StackOverflow users (32 occurrences) suggest that synchronizing states between thread/process might increase energy consumption. For example "*I need to maintain the data updated, according to the modify that may happens server-side. [...] If new data are available, other web service is called for obtaining them. This solution works fine for my [...], but this solution is too expensive in term of energy consumption [...].*"[19].

Background wallpapers. StackOverflow users (17 occurrences) have pointed out that wallpapers in mobile applications, may incur significant energy consumption. It could be even worse when the wallpaper uses animations at a high frame rate. Also, colors play an important role in energy consumption. StackOverflow users agree that the white color is the most energy inefficient, whereas black is the most energy efficient. As a StackOverflow user said "*To give you an idea, a static blue wallpaper (for instance a jellyfish in an aquarium) consumes more battery than the 3D galaxy live wallpaper*"[20].

Advertisement. According to StackOverflow users (11 occurrences), advertisement can increase energy consumption in mobile applications. "*to send advertisements, just turns on briefly to do a quick Tx/Rx. When scanning, the Rx is turned on for a relatively long time because you don't know at what time or on which frequency advertisements will arrive*"[21]. A recent study has discussed the impact of advertisement on different mobile applications [33]. The authors showed savings of up to 75% in applications that do not use advertisement.

[15]www.stackoverflow.com/questions/3092498
[16]www.stackoverflow.com/questions/4361967
[17]www.stackoverflow.com/questions/4361967
[18]www.stackoverflow.com/questions/422539
[19]www.stackoverflow.com/questions/14997997
[20]www.stackoverflow.com/questions/2902382
[21]www.stackoverflow.com/questions/13584367

High GPU usage. StackOverflow users (8 occurrences) believe that GPUs waste more energy than CPUs. The rationale behind this intuition is that GPUs usually have more cores (and more cooling devices as well), as stated by a user: "*[. . .] The higher the clock rate, the more power used. GPUs tend to have high clock rates so they can do lots of work;* "[22].

RQ3 Summary: As StackOverflow *users suggest, some of the causes for high energy consumption might include unnecessary resource usage, excessive synchronization between peers, background activities, advertisement, and even the use of bright colors.*

3.4 RQ4: What solutions do developers employ or recommend to save energy

We identified that only 11.07% (36 out of 325) of the questions ask about solutions to improve the energy consumption by doing modifications at the code level. However, it is interesting to observe that some respondents seem to know that a particular piece of code might have a substantial impact on energy consumption, which is not straightforward. Only recent studies have focused on understanding the relationship between code design and energy consumption in high-level applications [15, 23, 26]. We hypothesize that those StackOverflow users are experiencing significant battery drain while introducing new features.

We found a total of 8 themes in our qualitative analysis. We identified these themes by analyzing the answers about reducing the energy consumption through modifications at the code design level. We chose these themes due to the frequency with which they appear in our data. We briefly describe each theme next:

Keep IO to a minimum. StackOverflow users (29 occurrences) believe that accessing external devices increases energy consumption. In particular, this could be even more detrimental to mobile devices, which perform IO operations frequently via bluetooth, wifi, camera and GPS, in the same application, concurrently. One common suggestion is "*to keep IO to a minimum*"[23].

Bulk operations. A number of users (24 occurrences) argued that, if one needs to send a sequence of commands to a hardware device, it is preferable to buffer them up and send them out all at once. In this manner, the programmer may avoid multiple wakeup → send → sleep cycles. For instance, as a user pointed out "*Don't transfer say 1 file, and then wait for a bit to do another transfer. Instead, transfer right after the other. This reduces the amount of time the radios need to be active for, and hence conserves battery life*"[24]. It also includes other IO operations, such as read/write in files, wifi transmissions, GPS usage, synchronizing data through the network, among many others.

Avoid Polling. Several users (17 occurrences) have argued that polling is not a good design choice for energy consumption. With polling, the application is continuously busy, polling the hardware to check if the desired value is available, which may become a source of wasted CPU cycles. As one user said, "*Polling is imprecise by its nature. The higher your target precision gets, the more waste-*

[22]www.stackoverflow.com/questions/12332609
[23]www.stackoverflow.com/questions/2905958
[24]www.stackoverflow.com/questions/12120629

ful the polling becomes"[25]. A common suggestion is to use asynchronous communication instead. Since the recipient of asynchronous communication can remain dormant until an event occurs, an interrupt-driven approach can make efficient use of existing resources. Programmers should consider polling only if they cannot achieve the goal using interruptions, "*The best way I can think of to do this would to make an application entirely interrupt-driven*"[26].

Hardware Coordination. Some users (11 occurrences) claim that the programmer should know better the underlying hardware to save energy. Then, the programmer can (i) "*execute [the code] entirely in the processor cache, you'll have less bus activity and save power*"; (ii) "*If the processor has multiple levels of cache, try to fit in the lowest level of instruction or data cache possible.*"; (iii) "*if the code needs to use external memory, try to use it as little as possible*"; (iv) "*don't use [the] floating point unit or any instructions that may power up any other optional functional units unless you can make a good case that use of these instructions significantly shortens the time that the CPU is out of sleep mode*", and (v)"*Set unused memory or flash to 0xFF not 0x00*"[27].

Concurrent programming. Appropriate use of multithreading can greatly improve application performance. Some users (9 occurrences) have pointed out that concurrency could be used to improve energy efficiency as well. For example, "*using multiple threads can save energy when you have I/O waits. One thread can wait while other threads can perform other computations; instead of having your application idle*"[28].

Lazy Initialization. Some users (7 occurrences) pointed out that lazy resource initialization could save energy, *e.g.* the programmer needs to "*draw to the screen only when necessary rather than endlessly*"[29]. The application should initialize resources only when they are strictly necessary.

Race to Idle. Some of the StackOverflow users (7 occurrences) suggest that, to save energy, the programmer should "*do the work as quickly as possible, and then go to some idle state*"[30]. The rationale behind this is that faster programs will theoretically consume less energy because they will have the machine idle faster, waiting for events (or interrupts) to happen. Moreover, modern CPUs are very energy-efficient when idle [2].

Efficient Data structure. Finally, some users (5 occurrences) agree that the use of high level data structures, such as maps or concurrent implementations, should be avoided when unnecessary, and simple data structures should be preferred instead. As a StackOverflow user suggested, "*Use plain C data structures (instead of Foundation objects) and pack them*"[31]. The programmer saves energy by avoiding creating new internal objects.

[25] www.stackoverflow.com/questions/10929875
[26] www.stackoverflow.com/questions/3866746
[27] All suggestions stated in this paragraph are in the same question: www.stackoverflow.com/questions/61882
[28] www.stackoverflow.com/questions/6925572
[29] www.stackoverflow.com/questions/4361967
[30] www.stackoverflow.com/questions/61882
[31] www.stackoverflow.com/questions/4361967

RQ4 Summary: We have found 8 themes of design choices that the StackOverflow *users claim that could impact energy consumption. These solutions cover a wide spectrum of computational systems and include hardware manipulation, concurrent programming, reducing the amount of IO operations, among others.*

4. DISCUSSION

In this section, we summarize our findings, and provide additional discussion on the data presented in the previous section.

4.1 Overall Assessment

Our study reveals distinct characteristics of energy consumption questions and their answers.

Compared with the average of the questions in StackOverflow, discussions related to energy consumption can be considered more *interesting* and *challenging*:

- *interesting*: Questions from the base group are up-voted 1.1 times more frequently than the average StackOverflow question. In the same vein, they are answered 2.67 more frequently than the average StackOverflow question, viewed 1.68 times more frequently, and marked as favorites 3.89 more frequently.

- *challenging*: even though the questions related to energy consumption are more likely to be up-voted, their answers are not (the $\mathbb{S}$ value for the answers is 1.05 times lower than StackOverflow control set). These answers are also entailed by fewer comments (the $\mathbb{C}$ value is 1.16 times lower). We have examined these answers and noticed that some of them have evident problems. We discuss them in Section 4.2.

4.2 Misconceptions, panaceas, and lack of tools

We identified three recurring problems in the answers in our base group of questions: (i) misconceptions about software energy consumption and how it can be reduced, (ii) solutions that are applicable in certain contexts being presented as universal, and (iii) lack of tools, in particular, measurement tools. We examine each one in turn.

First, we find 37 users holding *misconceptions*, of which we describe three. (1) Some users confuse "power" and "energy." Scaling down CPU frequencies is often suggested as a direct solution to reduce *energy* consumption, which in reality has more direct impact on *power*. Scaling down the CPU frequency is effective in saving *power*. Saving *energy* however is more complex. Since energy consumption "E" is an accumulation of power consumption "P" over time "t", that is $E = P \times t$, a reduction of frequency may increase the execution time, keeping energy constant or even increasing it. (2) StackOverflow users often use performance as the primary indicator to estimate energy consumption, such as "*It may reduce execution time since HW accel is there, therefore power consumption may be lower*"[32]. According to numerous studies [4, 7, 29, 23, 24], power and performance are not always correlated. (3) Some StackOverflow users believe that switching to a managed language runtime, such as Java or .NET, might improve energy consumption, such as "*My take is that the best way is to mix*

[32] www.stackoverflow.com/questions/9924255

languages, use existing Java-based infrastructural tools [...] to build the performance critical parts and something more nimble to build complex but not that heavy business and presentation logic."[33]. However, recent studies show that energy consumption is heavily workload-dependent [7]. Also, managed languages have to deal with the energy consumption cost of their underlying services. Another recent study points out that the total energy consumption in VM services ranges from 9% to 82% of the overall energy consumption of an application [4]. Curiously, none of these answers were "down-voted."

Second, we identify some generic *panaceas* offered (by 23 users) as solutions to energy problems. They are not necessarily incorrect, but the generic folklore nature of such answers hardly qualifies them as problem solvers. For instance, one user asked "*[...] given this how can I write the code 'power' efficiently so as to consume minimum watts?*"[34]. Some suggested answers include:

- *Interrupts are your friends; Polling / wait() aren't your friends*
- *Do as little as possible*
- *Make your code as small/efficient as possible*
- *Turn off as many modules, pins, peripherals as possible in the micro*
- *Run as slowly as possible*

Part of the vagueness here is innate to online forums. Nonetheless, many other common questions frequently asked by StackOverflow users, such as *How to avoid memory leaks?* or *How to fix null pointer bugs?*, are often entailed by much more concrete and useful answers.

Third, we believe the *lack of tool support* poses significant hurdles to energy-aware software development. We observe that StackOverflow users are keenly aware of the multi-granularity multi-level nature of energy optimizations with diverse discussions on hardware devices, operating systems, and applications. From the 59 questions under the *Measurement* theme in Section 3.2, 38 of them are directly related with a lack of tool support. These programmers are faced with insufficient support of established tools, from lower-level ones for energy profiling, measurement, and estimation, to higher-level ones for energy management frameworks, software architectures, and refactoring tools. One example of it is the following question created by a StackOverflow user: "*JouleMeter is a tool from Microsoft for measuring power consumption by different processes on a windows machine. Please tell me if there is any similar tool on linux for getting information of energy consumption by different processes and applications on linux machine. Also I am looking for an open-source solution.*"[35], which has no answer.

4.3 Do researchers agree with the code design suggestions?

In Section 3.4 we categorized 8 strategies that StackOverflow users suggested as a way to save software energy consumption. In this section, we compare these strategies with the state of the art, in order to verify whether they are supported.

Keep IO to a minimum. A recent study showed that an IO-intensive benchmark could produce high energy consumption [23]. Similarly, network communication is a major consumer of energy [22], consuming between 10% and 50% of the total energy budget of a typical mobile application. Thus, reducing IO might save energy.

Bulk operations. This strategy could save energy by reducing IO overhead and network communication [15]. However, the degree of batching should be determined by the network conditions in place. For example, for a network with limited bandwidth (e.g., 3G or 4G), batching too many individual remote communications can saturate the network, which in turn can increase the aggregate latency of remote interactions, thus incurring high energy costs.

Hardware Coordination. Several authors support the idea of using memory optimization techniques in order to reduce the bus switching activity [3, 28, 10, 11]. Other software-oriented proposals focus on instruction scheduling and code generation, possibly minimizing memory access cost [8].

Concurrent programming. Several authors [6, 17, 23, 24, 29] have been studying the relationship between concurrent programming, performance and energy, but no consensus has emerged from it. For example, Liu [17] conducted an initial study on the energy consumption of a number of synchronization approaches. Furthermore, Pinto and Castor [23] investigated a group of high level concurrent constructs in different scenarios in order to identify when it is possible to switch from one construct to another. In both studies the authors suggest that concurrency cannot be seen as a one-size-fits-all solution.

Race to Idle. Race to idle is the assumption that, the faster the program executes, the more energy efficient it will be. Most of this belief comes from the use performance as a proxy for energy consumption. However, this is not an accurate approximation. For instance, mobile devices and modern CPUs, scale their voltage dynamically, which makes it more difficult to say beforehand which computation will consume less energy [16]. Also, hardware components have varied energy consumption patterns [11]. Concurrent software plays yet another role: it has more complex behaviour and overheads. Recent studies have shown that the race to idle principle does not always hold for multi-threaded applications. [23, 29].

Efficient Data structures. The idea of energy efficient data structures is not new. Daylight *et al.* [6] discussed some energy consumption patterns and trade-offs faced during the implementation of non-trivial energy efficient data structures. The authors showed gains in energy consumption with a relatively small overhead.

As for the remaining solutions proposed by the respondents (Section 3.4), we did not find studies in the research literature that directly support them, although there are related ones. For example, Sahin *et al.* [26] analyzed the impact of some well-known design patterns on energy consumption, and Menarini [18] provided important clues on how different e-services impact on energy consumption. We describe four more themes, from the researchers perspective, that we consider valuable for the development community.

Loop transformations. In Brandolese *et al.* [3], the au-

[33] www.stackoverflow.com/questions/1318851
[34] www.stackoverflow.com/questions/61882
[35] www.stackoverflow.com/questions/13286662

thors suggests 8 loop transformations to save energy. Some of those include (i) *loop fusion*: merges different loops to reduce control operations, and (ii) *loop interchange*: modifies the nesting ordering of the loops to change array accesses. Even though some of these can be achieved by automatic compiler optimization, programmers can greatly help in this area to compensate for the fundamentally conservative nature of compiler optimizations.

Data compression. Wilke [33] *et al.* showed that compressed image formats may save energy (*e.g.*, when transferring images through networks). With compressed image formats, programmers decrease the overall image size, whereas maintaining reasonably image quality.

Offloading methods. Few answers (4 occurrences) mention offloading techniques to save energy. When developing a mobile application, sending heavy calculations to a server [15] has the potential to save energy. However, a threshold should be defined when using this approach: if the calculation is too fast, it might be better to run it locally, rather than incurring the cost of network communication.

Approximated programming. Many software components can tolerate occasional "soft errors", *i.e.*, errors that may reduce the accuracy of the results. This technique might bring the benefits of fast and energy-efficient execution at the price of some controlled approximation [5].

5. THREATS TO VALIDITY

We divide our threats to validity discussion in terms of internal ones and external ones.

Internal: First, our study is restricted by the size of our dataset. Despite having mined the entire StackOverflow site, we only found 325 questions. Moreover, a number of studies have been conducted using StackOverflow in recent years [20, 30, 32]. In fact, StackOverflow is the most widely used Q&A website in the software development world. Studies show that StackOverflow users are greatly diverse with relation to their skills and age [20]. Second, not all questions that we find through querying the database using energy consumption keywords are related to energy consumption. We minimize the false-positive rate by investigating all questions and answers manually. Still, in order to reduce the number of false-negatives, we experimented a number of energy consumption-related keywords. Most of the keywords used were based on the study of Wilke *et al.* [34]. We examined each one of them and kept only the keywords that retrieved relevant questions. Third, we choose Thematic Analysis as our research method. While we achieved saturation (*i.e.*, no more themes could be absorbed) regarding the topics we focused on in our research, there may be additional themes that might add new insights. We also share our raw data (extracted questions and answers) to provide a means for replication and verification.

External: Our results only apply to application programmers interested in energy consumption on StackOverflow. It does not cover systems programmers, nor application programmers that use other Q&A websites. However, even though the results of this work might not be generalizable, we believe that our data source is reliable. Since no user has created more than one question and no user has produced more than three answers, our data demonstrates diversity. Finally, we are only considering English questions and answers.

6. RELATED WORK

Energy efficiency in software engineering practices is an emerging problem with few prior empirical studies. Hindle [13] investigated the relationship between software changes and power consumption. The author observed that intentional performance optimization introduced a steady reduction in power consumption. Pathak *et al.* [21] categorized energy bugs through analyzing the posts from 4 online forums. They produced a comprehensive taxonomy ranging from battery problems, SIM card problems, OS configuration problems, to no-sleep bugs. In comparison, our choice of StackOverflow — a programmer-centric website — allows us to zero in on energy-aware *software development*, instead of taking the more system-oriented view for mobile devices. The Q&A form of StackOverflow further offer rich contextual information of the energy-related keywords, providing insights such as what *solutions* software practitioners know about energy consumption problems. Heikkinen *et al.* [12] studied energy problems of mobile handsets through a questionnaire-based study. The focus in their study is on handset *users*, not *software developers*. Recently, Wilke *et al.* [34] analyzed the Apps from Google Play market place. Their study focuses on a domain complementary to ours: the impact of energy consumption problems on the post-programming stages of software lifecycle. Their work is particularly interesting in correlating energy consumption problems with the user ratings to the Apps, the pricing of Apps, and different natures of Apps.

More generally, Q&A websites have been well studied. Empirical studies over StackOverflow include Treude *et al.* [30], who analyzed 385 questions and labeled them into 10 categories, Wang *et al.* [32], who automatically analyzed 100,000 questions and identified the top-5 topics of interest, and Morrison *et al.* [20] who studied how age is related to programming skills, and results show that programming knowledge increase until the age of 50. More broadly, Barua*et al.* [1] proposed a semi-automatic approach to discover the main topics present in StackOverflow discussions. The authors observed several relationships between topics and trends over time. Also, they pointed out that web and mobile development are the most popular topics.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we investigated how StackOverflow users are interested in software energy consumption. We observed that energy consumption questions are more *interesting* and also more *challenging* than the average StackOverflow question. We identify 5 main themes regarding energy consumption questions, namely: *Measurement*, *General Knowledge*, *Code Design*, *Context Specific* and *Noise*. Questions that focus on source code modifications receive more attention (popularity and answers). We identify 7 major causes for energy consumption problems, varying from background activities to synchronization. We also discuss how to address each one of them and, when possible, compare with the state-of-the-art software energy consumption research.

As future work, we plan to extend the findings of this study with a survey, in order to observe if our findings could be generalizable to other data sources. Also, we plan to analyze commit messages and bug reports as ways to investigate how frequently energy bugs are being introduced during the software evolution process.

8. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful comments, and Irineu Moura for the valuable reviews. This work is supported by CAPES, CNPq (306619/2011-3, 487549/2012-0 and 477139/2013-2), FACEPE (APQ-1367-1.03/12), INES (CNPq 573964/ 2008-4 and FACEPE APQ-1037-1.03/08), a Google award, and by NSF (CCF-1054515).

9. REFERENCES

[1] A. Barua, S. Thomas, and A. Hassan. What are developers talking about? an analysis of topics and trends in stack overflow. *EMSE*, 2012.

[2] L. Bircher and L. John. Analysis of dynamic power management on multi-core processors. In *ICS*, 2008.

[3] C. Brandolese, W. Fornaciari, F. Salice, and D. Sciuto. The impact of source code transformations on software power and energy consumption. *Journal of Circuits, Systems and Computers*, 11(05), 2002.

[4] T. Cao, S. Blackburn, T. Gao, and K. McKinley. Yin and yang of power and performance for asymmetric hardware and managed software. In *ISCA*, 2012.

[5] M. Carbin, D. Kim, S. Misailovic, and M. Rinard. Proving acceptability properties of relaxed nondeterministic approximate programs. In *PLDI*, 2012.

[6] E. G. Daylight, T. Fermentel, C. Ykman-Couvreur, and F. Catthoor. Incorporating energy efficient data structures into modular software implementations for internet-based embedded systems. In *WOSP*, 2002.

[7] H. Esmaeilzadeh, T. Cao, X. Yang, S. Blackburn, and K. S. McKinley. What is happening to power, performance, and software? *IEEE Micro*, 32(3), 2012.

[8] K. I. Farkas, J. Flinn, G. Back, D. Grunwald, and J. M. Anderson. Quantifying the energy consumption of a pocket computer and a java virtual machine. In *SIGMETRICS*, 2000.

[9] J. Fereday. Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development. *International Journal of Qualitative*, 5, 2006.

[10] R. Ge, X. Feng, W. chun Feng, and K. Cameron. Cpu miser: A performance-directed, run-time system for power-aware clusters. In *ICPP*, 2007.

[11] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan. Estimating mobile application energy consumption using program analysis. In *ICSE*, 2013.

[12] M. V. Heikkinen, J. K. Nurminen, T. Smura, and H. Hammainen. Energy efficiency of mobile handsets: Measuring user attitudes and behavior. *Telematics and Informatics*, 29(4), 2012.

[13] A. Hindle. Green mining: A methodology of relating software change to power consumption. In *MSR*, 2012.

[14] A. Iyer and D. Marculescu. Power efficiency of voltage scaling in multiple clock, multiple voltage cores. In *ICCAD*, 2002.

[15] Y.-W. Kwon and E. Tilevich. Reducing the energy consumption of mobile applications behind the scenes. In *ICSM*, 2013.

[16] D. Li, S. Hao, W. G. J. Halfond, and R. Govindan. Calculating source line level energy information for android applications. In *ISSTA*, 2013.

[17] Y. Liu. Energy-efficient synchronization through program patterns. In *GREENS*, 2012.

[18] M. Menarini, F. Seracini, X. Zhang, T. Rosing, and I. Kruger. Green web services: Improving energy efficiency in data centers via workload predictions. In *GREENS*, 2013.

[19] A. Merkel and F. Bellosa. Balancing power consumption in multiprocessor systems. In *EuroSys*, 2006.

[20] P. Morrison and E. Murphy-Hill. Is programming knowledge related to age? an exploration of stack overflow. In *MSR*, 2013.

[21] A. Pathak, Y. C. Hu, and M. Zhang. Bootstrapping energy debugging on smartphones: A first look at energy bugs in mobile devices. In *HotNets*, 2011.

[22] K. Pentikousis. In search of energy-efficient mobile networking. *Comm. Mag.*, 48(1), 2010.

[23] G. Pinto and F. Castor. On the implications of language constructs for concurrent execution in the energy efficiency of multicore applications. In *SPLASH Companion*, 2013.

[24] K. K. Rangan, G.-Y. Wei, and D. Brooks. Thread motion: fine-grained power management for multi-core systems. In *ISCA*, 2009.

[25] H. Ribic and Y. D. Liu. Energy-efficient work-stealing language runtimes. In *ASPLOS*, 2014.

[26] C. Sahin, F. Cayci, I. Gutiérrez, J. Clause, F. Kiamilev, L. Pollock, and K. Winbladh. Initial explorations on design pattern energy usage. In *GREENS*, 2012.

[27] L. Singer, F. Figueira Filho, and M.-A. Storey. Software engineering at the speed of light: How developers stay current using twitter. In *ICSE*, 2014.

[28] V. Tiwari, S. Malik, and A. Wolfe. Power analysis of embedded software: A first step towards software power minimization. *IEEE Transactions on VLSI Systems*, 2:437–445, 1994.

[29] A. E. Trefethen and J. Thiyagalingam. Energy-aware software: Challenges, opportunities and strategies. *JCS*, 2013.

[30] C. Treude, O. Barzilay, and M.-A. Storey. How do programmers ask and answer questions on the web? (nier track). In *ICSE*, 2011.

[31] N. Vijaykrishnan, M. Kandemir, S. Kim, S. Tomar, A. Sivasubramaniam, and M. J. Irwin. Energy behavior of java applications from the memory perspective. In *JVM*, 2001.

[32] S. Wang, D. Lo, and L. Jiang. An empirical study on developer interactions in stackoverflow. In *SAC*, 2013.

[33] C. Wilke, C. Piechnick, S. Richly, G. Püschel, S. Götz, and U. Assmann. Comparing mobile applications' energy consumption. In *SAC*, 2013.

[34] C. Wilke, S. Richly, S. Gotz, C. Piechnick, and U. Assmann. Energy consumption and efficiency in mobile applications: A user feedback study. In *GreenCom*, 2013.

[35] Z. Zhuang, K.-H. Kim, and J. P. Singh. Improving energy efficiency of location sensing on smartphones. In *MobiSys*, 2010.