2017 IEEE 41st Annual Computer Software and Applications Conference

What Makes Software Energy-Efficient?
Make It Faster

Hiroyuki Matsuo, Shinsuke Matsumoto and Shinji Kusumoto
Graduate School of Information Science and Technology, Osaka University, Japan
Email: {h-matsuo, shinsuke, kusumoto}@ist.osaka-u.ac.jp

***Abstract*—In order to reduce software energy consumption, a lot of studies have been carried out focusing on the difference of implementation, such as API and algorithm. However, we hypothesize that there is a strong correlation between total energy consumption of a program and duration of its execution. If this hypothesis is correct, reducing energy consumption is equal to decreasing duration. Experimental results reveal that there is a strong positive correlation between them, and its correlation coefficient is higher than 0.9. We also find that memory usage is weakly correlated with total energy consumption. As a result, we conclude that if developers want to reduce software energy consumption, they should firstly decrease duration of execution, and secondly reduce memory usage.**

***Index Terms*—energy consumption, software, program, sorting algorithm, Java Collections class.**

I. Introduction

There have been several studies for optimizing or saving energy consumption of software at an implementation level. Our fundamental question here is follows. *Do any dominant factors exist that affect program energy consumption rather than duration of program execution?* In general, every hardware device consumes energy steadily during its operation. We assume that if a program accomplishes in a shorter period of time, its energy consumption will be certainly reduced. The usage of computational hardware resources (e.g., memory, hard drive and Wi-Fi) differs depending on implementation of a program. However, we believe that a difference of the resource usage cannot be a dominant factor of energy consumption; duration affects energy consumption significantly.

If the above hypothesis is correct, an emerging challenge for program energy consumption can be simply solved by applying techniques or tools which have already been achieved in a field of program performance optimization. Furthermore, measuring energy consumption requires preparing for a special environment such as GreenMiner [1]. On the other hand, if our hypothesis is true, the duration of execution, which is extremely easy to measure, can be introduced as a substitute for the special environment. This will be a useful fact for many software developers.

In this paper, we empirically study a relationship between total energy consumption of software and its duration. The experimental objects are small programs which have the same functional features but are implemented in different ways (e.g., bubble sort and quick sort). The concrete hypotheses are as follows:

H_1 There is a significant strong correlation between total energy consumption and duration.

H_2 There is a slight correlation between total energy consumption and memory usage.

The hypothesis H_1 is equivalent to the previous question. In the hypothesis H_2, we try to confirm that energy consumption cannot be completely predicted only by duration; usage of hardware resources may affect on it. In this paper, we measure a memory usage as one of the computational resources.

II. Related Works

Many studies have been carried out on the topic of energy consumption of software at a source code level. Bunse et al. studied differences in energy consumption between major sorting algorithms [2]. Their conclusion is that insertion sort is the most energy-efficient. Hasan et al. conducted an experimental study on energy consumption of usual API operations for popular Java Collections classes [3]. They concluded that TIntArrayList is the most energy-efficient list implementation.

From a perspective of a duration of program execution and its energy consumption, their conclusions seem to be contrary to each other. While, Bunse's study mentioned that there are no direct correlations between duration and total energy consumption, Hasan's study pointed out that increase in duration (i.e., low performance) may be a degradation factor for energy consumption. We consider that this contradiction may be derived from a difference of their experimental environments. Bunse et al. measured only a CPU's energy consumption, however, Hasan et al. measured whole hardware devices embedded in an Android device. For a practical purpose, we need to conduct a replication study under the same conditions.

In addition, Hasan's study gives only a discussion of a relation between duration and energy consumption. Their focus is only on a comparative study of Java Collections classes, so empirical and quantitative results of the relation have not been reported. They also concluded more investigation is required.

Based on the above discussion, we decide to conduct an additional study based on their two experiments under the following conditions:

- Measure a total energy consumption which is consumed by not only a CPU but the whole device.
- Measure a duration of program execution as well as a total energy consumption.

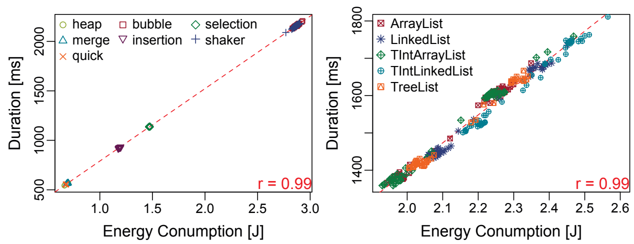
III. Experiment

A. Experiment design

We adopt two types of experimental subjects according to the related works [2] and [3]. The first type is sorting

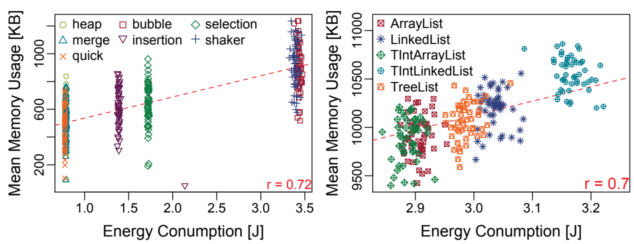
0730-3157/17 $31.00 © 2017 IEEE
DOI 10.1109/COMPSAC.2017.264

IEEE computer society



(a) Sorting algorithms (b) Java Collections classes

Fig. 1. Relation between energy consumption and duration



(a) Sorting algorithms (b) Java Collections classes

Fig. 2. Relation between energy consumption and memory usage

algorithm. This type includes seven major algorithms: heap, merge, quick, bubble, insertion, selection and shaker. The second type is Java Collections classes which implement list interface. This type also includes five Java classes: ArrayList, LinkedList, TIntArrayList, TIntLinkedList and TreeList. On both of each type, these subjects have the same functional features but have the different implementations.

The following two experiments are conducted for each experimental subjects. In the first experiment, total energy consumption and duration are measured to confirm the hypothesis H_1. We set a number of trials as 50. For sorting algorithms, 5,000 random integers are set and sorted on each trial. For Java Collections classes, we insert 5,000 integers at the middle of the list. The second experiment measures total energy consumption and memory usage to confirm the hypothesis H_2. The number of trials and program operations are the same as the first experiment. Please note that the results will contain more noise in the latter experiment because it requires an additional program execution for measuring a memory usage on a test-bed device.

Measurement infrastructure was built with reference to GreenMiner [1]. A Raspberry Pi device acts as a test-bed, and another Raspberry Pi device operates and measures the test-bed.

B. Results

The experimental results are shown in Figure 1 and Figure 2. Figure 1 shows a scatter diagram of energy consumption and duration. Figure 2 represents a scatter diagram of energy consumption and memory usage. In each diagram, the vertical axis indicates execution time [ms] and the horizontal axis indicates total energy consumption [J] or memory usage [KB], respectively. Each point shows the measured data on each trial. For example, one point in Figure 1(a) indicates a measurement result about quick sort of 20th trial. The red dash line means a regression line of each sample, and a calculated correlation coefficient r is also shown at the bottom right.

As a result, Figure 1 indicates that there is a significant correlation between total energy consumption and duration. A correlation coefficient r is greater than 0.9 on both experimental objects, so there is a strong positive correlation between them. We conclude that hypothesis H_1 is supported. This result indicates that duration of program execution can be a reliable metric to estimate its energy consumption without any preparations for special devices. In fact, one of Hasan's conclusions, the most energy-efficient list implementation is TIntArrayList, can be derived from only y-axis (duration) of Figure 1(b).

From Figure 2, memory usage has some variation compared to duration. A correlation coefficient r is about 0.7, so there is a positive correlation between them. We conclude that hypothesis H_2 is supported.

IV. Discussion and Conclusion

This paper studies a relationship between software energy consumption and duration of execution time. Our experimental results reveal that there is a strong correlation between them, so we conclude that software developers should make their software faster if they would like to reduce total energy consumption. We also found that there is a positive correlation between software energy consumption and memory usage, however, it is much weaker compared to one about duration.

These results are obtained on a Raspberry Pi, as we have mentioned. However, we think this trend can be also found on both larger computers and smaller ones. In the former case, such as on the computers with many GPUs or frequent network communications, there should be a marked tendency because such computers consume more energy in a unit of time. In the latter case, the smaller computers need less energy so we should find a weak trend on them.

As a future work, we will study a relationship between software energy consumption and other computational resources. We are also going to conduct practical experiments to confirm whether this study results are applied to general software libraries and applications.

Acknowledgment

This research was partially supported by JSPS Grant-in-Aid for Scientific Research (JP25220003, JP26730155).

References

[1] A. Hindle *et al.*, "Greenminer: A hardware based mining software repositories software energy consumption framework," in *Proc. Working Conf. Mining Softw. Repositories (MSR)*, 2014, pp. 12–21.

[2] C. Bunse *et al.*, "Choosing the "best" sorting algorithm for optimal energy consumption," in *Proc. Int'l Conf. Softw. and Data Technologies (ICSOFT)*, vol. 2, 2009, pp. 199–206.

[3] S. Hasan *et al.*, "Energy profiles of java collections classes," in *Proc. Int'l Conf. Softw. Eng. (ICSE)*, 2016, pp. 225–236.