A Preliminary Study of the Impact of Code Coverage on Software Energy Consumption

Adel Noureddine, Matias Martinez, Houssam Kanso

A Preliminary Study of the Impact of Code Coverage on Software Energy Consumption

Adel Noureddine*§, Matias Martinez†¶, and Houssam Kanso‡§
§*Universite de Pau et des Pays de l'Adour, E2S UPPA, LIUPPA*, Anglet, France
¶*Université Polytechnique Hauts-de-France, LAMIH UMR, CNRS 8201*, Valenciennes, France
*adel.noureddine@univ-pau.fr, †matias.martinez@uphf.fr, ‡houssam.kanso@univ-pau.fr

***Abstract*—Software testing plays an important role in building quality software and improving maintainability. However, there are no research studies to analyze its impact on energy efficiency. In this paper, we our hypothesis and research questions on the impact of software tests (in particular through coverage metrics) on the energy consumption of software. We also present our experimental methodology and our initial results.**

***Index Terms*—Software Energy, Software Testing, Power Consumption, Code Coverage**

I. Introduction

Software energy consumption is a major concern for software developers, practitioners [1] and architects [2]. An important issue is the lack of tools to monitor software energy, and limited knowledge in understanding the factors impacting the energy consumption of software [3]. In particular, the authors of [3] note the *lack of knowledge on how to write, maintain, and evolve energy-efficient software*. The authors also discussed the state of the art of energy-aware software testing, and found that few studies propose energy-aware software testing techniques. These techniques offer new approaches to reduce the energy consumption of test suites [4], including in Android [5], or detecting energy bugs through software tests [6].

Current approaches allow software developers to monitor the energy consumption of their devices' architectures [7], their applications [8], and within the source code [9], thus allowing to detect energy hotspots. With such tools and in-depth software energy knowledge, developers can detect and improve their software. However, the technical and psychological scalability of these approaches (such as resistance from developers to adopt new energy-aware coding behaviors, and the pressure of project deadlines and release) limits their effectiveness, as developers report a lack of proper tools and knowledge as shown in [3]. We argue that leveraging existing, more accepted, and adopted approaches in software development, to guide developers in writing energy-aware software is needed. In particular, we argue for leveraging software testing for energy efficiency. The advantages of software testing are well known in terms of improving software quality and maintainability, and reducing bugs. Previous works have previously studied the energy consumption of the testing activity (e.g., [10], [11]). In this experiment, we investigate the effects of software tests on energy efficiency. In particular, we analyze if software written with unit tests and having good code coverage (along with a few other test metrics), are more energy-efficient than software with no unit tests or low code coverage.

II. Hypothesis and Research Questions

We argue that a relation might exist between code quality and energy consumption. Although specific cases where good quality code might lead to energy inefficiency, we hypothesize that software written following the principles of testing (such as test-driven development) is more energy efficient.

Our intuition behind our hypothesis is the important impact of bugs on energy consumption as shown in multiple studies [6], [12], [13]. Untested code may contain bugs that are usually resource-consuming and can cause power-consuming problems (such as resource leak, deadlocks, and infinite loops). Therefore, we argue that writing better code will lead to more energy efficient software.

Therefore, the research question that guides our research is:

RQ: Is there a correlation between energy consumption of an application and the quality of its test suite?

To answer the research question, we set up the following hypotheses:

- Null hypothesis $H0$: there is no correlation between a metric that represents test quality and energy consumption of the application of the test.
- Alternative hypothesis $H1$: there is a correlation between a metric that represents test quality and energy consumption.

III. Proposed Methodology

To address our research question, we first need to measure the energy consumption of executing an application, and then collect the metrics related to the test cases. Finally, we analyze the correlation between energy consumption and those metrics.

A. Requirements on the evaluation dataset

We decide to study the energy consumption and test quality of a set of applications from the *same* domain, i.e., that implement the same functionalities (e.g., parse a file, compress a file, implements an arithmetic operation). The main reason for taking that decision is it allows us to fairly compare the energy across different implementations of a given functionality, and to remove the potential threats of comparing two functionally different applications with different energy

requirements. For example, an application that does more intensive CPU calculations could consume more than another one that simply reads files.

Last but not least, as we evaluate applications that implement at least one common functionality F, we can then execute all of them using the same input values I. Consequently, for each application A that we consider in this experiment, we measure the energy demanded by executing the functionality F from A, given I as input. This implies that we exercise the same functionality from each implementation using the same inputs.

B. Measuring Energy Consumption

To measure the energy consumption of an application A, we execute A given a set of input values I and measure the energy consumed between the beginning of the execution and the end. We call to those steps *test workload*.

We measure the energy consumption of the workloads using PowerJoular [1] which uses Linux kernel's Intel RAPL through the powercap interface [2]. In this preliminary study, we will focus on the energy consumed by the CPU (and ignore the energy consumed by other high-consuming components, such as the GPU). Therefore we choose libraries that only requires CPU computations.

For each test workload, we run it in a loop for 200 times and measure its energy consumption. We report the energy consumption per loop by dividing the total energy by the number of loops. This is the common strategy to mitigate the effect of external factors on energy consumption [14].

C. Collecting Test metrics

We use SonarQube[3] and JaCoCo[4] to collect test and coverage metrics for the applications under evaluation, as our initial software set is written in Java. We collected the following test and coverage metrics from SonarQube, such as branch coverage, line coverage, test errors and failures.

IV. Preliminary Results

In our study, we choose applications that focus on the *parsing of JSON files* and we measure the energy consumption of a single functionality F: parsing a JSON file from the disk and to create a representation of it in RAM memory. We selected 14 JSON libraries that were analyzed by [15].

Our preliminary study on those 14 libraries shows that there is a moderate positive correlation between Line coverage and energy consumption of test case execution, and moderate negative correlation with uncovered lines.

V. Conclusion and Future work

In this paper, we presented a method to correlate the energy consumption of software applications and their testing metrics. Our initial findings, which finds a moderate positive correlation between Line coverage and energy consumption, encourage the software engineering community to further study and confirm if a correlation exists between energy consumption and other code coverage metrics. For doing so, we aim to expand our initial study with more libraries that implement the same functionality (e.g., more libraries for parsing JSON files), more application domains (e.g., decompress files), other programming languages and platforms, and additional testing metrics.

[1] https://www.noureddine.org/research/joular/powerjoular

[2] https://www.kernel.org/doc/html/latest/power/powercap/powercap.html

[3] SonarQube: https://www.sonarqube.org/

[4] JaCoCo: https://www.eclemma.org/jacoco/

References

[1] I. Manotas, C. Bird, R. Zhang, D. Shepherd, C. Jaspan, C. Sadowski, L. Pollock, and J. Clause, "An empirical study of practitioners' perspectives on green software engineering," in *Proceedings of the 38th International Conference on Software Engineering*, ser. ICSE '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 237–248. [Online]. Available: https://doi.org/10.1145/2884781.2884810

[2] R. Bashroush, E. Woods, and A. Noureddine, "Data Center Energy Demand: What Got Us Here Won't Get Us There," {*IEEE*} *Software*, vol. 33, no. 2, pp. 18–21, 2016. [Online]. Available: https://doi.org/10.1109/MS.2016.53

[3] G. Pinto and F. Castor, "Energy efficiency: A new concern for application software developers," *Commun. ACM*, vol. 60, no. 12, p. 68–75, Nov. 2017. [Online]. Available: https://doi.org/10.1145/3154384

[4] D. Li, Y. Jin, C. Sahin, J. Clause, and W. G. Halfond, "Integrated energy-directed test suite optimization," in *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, 2014, pp. 339–350.

[5] R. Jabbarvand, A. Sadeghi, H. Bagheri, and S. Malek, "Energy-aware test-suite minimization for android apps," in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, 2016, pp. 425–436.

[6] A. Banerjee, L. K. Chong, S. Chattopadhyay, and A. Roychoudhury, "Detecting energy bugs and hotspots in mobile apps," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 588–598.

[7] M. Colmant, R. Rouvoy, M. Kurpicz, A. Sobe, P. Felber, and L. Seinturier, "The next 700 cpu power models," *Journal of Systems and Software*, vol. 144, pp. 382–396, 2018.

[8] D. Di Nucci, F. Palomba, A. Prota, A. Panichella, A. Zaidman, and A. De Lucia, "Software-based energy profiling of android apps: Simple, efficient and reliable?" in *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 2017, pp. 103–114.

[9] A. Noureddine, R. Rouvoy, and L. Seinturier, "Monitoring energy hotspots in software," *Automated Software Engineering*, vol. 22, no. 3, pp. 291–332, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10515-014-0171-1

[10] L. Cruz and R. Abreu, "On the energy footprint of mobile testing frameworks," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[11] S. Chowdhury, S. Borle, S. Romansky, and A. Hindle, "Greenscaler: Training software energy models with automatic test generation," *Empirical Softw. Engg.*, vol. 24, no. 4, p. 1649–1692, Aug. 2019. [Online]. Available: https://doi.org/10.1007/s10664-018-9640-7

[12] A. Pathak, A. Jindal, Y. C. Hu, and S. 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*, 2012, pp. 267–280.

[13] H. Jiang, H. Yang, S. Qin, Z. Su, J. Zhang, and J. Yan, "Detecting energy bugs in android apps using static analysis," in *International Conference on Formal Engineering Methods*. Springer, 2017, pp. 192–208.

[14] L. Cruz. Scientific guide to collect and analyse software energy consumption data. [Online]. Available: https://luiscruz.github.io/2021/08/20/measuring-energy.html

[15] N. Harrand, T. Durieux, D. Broman, and B. Baudry, "The behavioral diversity of java json libraries," *ArXiv*, vol. abs/2104.14323, 2021.