Towards an Engineering Discipline for Green Software *

João Paulo Fernandes[1] and João Saraiva[2]

[1] CISUC & Universidade de Coimbra, Portugal
[2] HASLab/INESC TEC & Depart. de Informática, Univ. do Minho, Portugal
jpf@dei.uc.pt , saraiva@di.uminho.pt

Abstract. This technical report describes the research developed at the Green Software Laboratory at Coimbra and Minho Universities, which was presented at the first teachers training meeting of the Erasmus+ project "Focusing Education on Composability, Comprehensibility and Correctness of Working Software". It presents both a green ranking for programming languages and data structures, and techniques to locate abnormal energy usage in software systems.

Keywords: Green Computing, Energy-aware Software, Source Code Analysis

1 Motivation

The current widespread use of non-wired but powerful computing devices, such as, smartphones, laptops, etc., is changing the way both computer manufacturers and software engineers develop their products. In fact, computer/software execution time, which was the primary goal in the last century, is no longer the only concern. Energy consumption is becoming an increasing bottleneck for both hardware and software systems. As a consequence, research on green software is a relevant and active area of research.

This report briefly describes the research that is being developed in green software in the Green Software Laboratory (GSL). GSL consists of various Portuguese research groups, including two sites of the project "Focusing Education on Composability, Comprehensibility and Correctness of Working Software". GSL is an initiative to develop techniques and tools aiming at reducing energy consumption across various computing systems (mobile, programs, databases, etc.). GSL specifically focus on the software side, where it applies (source code) analysis and transformation techniques to detect anomalies in energy consumption and to define optimizations to reduce such consumption.

* This work is financed by the ERDF European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project POCI-01-0145-FEDER-006961, and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciencia e a Tecnologia within project POCI-01-0145-FEDER-016718 and UID/EEA/50014/2013.

In the last century efficiency of a software system was mainly focused on execution time and memory consumption efficiency. Nowadays, software developers often ask the question "is a faster program also a greener program?". There are many aspects of a software system that influences its energy performance: the programming language and its execution model (compiled to binary code or to a virtual machine, interpreted code, lazy versus strict evaluation, use of runtime partial evaluation, etc). The efficiency of the memory model and language libraries also influence performance. The complexity of the algorithm used to implement the desired computer problem, also influences performance: if the implemented algorithm has to do more work than what is strictly needed, then, more CPU and energy will be used.

In this document we briefly report the research results achieved in the GSL, namely in analyzing the energy efficiency of programming languages (Section 2), data structure libraries (Section 3), and of software's source code (Section 4).

2 Greenness in Programming Languages

An interesting question that arises when discussing energy in programming languages is whether a faster language is also an energy efficient language, or not. Comparing software languages, however, is an extremely complex task, since the performance of a language is influenced by the quality of its compiler, virtual machine, garbage collector, etc.

In the Green Software Laboratory we studied, assessed and compared the performance of (a total of) 27 of the most widely used software languages. We used two different computer problem repositories: *Computer Language Benchmark Game* (CLBG)[3] and the *Rosetta Code*[4] repositories [1–3]. Both repositories define a set of computer tasks and provide implementations in a large group of programming languages. While CLBG was tailored to analyze execution time performance of languages, *Rosetta Code* was defined with more program comprehension purposes.

We compiled/executed such programs using the state-of-the-art compilers, virtual machines, interpreters, and libraries for each language. Then, we monitored the execution time, peak and overall memory consumption, and CPU/-DRAM/GPU energy consumption. We produced a energy ranking of the 27 languages and we also analyzed those results according to the languages' execution type (compiled, virtual machine and interpreted), and programming paradigm (imperative, functional, object oriented, scripting) used. For each of the execution types and programming paradigms, we compiled a software language ranking according to each objective individually considered (e.g., time or energy consumption). Our first experiments show expected results, like the C language being both the faster and greener language, however, it also show slower languages that are more energy efficient than others [2, 3].

[3] `http://benchmarksgame.alioth.debian.org/`
[4] `http://www.rosetta.org`

3 Greenness in Data Structures

Programming language/paradigm, and its powerful compiler optimizations, is not the only aspect that influences the energy consumption of a software system. In fact, a program may also become more efficient by "just" optimizing its libraries [4, 5]. Most languages offer powerful libraries to manipulate data structures. In GSL we studied the energy performance of two advanced data structures widely used in the Java and Haskell programming languages.

In Java, we conducted a detailed study in terms of energy consumption of the *Java Collections Framework* (JCF) library [5]. We considered the usual three different groups of data structures, namely Sets, Lists, and Maps, and for each of these groups, we studied the energy consumption of each of its different implementations and methods [4]. This JCF energy-awareness can not only be used to steer software developers in writing greener Java software, but also in optimizing legacy Java code. We have developed a Java data structure refactoring tool, named *jStanley*, which refactors Java source code when a greener collection is available [6]. We have also executed an initial evaluation with 7 publicly available Java projects where we were able to improve the energy consumption between 2% and 17%.

In Haskell, we studied the energy consumption of *Edison*[6], a fully mature and well documented library of purely functional data structures [7]. Edison provides different functional data structures for implementing three types of abstractions: *Sequences* (lists, queues and staks), *Collections* (sets and heaps) and *Associative Collections* (maps and finite relations). We analyzed 16 implementations of such data structures while measuring detailed energy and time metrics [5]. We further investigated the energy consumption impact of using different compilation optimizations. We have concluded that energy consumption is directly proportional to execution time and that the energy consumption of DRAM representing between 15 and 31% of the total energy consumption. Finally, we also concluded that optimizations can have both positive or negative impact on energy consumption.

4 Greenness in Source Code

Not only languages and data structure libraries do influence energy consumption, algorithms and programming practices also play a key role on the efficiency of programs. In GSL we have adapted well-know fault localization techniques to statically locate "energy leaks" (seen as energy inefficiency, thus, energy faults) in the source code of applications [8–11]. We defined SPELL - *SPectrum-based Energy Leak Localization* to determine red (energy inefficient) areas in software. A first experimental study shows that expert programmers, with access to the energy leaks detects by SPELL, were able to better optimize the energy consumption of the programs (between 15% and 74%), than experts with no information

[5] `docs.oracle.com/javase/7/docs/technotes/guides/collections/index.html`
[6] `hackage.haskell.org/package/EdisonAPI-1.3/docs/Data-Edison.html`

or the information provided by a standard programs (runtime) profiler. We have also studied the energy behaviour of C/C++ programs [12].

The widespread use of non-wired devices and the advent of the internet-of-things, is changing the way software engineers develop their software. Software has to run on a variety of mobile devices and energy consumption is a main concern when developing software. Software Product Lines (SPL) have emerged as an important software engineering discipline allowing the development of software that shares a common set of *features*. In GSL we have defined static analysis techniques to reason about energy consumption in SPLs based on conditional compilation. Such techniques allow software developers to identify (non) green *products* and/or *features* in a SPL [13].

Android is a widely used ecosystem for non-wired devices, and software energy analysis and optimization is an active area of research. The GSL team has developed several techniques [14, 15] and tools to analyze and optimize energy consumption in the source code of Android applications [16, 17].

Nowadays, most of the data stored in our mobile devices (files, photos, videos) is also stored in the cloud provided by the ecosystem of the device's operating system. Such cloud systems are data centers that daily run a large amount of data querying processes, monitored and controlled by highly sophisticated database management systems, which are responsible to establish efficient query processing plans to support them. Database systems usually rely on plans that optimize response time. We designed and developed an alternative method to define energy consumption plans for database queries [18, 19]. Our first experimental results show that the use of optimization heuristics allows for significant gains, both in terms of energy consumption and the time spent with the execution of queries.

5 Conclusions

This technical report described the research developed at the Green Software Laboratory, namely a green ranking of programming languages and data structures, techniques to detect energy inefficiency in a software system's source code, and an energy-aware query execution plan for database systems.

Acknowledgment

This paper is part of the Intellectual Output O2 of the Erasmus+ Key Action 2 (Strategic partnership for higher education) project No. 2017-1-SK01-KA203-035402: "Focusing Education on Composability, Comprehensibility and Correctness of Working Software". The information and views set out in this paper are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

References

1. Couto, M., Pereira, R., Ribeiro, F., Rua, R., Saraiva, J.: Towards a green ranking for programming languages. In: Proceedings of the 21st Brazilian Symposium on Programming Languages. SBLP (2017) 7:1–7:8 (best paper award).
2. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Energy efficiency across programming languages: How do energy, time, and memory relate? In: Proc. of the 10th ACM SIGPLAN Int. Conference on Software Language Engineering. SLE 2017, New York, NY, USA, ACM (2017) 256–267
3. Pereira, R., Couto, M., Ribeiro, F., Rua, R., Cunha, J., Fernandes, J.P., Saraiva, J.: Ranking programming languages by energy efficiency. Science of Computer Programming (2018) Submitted.
4. Pereira, R., Couto, M., Saraiva, J., Cunha, J., Fernandes, J.P.: The Influence of the Java Collection Framework on Overall Energy Consumption. In: 5th Int. Workshop on Green and Sustainable Software. GREENS '16, ACM (2016) 15–21
5. Melfe, G., Fonseca, A., Fernandes, J.P.: Helping developers write energy efficient haskell through a data-structure evaluation. In: Proceedings of the 6th International Workshop on Green and Sustainable Software. GREENS '18, New York, NY, USA, ACM (2018) 9–15
6. Pereira, R., Simão, P., Cunha, J., Saraiva, J.: jStanley: Placing a Green Thumb on Java Collections. In: 33rd ACM/IEEE International Conference on Automated Software Engineering. ASE 2018, New York, NY, USA, ACM (2018) 856–859
7. Lima, L.G., Melfe, G., Soares-Neto, F., Lieuthier, P., Fernandes, J.P., Castor, F.: Haskell in Green Land: Analyzing the Energy Behavior of a Purely Functional Language. In: Proc. of the 23rd IEEE Int. Conf. on Software Analysis, Evolution, and Reengineering (SANER'2016), IEEE (2016) 517–528
8. Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J.: Helping programmers improve the energy efficiency of source code. In: Proc. of the 39th Int. Conf. on Soft. Eng. Companion, ACM (2017)
9. Pereira, R.: Locating energy hotspots in source code. In: Proceedings of the 39th International Conference on Software Engineering Companion. ICSE-C '17, Piscataway, NJ, USA, IEEE Press (2017) 88–90 (ACM SRC silver award).
10. Pereira, R.: Energyware Engineering: Techniques and Tools for Green Software Development. PhD thesis, Depart. de Informática, Universidade do Minho (2018)
11. Pereira, R., Carção, T., Couto, M., Cunha, J., Fernandes, J.P., Saraiva, J.: Spelling out energy leaks: Aiding developers locate energy inefficient code. (2018) (submitted).
12. Santos, M., Saraiva, J., Porkolb, Z., Krupp, D.: Energy consumption measurement of c/c++ programs using clang tooling. SQAMIA'17 - CEUR Workshop Proceedings **1938** (2017)
13. Couto, M., Borba, P., Cunha, J., Fernandes, J.P., Pereira, R., Saraiva, J.: Products go green: Worst-case energy consumption in software product lines. In: Proceedings of the 21st International Systems and Software Product Line Conference - Volume A. SPLC '17, ACM (2017) 84–93
14. Couto, M., Carção, T., Cunha, J., Fernandes, J.P., Saraiva, J.: Detecting anomalous energy consumption in android applications. In Quintão Pereira, F.M., ed.: Programming Languages: 18th Brazilian Symposium, SBLP 2014, Maceio, Brazil, October 2-3, 2014. Proceedings. (2014) 77–91
15. Cruz, L., Abreu, R.: Performance-based guidelines for energy efficient mobile applications. In: 4th International Conference on Mobile Software Engineering and Systems. MOBILESoft '17, Piscataway, NJ, USA, IEEE Press (2017) 46–57

16. Couto, M., Cunha, J., Fernandes, J.P., Pereira, R., Saraiva, J.: Greendroid: A tool for analysing power consumption in the android ecosystem. In: 2015 IEEE 13th International Scientific Conference on Informatics. (Nov 2015) 73–78
17. Cruz, L., Abreu, R., Rouvignac, J.N.: Leafactor: Improving energy efficiency of android apps via automatic refactoring. In: IEEE/ACM International Conference on Mobile Software Engineering and Systems, MobileSoft 2017. (2017)
18. Gonçalves, R., Saraiva, J., Belo, O.: Defining energy consumption plans for data querying processes. In: 2014 IEEE International Conference on Big Data and Cloud Computing (BdCloud)(BDCLOUD). Volume 00. (Dec. 2015) 641–647
19. Belo, O., Gonçalves, R., Saraiva, J.: Establishing energy consumption plans for green star-queries in data warehousing systems. In: 2015 IEEE International Conference on Data Science and Data Intensive Systems. (Dec 2015) 226–231