Towards a Green and Sustainable Software

Hayri Acar, Gülfem Isiklar Alptekin, Jean-Patrick Gelas, Parisa Ghodous

Towards a Green and Sustainable Software

Hayri ACAR [a], Gülfem I. ALPTEKIN [c], Jean-Patrick GELAS [b], Parisa GHODOUS [a]
[a] *University of Lyon, LIRIS, France*
[b] *ENS Lyon, LIP, UMR 5668, France*
[c] *Galatasaray University, Turkey*

Abstract. Information and Communication Technologies (ICTs) are responsible around 2% of worldwide greenhouse gas emissions [1]. On the other hand, the use of mobile devices (smartphone, tablet, etc.) is continually increasing. Due to the accessibility of the Internet and the cloud computing, users will use more and more software applications which will cause even an increasing effect on gas emission. Thus, an important research question is "how can we reduce or limit the energy consumption related to ICT and, in particular, related to software?" For a long time, proposed solutions focused only on the hardware design, however in recent years the software aspects have also become important. Our first objective is to compare the studies in the research area of energy efficient/green software. Relying on this survey, we will propose a methodology to measure the energy consumed by software at runtime.

Keywords. Green Software, Green IT, Sustainable Software, Energy Efficiency.

1. Introduction

The availability of various services (i.e. eBank, eHospital) through the cloud has facilitated daily lives. It allows to make energy and money savings by preventing people from moving to accomplish a small task (for instance see his account at the bank). Furthermore, the availability of these services through mobile devices and their widely usage has a positive impact on energy saving. It is also worthwhile to consider technology addicts developing/using applications or software when estimating the growing impact of software on energy consumption.

The emission of greenhouse gases is being reduced thanks to technological progress. However, the increasing number of applications' users causes additional consumption. Therefore, in order to get a better efficiency developers needs to be guided to optimize their development to establish green software.

In this paper, we've made a state of the art for these research questions by summarizing related works in this field and then we compare them.

We aim at establishing an estimation model for the consumed energy. We then investigate its performance and accuracy on a development project. The model will be used as an energy consumption measurement tool that guides developers building greener software.

2. Related work

The hardware methods to measure energy consumption, in most cases, are based on measurement instruments such as power meter or printed circuits. Thus, it is impossible to measure virtual machines whose usage is becoming more widespread. In addition, the usage of these materials causes energy production and thus additional cost, which is not preferred in creating a measurement model.

On the other hand, software tools are based on computer models for energy consumption in order to provide with an estimation. The lack of accuracy and comprehensiveness can cause incorrect and unsatisfactory results because simplifications adopted in estimating the energy consumption for a specific area, will not be valid in another area. Therefore, when using such a tool, it is necessary to be more precise, by taking into account all components of a computing device, such as a PC, tablet, smartphone or server, that are likely to consume energy.

With these ideas in mind, we made a list of energy measurement tools that have been proposed in recent years (Table 1).

2.1. Joulemeter

The energy consumption of a virtual machine, a computer or software is estimated by Joulemeter which measures hardware resources (CPU, disk, memory, screen, etc.) that are used [2]. The tool makes an auto calibration by getting back the values of the power consumed in the idle state, with the maximal and minimal frequency and by the power of the monitor. These values can be manually seized. The calculations are then made by using the values of these parameters.
The energy consumption of the main components and the total power which is supposed to be consumed by the device are visualized.
The tool also measures the consumption of a very precise process by allowing to seize the name of this one which can be found in the task manager processes tab. Thus, in real time, the variation of the power due only to the CPU can be observed for this given process. It is possible to register in a file the power consumed by this particular process.

This tool only allows estimating the energy consumption of a process.

2.2. vEC

Virtual Energy Counters (VEC) estimates the energy consumption of a given process [3].

The main components such as cache, main memory, and buses are considered to provide a quick estimate of the energy consumption. The tool is built on top of the Library of the Perfmon user for the UltraSPARC platform, and authors argue that is easily extendable to other platforms.

2.3. Orion

Orion is also an estimation tool of energy consumed by an application [4]. This one, compared to other tools, takes into account the communication components except the processor and memory that have been neglected in many cases.

For various architectural components of on-chip networks, this tool is a suite of dynamic and leakage power models developed in order to enable rapid power-performance tradeoffs at the architectural level.

2.4. Span

Span is used to provide with live, real-time power information phases of running applications [5]. According to a power model, this approach aims to help developers and to perform synchronization between power dissipation and the source code. Furthermore, tool is a result of external API calls to correlate power estimation with the source code of the application.

This work is different from others because the author has studied the energy consumption at the source code. Unfortunately, developers must instrument manually the code.

2.5. PowerAPI

The energy consumption of the processes in real time is estimated by POWERAPI using information collected by the hardware (CPU and network) through the operating system [6].

The tool is used to estimate the power consumption of each running application based on their Ids. The tool is limited to measure the power consumption of the CPU and network card without taking into account disk and memory.

2.6. Other energy estimation tools

The area of study is new and estimation of energy consumption tools with software methods are limited. Moreover, existing tools only measure the energy consumption due to a program without providing specific details. Moreover, most often some components are neglected during the measurements.

Other tools:

- Framework in order to reduce power consumption proposed by P.K. Gupta and G. Singh [7].
- Wattch, a simulator that estimates CPU power consumption [8].
- GREENSOFT is a method to measure power consumption taking account a hardware part with a power meter and a software part with a data aggregator and evaluator in order to provide a report [9].

Table 1. Energy consumption measurement software tools

Tools	Power model	Acronyms	Appreciations
Joulemeter	$E = Ecpu + Ememory + Edisk$	$Ecpu$, $Ememory$ & $Edisk$: CPU, memory & disk energy usage.	Just estimates the energy consumption of an application.
vEC	$E = Ebus + Ecell + Epad + Emain$	$Ebus$: data and address bus energy between processor and cache, $Ecell$: cache energy, $Epad$: data and address pad energy between cache and main memory, $Emain$: the main memory energy.	Limited to only estimates the power consumption due to memory.

Orion	$E = Eread + Ewrite$	$Eread$: read energy, $Ewrite$: write energy.	Communication components are considered on this tool.
Span	$P(t_j, f_i)_{pret} = \Delta P(t_j, f_i)_{pret} + P(f_i)$	P: power dissipation, f: CPU frequency, t: training benchmarks.	Manually code added in the software code to show the parts of code involved on the energy consumption.
PowerAPI	$Psoftware = Pcomp + Pcom$	$Pcomp$: CPU power consumed by software, $Pcomp$: power consumed by network card for transmitting software's data.	Only CPU and network have been considered on this power consumption tool.

The previous table (Table 1) represents, for each tool, the power model used in the estimation of energy consumption and shows the limits about the accuracy due to incompleteness. So, in the next section, an improved tool based on a power model taking account all components will be defined.

3. Proposed Software Model

3.1. Green process

All development processes of a computer program requires following a specific sequence in order to complete the project. In addition, after each phase, a green analysis step can be involved in order to check if the considered step has respected all criteria that will allow reducing energy consumption. If the criteria of a phase are not validated by the green analysis, depending uncommitted specifications, a return to the previous step or even return until the requirement analysis step can be performed.

The process described in the work [10] gets comprehensively the progress of a development project.

Thus, we will offer our descriptive diagram in Figure 1.

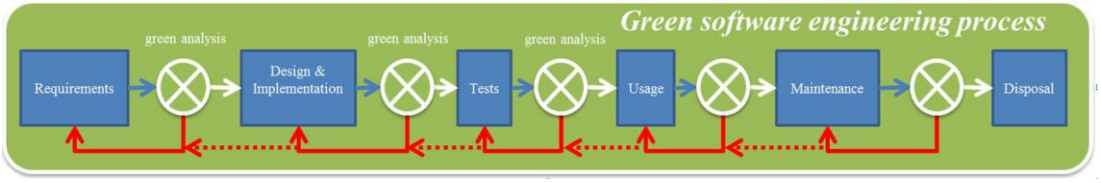

Figure 1. Green software engineering process.

- Requirements: First step in order to build a software product. This stage corresponds to the descriptions of the tasks that will be performed by the product. The aim is to meet customer demands.
- Design: The defined requirements are considered in order to create system architecture. The classes and the relationships among them are defined at this stage.

Implementation: In this step, the program is implemented in respect to its design. Developers should choose the most appropriate programming language.

- Tests: This step allows checking if the software meets its requirements, to discover faults or defects. The tests will be defined at the end of requirements phase (QCHP) before design and implementation step, to show that the specifications have been understood. Use of different tester will allow developers to see if the requirements are correct and consistent.

  The energy consumption measurement tool will be used in order to know whether the program can be improved.

- Usage: This step defines how the software product can be used by the user in a green manner. The responsibility belongs to the user but also to the engineers themselves. The user should trained to use the software, because the fact that improper handling can cause errors in the program.

- Maintenance: Newer versions or enhancements usually involve changes. The developers need to handle them. Furthermore, developers need to know the cost is proportional to the energy waste. Several types of errors in the program can cause to return to the implementation phase, but sometimes even more complicated errors can cause the developer to return to the first step of requirement analysis. The maintenance process must be carried out in the most energy efficient manner.

- Disposal: Software and hardware must be replaced when it is not profitable up to date, when it is no longer used, or when it has become obsolete. This step considers both the software and the hardware running the code. Disposal of old hardware also causes energy consumption.

- Green analysis: This step can be added at the end of each one in order to improve energy efficiency. This stage will evaluate the greenness of the software.

3.2. Power model

Each estimation tool of energy consumption is based on a power model that takes into consideration different electronic components depending on its area of operations.

Thus, in our case we establish a power model that takes into account all the components of the device, even if its consumption is negligible so that our tool can be a generic one. If the component does not exist in a particular case, then its consumption will be considered at zero.

Moreover, we establish formulas based on parameters determined by the provider to facilitate the calculations.

The power consumed by the software can be separated in two parts: static and dynamic, as given in Eq. (1):

$$P_{Software} = P_{Software,dynamic} + P_{Software,static} \quad (1)$$

The consumed dynamic power can be expressed as follows:

$$P_{Softwre,dynamic} = P_{CPU,dynamic} + P_{CPU,static} + P_{Memory,dynamic} + P_{Memory,static} + P_{Disk,dynamic} + P_{Disk,static} + P_{Network,dynamic} + P_{Network,static} \quad (2)$$

Integrating Eq. (2) into Eq. (1), we obtain:

$$P_{Software} = P_{CPU,dynamic} + P_{CPU,static} + P_{Memory,dynamic} + P_{Memory,static} + P_{Disk,dynamic} + P_{Disk,static} + P_{Network,dynamic} + P_{Network,static} + P_{Software,static} \quad (3)$$

According to Eq. (3), separating static and dynamic power, we deduce following equations:

$$P_{static} = P_{CPU,static} + P_{Memory,static} + P_{Disk,static} + P_{Network,static} + P_{Software,static} \quad (4)$$

$$P_{dynamic} = P_{CPU,dynamic} + P_{Memory,dynamic} + P_{Disk,dynamic} + P_{Network,dynamic} \quad (5)$$

As a result, we can redefine Eq. (1) like:

$$P_{Software} = P_{dynamic} + P_{static} \quad (6)$$

In our case, we cannot improve the static power due to the material components of devices produced by the manufacturer. Thus, we are interested only in the dynamic power consumed by software. So, we establish a power measurement formula to each component.

3.2.1. CPU

CPU power consumption depends on several factors. This power is approximately proportional to the CPU frequency, and to the square of the CPU voltage:

$$P_{CPU} = C \times V^2 \times F \quad (7)$$

where C represents a constant depending of Capacitance, V is the voltage and F is the frequency.

However, we only want to define the power consumed by the program. So, the usage percent of the process Id is determined and it is multiplied with the total consumed power:

$$P_{CPU,id} = P_{CPU} \times \frac{N_{id}}{100} \quad (8)$$

where N correspond to the CPU usage of the software.

3.2.2. Other components

For this preliminary study, the observations are limited to the power consumed by the CPU, but our energy measurement tool will be used for other components quickly and easily, in a near future.

3.3. TEEC (Tool to Estimate Energy Consumption): Design & Implementation

According to [6], Java programming language is stated as the language with the least energy consumption during compilation and execution stages. Thus, Java is chosen as the development language.

Sigar library [11] allows getting information about the CPU usage, including the percentage of usage of each process and the number of cores used. Thus, the id of the ongoing process can be identified and retrieved.

Moreover, the form of global variable data providers is formed that allows estimating the energy and assigning a corresponding value.

Java Agents are also utilized, which are the software components that provide with the instrumentation capabilities to an application, such as re-defining the content of class that is loaded at run-time.

Coding a Java Agent requires writing a Java class that has the premain() method with the following signature:

public static void premain(String args, Instrumentation inst);

The manifest file "MANIFEST.MF" has to contain at least:

Manifest-Version: 1.0
Premain-Class: package.Agent

To run the agent, the following command is used:

java –javaagent:Agent.jar –cp folder/sigar.jar package.MainApplication

The model can be illustrated as in Figure 2.

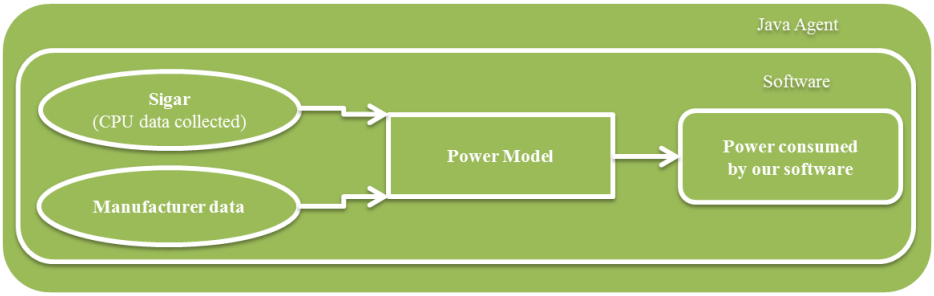

Figure 2. Operation of the proposed power model.

4. Experiments

First, the proposed tool is tested with a program that requires a lot of calculation, and therefore heavy use of CPU. As the proposed power tool, only measures the power consumed by the CPU, the measurement is more precise and accurate.

The Fibonacci sequence is implemented which corresponds to a sequence of integers in which each term is the sum of the two preceding terms.

The information that is obtained with the Sigar library on our machine is given in Figure 3:

```
CPU
Vendor: AMD
Clock: 1995Mhz
Model: Phenom(tm) II N930 Quad-Core Processor
CacheSize: -1
CorePerSocket: 4
TotalCores: 4
TotalSockets: 1
```

Figure 3. CPU information obtained with Sigar.

Furthermore, the task manager is seen before and after the execution of the program to demonstrate that only the CPU is impacted.

The usage of CPU is observed to increase from a few percent to thirty percent, and it stays around these levels until the end of program execution and returns back to a few percent.

With the proposed power model tool TEEC, the power consumption of Fibonacci sequence using recursive method and iterative method are estimated. The generated test calculates the first 45 values of the Fibonacci sequence with recursive method. For the iterative method, the calculations for the first 5000 value are performed.

The results are represented in Figure 4:

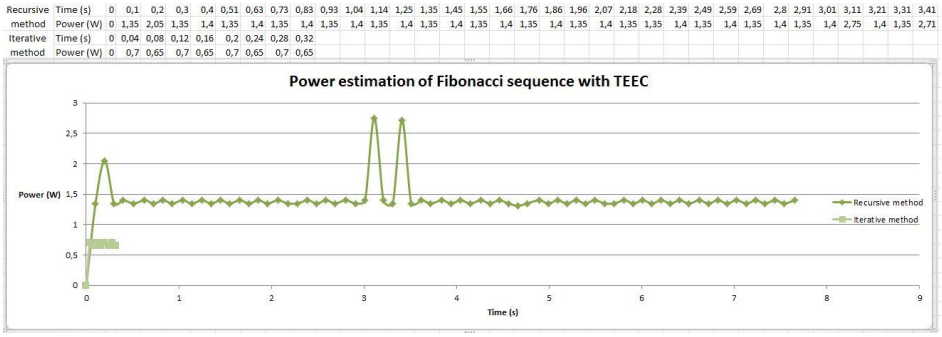

Figure 4. Power consumption of Fibonacci sequence with TEEC.

The results are compared to the results of Joulemeter application for a particular process with its Id and name (Figure 5).

Recursive method	Time (s)	0	0,077	0,154	0,231	0,308	0,385	0,462	0,539	0,616	0,693	0,77	0,847	0,924	1,001	1,078	1,155	1,232	1,309	1,386	1,463	1,54	1,617	1,694	1,771	1,848	1,925	2,002	2,079	2,156
	Power (W)	0	1,2	1,4	1,5	1,4	1,3	1,5	2,2	2,2	1,3	1,4	1,4	1,5	1,3	1,4	1,3	1,4	1,4	1,4	1,4	1,4	1,5	1,3	1,4	1,3	1,4	1,3	1,4	1,4
Iterative method	Time (s)	0	0,04	0,08	0,12	0,16	0,2	0,24	0,28	0,32																				
	Power (W)	0	0,8	0,8	0,8	0,5	0,6	0,8	0,5	0,5																				

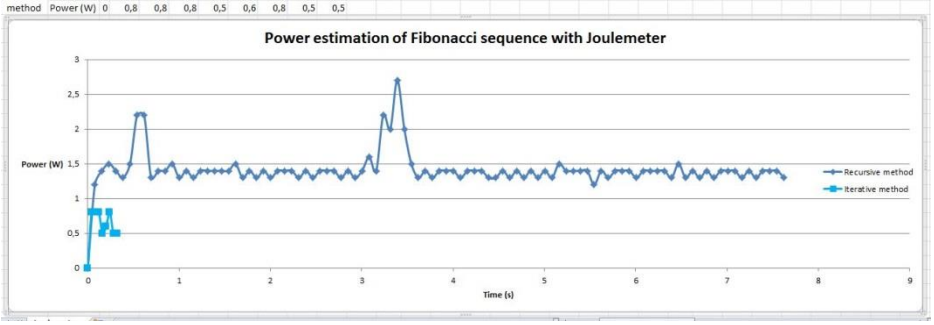

Figure 5. Power consumption of Fibonacci sequence with Joulemeter.

First, it is observable that quite similar results are obtained for the running application. It shows the effectiveness of the proposed tool and computational model.

Moreover, the results reveal that the iterative method is quicker and consumes less power than the recursive method.

As a future work, the measures will be validated on other applications to demonstrate the precision and accuracy of the proposed model.

5. Conclusion and Perspectives

The contribution to power measurement literature will continue by bringing improvement to the estimation of the consumption of other components; such as memory, disk and network, which are neglected in related models in literature. It will allow us to have a higher accuracy in estimating the energy consumption of a program.

Using Java agents, the methods will be re-implemented automatically in order to observe their energy consumption. We will seek to be more precise in locating the most intensive pieces of code in each function to help developers optimize their codes.

The similar energy estimation tools in literature are analyzed in the paper. The research area of green software development is relatively new, and major part of the tools only provides with an estimation of the energy consumption of an application without involving the source code. Moreover, the recent tools, which have began to take into account the source code, do not take into account all the components that consume energy and / or request to integrate the code manually. Hence, there is a lack of precision and a difficulty of using these tools.

After this state of the art, an energy consumption estimation tool is proposed. It has been implemented so as to measure only the consumption due to the CPU, but it may be used for other components quickly and easily, in the future studies.

The proposed tool is expected to be improved, and it is planned to dynamically identifying the locations of the head of the largest energy consumer code. This will allow developers to optimize their own codes to obtain greener software.

References

[1] Gartner, Green IT: The New Industry Shock Wave, Gartner, Presentation at Symposium/ITXPO Conference, 2007.

[2] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka Bhattacharya, Virtual Machine Power Metering and Provisioning, ACM Symposium on Cloud Computing (SOCC), 2010.

[3] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykrishnan, M.J. Irwin, and A. Sivasubramaniam, vEC: Virtual Energy Counters, ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering, 2001, 28-31.

[4] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, Sharad Malik, Orion: A Power-Performance Simulator for Interconnection Networks, 35th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-35), 2002.

[5] Shinan Wang, Hui Chen, Weisong Shi, SPAN: A software power analyzer for multicore computer systems, Sustainable Computing: Informatics and Systems,Volume 1, Issue 1, March 2011, 23–34.

[6] Adel Noureddine, Aur_elien Bourdon, Romain Rouvoy, Lionel Seinturier, A Preliminary Study of the Impact of Software Engineering on GreenIT. First International Workshop on Green and Sustainable Software, Jun 2012, Zurich, Switzerland. 21-27.

[7] P.K. Gupta and G. Singh , A Framework of Creating Intelligent Power Profiles in Operating Systems to Minimize Power Consumption and Greenhouse Effect Caused by Computer Systems, Journal of Green Engineering (2011), 145–163.

[8] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, The Design and Use of SimplePower: A cycle-Accurate Energy Estimation Tool, Design Automation Conference, 2000.

[9] Eva Kern1, Markus Dick2, Stefan Naumann1, Achim Guldner1, Timo Johann2, Green software and green software engineering–definitions, measurements, and quality aspects, ICT4S 2013: Proceedings of the First International Conference on Information and Communication Technologies for Sustainability, Zurich, February 14-16, 2013.

[10] Sara S. Mahmoud and Imtiaz Ahmad, A Green Model for Sustainable Software Engineering, International Journal of Software Engineering and Its Applications Vol. 7, No. 4, July, 2013.

[11] Ryan Morgan and Doug MacEachern ,https://support.hyperic.com/display/SIGAR/Home, 2010.