# TEEC: Improving power consumption estimation of software

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

# TEEC: Improving power consumption estimation of software

Hayri Acar[1], Gülfem I. Alptekin[2], Jean-Patrick Gelas[3], Parisa Ghodous[4]

## 1. Introduction

Recently, researchers have begun to give importance at the energy consumed by software. But, to solve this problem, they propose often a hardware study of different devices. For this, they used hardware devices like powermeter or printed circuit. The main advantage of this methodology is the fact that we can obtain accurate results because we measure the energy consumed by components. But, we are limited at the fact that we can't measure the power consumed by VM (Virtual Machine) and the cost of this process itself can be expansive. Estimating power consumption of software has begun a popular research field. Several tools have been presented in academic literature, however, these tools have the capacity to estimate only specific component's consumption.

ICT (Information and Communications Technologies) constitutes 2% of such gas emissions, and it is projected an increase to 4% by 2020, if nothing is done [1]. In fact, recent trends such as cloud computing and internet of things even increase the number of devices, and consequently the software running on them. Hence, software became a fundamental actor of efficiency plans that aims at reducing greenhouse gas emission.

In this paper, we propose a tool, called TEEC (Tool to Estimate Energy Consumption), in order to estimate the power consumption of a given software at runtime by taking into account CPU, memory and disk power consumptions. Using TEEC, we expect to be able to obtain software/applications having some functionality and consuming less power.

## 2. Components Modelization

### 2.1 Related Work

Similar to our recent study [2], there are several tools that are able to estimate the power consumption using a program:

- Joulemeter [3] which provides components' power usage for all processes. It also estimates power consumed by CPU for a given process.
- Span [4] gives power information about running programs. With specific functions, developers can manually instrument source code.

1 LIRIS, University of Lyon 1, Lyon France, hayri.acar@univ-lyon1.fr
2 Galatasaray University, Istanbul Turkey, gisiklar@gsu.edu.tr
3 ENS Lyon, LIP, UMR 5668, Lyon France, jean-patrick.gelas@univ-lyon1.fr
4 LIRIS, University of Lyon 1, Lyon France, parisa.ghodous@univ-lyon1.fr

- GREENSOFT [5] is a hybrid solution, which takes into account hardware devices and software part during the power estimation.

For a power estimation to be accurate, it is required that all components need to be taken into account during the software runtime. This estimation will guide software developer to improve their code.

Thus, we propose a detailed study of the components: CPU, memory and hard disk in order to provide an accurate modelization. Accordingly, we use mathematical formula for measuring energy consumption for each component.

## 2.2 CPU

For a long time the CPU was considered the largest energy consumer component [6] in a computer. That is the reason why in most of the works, the modelization has taken into account only the CPU to estimate the energy consumed by computer program. Several models and factors have been proposed for CPU power consumption, but in this study, we will consider the following equation (1):

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

where $P_{CPU,dynamic}$ represents dynamic power consumption, while $P_{CPU,static}$ corresponds to static power consumption. Only the manufacturer can reduce the static power consumption because it depends on the architectural characteristics of each component. In our case, we want to reduce the energy consumed by software. Therefore, we only consider the dynamic power consumption to get more accurate results. The CPU, like many integrated circuit, is a set of gates. Hence, the main power consumption in CPU is due to capacitors charge and discharge during computations. We assume that in a switching cycle, there are low-to-high and high-to-low transitions. For N gates, the power need to be multiplied by N. Hence, we can define a parameter $\alpha < 1$ as the average fraction of gates that commute at each cycle. To determine the power consumed by the program, we multiple by the percentage of the process id $N_{id}$. The equation of the power becomes (2):

$$P_{CPU,dynamic} = C.f.V_{dd}^2 \quad (2)$$

where $C = C_L.N.\alpha$, $V_{dd}$ is the voltage and f is the frequency.

## 2.3 Memory

According to [7], the power used on servers is increasing and two largest consumers of power are the processors and RAM chips. Especially because memory RAM size increase a lot this last decade. Several works have the objective to optimize systems to reduce DRAM power consumption [8, 9, and 10].

In this work, we decide to study the DRAM in order to model its power consumption. Doing so, we use datasheet values from DRAM manufacturer to build the power consumption equation. Similar to CPU, we only interest in the dynamic power, since it is the only part for energy saving. Based on Micron [11], we assume that the dynamic power is composed of: Activate, Precharge, Read and Write power.

Accordingly, total power consumption of DRAM can be give as (3):

$$P_{DRAM,id} = \left(P_{Activate} + P_{Precharge} + P_{Read} + P_{Write}\right).M_{id} \quad (3)$$

where $M_{id}$ is the usage percent of the process id.

### 2.4 Hard disk

According to the International Data Corporation (IDC), storage capacity increases each year with a rate of 60% [12].

There are several tools proposed to test the performance of a hard drive. The most known are CristalDiskMark and HD Tune [13]. However, these programs allow only measuring read/write speeds. They do not give information about power consumption. That is the reason why researchers tried to take into account power consumption due to hard disk during the runtime. Some of the tools that are built for this purpose are: DiskSim [14], Tempo [15] and SODA [16].

In an Hard Disk Drive, there are four power management states are supported: active, idle, standby and sleep. Hard disk is also composed of dynamic (4) and static power:

$$P_{Disk,dynamic} = P_{Active} = P_{Read} + P_{Write} \quad (4)$$

## 3. Experiments

In order to realize our experiments, we develop a tool called TEEC. This tool has been developed in Java which is one of informatic language most used. We use Sigar library [17] to automatically obtain information about CPU, memory and disk. Besides, static manufacturer data are saved and used. So, we can provide an estimation of power consumption due to each component during runtime of software.

We perform our test on a notebook ASUS N751JK-T7238H, running Windows 8. Different tests have been executed with unoptimized and optimized code source in order to see the impact on each component.

Loops have an important effect on the performance of a program and provide efficient way for repeating a piece of code as many times as required. Java has three types of loop control structures which are: for, while and do-while. So, it is interesting to study some functions, as represented in Table 1, that are used during a development of a program in order to examine possible optimizations.

| **Unoptimized** | **Optimized** |
|---|---|
| *Locality of reference* | |
| for(int i=0;i<1000000;i++){<br>int sum = 0;<br>for(int x=0;x<50000;x+=100){<br>sum += values[x]; }} | for(int i=0;i<1000000;i++){<br>int sum = 0;<br>for(int x=0;x<500;x++){<br>sum += values[x]; }} |
| *Compare array to array list* | |
| for(int i=0;i<1000000;i++){<br>int sum = 0;<br>for(int v=0;v<list.size();v++) | for(int i=0;i<1000000;i++){<br>int sum = 0;<br>for(int v=0;v<array.length;v++) |

| | |
|---|---|
| sum += list.get(v);} | sum += array[v];} |
| *Compare integer list loop* | |
| for(Integer i:list) count++; | int size=list.size();<br>for(int i=0;i<size;i++) count++; |
| *Char array StringBuilder* | |
| for(int i=0;i<1000000;i++){<br>StringBuilder builder=new<br>StringBuilder();<br>for(int v=0;v<1000;v++)<br>builder.append('?');<br>String result=builder.toString();} | for(int i=0;i<1000000;i++){<br>char[] array=new char[1000];<br>for(int v=0;v<1000;v++)<br>array[v] = '?';<br>String result=new String(array);} |
| *Binary search* | |
| for(int i=0;i<10000000;i++){<br>int index = -1;<br>for(int j=0;j<values.length;j++)<br>if (values[j] == 80)<br>index = j;<br>break; }}} | for(int i=0;i<10000000;i++)<br>int index<br>=Arrays.binarySearch(values,80); |

Table 1: Unoptimized and optimized functions

## 3.1 Summary

We develop two JAVA projects in order to regroup all the unoptimized and optimized functions previously described. We obtain the following power and energy related relationships (Figure 1 and 2).

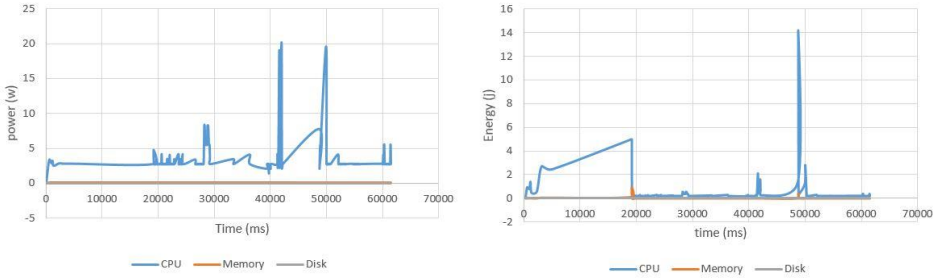


Figure 1: Unoptimized Power and Energy

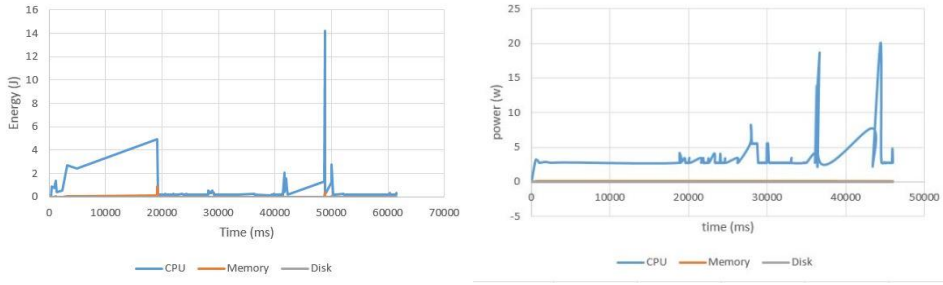


Figure 2: Optimized Power and Energy

Therefore, we observe, without surprise, that in general the power consumption of CPU dominate memory or disk consumption. If we analyze the results obtained each 50 ms, we can say that the power consumption of disk can be neglected for these cases, but in some cases power consumption of memory must be taken into account. Moreover, we can say that the power consumption of the unoptimized code is higher than the one of the optimized code and the total execution time of optimized code is less than the one of the unoptimized code. Consequently, it is of great interest to develop optimized parts of code in order to obtain green and efficient software.

## 4. Validation

To validate our experiments, we use a powermeter 'wattsup ?PRO'. We connect this device to the notebook via USB port. This device saves in his memory the power consumed by all process in runtime. So, we connect WattsUp to the notebook and then we wait until the power reach a stationary state. Then, we launch the unoptimized program, followed by the optimized program. We then transfer the results using the application WattsUpUSB and the results are depicted in Figure 3.

Comparing to the results obtain with TEEC, even if we make a measurement in each second, we can say that in most of the case, optimized code test is faster and reveals less power than unoptimized code test. Each optimized and unoptimized curves present some increase of power as we observed with TEEC.

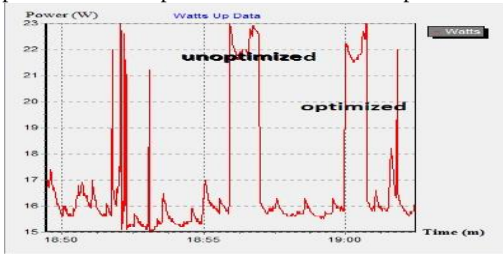


Figure 3: Unoptimized and Optimized Power

## 5. Conclusion

In addition to the CPU, a modelization of memory and hard disk has been made to describe the behavior of each component. Mathematical expressions have been established in order to calculate the

power consumption of each component. The tool we designed, named TEEC, take into account CPU, memory and hard disk power consumption.

The accuracy of TEEC has been tested over several optimized and unoptimized functions and validated against a real powermeter.

We stated that power consumption of memory should not always be neglected in front of the CPU power consumption whereas power consumption of hard disk can be neglected.

We observed that the optimizations of source code are required in order to contribute to the reduction of the greenhouse gas emissions.

In a short term, we wish to extend the capability of TEEC by integrating others components power consumption (such as network interface cards, etc.). In a long term, we will use the output of TEEC to guide developers in order to build greener software in real time.

## References


[1] GreenTouch, ICT Industry Combats Climate Change. http://www.greentouch.org/?page=how-the-ict-industries-can-help-the-world-combat-climate-change

[2] Acar, H., Alptekin, G.I., Gelas, J.-P. and Ghodous, P. 2015. Towards a Green and Sustainable Software. In Proceedings of the 22nd ISPE International Conference on Concurrent Engineering (Delf, The Netherlands, July 20 – 22, 2015).

[3] Kansal, A., Zhao, F., Liu, J., Kothari, N. and Bhattacharya, A. 2010. Virtual Machine Power Metering and Provisioning. In Proceedings of the 1st ACM Symposium on Cloud computing. (New York, USA, 2010).

[4] Wang, S., Chen, H. and Shi, W., 2011. SPAN: A software power analyzer for multicore computer systems. Sustainable Computing: Informatics and Systems,Volume 1, Issue 1, March 2011, 23–34.

[5] Kern, E., Dick, M., Naumann, S., Guldner, A. and Johann, T. 2013. Green software and green software engineering–definitions, measurements, and quality aspects. In Proceedings of the First International Conference on Information and Communication Technologies for Sustainability (Zurich, Switzerland , February 14-16, 2013).

[6] Kim, M., Ju, Y., Chae, J., Park, M., 2014. A Simple Model for Estimating Power Consumption of a Multicore Server System. International Journal of Multimedia and Ubiquitos Engineering.

[7] Minas, L., Ellison, B., 2009. The Problem of Power Consumption in Servers. Intel Press.

[8] Hur, I. and Lin, C., 2008. A comprehensive approach to DRAM power management. International Symposium on High Performance Computer Architecture.

[9] Kang, U. et al., 2010. 8 Gb 3-D DDR3 DRAM Using Through-Silicon-Via Technology. Journal of SolidState Circuits.

[10] Vogelsang, T., 2010. Understanding the energy consumption of dynamic random access memories. Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture.

[11] Micron, 2007. Calculating Memory System Power for DDR3.

[12] IDC serves up 10 storage predictions for 2008, 2007. http://www.cio.co.uk/news/index.cfm?articleid=2455

[13] HD Tune. 2015. http://www.hdtune.com/index.html

[14] Bucy, J., Schindler, J., Schlosser, S., Ganger, G. and Contributors. 2008. The DiskSim Simulation EnvironmentVersion 4.0 Reference Manual. Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, PA 15213.

[15] Molaro, D., Payer, H., Le Moal, D. 2009. Tempo: Disk Drive Power Consumption Characterization and Modeling. In Proceedings of the IEEE 13th International Symposium on Consumer Electronics.

[16] Sankar, S., Zhang, Y., Gurumurthi, S., Stan, M. 2008. Sensitivity Based Optimization of Disk Architecture. In IEEE Transactions on Computers.

[17] Morgan, R. and MacEachern, D. 2010. https://support.hyperic.com/display/SIGAR/Home