Improving the Energy Efficiency of Software Systems for Multi-Core Architectures

Maxime Colmant, Romain Rouvoy, Lionel Seinturier

Improving the Energy Efficiency of Software Systems for Multi-Core Architectures

Maxime Colmant
ADEME
University Lille 1
Inria
maxime.colmant@inria.fr

Romain Rouvoy
University Lille 1
Inria
romain.rouvoy@inria.fr

Lionel Seinturier
University Lille 1
Inria
IUF
lionel.seinturier@inria.fr

ABSTRACT

The ICT has an huge impact on the world CO_2 emissions and recent study estimates its account to 2% of these emissions. This growing account emissions makes IT energy efficiency an important challenge. State-of-the-art has proven that the processor is the main power consumer. Processor are nowadays more and more complex and they are used in many hardware systems, such as computers or smartphones. This thesis is thus focusing on the software energy efficiency for multi-core systems. In this paper, we therefore report our motivations to understand deeply their architectures for improving their energy efficiencies. Manufacturers have worked tremendously to improve performance and reduce power consumption of their processors. However a lot of things remains to do in the software side. We claim that energy-efficient softwares can play a deterministic role to reduce the IT carbon footprint. To answer this challenge, we are believing on the software-metric approach with a minimal hardware investment. For this purpose, an efficient, scalable and non-invasive tool is needed. As a result, we created PowerAPI, to provide fine-grained power estimations at process and code-level for optimizing the software enegergy efficiency automatically. This solution will help to identify clearly the energy leaks for optimizing automatically the power consumed by software.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

Keywords

CPU power model, Middleware toolkit, Power monitoring

1. INTRODUCTION

The increasing usage of IT devices (*e.g.* computers, smartphones, electronics) takes a non-negligible part of our global carbon emissions. According to [12], the *Information and Communications Technology* (ICT) carbon footprint was estimated at 0.83 $GtCO_2$ in 2007 and is estimated to 1.43 $GtCO_2e$ for 2020, representing 6% of our overall emissions per year. This growing carbon footprint makes IT energy efficiency an important challenge. Power estimation of software processes is considered as a cornerstone of power-efficient systems. It is particularly useful when operating at level of individual tasks for identifying the largest power consumers and make informed decisions during the scheduling. Modern processors are increasingly complex and such systems do not provide coarse-grained solutions to estimate their own power consumptions. However some manufacturers are going in this direction. For example, Intel developed such kind of solution [3, 14] but it is limited at some processor generations. A lot of work still remains to do for proposing a solution that will work on any architecture without doing any assumptions.

To have the opportunity to address this problem, a powerful and a non-invasive monitoring solution is needed. Then, we will be able to act and to optimize their energy consumptions by playing with the scheduling or by improving their resource uses.

In this paper, we thus explain why this problem is a key challenge and why it is important to address it in priority. For estimating the software power consumption we also present our middleware toolkit, PowerAPI[1], written in Scala. Based on the concept of distributed actors, PowerAPI is a scalable, non-intrusive and efficient way to estimate the power consumption of running processes, in real-time, with a minimal hardware investment. Our library automatically learns the processor energy profile by stressing it in several dimensions for capturing the supported features. To get representative and accurate metrics, the *Hardware Performance Counters* (HPC) are used. They are mostly available on modern processors. With the help of a regression approach, we used them associated to the power consumptions for building an efficient and lightweight energy profile. As a preliminary experiment, we evaluate our tool on a well-known benchmark, SPECJbb2013, to compare our power estimations with real measurements.

The rest of this paper is organized as follows. First we discuss the motivations behind this thesis in Section 2. Next, we provide a clearly description of our research methodologies in Section 3. We then introduce our preliminary results in Section 4 and finally, we conclude in Section 5.

2. MOTIVATION

[1]Available as open source from `http://bit.ly/PowerAPI`.

Middleware 2014 Doctoral Symposium, December 8-9, Bordeaux, France

http://dx.doi.org/10.1145/2684080.2684081 ...$15.00.

During the last decade, developers have been developing software products without paying a careful attention to the resources they were consuming. In particular, the software performance was generally preferred. However, recent trend in GreenIT raises the key issue of the impact of the ICT on the world CO_2 emissions. Nowadays, the total of dioxide emissions reached 9.1 billions tons [9] and current estimates account the ICT for 2% of world CO_2 emissions [12] which is comparable to the civil aviation.

The main source of emissions come from the hardware components. Particularly, one can mention the massive impact of the processor on the power consumption. Nowadays, processors are becoming complex computational units regarding to the Moore's law. The most important effect of this law is the development of multi-core processors. They are widely used whether in computer, smartphones and so on. A lot of work was done on the hardware side to improve their performances and to reduce their energy consumptions. One can mention the *Simultaneous Multi-Threading* (SMT) principle which allows to improve the performance by separating each core to two logical HyperThreads or reduce the energy consumption by sharing more effectively resources. The *Dynamic Voltage/Frequency Scaling* (DVFS) is also very important because it can improve drastically the energy efficiency of a system. In particular, it allows the processor to adjust its clock speed, and thus run at different frequencies whether is necessary in order to drop down its power consumption by reducing the frequency of underused cores. Moreover, we can also mention the *C-States* that were introduced to save energy and allow the CPU to use low power modes. The main idea of this principle is to lower the clock speed, turn off some units to reduce the amount of energy consumed.

Modern processors are really well-optimized but if the softwares running on them are wasting time and resources, all these optimizations are useless. We therefore believe they can play a deterministic role in the development of a greener planet. This does not only encompass the development of software for better controlling the energy distribution (*e.g.*, SmartGrid), but also the emergence of green software as a more sustainable solution. From a scientific perspective, this challenge requires a deep understanding and analysis of the energy consumption of nowadays systems to reduce their resource consumption and to cause fewer carbon emissions. We argue it is very important to identify the main power consumers and try to make decisions based on it, for example by using different scheduling algorithms.

Power monitoring is usually done with the help of hardware measurement equipments, such as power meters or specialized integrated circuits. This technique is not suitable at large because it will require costly investments. Furthermore, it provides the power consumption of the whole system and not directly the energy consumption of software components and running applications. As a consequence, we are targeting a scalable approach that can be achieved by a software-centric approach. Monitoring the energy consumption of softwares with this approach requires to tackle several challenges. Contrarily to the hardware approach, the software-centric approach requires power models to provide accurate power estimations. Such solutions need to find the best trade-off between accuracy and overhead. Software solutions are already provided to estimate the processor power consumption. As an example, the *Running Average Power Limit* (RAPL) [3, 14], developed by Intel since the Sandy Bridge architecture, provides different packages and one of them allows to get the power consumption of the whole CPU package by reading registers (MSR). However, this solution is architecture dependent and is limited to few architectures. Moreover, most papers are targeting specific softwares and architectures [1, 2, 4, 10, 14]. They do not provide a general approach which could be directly applied in most situations. Our main goal is to provide a way to obtain reliable power estimations on all recent architectures without important hardware investments.

By establishing a clear knowledge of usual energy leaks, we intend to identify green patterns as a methodological guideline that can assist the developers in building energy-efficient software. Additionally, the emergence of renewable energies is introducing the need for the development of adaptive strategies that can cope with the sporadic nature of these energy feeds.

3. RESEARCH METHODOLOGY

We aim to answer a crucial problem for the future of ICT: to reduce their carbon footprint. Despite it can be considered as ambitious, we believe we can achieve significant and encouraging results quickly. We decided to target multi-core architectures because they are widely used in nowadays computers or smartphones. Current platforms do not provide directly the software power consumption although manufacturers are making efforts to address this problem (*e.g.* RAPL). Thus, an efficient profiling tool is needed. When considering the state-of-the-art, all the provided tools are architecture and software dependent [1, 2, 3, 4, 10, 14]. Based on the work presented in [6], we decided to create a generic and software independent tool that is able to estimate the software power consumption on modern CPU architectures. To answer this challenge, the first step is to build power models which trustfully represent the power consumption. They are used for estimating and splitting the power consumption between all the system components (*i.e.* CPU, GPU, memory, disk, network). Most of the time, sampling step and regression analysis are used to learn the energy profiles. Both power measurements and raw metrics are gathered and injected in various regression techniques and the resulting models are mostly linear [5]. Different metrics can be used to represent the processor activities. For example, Versick *et. al* [13] use the CPU load to represent the processor activity or the others use HPC that provide more informations about the type of instructions executed [1, 2, 4, 10, 14]. Regarding the state-of-the-art, the second one is considered as a better solution as these performance counters can capture all the processor activities while the CPU load mostly indicates whether the processor executes a job. These HPC are very useful but they are architecture dependent. In fact, each architecture can have different HPC and thus the state-of-the-art generally tries to find the most relevant counters to model the power consumption. Hence, each paper identifies a different set of HPC and it is then difficult to adapt their solutions. Therefore, we decided to find a processor power model that could be used on any modern architectures (*i.e.* Intel, AMD). Our process used to learn the energy profile of a modern processor, regardless of its architecture, is presented in Figure 1.

First, we defined specific CPU and memory intensive work-

loads to identify and capture the relationship between the kind of operations executed and the power consumption. These benchmarks are exectued for each frequency made available by the processor.

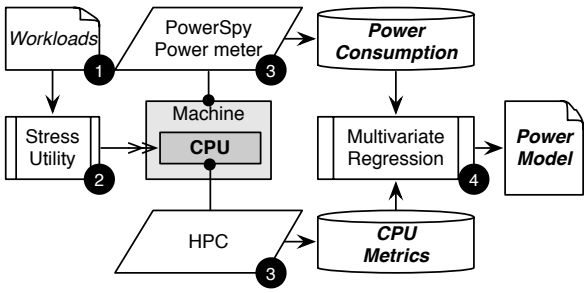

Figure 1: Power model learning process.

We selected several hardware performance counters according to two criteria: their availability on a large family of architectures and the overhead imposed by their exploitation. We studied the evolution of all the generic counters [8] to identify the counters `instructions`, `cache-references`, `cache-misses` as the ones which are the most correlated with the power consumption on multi-core systems. We use libpfm4[2] library for accessing to the HPC.

To collect the power consumption, we use a bluetooth power meter, PowerSpy[3].

To improve the accuracy of the power model, the workloads are executed several times and then the values are correlated using a multivariate regression. One power model is computed per frequency.

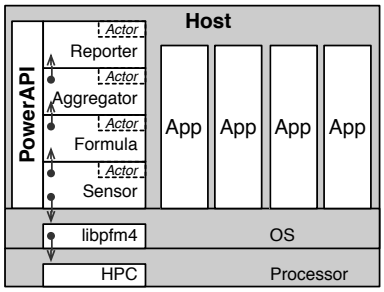

Figure 2: The PowerAPI architecture.

Our approach is implemented in PowerAPI, its global architecture is depicted in Figure 2. It includes at least four actor components: *Sensor*, *Formula*, *Aggregator*, *Reporter*. An actor is a lightweight entity that runs concurrently and processes messages using an event-driven model[4]. It can handle millions of messages per second, depending on their work, which is a key property for supporting real-time power estimations.

The components are described as follows:

Sensor monitors the metrics of a given process and then publish a sensor message to the event bus,

Formula get the sensor messages from the event bus in order to estimate the power consumption of a given process,

Aggregator aggregates the power estimations according to a dimension, like the PID or the timestamp,

Reporter converts the power estimations produced by the library into a suitable format.

[2] `http://perfmon2.sourceforge.net`
[3] `http://www.alciom.com/en/products/powerspy2.html`
[4] `http://akka.io`

4. PRILIMINARY RESULTS

Vendor	**Intel**
Processor	**i3**
Model	2120
Design	4 threads
Frequency	3.30 GHz
TDP	65 W
SpeedStep (DVFS)	✓
HyperThreading (SMT)	✓
TurboBoost (Overclocking)	✗
C-states (Idle states)	✓
L1 cache	64 KB / core
L2 cache	256 KB / core
L3 cache	3 MB

Table 1: Intel Core i3 2120 specifications.

As a priliminary experiment, we used a memory intensive benchmark, SPECjbb2013 on a Intel Core i3 2120 processor. Its specifications are described in Table 1.

Regarding the process described in Figure 1, the equation below reflects the power model on this processor:

$$Power = 31.48 + \sum_{f=1.6}^{3.30} Power_f$$

Where the overall power consumption over a period is computed as the sum of the power consumptions in the different frequencies f supported by the processor, including the TurboBoost ones when availables. The constant (31.48) isolates the idle power of the machine. As an example, the formula at maximum frequency is described as follows:

$$Power_{3.30} = \frac{2.22 \cdot i}{10^9} + \frac{2.48 \cdot r}{10^8} + \frac{1.87 \cdot m}{10^7}$$

Where i, r, and m refers to the HPC instructions, cache-references, and cache-misses, respectively. One can observe that the cache activities tend to lead the power consumption.

As we can observe in Figure 3, the estimations produced by our power model follow the same trend as the real power consumption and exhibit a median error of 15%.

By comparison, Bertran *et. al* [1] use a sampling phase to gather data related to the HPC. Their CPU energy model show an average error of 4.63% on six applications taken from the SPEC CPU2006 suite on the processor Intel Core 2 Duo which is a simple architecture without any features for improving performances (no HyperThreading, no TurboBoost). In [14], the authors proposed an HyperThread aware model for reducing the error models. They evaluated their new model on private Google benchmarks and have an average error of 7.5%. Unfortunately, neither their experiments nor the power model they proposed can be reproduced.

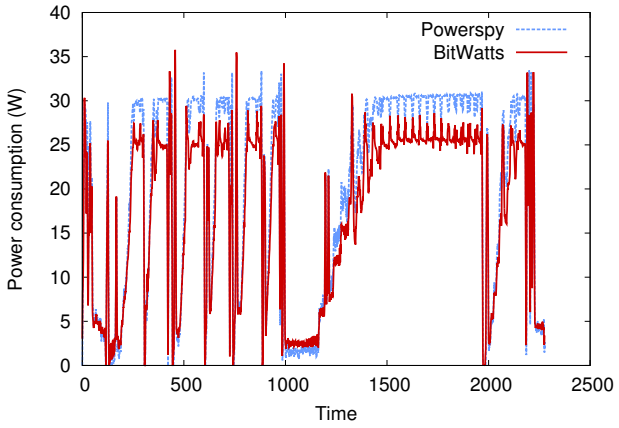

Figure 3: Preliminary experiment on SPECJbb2013 benchmark.

5. CONCLUSION

In this paper, we presented our researches to improve the software-energy efficiency on multi-core systems. Our motivations were driven by the huge impact of the ICT on the world CO_2 emissions which represents 2% of them. It is important to target such architectures because they are widely used in nowadays computers and smartphones. Lot of work has been done on the hardware side with several features, like the DVFS one. But there is still a lot of work to do on the software side and we showed why this is a key challenge. Developers start to consider the potential power consumption of their software. As a result, several tools were created to provide such informations to the developers [7, 11].

We also presented our approach to learn the modern CPU energy profiles. First, it stresses the processor features by applying CPU and memory intensives workloads. Then, the power consumption and raw metrics are gathered to build a power model by using a multivariate regression technique. With the help of the state-of-the-art, we have clearly identified the hardware performance counters as the the most relevant metrics to estimate the software power consumptions. In fact, they allow to capture all the various activities executed by modern processors.

Based on these models, we propose PowerAPI, an architecture and application independent middleware toolkit for estimating the power consumption at process-level.

Our results show that only consider the generic counters is not the necessarily the most reliable solution leading to high errors. This is why we plan to improve our learning algorithm by using the Spearman rank correlation for finding automatically the most correlated ones with the power consumption. Then, we will work on the software optimizations, and one of the suitable examples could be the virtual machines. Indeed, they are more and more used and a lot of work still remains to optimize their power consumptions.

Acknowledgements

These researches are co-funded by the French Agency Of Environment (ADEME), University Lille 1 and Inria.

6. REFERENCES

[1] Bertran, R., Gonzalez, M., Martorell, X., Navarro, N., and Ayguade, E. Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of the 24th ACM International Conference on Supercomputing* (New York, NY, USA, 2010), ICS '10, ACM, pp. 147–158.

[2] Bircher, W. L., and John, L. K. Complete system power estimation: A trickle-down approach based on performance events. In *Performance Analysis of Systems & Software, 2007. ISPASS 2007. IEEE International Symposium on* (2007), IEEE, pp. 158–168.

[3] Hähnel, M., Döbel, B., Völp, M., and Härtig, H. Measuring energy consumption for short code paths using rapl. *SIGMETRICS Perform. Eval. Rev. 40*, 3 (Jan. 2012), 13–17.

[4] Lim, M. Y., Porterfield, A., and Fowler, R. Softpower: Fine-grain power estimations using performance counters. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (New York, NY, USA, 2010), HPDC '10, ACM, pp. 308–311.

[5] McCullough, J. C., Agarwal, Y., Chandrashekar, J., Kuppuswamy, S., Snoeren, A. C., and Gupta, R. K. Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conference* (2011).

[6] Noureddine, A. *Towards a Better Understanding of the Energy Consumption of Software Systems*. PhD thesis, University Lille 1, 2014.

[7] Noureddine, A., Rouvoy, R., and Seinturier, L. Unit Testing of Energy Consumption of Software Libraries. In *Symposium On Applied Computing* (Gyeongju, Korea, Republic Of, Mar. 2014).

[8] perf-event-open man page. `http://web.eece.maine.edu/~vweaver/projects/perf_events/perf_event_open.html`. Accessed: 2014-05-21.

[9] Peters, G. P., Marland, G., Le Quere, C., Boden, T., Canadell, J. G., and Raupach, M. R. Rapid growth in co2 emissions after the 2008-2009 global financial crisis. *Nature Clim. Change 2*, 1 (01 2012), 2–4.

[10] Spiliopoulos, V., Sembrant, A., and Kaxiras, S. Power-sleuth: A tool for investigating your program's power behavior. In *MASCOTS* (2012), pp. 241–250.

[11] Sterling, C. Energy consumption tool in visual studio 2013, July 2013.

[12] The Climate Group. SMART 2020: Enabling the low carbon economy in the information age.

[13] Versick, D., Wassmann, I., and Tavangarian, D. Power consumption estimation of CPU and peripheral components in virtual machines. *SIGAPP Appl. Comput. Rev. 13*, 3 (Sept. 2013), 17–25.

[14] Zhai, Y., Zhang, X., Eranian, S., Tang, L., and Mars, J. Happy: Hyperthread-aware power profiling dynamically. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)* (Philadelphia, PA, June 2014), USENIX Association, pp. 211–217.