BitWatts: A Process-level Power Monitoring Middleware

Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loïc Huertas, Romain Rouvoy, Anita Sobe

▶ **To cite this version:**

Maxime Colmant, Mascha Kurpicz, Pascal Felber, Loïc Huertas, Romain Rouvoy, et al.. BitWatts: A Process-level Power Monitoring Middleware. Middleware 2014, Dec 2014, Bordeaux, France. 2014. hal-01078825v1

HAL Id: hal-01078825

https://inria.hal.science/hal-01078825v1

Submitted on 31 Oct 2014 (v1), last revised 8 Dec 2014 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

BitWatts: A Process-level Power Monitoring Middleware

Maxime Colmant
ADEME / Univ. Lille 1 / Inria

Mascha Kurpicz
Univ. Neuchâtel

Pascal Felber
Univ. Neuchâtel

Loïc Huertas
Inria / Univ. Lille 1

Romain Rouvoy
Univ. Lille 1 / Inria

Anita Sobe
Univ. Neuchâtel

ABSTRACT

Power estimation of software processes provides critical indicators to drive scheduling or power capping heuristics. State-of-the-art power estimation solutions only provide coarse-grained support for power estimation. In this paper, we therefore propose a middleware for assembling and configuring software-defined power meters. Software-defined power meters provide real-time and accurate power estimation of software processes. In particular, our solution automatically learns an application-agnostic power model, which can be used to estimate the power consumption of applications.

Our approach, named BitWatts, builds on a distributed actor middleware to collect process usage and infer fine-grained power consumption without imposing any hardware investment (*e.g.*, power meters).

Categories and Subject Descriptors

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

Keywords

CPU power model, Middleware toolkit, Power monitoring

1. INTRODUCTION

Energy-efficient computing is becoming increasingly important, which requires power estimation as a key mechanism for understanding and reducing the energy consumption of software. It is particularly useful to identify the largest power consumers and make informed decisions when scheduling tasks at the level of individual application processes. However, modern processors are increasingly complex and only a few manufacturers, like Intel with *Running Average Power Limit* (RAPL) [1, 6], start to embed power-aware interfaces in their latest generation of processors. To overcome this limitation, we propose to a middleware , named BitWatts[1], to build software-defined power meters. Based on the concept of distributed actors, our software-defined power meters provide a scalable, non-intrusive and efficient way to estimate the power consumption of running processes, in real-time, with minimal hardware investment. Besides, our solution automatically learns the processor power profile by stressing it in several dimensions for capturing the supported features. To get representative and accurate metrics, BitWatts uses *Hardware Performance Counters* (HPC), which are mostly available on modern processors. The underlying CPU power model is obtained by applying a multivariate regression on both HPC metrics and hardware power measures collected under various stress setups. As a preliminary experiment, we evaluate our tool on a well-known benchmark—SPEC CPU2006—to compare our power estimation with real measurements. The remainder of this paper introduces the CPU model learning phase (cf. Section 2) as well as our middleware (cf. Section 3) that we used to perform our preliminary experiments (cf. Section 4).

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Middleware 2014, Poster session December 8-9, Bordeaux, France
Copyright 2014 ACM 978-1-4503-3220-0/14/12 ...$15.00.

2. LEARNING THE CPU POWER MODEL

The hardware performance counters used to estimate the power consumption of processors have to be carefully selected according to two criteria: their availability on a large family of architectures and the overhead imposed by their exploitation. We therefore studied the evolution of all available counters under different workloads to identify the counters `cycles`, `instructions`, `cache-references`, and `cache-misses` as the ones which characterize as best the power model of multi-core architectures. While the counter `cycles` provides a clear indication on the current frequency of the cores, the three others counters reflects the processors' activity. The hardware performance counters and power consumption values collected along the execution of different workloads are then correlated using a multivariate regression to connect the evolution of the power consumption with the variations of a vector made of the hardware performance counters we identified. In practice, the equation below reflects the power model we obtained for an Intel Xeon processor:

$$Power_{Xeon}(host) = 90.23 + \sum_{\substack{f=1.6,\\ f \in Freq}}^{2.93} Power_f(host)$$

Where the power consumption over a period of time is computed as the sum of the power consumption in the different frequencies f supported by the processor, including the TB special frequency when available. The constant (90.23) isolates the idle power of the machine. Then, the power model

per frequency is defined as follows:

$$Power_{2.93}(host) = \frac{5.37 \cdot i}{10^9} + \frac{7.67 \cdot r}{10^8} + \frac{3.23 \cdot m}{10^7}$$

Where i, r, and m refers to the HPC instructions, cache-references, and cache-misses, respectively.

3. BITWATTS MIDDLEWARE

We built BitWatts as a modular middleware to assemble software-defined power meters. Software-defined power meters are expected to deliver power consumption reports at high frequency—between 10 and 100 Hz—in order to closely monitor the activity of an application running on a system. We therefore adopted the actor programming model to develop a middleware that can scale with the frequency and the number of applications to be monitored. An actor is a lightweight entity that executes concurrently with other actors and processes messages asynchronously using an event-driven model. Actors provide a scalable solution to process millions of messages per second, which is a key property for supporting real-time power estimation.

In BitWatts, we use the Scala programming language[2] and the Akka library[3] to develop actors with the following roles:

Sensor connects the software-defined power meters to the underlying system in order to collect raw values of system activity. Raw values can be coarse-grained power consumption reported by third-party power meters and embedded probes, or CPU activity statistics as delivered by the process file system (`ProcFS`). Sensors are triggered according to the requested monitoring frequency and forward raw values to the appropriate formula.

Formula uses the raw values received from the sensor to perform the power estimation. A formula therefore implements a specific power model [4, 3, 5] to convert raw values into actual power consumption values. The granularity of the power consumption reported by the formula (machine, core, process) depends on the granularity of the values forwarded by the sensors.

Aggregator is in charge of aggregating power consumption, according to a specific dimension like the *process identifier*, to compute the power consumption, or *timestamp*, to group the power consumption of several applications.

Reporter finally formats the power consumption produced by the formula or the aggregator into a suitable format. Such reports can be provided for instance via a Web interfaces or a virtual file system (*e.g.*, based on FUSE).

4. PRELIMINARY EXPERIMENTS

To assess our solution, we compare the energy consumption measured by the power meter with the estimations produced by the power models we learned. This comparison uses SPEC CPU2006 [2] as an acknowledged benchmark suite, which includes many CPU-intensive workloads. In particular, we report on the execution of five representative benchmarks covering both integer and floating point operations. Figure 1 depicts the measured energy consumption compared to the estimated one. The straight line $y = x$ represents the best fit energy consumption, while the upper and lower lines indicate an absolute error of 6 %.

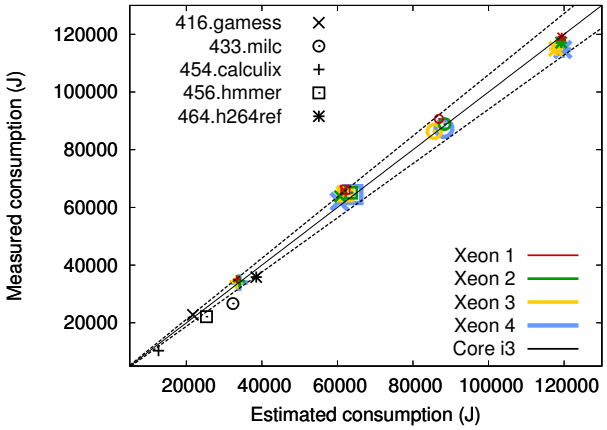

Figure 1: Energy consumption.

5. CONCLUSION

BitWatts is a middleware to build software-defined power meters. BitWatts learns the processor characteristics and creates a CPU power model of a multi-core architecture by means of sampling and multivariate regression. We evaluated the performance of BitWatts in different settings and showed that it performs well on a variety of applications.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007–2013] under the ParaDIME Project (`www.paradime-project.eu`), grant agreement no. 318693.

6. REFERENCES

[1] 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.

[2] Henning, J. L. SPEC CPU2006 benchmark descriptions. *SIGARCH Comput. Archit. News 34*, 4 (Sept. 2006), 1–17.

[3] Kansal, A., Zhao, F., Liu, J., Kothari, N., and Bhattacharya, A. A. Virtual machine power metering and provisioning. In *1st ACM Symposium on Cloud computing* (2010), ACM, pp. 39–50.

[4] Koller, R., Verma, A., and Neogi, A. WattApp: an application aware power meter for shared data centers. In *7th International Conference on Autonomic computing* (2010), ACM, pp. 31–40.

[5] 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.

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

[2] `http://scala-lang.org`
[3] `http://akka.io`