The Journal of Open Source Software

PowerAPI: A Python framework for building software-defined power meters

Guillaume Fieni [1], **Daniel Romero Acero** [1], **Pierre Rust** [3], and **Romain Rouvoy** [1,2]

1 Inria, France **2** University of Lille, France **3** Orange Labs, France

DOI: 10.21105/joss.06670

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @varshaprasad96
- @EpicStep

Submitted: 16 April 2024
Published: 04 June 2024

Summary

Software that we use daily for accessing digital services from connected devices has a negative impact on the environment as it consumes energy. These digital services, hosted by physical machines around the world, also contribute to planetary pollution. Unfortunately, providers of these online services mostly focus on hardware efficiency to reduce the environmental impact without considering the software they host. For this reason, we propose `PowerAPI` (G. Fieni, Romero, et al., 2024), a software-defined solution that delivers real-time estimations of software power consumption to spot opportunities to reduce it and therefore to limit their impact on the planet beyond hardware improvements.

Statement of need

State-of-the-art power meters only measure the global consumption of power of the machines hosting the software. To deal with this limitation, several software-defined power meters have been proposed. However, they are statically designed and coupled to a specific processor family or they are mainly based on power models that need a lot of data and time to be trained to predict power consumption accurately (Colmant et al., 2018; LeBeane et al., 2015). Power models designed in this way are only suitable for environmental conditions (i.e., memory usage, CPU usage, kinds of running applications) similar to those when the data for the training was collected. If these conditions change, power models become deprecated and have to be trained again, which make them unsuitable for production environments of Cloud providers.

To deal with this constraint, we developed `PowerAPI` (G. Fieni, Romero, et al., 2024), a software toolkit for assembling software power meters, enabling developers/IT administrators to monitor power consumption of software that they write/deploy. Software-defined power meters created with `PowerAPI` enable power consumption estimations at different granularity levels: process, thread, container, virtual machine, etc. Furthermore, power models used by `PowerAPI` are continuously self-calibrated to consider current execution conditions of the machine hosting the software.

Figure 1 present the general idea behind a Software Power Meter in `PowerAPI`:

1. A Sensor collects raw metrics related to a machine hosting applications/software to be monitored in terms of energy consumption.
2. These metrics are stored in a database to be consumed by a software power model.
3. This power model uses machine learning techniques to estimate energy consumption of applications/software with the raw metrics and it calibrates itself when required.
4. Estimations produced by the power model are stored in another database.
5. The stored estimations are used to optimize concerned applications/software.

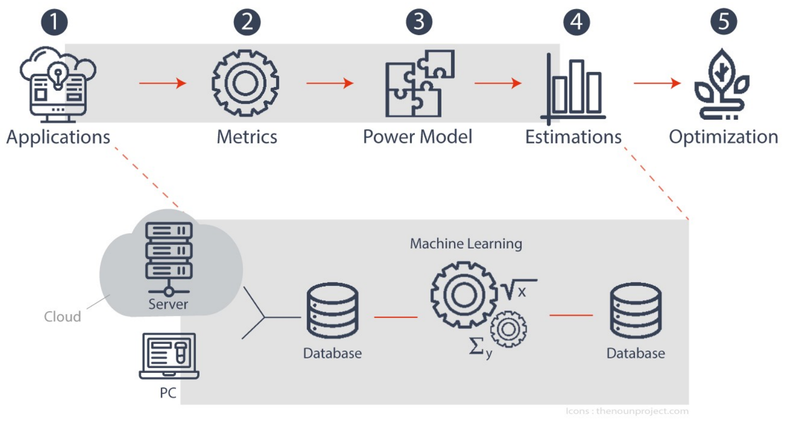

Figure 1: Power consumption estimation in PowerAPI.

Currently, `PowerAPI` offers a software-defined power meter composed by `Hardware Performance Counter` (`HwPC`) Sensor (G. Fieni & Rouvoy, 2024) and `SmartWatts` Formula (Guillaume Fieni et al., 2020; G. Fieni, D.Romero, et al., 2024). `HwPC` Sensor is written in `C` and is based on the *Running Average Power Limit* (RAPL) interface (Corporation, 2023), available on Intel and AMD processors, and the `perf` tool and `cgroup` from the Linux kernel. `SmartWatts` Formula, developed in Python, defines a power model based on a linear regression from the sckit-learn library (Pedregosa et al., 2011), which is self-calibrated by using suitable performance counters and an error threshold provided by the Power Meter user. It is possible to use a `CSV` files, `MongoDB`, or a `HTTP` Socket to store metrics from Sensors. For storing estimations, `PowerAPI` offers `CSV` files, `MongoDB`, `InfluxDB`, and `Prometheus`.

`PowerAPI` is based on the actor model (Agha, 1986; Hewitt et al., 1973), which means that there is a low coupling between different architectural elements in a software-defined power meter. This fosters the creation of new `Formulas` and/or `Sensors` actors to tune software-defined power meters according to requirements.

Tool Demonstration

A demonstrator of `PowerAPI` is made available to illustrate the deployment of a software-defined power meter on a Xeon CPU E5-2407 Intel processor (Sandy Bridge Family) running Debian 11. The selected configuration includes `MongoDB` for storing metrics and `InfluxDB` for the estimations. In this demonstration, we monitor, in real-time, the power consumption of two web applications based on a microservice architecture and hosted by a bare-metal server: a social network from DeathStarBench and tea store online developed by the Descartes Research group. Figure 2 and Figure 3 depicts the grafana dashboard for these web applications.

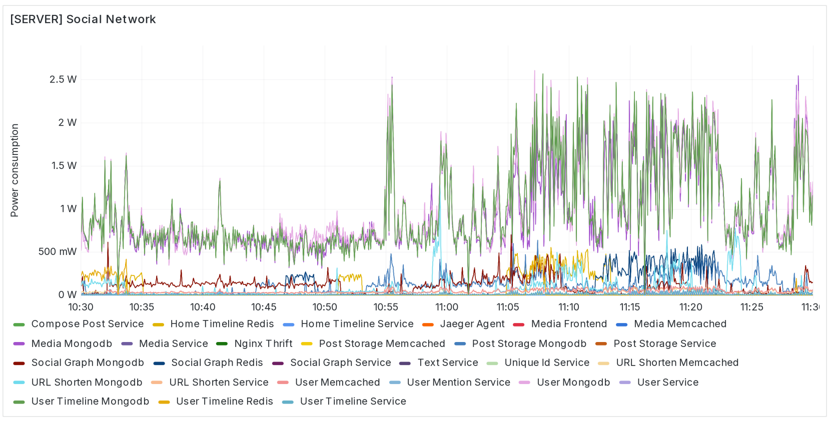

Figure 2: Social Network Power consumption, server side.

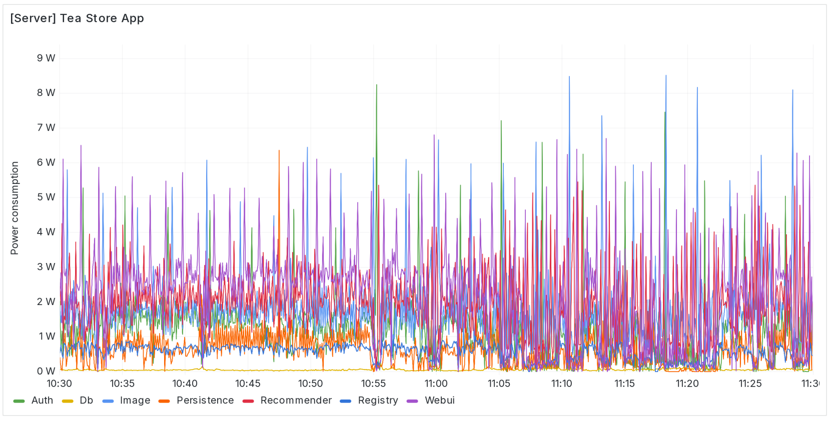

Figure 3: Tea Store Power consumption, server side.

In the tool demonstration, we also monitor, in real-time, a client laptop accessing the above-mentioned web applications from Firefox and Chromium. The laptop uses an Intel i5-6300U (Skylake Family) processor and runs Ubuntu 20.04.5 LTS. Figure 4 and Figure 5 show the grafana dashboard for the client running 2 instances of each browser.

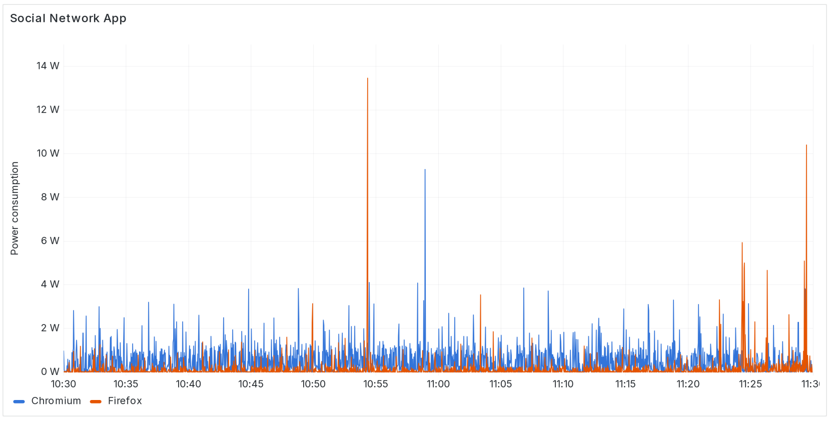

Figure 4: Social Network Power consumption, client side.

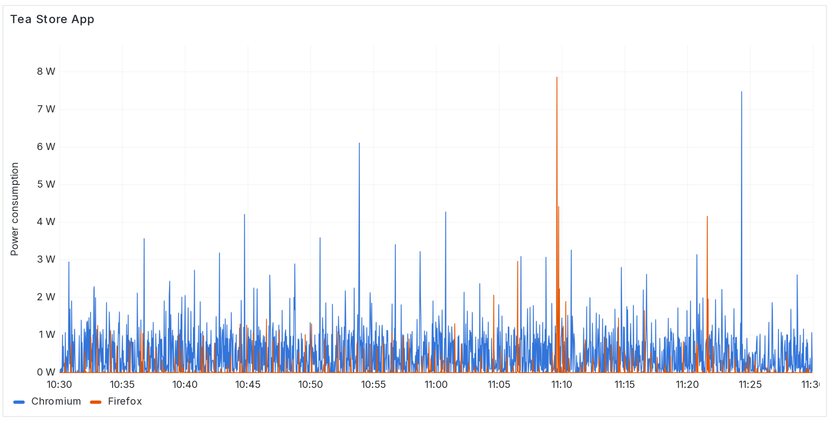

Figure 5: Tea Store Power consumption, client side.

Publications and Projects

The software toolkit has results from contributions described in several publications (Colmant et al., 2017, 2018; Guillaume Fieni et al., 2020, 2021) and is already exploited in several research projects:

- Distiller ANR project searches how to reduce energy consumption of Cloud applications.
- Défi FrugalCloud includes the optimization of the energy footprint of cloud infrastructures operated by OVHcloud.
- Défi Pulse studies how to valorize emissions from High Performance Computing (HPC) using as use case Qarnot Computing's offers.

Acknowledgements

Currently, the maintenance of the toolkit is funded by Inria, Orange Labs, OVH Cloud, Davidson Consulting and La Poste Groupe.

This work received support from the French government through the *Agence Nationale de la Recherche* (ANR) under the France 2030 program, including partial funding from the CARECloud (`ANR-23-PECL-0003`), DISTILLER (`ANR-21-CE25-0022`), and GreenAct (`ANR-21-HDF1-0006`) projects.

References

Agha, G. (1986). *Actors: A model of concurrent computation in distributed systems*. MIT Press. ISBN: 0262010925

Colmant, M., Felber, P., Rouvoy, R., & Seinturier, L. (2017). WattsKit: Software-defined power monitoring of distributed systems. In F. Capello, G. Fox, & J. Garcia-Blas (Eds.), *17th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid)* (p. 10). IEEE. https://doi.org/10.1109/ccgrid.2017.27

Colmant, M., Rouvoy, R., Kurpicz, M., Sobe, A., Felber, P., & Seinturier, L. (2018). The next 700 CPU power models. *Journal of Systems and Software*, *144*, 382–396. https://doi.org/10.1016/j.jss.2018.07.001

Corporation, I. (2023). *Intel 64 and IA-32 architectures software developer's manual - combined volumes: 1, 2A, 2B, 2C, 2D, 3A, 3B, 3C, 3D, and 4*. Intel Corporation.

Fieni, G., D.Romero, & Rouvoy, R. (2024). SmartWatts formula. In *GitHub repository*. GitHub. https://github.com/powerapi-ng/smartwatts-formula

Fieni, G., Romero, D., & Rouvoy, R. (2024). PowerAPI core. In *GitHub repository*. GitHub. https://github.com/powerapi-ng/powerapi

Fieni, G., & Rouvoy, R. (2024). Hardware performance counters (HwPC) sensor. In *GitHub repository*. GitHub. https://github.com/powerapi-ng/hwpc-sensor

Fieni, Guillaume, Rouvoy, R., & Seinturier, L. (2020, May). SmartWatts: Self-calibrating software-defined power meter for containers. *CCGRID 2020 - 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*. https://doi.org/10.1109/CCGrid49817.2020.00-45

Fieni, Guillaume, Rouvoy, R., & Seinturier, L. (2021, May). SELFWATTS: On-the-fly Selection of Performance Events to Optimize Software-defined Power Meters. *CCGRID 2021 - 21th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*. https://doi.org/10.1109/ccgrid51090.2021.00042

Hewitt, C., Bishop, P., & Steiger, R. (1973). A universal modular ACTOR formalism for artificial intelligence. *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, 235–245.

LeBeane, M., Ryoo, J. H., Panda, R., & John, L. K. (2015). Watt watcher: Fine-grained power estimation for emerging workloads. *2015 27th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, 106–113. https://doi.org/10.1109/SBAC-PAD.2015.26

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., & others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.