http://excel.fit.vutbr.cz

Profiling energy consumption on Linux systems

Ondřej Míchal*

Abstract
Software resource consumption is a widely and actively researched area. However, of the many resources in software which can be profiled, energy consumption has long been the one resource without any generic, and yet comprehensive profilers. In the age of mobile devices and efficient processing units the need for such profiles is continuously increasing. In this work we present methods for accurate measurement of energy consumption of software, describe a novel open-source profiler and also design a visualizer of the profiled data. With the profiler we then conduct a list of experiments on GNOME Shell with various workloads to showcase its capabilities and demonstrate the usefulness of knowing energy consumption.

*xmicha80@stud.fit.vutbr.cz, Faculty of Information Technology, Brno University of Technology

Nowadays, resource consumption of software is increasingly becoming a concern for users as well as software developers. With mobile devices dominating the electronics market the need for having optimal resource consumption is more than obvious. But, the same need applies to other types of devices as well. Personal desktops, servers (mainly in the HPC space), IoT devices or other smart machines need to perform efficiently as well.

There is quite a high number of different resources a software can consume. For many of these one can use an abundance of existing tools analysers to cover most, if not all, use cases. However, many of the resources are yet to be properly research. A particular resource lacking well established tooling is energy consumption. Resources like CPU or memory are being optimized for in most software as their are the easiest to optimize for. If a software developer had in their toolbox a tool for quick, reliable and detailed analysis of their software energy consumption, they could start optimizing the software for optimal energy consumption as well. Ignoring performance aspects of ones software can often lead to overheating and in-general unfavourable execution time/energy consumed ratio which optimizing for energy consumption could help to mitigate.

Currently, most Linux tools that support measuring energy consumption do not provide detailed enough information and lack any context, i.e., functions/methods or lines of code which can help the developer to locate the culprit of potential problems. Among the existing tools are powertop [1] or turbostart [2]. We see this as an opportunity to make use of current state-of-the-art technology and existing research to create an open-source profiler for detailed profiling of energy consumption. We believe such profiler could be used as a stepping stone for further research, especially if employed in some wider performance analyses project. Such a project is Perun [3], an open-source performance version system for continuous tracking of a project's performance developed by the VeriFIT research group at BUT FIT.

In this work we build a prototype of an open-source energy profiler to that is expected to be integrated in Perun together with methods for visualizing the profile data it generates. To monitor the energy consumption of a system one could use an external hardware monitor but those lack the required granularity, can only measure energy consumption as a whole and are an uncommon external hardware. Instead, we use a purely software solution utilizing Running Average Power Limit (RAPL), a capability introduced in modern Intel processors (introduced in the Sandy Bridge processor generation) and currently supported by some AMD processors as well. This interfaces is proven [4, 5, 6] to provide high-quality energy consumption readings with high level of granularity. To provide runtime context to energy readings we propose to trace system calls. Existing research [7, 8] shows the potential of system calls to provide the needed context for energy consumption. Implementing the results of the mentioned research in full is out of the scope of this work, so we use a much simpler approach which still demonstrates the potential of the energy profiler. To trace system calls and sample energy consumption readings we use the eBPF instrumentation framework. It allows running sandboxed programs in a privileged context like the Linux kernel. By using eBPF we achieve high granularity of sampled data by only marginally affecting the performance of the profiled software. The profiler runs loads eBPF programs into the kernel, samples the data and at the end outputs a performance profile in JSON format, which contains the sampled performance events (i.e., energy consumption samples and all traced system calls). In testing we could sample energy consumption at a rate of thousands of samples per second without significantly affecting the profiled process's performance. Together with the profiler we implement a tool for visualizing the profile data created by the profiler using Python modules Pandas, Seaborn and Matplotlib. We build on these libraries, since they are already supported by the Perun project and they are well-maintained and efficient. In particular, we propose to visalize the data using (1) heat maps (which shows higher energy consumptions using "warmer" colours in grid-like graphs) and (2) waterfall graph (that presents the consumption of different calls in time in a single plot).

We demonstrate the profiler on a series experiments showing its capabilities. We chose GNOME Shell, a graphical shell, as the subject of the testing as it is a non-trivial software making use of all general components (CPU, RAM, GPU) and which is highly susceptible to the overhead introduced by profilers. For the experiments we used a single testing scenario but changed the execution environment to see how does it affect the results of the profiler, to see how susceptible it is to noise. In particular, we (1) run GNOME Shell without any particular workloads, and (2) run GNOME Shell with Firefox browser playing a YouTube video. We ran both experiments for about 30s and then compared the results using our visualizations. The experiments showed that the profiler can successfully locate the sources of high consumption of the profiled software, and differences of the runtime environments as well as anomalies in energy consumptions can affect the quality of the resulting profiles.

Acknowledgements

I would like to thank my supervisors Jiří Pavela, Ing. and Tomáš Fiedor, Ing., Ph.D. from the VeriFIT performance team for their guidance, honest feedback and support while making this work.

References

[1] Arjan van de Ven. PowerTOP, April 2023.

[2] Linus Torvalds. Torvalds/linux, April 2023.

[3] Tomas Fiedor. Perun: Lightweight Performance Version System, July 2022.

[4] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In 2015 IEEE International Parallel and Distributed Processing Symposium Workshop, pages 896–904, May 2015.

[5] Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. RAPL in Action: Experiences in Using RAPL for Power Measurements. ACM Transactions on Modeling and Performance Evaluation of Computing Systems, 3(2):9:1–9:26, March 2018.

[6] Joe A. Garcia. Exploration of Energy Consumption Using the Intel Running Average Power Limit Interface. In 2019 IEEE Space Computing Conference (SCC), pages 1–10, July 2019.

[7] Abhinav Pathak, Y. Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: Fine grained energy accounting on smartphones with Eprof. In Proceedings of the 7th ACM European Conference on Computer Systems - EuroSys '12, page 29, Bern, Switzerland, 2012. ACM Press.

[8] Karan Aggarwal, Chenlei Zhang, Joshua Charles Campbell, Abram Hindle, and Eleni Stroulia. The power of system call traces: Predicting the software energy consumption impact of changes. In Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, CASCON '14, pages 219–233, USA, 2014. IBM Corp.