Techniques and tools for measuring energy efficiency of scientific software applications

David Abdurachmanov[1], **Peter Elmer**[2], **Giulio Eulisse**[3], **Robert Knight**[4], **Tapio Petteri Niemi**[5], **Jukka Nurminen**[6], **Filip Nyback**[6], **Goncalo Marques Pestana**[6], **Zhonghong Ou**[6]

[1] Digital Science and Computing Center, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania
[2] Department of Physics, Princeton University, Princeton, NJ 08540, USA
[3] Fermilab, Batavia, IL 60510, USA
[4] Research Computing, Office of Information Technology, Princeton University, Princeton, New Jersey 08540, USA
[5] Helsinki Institute of Physics, PO Box 64, FI-00014, Helsinki, Finland
[6] Aalto University, PO Box 11100, 00076 Aalto, Finland

E-mail: `Peter.Elmer@cern.ch`

Abstract. As both High Performance Computing (HPC) and High Throughput Computing (HTC) are sensitive to the rise of energy costs, energy-efficiency has become a primary concern in scientific fields such as High Energy Physics (HEP). There has been a growing interest in utilizing low power architectures, such as ARM processors, to replace traditional Intel x86 architectures. Nevertheless, even though such solutions have been successfully used in mobile applications with low I/O and memory demands, it is still unclear if they are suitable and more energy-efficient in the scientific computing environment. Furthermore, there is still lack of tools to derive and compare power consumption for these types of workloads, and eventually to support software optimizations for energy efficiency.

To that end, we have performed several physical and software-based measurements of workloads from CERN running on ARM and Intel architectures, to compare their power consumption and performance. We leverage several profiling tools to extract different aspects of the experiments, including hardware usage and software characteristics. We report the results of these measurements and the experience gained in developing a set of measurement techniques and profiling tools to accurately assess the power consumption for scientific workloads. [Version of 17 September 2014]

1. Introduction

The most recent scientific applications have to process and store considerable volume of data. It is foreseeable that the volume of data will increase considerably in the future, as technology and requirements enhance. Thus, energy consumption has become a major concern amongst the scientific community.

The Large Hadron Collider (LHC) [1] at the European Laboratory for Particle Physics (CERN) in Geneva, Switzerland, is one of the scientific projects which computational requirements are too massive for resources to be processed and held in one single infrastructure. Hence, data processing and storage are distributed across the Worldwide LHC Computing Grid (WLCG) [2], which uses resources from 160 computer centers in 35 countries. The access to such computational resources have made possible CMS [3] and ATLAS [4] experiments to achieve important results, such as the discover of the Higgs Boson [5, 6]. While enabling this and other discoveries, the WLHC consumes massive amount of computational resources and, proportionally, energy. Only CMS experiment used approximately 80,000 to 100,000 x86-64 cores of capacity in 2012, according to [7, 8]. In the future, with the improvement of the detector's luminosity, the dataset size will increase by 2-3 orders of magnitude [7, 8], presenting even more challenges from the energy consumption point of view.

In order to find and develop better solutions to improve energy efficiency in High Energy Physics (HEP), it is important to understand how energy is used by the HEP systems themselves. There are few tools and techniques that facilitate researchers to reach that goal. Some of these tools and techniques are outlined and described in this article.
As energy efficiency becomes a concern, new solutions have been considered to develop energy efficient systems. One potential solution is to replace the traditional Intel x86 architectures by low power architectures such as ARM. A comparison of the energy efficiency between ARMv7 and x86 Intel architecture is conducted in this article. The experiments use CMS workloads and rely on the techniques and tools described earlier to perform the measurements.

This article is structured as following. Firstly, we describe where is energy consumed in a HTC system. Secondly, we outline some of the tools and techniques available to measure and monitor energy consumption on HTC systems. Finally, we present the results of a comparison between ARMv7 and Intel Xeon architecture using CMS workloads.

2. Tools and techniques for energy measurement

As pictured in Figure 1, a HTC system is composed by several components. It is important to grasp where and how energy is consumed by the system, so that researchers can trace, identify and improve energy bottlenecks. Therefore, given the specificities of those components, different types of tools and techniques should be used to measure energy consumption, according to what is suppose to measure.. In this section, we will outline and describe different approaches to measure energy consumption of the different components on a HTC system.

2.1. External probing devices

External probing devices measure the energy consumed by the whole system, without breaking it down in components pictured on Figure 1.

2.1.1. Noninvasive clamp meters and plug-in energy monitors are probes that allow to measure electrical current pulled by the system without making physical contact or interfering with it. Generally, the accuracy of these tools is around 3%, whereas its precision is in the order of seconds (Hz).

2.1.2. Power distribution units (PDU) are devices designed to distribute electric power to systems. They are broadly used in server's rack and usually have power monitoring capabilities embedded, which makes it useful for data warehouses and big computing facilities. The accuracy and precision of such devices are similar to clamp meters described above.

can we generalize the accuracy and precision based on the tools we used ? are there any references stating this ?

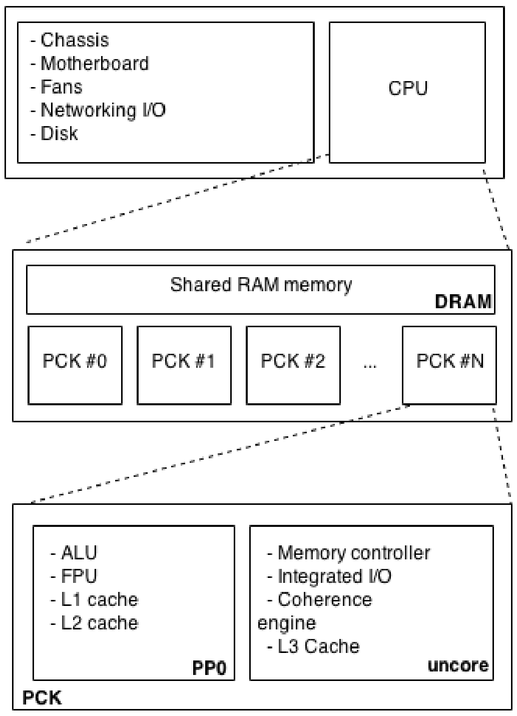

Figure 1. Components that contribute for power consumption in HPC

2.2. Internal probing chips

2.2.1. Running Average Power Limit Running Average Power Limit (RAPL) provides a platform for monitoring and limiting power of systems on chip (SoC). It is an Intel technology which was introduced initially on the Sandy Bridge processors. RAPL platform exposes chip's energy measurements via the MSR registers. According to [9], this technology offers power measurements of the system at a granularity impossible to reach before with other tools.

As documented by Intel in [10], there are 3 different domains to sample energy consumed by different SoC components on a server. The domains are **package** (pck), which measures energy consumed by the system's sockets, **power plane 0** (pp0), which measures energy consumed by the CPU core, and **dram**, which accounts for the sum of energy consumed by memory in a given socket, excluding the core caches. The measurements are dumped in the MSR registers at a frequency of 1 kHz and are exposed to the user via /dev/cpu/*cpu_nr*/msr. It is also possible to read and write data from the MSR register using Intels open source tool msr-tools [11].

In addition to power monitoring of the sockets, it can limit the power consumed by the different domains. This feature, usually referred as power capping, allows the user to define the average power consumption limit of a domain in a defined time window. For more information about RAPLs features and configurations, refer to section 14.9.1 of Intels Developers manual [10].

The advantages of using RAPL for measuring power consumption are a straightforward and already installed tool to perform fine grained measurements of energy consumption on SoC and its components.

On the other hand, the drawbacks are lack of documentation available about the monitoring chip. To the knowledge of the authors, specifications such as error degrees, accuracy and implementation diagrams are not publicly available. In addition, the RAPL technology is vendor locked. Considering those two points, it is difficult to accurately compare and reason power measurements between SoCs from Intel and other vendors.

2.2.2. Chip monitors are

2.3. Software based measuring tools

There are software tools that provide insights about energy consumed by the system. In addition, they can interact directly with the software. This capability enables to map applications' code and energy consumption and help developers to develop energy efficient applications. There are several examples of software based measuring tools, such as powertop [12] and IgProf [13], a general-purpose and open-source application profiler that was recently upgraded with energy measuring capabilities.

2.4. IgProf

IgProf is an application profiler that profiles mainly performance and memory usage. The profiler collects data about the resources an application uses. The resource of interest can be for example execution time, memory or file descriptors. The main idea behind profiling is to find the parts of the application where a resource is used the most, i.e. possible bottlenecks. When the bottlenecks have been located, it makes sense to make optimisations where the bottlenecks are, because resource usage is usually not uniformly distributed over the whole application. [13] [14] [15]

The profiler was available on the Intel x86 and x86-64 architectures, as well as on 32-bit ARM, but support for 64-bit ARM was missing. The port of IgProf to 64-bit ARM enables developers to evaluate how applications execute on the new architecture with regard to performance and memory usage.

A simple energy profiling module extends the functionality of IgProf. The energy profiling module is based on sampling and uses the PAPI library to obtain energy measurements from the RAPL (Running Average Power Limit) interface present on recent Intel processors. [16]

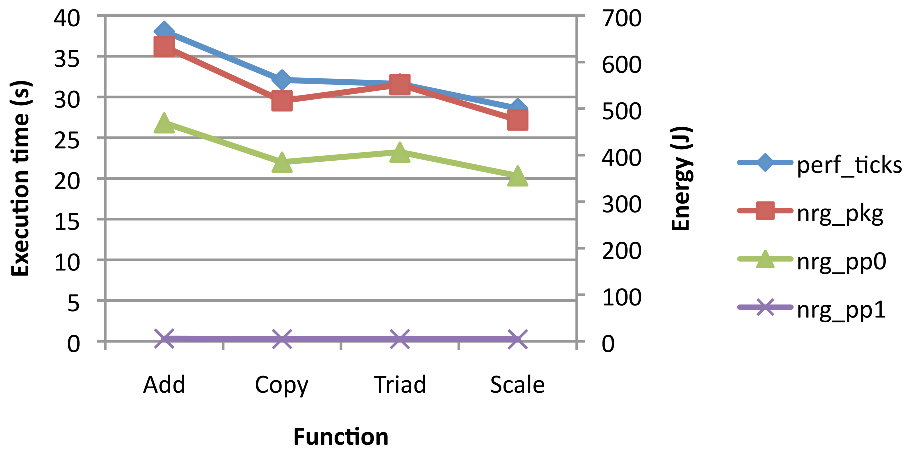

Figure 2. The results of performance and energy profiling of the stream tool.

Figure 2 shows the results from performance and energy profiling of the *stream* benchmarking tool. The X-axis describes the four main functions contributing to the execution time and energy consumption of the stream tool: Add, Copy, Triad and Scale. The left scale of the Y-axis and the perf_ticks series describe the execution time spent in each function, whereas the right scale of the Y-axis and the nrg_pkg, nrg_pp0 and nrg_pp1 series describe the amount of energy spent in each function. The energy consumption of the processor package domain and the power plane 0 (describing the CPU cores) seem to follow the time spent in the functions, whereas the energy consumption of power plane 1 (describing the GPU) seems to be fairly constant. [17]

The profiling results of a simple single-threaded application seem to show a correlation between the execution time and the energy spent in a function. The energy profiling module is, however, still rather limited; e.g. it does not seem to profile multi-threaded applications correctly.

3. Use case

- introduce section as an example of the tool's utilization
- mention that ARM might be a good solution to energy efficient HTC architecture.
- references to previous works

3.1. Experiments setup

- explain ParfullCMS
- machine's specs
- experiment's setup

3.2. ARM results

Result of measurements done in ARM

3.3. Intel results

Result of measurements done in Intel

3.4. Analysis

- conclusions of the experiment

4. Conclusions

Acknowledgements

This work was partially supported by the National Science Foundation, under Cooperative Agreement PHY-1120138, and by the U.S. Department of Energy.

References

[1] Lyndon Evans and Philip Bryant. LHC Machine. *JINST*, 3:S08001, 2008.

[2] Bird I. Computing for the Large Hadron Collider. *Annual Review of Nuclear and Particle Science*, 61, 2011.

[3] S. Chatrchyan et al. The CMS experiment at the CERN LHC. *JINST*, 3:S08004, 2008.

[4] Aad G et al (Atlas Collaboration). The Atlas Experiment at the CERN Large Hadron Collider. *JINST*, 3, S08003.

[5] S. Chatrchyan et al. Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC. *Phys.Lett.*, B716:30–61, 2012.

[6] G. Aad et al. Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys.Lett.*, B716:1–29, 2012.

[7] David Abdurachmanov, Peter Elmer, Giulio Eulisse, and Shahzad Muzaffar. Initial explorations of arm processors for scientific computing. *Journal of Physics: Conference Series*, 523(1):012009, 2014.

[8] David Abdurachmanov, Kapil Arya, Josh Bendavid, Tommaso Boccali, Gene Cooperman, Andrea Dotti, Peter Elmer, Giulio Eulisse, Francesco Giacomini, Christopher D Jones, Matteo Manzali, and Shahzad Muzaffar. Explorations of the viability of arm and xeon phi for physics processing. *Journal of Physics: Conference Series*, 513(5):052008, 2014.

[9] Balaji Subramaniam and Wu chun Feng. Towards energy-proportional computing for enterprise-class server workloads. *CPE '13 Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, 2012.

[10] *Intel 64 and IA-32 Software Developer Manuals - Volume 3.* www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html.

[11] Msr tools. `https://01.org/msr-tools`. Last time accessed: 28-07-2014.

[12] Powertop. `https://01.org/powertop`. Last time accessed: 17-09-2014.

[13] Giulio Eulisse and Lassi Tuura. IgProf, the Ignominous Profiler. Web page. `http://igprof.org/`. Accessed 2014-03-27.

[14] Lassi Tuura, Vincenzo Innocente, and Giulio Eulisse. Analysing CMS software performance using IgProf, OProfile and callgrind. *Journal of Physics: Conference Series 119*, 2008. DOI 10.1088/1742-6596/119/4/042030.

[15] Martin Fowler. Yet another optimization article. *IEEE Software*, 1(May/June):20–21, 2002. ISSN 0740-7459.

[16] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. Measuring energy and power with PAPI. *Proceedings of the International Conference on Parallel Processing Workshops*, pages 262–268, 2012. DOI: 10.1109/ICPPW.2012.39.

[17] John D. McCalpin. STREAM: Sustainable memory bandwidth in high performance computers. Web page. `http://www.cs.virginia.edu/stream/`. Accessed 2014-09-04.