A User Perspective on Energy Profiling Tools in Large Scale Computing Environments

Fahimeh Alizadeh Moghaddam
SNE and S2 groups
University of Amsterdam and VU University Amsterdam
Amsterdam, The Netherlands
Email: f.alizadehmoghaddam@uva.nl

Patricia Lago
Software and Services group (S2)
VU University Amsterdam
Amsterdam, The Netherlands
Email: p.lago@vu.nl

Thomas Geenen
SURFsara
Amsterdam, The Netherlands
Email: thomas.geenen@SURFsara.nl

Paola Grosso
System and Network Engineering group (SNE)
University of Amsterdam
Amsterdam, The Netherlands
Email: p.grosso@uva.nl

***Abstract*—The growth of power consumption in ICT infrastructures emphasizes the importance of monitoring their usage and finding available room for improvement. Users can choose among several existing tools to determine the energy profile of a running application in order to provide more sustainable software. We conducted a field study in the SURFsara data center and we experimented with the tools available, assessing them in light of their informative power. We derived some recommendations for ICT users and infrastructure operators that highlight the relation between the intended use of the profile and the easiness of running a specific tool. We categorize users in two types: *the generic user*, who is interested in summary statistical results on power measurements and *the software developer* that intends to delve in the code details in order to reprogram the application more efficiently. We concluded that SLURM and Score-P are suitable for both types of users when their requirements are thoroughly studied and taken into account.**

I. Introduction

ICT infrastructures are being used more extensively every day because of their wide range of applicability. Consequently, efficiency of power consumption and energy management in these environments is important. Large scale computing environments and data centers as one of the major parts of the ICT technologies play a remarkable role in energy consumption. As DatacenterDynamics 2012 Global Census [1] indicates, the power consumption of data centers globally has been 38GW and has a significant impact on environmental and economical resources.

In order to make ICT infrastructure and data centers more energy efficient, an important step is to profile the power consumption of running applications. Energy profiling of software is in fact becoming a major focus area in the wider field of "green" ICT. The knowledge of power behavior of the hardware components can lead to a more efficient selection of ICT architectures; similarly knowledge of the power behavior of software can lead to revision of code to better exploit the underlying hardware characteristics.

Recently, a number of profiling tools have been introduced that are able to monitor the energy consumption of running software applications. These tools vary in terms of accuracy, sampling rate and overhead. Processing measurements will also exhibit varying degrees of post-processing, visualization and interpretation feedback. Given this plethora it is not always clear from the start for users (from cloud providers to end users) what to use, and under which circumstances. As we will show in this article the choice for a tool will depend on the final purpose for the energy information retrieved. We can distinguish between two types of users that are categorized based on their requirements:

- *The generic user* that is interested in summary results, with the goal of profiling ICT resources and running applications and determining their footprint;
- *The software developer* who uses the footprint information to improve the running code in terms of energy efficiency.

We assess the usability of a profiling tool based on a set of initial user requirements identified out of experience:

- **Expertise level**: It refers to the level of knowledge of the infrastructure, and the software/hardware interactions that a user has.
- **Documentation provided**: We consider that all users work with the profiling tools for the first time. It is therefore important that they get enough information from the tool providers.
- **Access to support team**: Sometimes users face problems to get the tool working and they need a point of contact, e.g. tools mailing lists.
- **Easiness to export data**: Some users need to transfer their data to other tools to process them.
- **Accuracy of provided information**: The accuracy requirement should take into account errors that will occur in collecting and reporting the data.

The main goal of the research we present here is to evaluate, from an end user perspective, the usability of a series of commonly available monitoring tools. Namely we are interested in answers to the questions:

978-3-901882-70-8

Under which circumstances should a user choose for an energy profiling tool?

What will be the consequences in terms of accuracy and overhead of this choice?

We focus on the profiling tools provided by SURFsara, the Dutch national supercomputer center that supports computing-based research from scientists all over the Netherlands. We assume that SURFsara offering in terms of energy profiling tools is a representative example for a generic cloud-based environment, and as such our conclusions are of general applicability.

II. Measurement Mechanisms

If we consider the Intel Sandy Bridge and Haswell architectures used at SURFsara we can distinguish between two different methods of energy information:

- Collecting data from CPU and DRAM
  - Running Average Power Limit (RAPL);
  - Performance Application Programming Interface (PAPI).
- Collecting data from other components (including CPU)
  - Baseboard Management Controller (BMC);
  - Intelligent Platform Management Interface (IPMI).

RAPL - Running Average Power Limit - sensors provide energy consumption information on Intel Sandy Bridge architectures. RAPL is not an analog power meter, but it uses a software power model running on a helper controller. The energy is estimated using hardware performance counters temperature, leakage models and I/O models. Values are exposed to users via model-specific register (MSR). Several types of readings are reported by RAPL on the level of socket but not on the level of individual core:

- PACKAGE ENERGY: total usage by entire package;
- PP0 ENERGY: sum of energy of all cores and caches (not each core individually);
- PP1 ENERGY: on original Sandy Bridge this includes the on-chip Intel GPU;
- UNCORE ENERGY: the energy usage of uncores;
- DRAM ENERGY: DRAM energy usage.

The division of readings helps fine-grained analysis on different domains. Depending on client (Desktop) or server platforms, some readings might not be supported:

$$Client(Desktop)Platform : PACKAGE = PP0 + PP1$$

$$ServerPlatformPACKAGE = PP0 + UNCORE$$

Recently RAPL has been included in the PAPI library [2]. It has become possible to access the RAPL counter using the across-platform PAPI [3]. PAPI is platform independent and is able to collect energy data through external and internal measurements. RAPL is considered as an internal measurement mechanism through PAPI, which does not need additional changes to the hardware.

IPMI provides a set of standardized interfaces to monitor computer systems. For energy measurements the IPMI relies on on-board BMC specification. The sensors of the BMC measure a wider range of variables in comparison to RAPL counters such as temperature, fan, voltage and hardware errors.

III. Related Work

Extensive work exists on RAPL. The Intel documentation indicates that energy readings are updated roughly every millisecond (1 kHz). Rotem et al. [4] performed an extensive study on the energy capability of Intel processors and showed that the RAPL results match actual hardware. In further investigation, such as [5] it has been noticed that there are small deviations under different types of loads.

SLURM has been introduced in 2003 [6], without the support of energy measurement plugins. Recently energy accounting per job through SLURM has been added. Georgiou et al. [7] represents an evaluation on both available energy measurement plugins (RAPL and IPMI) in this tool. Also [8] provides a literature study on two fundamental power management techniques in HPC environments: Metrics and Profiling. They do not focus on specific tools from a user perspective.

In [9], Score-P is described as a performance measurement infrastructure. Furthermore, they make evaluations among some performance analysis and optimization tools specific for parallel applications. However, experiments do not include power measurements.

The original contribution of our work compared to the research presented above is that we specifically focus on data processing tools from usability point of view.

IV. Available Tools

There is a set of data processing tools available at SURFsara, which use data collection mechanisms as a basis and provide visualizations and functionality analysis.

- **SLURM** (Simple Linux Utility for Resource Management) is an open-source RJMS (Resource and Job Management System)[1]. SLURM schedules incoming job submissions and put them in a queue until a compute node is available. The data collected from different sensors are stored in files. The user can choose between HDF5 or CSV output file formats. While output files can be merged easily they cannot be accessed during runtime. HDF5 file, a hierarchical file format, structures the data in two formats: *Time Series* to show the power consumption of a node based on 1 second intervals, and *Totals* to show the summary statistical information.
  SLURM is able to use both RAPL and IPMI plugins to collect energy measurements but currently does not allow making use of both at the same time for each individual node. At SURFsara, SLURM is used as the main scheduler and as such collects data on all the scheduled jobs.
- **Score-P and CUBE** Score-P[2] provides an instrumentation framework for applications, in which users are able to utilize PAPI in order to profile power measurements. The

[1] https://computing.llnl.gov/linux/slurm/slurm.html
[2] http://www.vi-hps.org/projects/score-p/

profiled data can be visualized afterwards by some interfaces. At SURFsara, CUBE is used as the visualization analysis tool. CUBE Uniform Behavioural Encoding is a program used to create an intuitive GUI for data. CUBE utilizes Scalasca (Scalable performance analysis of large scale applications), which in turn utilizes the Score-P library. The tool can be used to find opportunities for improving the energy efficiency of an application. For each run of the application, Score-P collects the statistical data in the CUBE4 file format. CUBE tool displays three hierarchical browsers: Metric, Program and System. The Metric browser identifies the summary results for all the measured metric variables. The Program browser allows the user to see how the metric values (for example power consumption) map to what part of the application. Finally the System browser concentrates on machines, nodes, processes and threads involved in the application run. One of the key functionality in CUBE is usage of colors to create heat maps of selected sets of data.

V. Infrastructure Setup

To perform our experiments we run the HPC Challenge benchmark in the SURFsara infrastructure because it tests multiple aspects of a computing system.

a) HPC Challenge benchmark: [3] consists of a set of 7 tests for HPC environments to stress out different parts of the system. The main purpose is to measure the performance of HPC systems in realistic scenarios. The tests included in HPCC are:

- **HPL**: It is a portable implementation of High Performance Linpack Benchmark. This test stresses the floating-point performance and accuracy of solving a linear system of equations. This extremely computational heavy test will give insight to the behavior of the CPU under peak load.
- **DGEMM**: It measures the floating-point performance of matrix multiplication.
- **FFT**: It stresses the floating-point performance of computing Discrete Fourier Transform with different complexities.
- **Latency/Bandwidth**: It focuses on latency and bandwidth measurements of simultaneous communication patterns.
- **PTRANS**: It concentrates on parallel processes where pairs of processors communicate simultaneously.
- **RandomAccess**: It stresses the performance of integer random updates of memory.
- **STREAM**: It measures memory bandwidth of the system.

Some of the tests are run in up to 3 variants: Single, Star and MPI. In the single version, the test is run on a single process. More than one process runs the identical single process in the star version. In the MPI version, processes cooperate to execute the test by contributing to part of it.

b) SURFsara infrastructure: As mentioned before, we use the infrastructure established at SURFsara to conduct our research. Cartesius[4] is a clustered symmetric multiprocessing system built by Bull and deployed as supercomputer in the infrastructure. There are different types of nodes built into the Cartesius system: thin, fat and GPU, which are meant for different functionalities.

We use HPCC subtests to collect power measurements data for further analysis. We choose GPU nodes at SURFsara to run our experiments, as power measurements are currently available on this type of nodes. SLURM is the main scheduler there that receives all the job submissions. However, it is possible to profile power consumption using other tools as such Score-P.

VI. Results

We run a series of experiments with the purpose of comparing accuracy and usability of the various tools:

A. Experiment 1: baseline experiment collecting data directly via PAPI/RAPL

We use the rapl_plot application to read RAPL sensors directly. Using this application we can set the sample rate in microseconds. We examine the results with four sample rates: every 1, 10, 100 and 1000 milliseconds. Figure 1 shows the wattage of HPL test of the HPCC benchmark over time. As the sample rate decreases, higher number of details is missing from the plots. For example in the bottom plot there is an instant peak between 1 and 2 seconds of runtime while it is completely missed in the top plot with the lowest sample rate. Changing the sample rate does not have a known influence on the power measurements, as this essentially depends on the application behavior itself. However, there is around 3,5% increase in run time duration with higher sample rates.

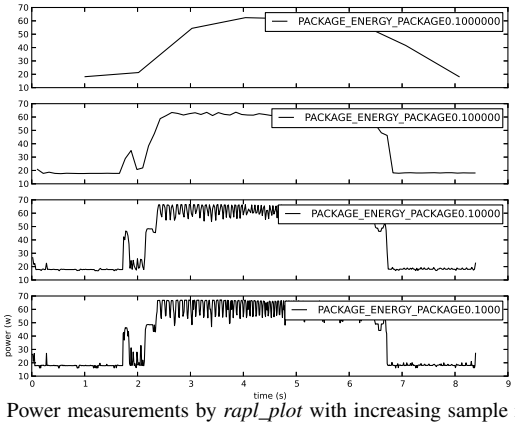

Fig. 1. Power measurements by *rapl_plot* with increasing sample rates from every 1 seconds (top plot) to every 1 milliseconds (bottom plot)

B. Experiment 2: direct submission via the SLURM scheduling system

SLURM is able to collect energy data from two plugins: RAPL and BMC. In this work to make a sound comparison between these tools, we run our experiments only with RAPL configuration. SLURM logs the energy profiling data and information on spent time, wattage and selected node in a hierarchical structure. The output HDF5 file contains energy data for intervals of 1 second. We take the results from the rapl_plot application as our baseline to evaluate the accuracy of SLURM data. It should be noted that sample

[3] http://icl.cs.utk.edu/hpcc/

[4] https://SURFsara.nl/systems/cartesius

rate of the rapl_plot application is set as 1 second as well. As for rapl_plot, we get several output files from the RAPL sensors: DRAM, PACKAGE, PP0 and UNCORE (See fig. 2). Since sensors are collecting measurements from different components, there are variations in ups and downs of different lines. In case of the HPCC benchmark, the instant changes of different lines happen symmetrically based on the starting time and stopping time of included subtests.

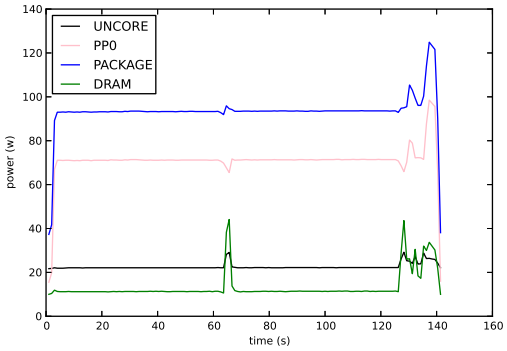

Fig. 2. Power measurements collected by different *rapl_plot* sensors: DRAM, PACKAGE, PP0 and UNCORE

Figure 3 shows the aggregation of the rapl_plot sensors data (PACKAGE ENERGY PACKAGE + DRAM ENERGY PACKAGE) and SLURM data. As it displays, two measurements are almost identical in x-axis (runtime) and y-axis (wattage). The overhead added to the values reported by SLURM is more on some small delays for reporting the sudden changes in power consumption.

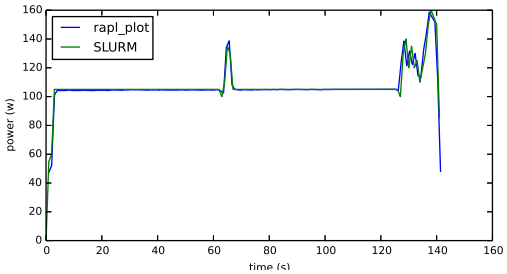

Fig. 3. Power measurements from SLURM and the *rapl_plot* sensors

C. Experiment 3: benchmark run with Score-P

We compiled the HPCC benchmark application with Score-P, which will provide us with energy measurements. Unlike SLURM, Score-P does not read the sensors with a specific sample rate. Energy consumption information and other variables are collected twice per function: one time at entry and once at exit. Consequently, Score-P does not provide time series that can inform us on the behavior of applications during run time. The granularity of data collected by Score-P is dependent on the function calls in the program.

However, the cumulative values of energy consumption are collected and calculated with a sample rate of 1000 times a second. Therefore, to compare Score-P values with rapl_plot, we choose the same sampling rate for rapl_plot as well.

In order to compare the reported summary results by CUBE, the graphical user interface provided along with Score-P, with the rapl_plot results, we have to start the rapl_plot application at the same time we start the HPCC benchmark.

TABLE I. ACCURACY OF SCORE-P TOOL IN DIFFERENT TEST RUNS

#	*Test*		*Power Consumption Overhead*
1	**HPL**		6%
2	**DGEMM**	Single	22.6%
		Star	47.3%
3	**FFT**	MPI	13.2%
		Single	31.7%
		Star	28.9%
4	**LatencyBandwidth**		34.9%
5	**PTRANS**		29.5%
6	**RandomAccess**	MPI	1.7%
		Single	24.4%
		Star	50.9%
7	**RandomAccess_LCG**	MPI	1.9%
		Single	21.8%
		Star	46.1%
8	**STREAM**	Single	-23.3%
		Star	49.2%

Table I displays the accuracy of the Score-P measurements in terms of the overhead in different test runs.

The following formula shows how the overhead is measured in table I, where x shows the total power consumption calculated from the rapl_plot results and y shows the summary result reported by Score-P: $Overhead(\%) = (y - x) * 100/x$. The total power consumption from the rapl_plot data is calculated by: $TotalPower(w) = (\sum w_i) * \Delta t/T$, where w_i represents the periodically read values from sensors. Δt represents the sample rate and T shows the total run time.

As seen in the table, the power measurements overhead of Score-P varies from 1.7% to 50.9%. Just in one case we observed a smaller power consumption than in rapl_plot, i.e. for the STREAM subtest with Single version (-23.3%). In essence the Score-P results have different level of accuracy, reflected in the different values of overhead, and this accuracy is application-dependent.

Still it is possible to benefit from the results. Using CUBE one can retrieve the power consumption for individual sensors and for each function call, which is not possible in SLURM.

VII. DISCUSSION

According to our findings, the results reported by SLURM are very close to the aggregated rapl_plot data. Therefore, SLURM is very suitable for discovering the trend of power consumption of an application. To give more informative feedback to the user, additional visualization tools can be used as add-ons to SLURM; these tools will produce plots based on the output files. A generic user, who is interested in the total power consumption, can benefit from the SLURM output data that is presented in the *Totals* subtree of the HDF5 file. This is helpful when the user is concerned about the power consumed by the various resources in the infrastructure, and not in the behavior of the different application components. A software developer, who is more concerned about the application itself, can make use of *Time Series* provided by SLURM. Still, it is not possible to see fine-grained information on each sensor.

Score-P introduces varying amounts of overhead to the summary results, which makes the reported data inaccurate. It is possible to create filter files (scorep.filter) in Score-P that contain filtering rules in order to reduce the overhead. Using filter functions Score-P stops instrumenting some parts of the code to restrict the amount of generated data.

TABLE II. EVALUATION OF SLURM AND SCORE-P FOR DIFFERENT TYPES OF USERS WITH RESPECT TO THEIR REQUIREMENTS

Type of the User	*Expertise level*	*Documentation provided*	*Access to support team*	*Easiness to export data*	*Accuracy of provided information*	*Description*
SLURM						
generic user	✗	✓	✓	✗	✓	The user can benefit from the summary information provided by SLURM although the reported information is very coarse-grained.
software developer	✓	✓	✓	✓	✓	The user gets fine-grained information from time series collected from the RAPL sensors.
Score-P						
generic user	✗	✗	✓	✗	✗	Summary results are provided to the user. The summary information by this tool is more fine-grained than the summary information by SLURM as it shows the collected data of the RAPL sensors separately.
software developer	✓	✗	✓	✗	✗	The user can get energy related information for different parts of the application.

However, it is possible to get fine-grained information on which parts of the application contributed how much using CUBE. For the generic user, the expectations are not met because the provided summary results contain errors. On the other hand CUBE fulfills the software developer requirements. Because CUBE displays the contribution of each function call in total power consumption using colors and exact numbers and the software developer, who cares about the behavior of power consumption in the application, benefits from it. Also it is possible to see the collected information from each sensor individually.

Table II summarizes pros and cons of the two investigated profiling tools in cloud-based environments and how the requirements are met for different types of users. Our results show that Score-P does not provide enough accuracy while SLURM output can be considered as precise enough. On the other hand, the reported data from the two profiling tools can be taken into account to find patterns in the power consumption of applications over time or per function call which is of interest of the software developer. In terms of documents and tutorials provided to users, SLURM outperforms Score-P.

VIII. CONCLUSION

We studied two power profiling tools for large scale computing environments from different perspectives. We ran extensive experiments using the HPCC benchmark in the SURFsara infrastructure in order to classify these tools according to the initial users requirements. Both SLURM and Score-P as our target profiling tools are able to make use of RAPL for collecting power consumption data. Furthermore, we collected power measurements directly from the PAPI libraries using the rapl_plot application as our baseline.

Although our findings show that both tools introduce some varying inaccuracy on the reported data compared to rapl_plot, the power measurements are provided to the users in a more structured way. Given that RAPL does not measure power consumption of the entire node, the collected data cannot be considered equivalent to the data provided by external sensors. On the other hand, both tools provide a satisfying precision level of collected data in terms of sensitivity in changes of power consumption over time or per function call.

We provided a thorough assessment of the tools with respect to the user requirements. It is our conclusion that both SLURM and Score-P can be used by different types of users, as long as their shortcomings and advantages are carefully evaluated against the user expectation. As future work, we aim to examine to what extent these profiling tools can be used in order to introduce improvements in running applications.

ACKNOWLEDGMENT

This work has been sponsored by the European Fund for Regional Development under project MRA Cluster Green Software and by the RAAK/MKB-project “Greening the Cloud".

REFERENCES

[1] A. Venkatraman, “Global census shows datacentre power demand grew 63% in 2012,” October 2012, [Online; accessed 2-July-2014].

[2] V. Weaver, D. Terpstra, H. McCraw, M. Johnson, K. Kasichayanula, J. Ralph, J. Nelson, P. Mucci, T. Mohan, and S. Moore, “Papi 5: Measuring power, energy, and the cloud,” in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, April 2013, pp. 124–125.

[3] V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, “Measuring energy and power with papi,” in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*, Sept 2012, pp. 262–268.

[4] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, “Power-management architecture of the intel microarchitecture code-named sandy bridge,” *Micro, IEEE*, vol. 32, no. 2, pp. 20–27, March 2012.

[5] D. Hackenberg, T. Ilsche, R. Schone, D. Molka, M. Schmidt, and W. Nagel, “Power measurement techniques on standard compute nodes: A quantitative comparison,” in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, April 2013, pp. 194–204.

[6] A. B. Yoo, M. A. Jette, and M. Grondona, “Slurm: Simple linux utility for resource management,” in *Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 44–60.

[7] Y. Georgiou, T. Cadeau, D. Glesser, D. Auble, M. Jette, and M. Hautreux, “Energy accounting and control with slurm resource and job management system,” in *Distributed Computing and Networking*. Springer, 2014, pp. 96–118.

[8] Y. Liu and H. Zhu, “A survey of the research on power management techniques for high-performance systems,” *Software: Practice and Experience*, vol. 40, no. 11, pp. 943–964, 2010.

[9] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, *et al.*, “Score-p: a joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir,” in *Tools for High Performance Computing 2011*. Springer, 2012, pp. 79–91.