An experimental comparison of software-based power meters: focus on CPU and GPU

Mathilde Jay, Vladimir Ostapenco, Laurent Lefèvre, Denis Trystram, Anne-Cécile Orgerie, Benjamin Fichel

An experimental comparison of software-based power meters: focus on CPU and GPU

Mathilde Jay*†, Vladimir Ostapenco†, Laurent Lefevre†, Denis Trystram*, Anne-Cécile Orgerie‡, Benjamin Fichel§
* *Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG, Grenoble, France*
{mathilde.jay,denis.trystram}@univ-grenoble-alpes.fr
† *Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, Lyon, France* - {laurent.lefevre,vladimir.ostapenco}@ens-lyon.fr
‡ *Univ. of Rennes, Inria, CNRS, IRISA, Rennes, France* - anne-cecile.orgerie@irisa.fr
§ *OVHCloud, France* - benjamin.fichel@ovhcloud.com

***Abstract*—The global energy demand for digital activities is constantly growing. Computing nodes and cloud services are at the heart of these activities. Understanding their energy consumption is an important step towards reducing it. On one hand, physical power meters are very accurate in measuring energy but they are expensive, difficult to deploy on a large scale, and are not able to provide measurements at the service level. On the other hand, power models and vendor-specific internal interfaces are already available or can be implemented on existing systems. Plenty of tools, called software-based power meters, have been developed around the concepts of power models and internal interfaces, in order to report the power consumption at levels ranging from the whole computing node to applications and services. However, we have found that it can be difficult to choose the right tool for a specific need. In this work, we qualitatively and experimentally compare several software-based power meters able to deal with CPU or GPU-based infrastructures. For this purpose, we evaluate them against high-precision physical power meters while executing various intensive workloads. We extend this empirical study to highlight the strengths and limitations of each software-based power meter.**

***Index Terms*—Energy measurement, Power measurement, Software evaluation, Experimental comparison**

I. Introduction

The usage of information and communication technologies (ICT) continues to drastically grow due to the democratization of artificial intelligence (AI), web services, Cloud storage, and other applications like online entertainment and cryptocurrencies. In 2021, the Shift Project [1] pointed out that the part of digital activities in the global energy demand was already 6% in 2019 and had been increasing by 6.2% per year from 2015 to 2019. The part of digital activities in the global greenhouse gas emissions was 3.5% in 2019, which was more than civil aviation.

As the global available energy is limited and the demand for ICT technologies is in constant growth, the energy efficiency of data centers and cloud services is a growing concern. In such a perspective, developers, users, and data center managers need to better understand the impacts of their activities and be aware of their environmental footprint. Whether it is a data scientist trying to understand the energy consumption of her/his GPU while performing AI model training, a data center manager wanting to perceive the consumption of CPU and RAM-intensive cloud workloads, or a user wishing to understand the power consumption of her/his laptop, energy and power measurement needs are numerous. The most mature and less intrusive way to estimate the power consumption of a computing node is through the use of physical power meters. But they require the deployment of an additional measuring infrastructure. In addition, data must be collected and made available using specific software often imposed by the power meter supplier. Finally, a physical power meter only measures the overall consumption of the computing node. It does not detail the consumption of the various components or services launched on this computing node.

On the other hand, numerous power models and internal interfaces have been developed in order to provide consumption metrics at multiple levels. They are able to provide energy consumption data with granularities ranging from the overall consumption of the computing node to the consumption of a single operating system process. Moreover, these technologies are already available or can be implemented on already existing systems, not requiring any additional financial investment or specific hardware setup.

Nevertheless, implementing power models and using these internal interfaces directly to retrieve power consumption metrics requires detailed knowledge of the underlying hardware being used. A multitude of tools has been developed in order to facilitate this task. In this paper, we call these tools *software-based power meters* and divide them into three categories: energy calculators, energy measurement software, and power profiling software. Energy calculators estimate energy consumption using thermal design power (TDP)-based modeling. Energy measurement and power profiling software can report respectively the energy and power consumption of the CPU (Central processing unit), DRAM (Dynamic random-access memory), and/or GPU (Graphics processing unit) as retrieved from the internal interfaces. Some of them additionally implement power models in order to give an estimation of the consumption at the level of a process, a container, or a virtual machine.

However, each tool is not equal in terms of the available feature set, the supported sampling rates, or the quality of the estimation. We have found that it can be difficult to choose

the right tool for a specific need.

The purpose of this paper is to explore and compare a selection of software-based power meters. We study them from various angles such as hardware compatibility, underlying technologies used, estimation models used, intrusiveness, quality of estimation, quality of documentation, their strengths, and their limitations. In order to evaluate some characteristics such as the quality of the estimation and the intrusiveness, we executed a set of benchmarks on a computing node equipped with high-precision external power meters.

The remainder of this article is organized as follows. Section II provides background information on methods to measure the energy consumption of computing nodes, as well as introduces the internal interfaces and power models used by the studied tools. Section III presents the existing tools and which ones have been selected for this study. It also includes a qualitative comparison of this selection. Section IV describes the methodology we followed to experimentally evaluate the software-based power meters. Finally, in Section V, we analyze the obtained results and make a quantitative analysis of the studied tools before concluding in Section VII.

II. Background and related works

This section begins with an overview of existing methods for measuring the energy consumption of computer nodes at different granularities, followed by a summary of related works.

A. Methods to measure the energy consumption

1) External Devices: External devices, commonly known as power meters, are equipment that is not embedded in computing nodes. They are generally positioned between the wall socket and the power supply unit of a computing node. These devices include external power meters and Power Distribution Units (PDUs) with measuring capabilities. Nowadays, a large number of external power meters are available on the market, such as OmegaWatt, Raritan intelligent power distribution units, and Eaton power meters. These devices measure the consumption of the entire computing node and have little impact on the monitored system. However, their performance (accuracy, measurement rates) can vary depending on the device and they require an additional financial investment.

2) Intra-node devices: Intra-node devices can be defined as equipment placed inside computing nodes. These type of devices includes Baseboard Management Controllers (BMC) embedded in computing nodes, devices placed between a computing node's power supply and main board as PowerMon2 [2], and devices for component-level instrumentation as PowerInsight [3]. These devices can provide consumption of individual computing node components, but also need an additional financial investment and lack user-friendliness.

3) Hardware sensors and software interfaces: Computing node vendors and component manufacturers embed digital sensors, onboard measurement circuits, and interfaces that measure the power consumption of the entire system, the processor socket, the memory, and other computing node components.

a) Intel CPU RAPL: The RAPL (Running Average Power Limit) interface was introduced by Intel in 2011 [4] in the Intel Sandy Bridge architecture. The first implementation of RAPL used a software power model in order to estimate energy usage based on "a set of architectural events from each Intel architecture core, the processor graphics, and I/O" [5]. The second implementation of RAPL, introduced with the Haswell architecture, is based on fully integrated voltage regulators and enables actual power measurement, improving the accuracy of RAPL measurements [6].

RAPL reports energy consumption and is able to limit the power consumption on different levels or power domains: entire CPU socket (PKG), all CPU cores (PP0), integrated graphics (PP1), dynamic random-access memory (DRAM), and entire SoC (PSys). The availability of power domains may vary between architectures and processor models. RAPL energy consumption reports can be accessed through MSR (Model-specific registers). The values in these registers are expressed in energy units and represent the energy consumed in microjoules since the processor was started.

RAPL registers have a high update frequency and low-performance overhead [7]. Their energy counters are updated approximately every 1 ms (1000 Hz). RAPL is an always-running interface and starts to work when the processor boots. However, the RAPL interface lacks detailed low-level implementation documentation. Thus the exact methodology of the RAPL calculations remains unknown. The RAPL registers are not updated precisely every 1 ms [8], have no timestamps attached [9], and are updated in non-atomic way [7]. The RAPL registers are limited to 32 bits and can overflow. This must therefore be taken into account when reading the RAPL values directly.

In the Zen architecture (17h family), AMD introduced a version of RAPL [10] whose implementation is similar to the first software model-based Intel RAPL and can show inconsistent results [11].

b) Nvidia GPU NVML: NVIDIA provides users with an API called NVIDIA Management Library (NVML) [12]. It provides access to GPU device metrics such as current utilization, temperature, and power draw. According to the official documentation, the method `nvmlDeviceGetPowerUsage()` returns the current power draw within an accuracy of 5% of the GPU and its associated circuitry (e.g. memory). Retrieval of current power draw is supported by Nvidia GPUs of the Fermi generation and newer. Even if there is no information on whether the power is measured or estimated, several works mention that it is measured from onboard sensors [13], [14].

4) Power and energy modeling: Power and energy modeling is useful when the previously discussed technologies are not available or don't provide enough detail. In this section, we only describe methods relevant to our study, while Lin et al. [15] provides a more detailed overview of existing power models.

a) Usage-based modeling: M. Etinski, J. Corbalan, J. Labarta, and M. Valero [16] show that the computing node

power consumption is significantly impacted by running workload and CPU frequency. Therefore, the CPU utilization is correlated to the computing node's power [17]. This has led to the emergence of multiple computing node power models based on resource usage. Some models are linear and based on the assumption that power consumption increases linearly as CPU usage increases [18]. But since the global computing node power consumption is not always linearly related to CPU utilization [19], other models attempt to add non-linearity in power models to reduce estimation errors [17], [18], [20].

The thermal design power (TDP) is provided by the manufacturer and represents the maximum amount of heat generated by a component under a steady workload. Even though the actual power consumption can exceed the TDP [21], [22], it can be used as a good approximation of component power consumption at maximum use. The total CPU power consumption can be estimated as the product of the TDP, the average CPU usage, and the total execution time.

b) Process-level modeling: In cloud environments, computing resources are executing multiple workloads simultaneously usually in the form of Virtual Machines (VM) and containers. This requires the ability to estimate consumption at the process level.

Usage-based process-level modeling: Because RAPL energy meters contain the overall energy consumption of a specific component (CPU, DRAM, Integrated GPU), the CPU utilization can be used to estimate the share of energy consumption of each process. Many works [23]–[25] use CPU-utilization-based modeling in order to estimate the consumption of workloads including VMs and containers with encouraging results in terms of precision. Another study shows the limits of this approach by pointing out that the CPU utilization rate directly calculated by the kernel often lacks precision [26].

Performance event based regression modeling: Another promising way to estimate the energy consumption of a process is to use RAPL counters in combination with regression modeling based on hardware performance counter events. Some examples of hardware performance counter events are CPU clock cycles, the number of instructions retired, and cache misses. This type of modeling may be self-calibrated using RAPL metrics as ground truth [27] or may require an initial calibration procedure with a physical power meter [28] in order to estimate the energy consumption of a process.

B. Related works

The study [29] the most related to our work proposes an experimental evaluation of a few tools on NLP models. Several surveys [30], [31] reference existing models, technologies, and tools, while not making any experimental comparison. F. Almeida et al. [32] focus on physical power measurement devices and power modeling approaches, which sometimes require prior training before use. Two studies [33], [34] respectively published in 2012 and 2013 compare tools that are only based on energy modeling approaches. None of these works present an experimental comparison of recent tools which can be used without any type of additional physical equipment, are suitable for cloud environments, support GPU consumption metrics, and report metrics at the level of a process.

III. Software-based power meters

Most of the methods introduced in Section II require either specific hardware equipment or detailed knowledge of the underlying hardware being used. They can't be used as they are and often require additional implementation. For example, integrated technologies need to be queried and processed in order to get energy metrics. Fortunately, software packages or programs were developed to simplify the process for the users.

A. Selected tools

We selected tools able to estimate energy consumption with varying levels of granularity and representative of the set of features offered by existing tools. We compare these tools in Table I. The version of each tool studied is indicated in parentheses.

1) Energy calculators: Energy calculators are software-based power meters available on the web relying on TDP usage-based modeling to compute the energy consumed by equipment.

ML CO2 impact (version of Jul 5, 2022) [35] was developed for cloud-based experiments and only supports one piece of equipment at a time. **Green Algorithms** (version 2.2) [36] has more parameters and provides the user with more metrics than the carbon footprint.

2) Energy measurement software: Energy measurement software packages return the total energy consumed by the computing node during the execution of the program. The three software packages we selected are all Python packages based on Intel RAPL and Nvidia NVML interfaces.

Carbon Tracker (version 1.1.6) [37] can be integrated into a machine learning model training to predict the energy consumed by the whole training from the training of one epoch. **Code Carbon** (version 2.0.0) [38] automatically retrieves the TDP coefficient from an internal database if Intel RAPL and Power Gadget are not available. **Experiment Impact Tracker** (version of June 4, 2021) [39] distinguishes itself by separating the consumption of the Python script execution process from the rest of the system.

3) Power profiling software: Finally, we study tools returning the power profile of a program and that are based on the Intel RAPL and/or Nvidia NVML interfaces.

PowerAPI (version 1.0.7) [40] is an open-source toolkit for building power meters. Smartwatts (version of Feb 28, 2022) is a software-based power meter implemented with PowerAPI able to estimate consumption at a process level. Smartwatts process-level estimates are based on performance events generated by each monitored process. **Perf** (version 5.4.203) is a command-line utility designed for Linux profiling with performance counters capable of reporting energy consumption gathered from RAPL. **Scaphandre** (version 0.4.1) [41] is a monitoring agent able of reporting power consumption at the process level based on usage-based process-level modeling and CPU utilization metrics. **Energy Scope** (version of January

2022) [42] is an energy monitoring and application profiling software package taking into account the power consumption of Nvidia GPUs.

B. Other available tools

nvidia-smi, **Likwid**, [43] **PowerTOP**, and **powerStat** are command line tools highly similar to those we selected in their functionality. **powermetrics** and **Intel Power Gadget** are supported only by MacOS or Windows operating systems. **PAPI** [44] is an interface to power-related performance counters. Other tools such as **PyJoules** [45] and **Cumulator** are Python packages highly similar to those studied in this work. **Kepler** is another promising software-based power meter developed to work exclusively in the Kubernetes environment, which is beyond the scope of this work.

C. Comparison criteria

We describe the various criteria we used to qualitatively compare the selected software-based power meters in Table I.

1) Development: By whom and when the tool was developed.

Citation Who developed it.

First / latest release date When the tool was released for the first time and when was made the latest release (as of 6 September 2022).

2) Environment: What are the hardware requirements and in which environment they can be used.

Hardware compatibility Which technologies must be available at the computing node to run tools.

Scope At which granularity the solution provides consumption values. Levels: machine, resource (CPU, GPU, DRAM(CPU)), cgroups, processes.

Virtualization support Whether the tool can be deployed and used in a virtualized environment.

Job management system support Whether the tool has built-in job management system support (OAR, SLURM).

3) Functionality: How each tool works internally.

Hardware technology used The technology used to measure or estimate energy consumption. For example RAPL, and NVML.

Software power model used If a model is used on top of the hardware technology. For example, to estimate the energy consumed at the process level.

Default sampling frequency At which frequency the solution samples energy consumption values by default.

Online reporting If the data are available in real-time or if it is provided only at the end of the execution.

Power profiling If the solution has power profiling capabilities in form of time series or if the solution reports only total resulting energy consumption.

4) User-friendliness: How easy it is to use and configure.

Configurability "Poor", "Fair" or "Good" depending on the number of configurable parameters supported among the following list: acquisition frequency, result data form, used model, and result data (power, energy, or carbon emission).

Availability of source code Whether the source code of the tool can be found online and with what license the tool is distributed.

Ease of use How easy it is to install and use the solution. "Poor" if one needs an understanding of the architecture of the tool or its environment to be able to configure it and collect its results. "Fair" if the tool doesn't need any architecture-dependent configuration but an additional mechanism is required to retrieve results. "Good" if no configuration nor additional mechanism is needed to retrieve results. "Very good" if no installation or software skill are required.

Quality of documentation "Poor" if the documentation is not sufficient to use and configure the tool. "Fair" if the documentation is sufficient, but an effort is needed to understand how it works. "Good" if the available documentation addresses usage questions such as parameter settings [29].

Resulting data format In which format the result data are provided. For example, as a value stored in a code variable (Code) or written into a database back-end (Prometheus, InfluxDB, MongoDB, Riemann, Warp10), sent via a socket connection (Socket), a text file (JSON, CSV, Latex), available on a web page (Web) or just printed in the standard output (Stdout).

Data visualization possibilities Dashboards (Grafana, Comet, custom) or online web resources.

IV. Methodology

After qualitatively comparing the software-based power meters, we conducted a group of experiments to verify how the selected tools work and evaluate the quality of their outputs.

A. Environment

1) Infrastructure: We executed all experiments on a machine from the Gemini cluster of large-scale test beds for experimental research called Grid'5000 [47]. This cluster was selected because it contains nodes with multiple recent GPUs supporting power metric retrieval with Nvidia NVML, two CPUs with the second implementation of Intel RAPL, and high-performance external power meters.

The nodes in this cluster have the following specifications:

- System model: Nvidia DGX-1
- CPU: 2 x Intel Xeon E5-2698 v4 (Broadwell, 2.20GHz, 20 cores/CPU)
- Memory: 512 GiB
- GPU: 8 x Nvidia Tesla V100-SXM2-32GB (32 GiB)

a) External power meter: The power consumption of each node in the Gemini cluster is individually monitored by an Omegawatt [48] power meter. This power meter has a maximum sampling frequency of 50 Hz and a precision of 0.1 W. We used them with a sampling frequency of 1 Hz.

b) BMC: In the Gemini cluster, each node has a BMC that reports the power consumption of the entire host system with a sampling frequency of 0.2 Hz.

TABLE I: Qualitative comparison of selected software-based power meters.

	External and intra-node devices		**Power profiling software**				**Energy measurement software packages**			**Energy calculators**	
	OmegaWatt	***BMC***	***Power API***	***Scaphandre***	***Energy Scope***	***Perf***	***Code Carbon***	***Experiment Impact Tracker***	***Carbon Tracker***	***Green Algorithms***	***ML CO2 Impact***
Developement											
Citation			[40]	[41]	[42]	[46]	[38]	[39]	[37]	[36]	[35]
First (latest) release date			Jul. 2019 (Aug. 2022)	Dec. 2020 (May 2021)	2021	Sept. 2009 (Jan. 2023)	Nov. 2020 (Sept. 2022)	Dec. 2019 (Jan. 2020)	Apr. 2020 (Jul. 2021)	Jul. 2020 (Jun. 2022)	Aug. 2019 (Jul. 2022)
Environment											
Hardware compatibility	Any	Any	Intel RAPL	Intel RAPL	Intel RAPL, Nvidia NVML	Intel RAPL	Any	Intel RAPL, Nvidia NVML	Intel RAPL, Nvidia NVML	Any	Any
Scope	Machine	Machine	CPU, DRAM, process	CPU, DRAM, process	CPU, DRAM, GPU	CPU, DRAM	CPU, DRAM, GPU	CPU, DRAM, GPU, process	CPU, DRAM, GPU	Machine	Machine
Virtualization support			Yes	Yes	No	No	No	No	No		
Job management support			No	No	OAR, SLURM	No	No	No	No		
Functional											
Hardware technology used			RAPL	RAPL	RAPL, NVML	RAPL	RAPL, NVML, TDP	RAPL, NVML	RAPL, NVML	TDP	TDP
Software power model used			Regression based on perf events	CPU usage based				GPU, CPU and RAM usage based			
Default sampling frequency (Hz)	1	0.2	1	0.1	2	10	1/15	1	0.1		
Online reporting	Yes	Yes	Yes	Yes	No	Yes	No	No	No	No	No
Power profiling	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No
User-friendliness											
Availability of source code (License)			Yes (BSD 3-Clause)	Yes (Apache 2.0)	No	Yes (GNU GPL)	Yes (MIT)	Yes (MIT)	Yes (MIT)	Yes (CC-BY-4.0)	Yes (MIT)
Ease of use	Poor	Poor	Poor	Fair	Good	Good	Good	Good	Good	Very good	Very good
Quality of documentation			Good	Good	Good	Good	Fair	Fair	Good	Good	Fair
Configurability	Fair	Poor	Good	Good	Good	Good	Poor	Fair	Poor	Poor	Poor
Resulting data format	HTTP end-point	HTTP end-point	MongoDB, InfluxDB, Prometheus, CSV, Socket, File	Prometheus, Warp10, Riemann, JSON, Stdout	JSON	CSV, Stdout, File	CSV	JSON, Code	File, Code	Web	Web, Latex
Data visualisation possibilities	Grafana (Kwollect)	Grafana (Kwollect)	Grafana (InfluxDB, Prometheus)	Grafana (Prometheus)	Custom Dashboard		Comet			Graphs on the web page	

2) Operating system and configuration: For all experiments, we used the minimal variant of Ubuntu 20.04 available on the Grid'5000 testbed. In order to increase consumption stability and the consistency of our results, we have disabled Hyper-Threading and Turbo-Boost technologies and set the CPU frequency to the maximum supported. We also installed an Nvidia GPU driver when relevant, with default power management configuration.

B. Selected benchmarks

In order to evaluate software-based power meters, we executed benchmarks representative of typical workloads, well known by the community, and implemented for both CPU and GPU. For this purpose, we chose the NAS parallel benchmarks in their classic implementation [49] and implemented for GPU in CUDA [50].

We selected three NAS benchmark kernels in order to simulate an intensive use of different computing node components. The first kernel is the EP (Embarrassingly parallel) kernel that generates pairs of Gaussian random deviates thus making intensive use of the CPU (and respectively GPU). The second kernel is the MG (Multi-Grid) kernel. This kernel performs a V-cycle multigrid algorithm and tests both short and long-distance data communication, thus is memory intensive. The third kernel is the LU (Lower-Upper Gauss-Seidel solver) kernel. This kernel is a pseudo-application that performs a synthetic computational fluid dynamics (CFD) calculation. LU is less CPU intensive than EP but also uses memory.

In the process level estimation study, we also used the IS (Integer Sort) kernel, which is similar to MG but performs memory operations differently.

Every parallel NAS benchmark has a class that can be thought of as a problem size. For instance, for the MG benchmark, classes from A to E will have increasing grid size, iteration number, and therefore execution time. The class for each benchmark kernel was chosen empirically for our experiments in order to have suitable execution times. Therefore, the classes of the chosen CPU and GPU benchmarks differ.

C. Experiment methodology

We evaluated the consumption results given by the software power meters listed in Section III while executing the NAS benchmark kernels presented in Section IV-B. We selected a one-minute interval between each benchmark execution to let the computing node component cool down after each benchmark run and prevent the power consumption of subsequent executions from being impacted.

1) Tools configuration: Most of the software-based power meters studied in this work were used with the default configuration. However, some of them have multiple configuration possibilities or the default values are not suitable for our experiments. We used **PowerAPI** with Hwpc-sensor as a sensor, Smartwatts as a formula, and MongoDB as a database. The Hwpc-sensor and Smartwatts formula was configured with a sampling frequency of 1 Hz. We also enabled the DRAM formula in Smartwatts and configured the Hwpc-sensor to collect and send DRAM RAPL events to the database. **Scaphandre** was used with the Prometheus exporter, which exposes power metrics on an HTTP endpoint. The default power data fetch interval of 10 seconds is insufficient in the context of our study, so we configured Prometheus to retrieve power data every five seconds (0.2 Hz). **Perf** was used with a sampling frequency of 10 Hz and with a CSV file as the data output. **Experiment Impact Tracker** and **Carbon Tracker** multiply the computed energy by a Power Usage Effectiveness (PUE) ratio by default. It was systematically removed.

2) Benchmarks execution: For the CPU benchmarks, the NAS benchmarks were launched on all cores of both CPUs by keeping the default thread number configuration. Each GPU NAS benchmark instance can be launched on only one GPU at a time. We ran a separate NAS benchmark instance on each GPU.

3) Reproducibility: In order to make the results more reliable, all the experiments were carried out ten times. We have automated the execution of all the experiments as well as the processing of their results. The source code and the experiment results are available and are described in Appendix A.

To be transparent about the impact of this work, we calculated the total energy consumed during all experiments and tests using the Omegawatt [48] external power meter. We obtained a total consumption of nearly 480 kWh.

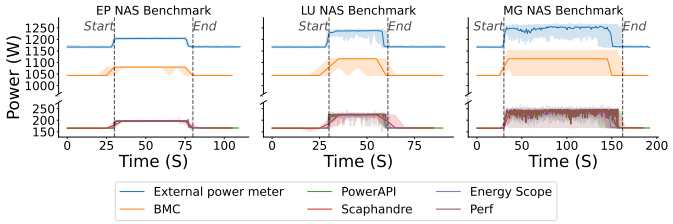
V. Results

A. Total computing node consumption

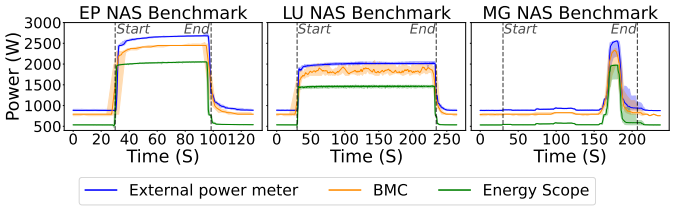
1) Power profile: Energy Scope, PowerAPI, Scaphandre and Perf are the only tools providing us with power profiles.

Fig. 1a displays the CPU benchmark profiles as given by the external power meter, the BMC, Energy Scope, PowerAPI, Scaphandre and Perf while Fig. 1b shows the GPU benchmark profiles as reported by the external power meter, the BMC and Energy Scope. To simplify our graphs, only the minimum, maximum, and median of the ten power values obtained for each timestamp are represented.

In both figures, we can see that the evolution of the Energy Scope, PowerAPI, Scaphandre and Perf power profiles are visually similar to the evolution of the external power meter and BMC profiles, despite a non-negligible offset between the three instruments. This offset depends on which component of the computing node is included in the reports of each tool. The external power meter is installed between the computing node and the wall socket. Thus, it reports the overall consumption of the computing node with all its components. The BMC is installed internally after the power supply unit (PSU) unit and the reporting scope differs according to its implementation. The values given by BMC are therefore predictably lower. The software-based power meters studied in this section are based on Intel RAPL and Nvidia NVML, which only include the consumption of the CPU, DRAM, or/and GPU components. The consumption of other components, such as fans, storage, and network interfaces, is not included. CPU benchmarks are carried out on computing nodes containing several idle



(a) [CPU Benchmarks]



(b) [GPU Benchmarks]

Fig. 1: Power profiles provided by Energy Scope, Scaphandre, Perf, PowerAPI, BMC, and the External power meter on both the CPU and GPU benchmarks.

GPUs whose consumptions are not included in the software-based power meter profiles. As a result, the offset with the external power meter values is much larger than for the GPU benchmarks.

Both figures show that the evolutions of the benchmarks are well captured by each tool. However, each tool captures them differently. Tools with a higher acquisition frequency like Perf, Energy Scope, PowerAPI, and external power meter better capture consumption changes when running the benchmark, resulting in a more precise power profile. While tools with low acquisition frequency like Scaphandre and the BMC are not able to capture all consumption details and will therefore give a less accurate power profile.

In Fig. 1b, we notice that the power reported by the external power meter does not go down instantly at the end of the computation phase, notably for the EP and LU benchmarks. The power takes some time to return to idle values. This additional consumption could be due to the fans running at high speed in order to cool the components after the execution of the benchmark. Since the power profile given by software-based power meters excludes fan consumption, this phenomenon is not observed by Energy Scope.

2) Correlation and offset with external power meter: We previously observed that the software-based power profiles are similar and are visually strongly correlated with the external power meter and the BMC profiles. We will study this correlation in more depth in this section.

The tools do not have exactly the same timestamps and sampling frequencies. To do a point-by-point correlation study between power profiles, we had to fit the higher frequencies data points to the lower ones by averaging. To compute the correlation, we used the library Pandas [51] and the default Pearson correlation.

The Pearson correlation coefficient between each tool and the external power meter is around 0.95 for all benchmarks, which is a highly strong correlation. Furthermore, the more stable the execution, the higher the correlation. On the CPU benchmarks, the correlation seems to be impacted by the sampling frequency in most cases. The higher the sampling rate, the stronger the correlation. Energy Scope has a sampling frequency of 2Hz and has the highest correlation (0.972), especially for benchmarks with a more volatile evolution. Scaphandre and PowerAPI have sampling frequencies of 0.5 Hz and 1 Hz, respectively, resulting in correlations of 0.947 and 0.956. On the other hand, Perf with a sampling frequency of 10 Hz exhibits a slightly lower correlation of 0.93 which is probably due to more aggressive averaging.

It can be noticed in Fig. 1 that the offset between the software-based power meters and the external power meter is not constant. To verify this assumption, we computed the regression using the linear regression method of scikit-learn [52]. We have found that the regression slope was respectively 1.17 and 1.18 on the CPU and GPU benchmarks, with the external power meter values being the response variable. We suppose that this offset increase is related to the components whose consumption is not included by the tools, such as the power supply unit and the fans. We believe it can be generalized that the relation between the power reported by the external power meter and the software-based power meters is not constant. Thus, estimating the total power consumption from the power reported by the tools can't be done by only adding a constant offset. Furthermore, this relation cannot be generalized and must be studied for each compute node architecture or even for each individual compute node.

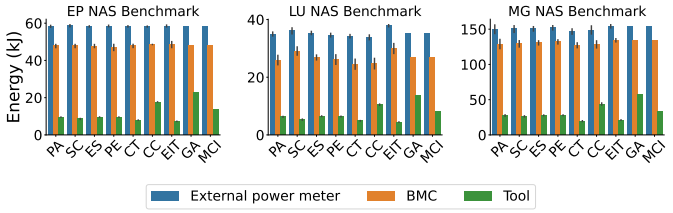
3) Total energy consumption: After studying the power profiles, we looked into the total energy consumed during the execution of the benchmarks. For the external power meter, the BMC and the software-based power meters only supporting the power profile as an output, the total energies are calculated by integrating the power time series.

Fig. 2a and 2b show the total energy in joule spent per benchmark as provided by the tools, the external power meter, and the BMC. The error lines on top of the bars indicate the standard deviation of the energy.

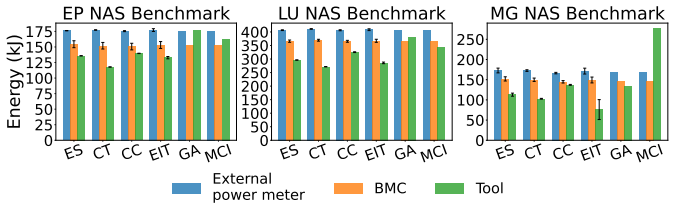
The external power meters report higher energy than the tools and the higher variability of the values provided by BMC leads to a greater standard deviation compared to the external power meter. The standard deviation of the energy reported by the tools is not significant in general which means that the tools are consistent.

For CPU benchmarks, almost all studied tools report similar energy consumption on every benchmark, except Code Carbon and online energy calculators (Green Algorithm, ML CO2 Impact). Code Carbon tends to report power consumption values almost twice as high as other tools based on the same interface for obtaining consumption values (Intel RAPL).

For GPU benchmarks, the differences between tool reports are larger, which is due to a higher difference in sampling frequency (see Table I). Experiment Impact Tracker has a



(a) [CPU Benchmarks]



(b) [GPU Benchmarks]

Fig. 2: Total energy consumed by the benchmarks as reported by the power meters. Tools: PowerAPI (PA), Scaphandre (SC), Energy Scope (ES), Perf (PE), Carbon Tracker (CT), Code Carbon (CC), Experiment Impact Tracker (EIT), Green Algorithm (GA), ML CO2 Impact (MCI)

higher variability on all benchmarks but especially on the MG benchmark.

For all benchmarks, the online calculators (Green Algorithm, ML CO2 Impact) tend to report energies closer to the external power meters than the tools based on RAPL and NVML, but not on the MG NAS benchmark. This suggests that those calculators work better for constant workloads too, and when the average usage is known.

4) Overhead:

a) Energy overhead: To determine the energy overhead of each tool, we executed the selected benchmarks without any software-based power meter running. We compare the consumption values reported by the external power meter with and without software-based power meters. The energy overheads of each studied tool are mostly insignificant. We obtain an energy overhead under 1% on average with a maximum of around 2% for both CPU and GPU benchmarks. Only Scaphandre (0.85%), Carbon Tracker (0.76%), and Experiment Impact Tracker (0.68%) are, on average, slightly outside the expected range of variation.

b) CPU overhead: In order to determine the overhead in CPU of each software-based power meter, we isolate a node CPU core from the operating system's task scheduler. Each tool was launched exclusively on that core while we tracked the utilization rate of that core. We were only able to assess the CPU overhead of Scaphandre and PowerAPI since only these two solutions have a daemon that can be assigned to a particular CPU core. All the studied solutions have an almost negligible CPU overhead (under 2%) while operating with default sampling frequencies. The maximal CPU overhead observed for PowerAPI is 3.7% with the maximum supported sampling frequency of 10 Hz. Scaphandre has a higher CPU overhead at the same sampling frequency. At a frequency of 1 Hz, the maximum CPU overhead of Scaphandre is close to 6% while that of PowerAPI is less than 1%. This is most likely due to the fact that Scaphandre is the only solution that performs all estimation calculations on the host machine.

5) Maximum supported sampling frequency: PowerAPI, Scaphandre, Perf and Energy Scope are the only software-based power meters supporting the configuration of the sampling frequency by the user. For each tool, we started our experiments with the default sampling frequency and gradually increased it until the studied tool stopped giving results, started having cumulative delays, or until the configuration is no longer possible.

Among the solutions studied, Perf and Energy Scope support the highest maximum sampling frequencies. Energy Scope reports values without any issues with the frequency up to 50Hz. Perf supports 1000 Hz, but according to the official manual page [53], using such a high sampling frequency can severely impact tool overhead. PowerAPI is able to give online (or real-time) energy consumption values for a single process at 10 Hz. Additionally increasing the sampling frequency leads to delays in its estimates. Scaphandre supports 0.5 Hz at the highest, after which it stops returning values. The maximum sampling frequency seems to depend on the used exporter. We tested the Stdout and JSON exporters and they give estimates with a frequency of up to 1Hz.

B. Computing node components consumption

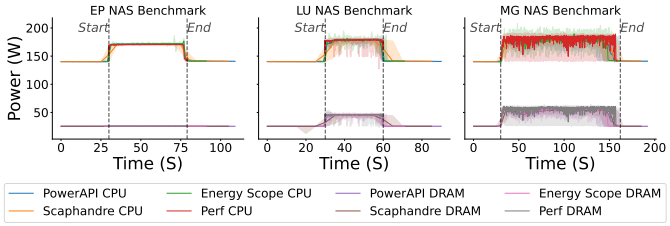
Power API [40], Scaphandre [41], Perf [46], and Energy Scope [42]have an additional feature to provide us with power profiles of the following components of the computing node: CPU, DRAM, and GPU (only for Energy Scope).

Fig. 3b and Fig. 3a show the power profiles for every CPUs, DRAMs and GPUs as provided by the above-mentioned software. The CPU benchmark power profiles are highly similar. The differences in the power profiles are due to different sampling frequencies. We can also mention that the DRAM consumption reported by each tool seems to reflect the actual DRAM usage by each benchmark. For memory-intensive benchmarks such as LU and MG, we see the change in DRAM consumption reports while the benchmark is running. Whereas, for the EP benchmark which uses no memory, the DRAM consumption data remain unchanged.

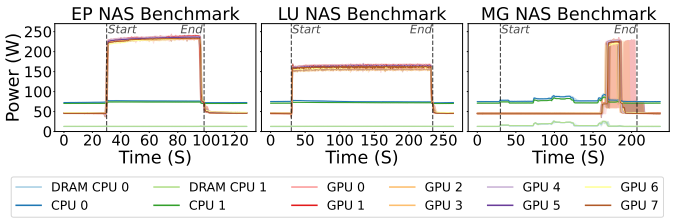
The GPU benchmark profiles confirm that the benchmarks are GPU intensive, except for MG which has a phase when only the CPU and the DRAM are working. Within this phase, the intensity in memory seems to prevent the GPUs from working at maximum utilization.

C. Application process consumption

The only tools supporting power estimation at individual process granularity are PowerAPI and Scaphandre. PowerAPI uses the performance event-based regression modeling described in Section II-A4b while Scaphandre uses the usage-based process-level modeling described in Section II-A4b. We performed several experiments to reveal similarities and



(a) [CPU Benchmarks] Profiles of total CPU and DRAM power consumption as provided by Energy Scope, Scaphandre, and PowerAPI.



(b) [GPU Benchmarks] Power profiles for all CPUs, GPUs and DRAMs from Energy Scope.

Fig. 3: Power profiles of specific components: DRAMs, CPUs, and GPUs.

differences in how each solution estimates consumption at the process level.

We first compare the power profiles and power consumption values reported by the tools for a single process. Then, we evaluate how each tool estimates multiple processes in parallel.

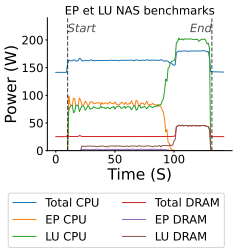
1) Single process: PowerAPI is able to estimate the CPU and DRAM components consumption, while Scaphandre can only estimate the CPU consumption per process. Each software-based power meter gives a similar power profile of the CPU part of the process. However, PowerAPI has a higher sample frequency and its power profile seems to better reflect the actual consumption of a process. As expected, the CPU and DRAM power profiles of the benchmark processes reported by both tools are almost equal to those of the RAPL profiles.

2) Parallel processes:

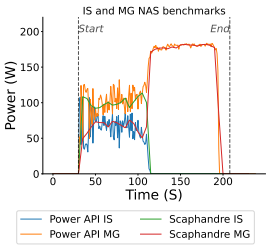
a) Identical benchmarks: First, we study two identical MG benchmarks executed in parallel. PowerAPI and Scaphandre give similar consumption estimates for both benchmarks. For both tools, the sum of the two process CPU energy estimates appears to be equal to the total CPU consumption. The DRAM consumption values reported by PowerAPI also seem consistent, as the sum is also equal to the total DRAM values. We conducted the same experiments with two LU and EP NAS benchmarks launched in parallel and we obtained similar results. PowerAPI and Scaphandre process-level estimates are consistent with each other while executing identical benchmarks in parallel.

b) Different benchmarks: We conducted experiments to evaluate how the studied tools are able to estimate the power of non-identical benchmarks running in parallel that solicit different components of a computing node.

We started by executing EP and LU benchmarks in parallel, one being CPU-intensive and the other both CPU and memory-



(a) Power profile of EP and LU Benchmarks given by PowerAPI.



(b) CPU power profile of MG and IS Benchmarks given by PowerAPI and Scaphandre.

Fig. 4: [CPU Benchmarks] Power profiles of benchmarks launched in parallel given by tools.

intensive. The power profile given by PowerAPI can be seen in Fig. 4a. The DRAM estimation given by PowerAPI doesn't seem consistent as it doesn't begin from the start of the benchmark run and gives non-zero values for the EP benchmark. Moreover, in this experiment, PowerAPI overestimates the CPU consumption of the LU benchmark after the end of the EP benchmark (90 seconds mark). PowerAPI's CPU estimates are even higher than the total consumption retrieved from RAPL. Scaphandre gives more stable estimations and its power profile follows the expected evolution. For clarity reasons, we have decided not to present it in the figure.

We also executed IS and MG benchmarks in parallel, both being memory-intensive while performing non-identical memory operations. The CPU power profile given by PowerAPI and Scaphandre during this experiment can be seen in Fig. 4b. Both tools report stable values, but their estimates are not exactly the same. When two benchmarks run in parallel (at the 30-second mark), PowerAPI estimates that the MG benchmark consumes more than the IS benchmark while Scaphandre gives the opposite estimate.

In conclusion, we have found that while estimations are coherent in most situations, estimates can also highly differ. We cannot conclude which estimation algorithm gave the best estimate because we have no reference.

3) Maximum number of simultaneously profiled processes: The maximum number of simultaneously profiled processes can be defined as the maximum number of processes for which each software-based power meter gives consistent power estimates without any significant delay. To find this number empirically, we gradually increase the number of processes in parallel until the studied tool stops giving results, reports inconsistent results, or starts to have cumulative delays.

We observed that PowerAPI was able to correctly estimate 4 parallel processes. Starting from 5 parallel processes, issues with the estimated values were noticed and PowerAPI completely failed to estimate 6 benchmarks running in parallel with very few returned values. Scaphandre managed to estimate more than 100 processes launched in parallel.

The difference in the number of processes estimated simultaneously between PowerAPI and Scaphandre seems to be due to the fact that Scaphandre uses a much simpler algorithm to

estimate consumption per process. As the number of processes estimated simultaneously by PowerAPI seemed abnormally low to us, we communicated our results to the PowerAPI developers. The PowerAPI team recognized this issue and provided us with a new version of SmartWatts (v0.9.2). We found that the new version is able to estimate nearly 18 processes simultaneously online without any delay.

VI. Lessons learned

Goals in measuring the power or the energy can be numerous. We identify a few scenarios and give recommendations on which tool to select based on our study.

When the objective is to reduce the amount of energy needed to execute a workload, any energy calculator, energy measurement software package, Energy Scope, or Perf can be used since they all report the energy. We have found that the energy calculators Green Algorithm and ML CO2 Impact can give very good estimations on the conditions of having a constant workload and an in-depth knowledge of the execution (duration and average usage). There is no guarantee of the quality of the estimation. The three energy measurement Python libraries are highly similar in their energy reports. Code Carbon has the lowest offset in all experiments. Carbon Tracker computes the energy on the fly to be able to make predictions on the whole program. It is also well documented in our opinion. Experiment Impact Tracker has higher variability in the estimations. Energy Scope and Perf additionally provide the user with a power profile and can be used with any workload as long as it is executed in a Linux environment. Unfortunately, Energy Scope is not open-source at this time. Perf is a command line tool that can be installed through the default package manager on most Linux systems and used without further configuration.

If one wants to monitor power consumption in real time, PowerAPI, Scaphandre and Perf are the only available choices.

No software-based power meter allows one to exactly measure the complete energy or the power consumed by the computing node while executing a workload, as we have shown that the relationship between the external power meter and the tool power is not constant.

Readers could also be interested in monitoring cloud services, for which process-level estimation is required. We have seen that both PowerAPI and Scaphandre have this functionality. They mostly give similar results for process-level estimation with a slight discrepancy when estimating multiple RAM-intensive processes running in parallel. Scaphandre is able to estimate more processes running in parallel and it has a less complicated and more intuitive setup process for working in a virtualized environment.

Whether the workload to study is a long-running job or short execution is of importance. Monitoring short executions demand a high sampling frequency. Energy Scope and Perf have the highest maximum sampling frequencies. PowerAPI, Perf, and Scaphandre allow the configuration of the sampling frequency, with Perf and PowerAPI having the more extensive available ranges.

For workloads relying on GPU computations like machine learning algorithms, Carbon Tracker, Code Carbon, Energy Scope and Experiment Impact Tracker are suitable fits.

Finally, when looking for a tool easy to use and when the precision is not critical, we would recommend Green Algorithms or ML CO2 Impact as they don't require any additional development or tool installation.

VII. Conclusion

In this paper, we propose a comparison of nine software-based power meters: Carbon Tracker [37], Code Carbon [38], Energy Scope [42], Experiment Impact Tracker [39], Perf [46], Power API [40], Green Algorithms [36], ML CO2 Impact [35], and Scaphandre [41]. We detail the existing methods to measure or estimate the energy consumption of a computing node or an application execution. Then we present and compare the selected tools qualitatively. An experimental study investigates the quality of the power profiles and the energy estimations under computation and/or memory-intensive benchmarks. Those experiments allow us to evaluate other characteristics of the software-based power meters, such as their overhead, the maximum sampling frequency, and the maximum number of processes they support. We experimentally validate the consistency of the power or energy consumption reported by the software-based power meters by comparing them with high-performance external power meters. We find that the profiles are highly correlated. We show and explain why the offset between the external power meter and the software-based power meters is significant and not constant. We have found that the software-based power meters based on Intel RAPL and Nvidia NVML can be used to estimate consumption at a fine granularity with low overhead. All studied tools give relatively similar consumption values. The main differences between them are the supported sampling frequencies, the user-friendliness, the environment in which they can be used, and the ability to estimate the power at various granularities.

For future work, we plan to conduct experiments on another type of computing node to generalize our results. We also plan to evaluate software-based power meters while running another set of workloads, for example, machine learning algorithms or cloud workloads.

Acknowledgments

This research is partially supported by the FrugalCloud collaboration between Inria and OVHCloud, by MIAI (ANR-19-P3IA-0003), and by the BATE project (BATE-UGA-REG21A87) of the Auvergne Rhône-Alpes french region. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] T. S. Project. (2021, Mar.) Environmental impact of digital: 5-year trends and 5G governance [Impact environnemental du numérique : tendances à 5 ans and gouvernance de la 5G]. Accessed: 2022-05-30. [Online]. Available: https://theshiftproject.org/wp-content/uploads/2021/03/Note-danalyse_Numerique-et-5G_30-mars-2021.pdf

[2] D. Bedard, M. Y. Lim, R. Fowler, and A. Porterfield, "PowerMon: fine-grained and integrated power monitoring for commodity computer systems," in *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, Mar. 2010, pp. 479–484.

[3] J. H. Laros, P. Pokorny, and D. DeBonis, "PowerInsight - a commodity power measurement capability," in *2013 International Green Computing Conference Proceedings*, Jun. 2013, pp. 1–6.

[4] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "RAPL: memory power estimation and capping," in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, Aug. 2010, pp. 189–194.

[5] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the Intel microarchitecture code-named Sandy Bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar. 2012.

[6] D. Hackenberg, R. Schone, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer, "An energy efficiency feature survey of the Intel Haswell processor," in *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. Hyderabad, India: IEEE, May 2015, pp. 896–904.

[7] K. Khan, M. Hirki, T. Niemi, J. Nurminen, and Z. Ou, "RAPL in action: experiences in using RAPL for power measurements," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, vol. 3, Jan. 2018.

[8] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, "Measuring energy consumption for short code paths using RAPL," *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, pp. 13–17, Jan. 2012.

[9] D. Hackenberg, T. Ilsche, R. Schone, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: a quantitative comparison," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Austin, TX, USA: IEEE, Apr. 2013, pp. 194–204.

[10] AMD, *Processor programming reference (PPR) for AMD family 17h model 01h, revision B1 processors*, Apr. 2017.

[11] R. Schöne, T. Ilsche, M. Bielert, M. Velten, M. Schmidl, and D. Hackenberg, "Energy efficiency aspects of the AMD Zen 2 architecture," *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 562–571, Sep. 2021.

[12] N. Corporation., "Nvidia management library (nvml)," 2019. [Online]. Available: https://developer.nvidia.com/nvidia-management-library-nvml

[13] Y. Arafa, A. ElWazir, A. ElKanishy, Y. Aly, A. Elsayed, A.-H. Badawy, G. Chennupati, S. Eidenbenz, and N. Santhi, "Verified onstruction-level energy consumption Measurement for NVIDIA GPUs," *Proceedings of the 17th ACM International Conference on Computing Frontiers*, pp. 60–70, May 2020.

[14] S. Sen, N. Imam, and C.-H. Hsu, "Quality assessment of GPU power profiling mechanisms," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. Vancouver, BC: IEEE, May 2018, pp. 702–711.

[15] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, "A taxonomy and survey of power models and power modeling for cloud servers," *ACM Computing Surveys*, vol. 53, no. 5, pp. 1–41, Oct. 2020.

[16] M. Etinski, J. Corbalan, J. Labarta, and M. Valero, "Understanding the future of energy-performance trade-off via DVFS in HPC environments," *Journal of Parallel and Distributed Computing*, vol. 72, no. 4, pp. 579–590, Apr. 2012.

[17] C.-H. Hsu and S. W. Poole, "Power signature analysis of the SPECpower_ssj2008 benchmark," in *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*, Apr. 2011, pp. 227–236.

[18] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proceedings of the 34th annual international symposium on Computer architecture*, ser. ISCA '07. New York, NY, USA: Association for Computing Machinery, Jun. 2007, pp. 13–23.

[19] R. Ferreira da Silva, H. Casanova, A.-C. Orgerie, R. Tanaka, E. Deelman, and F. Suter, "Characterizing, modeling, and accurately simulating power and energy consumption of i/o-intensive scientific workflows," *Journal of Computational Science*, vol. 44, p. 101157, Jul. 2020.

[20] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proceedings of the 2008 conference on Power aware computing and systems*, ser. HotPower'08. USA: USENIX Association, Dec. 2008, p. 3.

[21] Intel, "Thermal design power (TDP) in Intel® processors," 2019, Accessed: 2022-05-25. [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000055611/processors.html

[22] I. Cutress, "Why Intel processors draw more power than expected: TDP and Turbo explained," 2018.

[23] M. Colmant, M. Kurpicz, P. Felber, L. Huertas, R. Rouvoy, and A. Sobe, "Process-level power estimation in VM-based systems," in *Proceedings of the Tenth European Conference on Computer Systems*. Bordeaux France: ACM, Apr. 2015, pp. 1–14.

[24] S. S. Tadesse, F. Malandrino, and C.-F. Chiasserini, "Energy consumption measurements in Docker," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, Jul. 2017, pp. 272–273.

[25] J. Phung, Y. C. Lee, and A. Y. Zomaya, "Lightweight power monitoring framework for virtualized computing environments," *IEEE Transactions on Computers*, vol. 69, no. 1, pp. 14–25, Jan. 2020.

[26] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proceedings of the 1st ACM symposium on Cloud computing*, ser. SoCC '10. New York, NY, USA: Association for Computing Machinery, Jun. 2010, pp. 39–50.

[27] G. Fieni, R. Rouvoy, and L. Seinturier, "SmartWatts: self-calibrating software-defined power meter for containers," in *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*. Melbourne, Australia: IEEE, May 2020, pp. 479–488.

[28] F. Bellosa, "The benefits of event: driven energy accounting in power-sensitive systems," in *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, ser. EW 9. New York, NY, USA: Association for Computing Machinery, Sep. 2000, pp. 37–42.

[29] N. Bannour, S. Ghannay, A. Névéol, and A.-L. Ligozat, "Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools," in *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. Virtual: Association for Computational Linguistics, Nov. 2021, pp. 11–21.

[30] S. Labasan, "Energy-efficient and power-constrained techniques for exascale computing," *Semanticscholar: Seattle, WA, USA*, 2016.

[31] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *Journal of Parallel and Distributed Computing*, vol. 134, pp. 75–88, Dec. 2019.

[32] F. Almeida, M. D. Assunção, J. Barbosa, V. Blanco, I. Brandic, G. Da Costa, M. F. Dolz, A. C. Elster, M. Jarus, H. D. Karatza, L. Lefèvre, I. Mavridis, A. Oleksiak, A.-C. Orgerie, and J.-M. Pierson, "Energy monitoring as an essential building block towards sustainable ultrascale systems," *Sustainable Computing: Informatics and Systems*, vol. 17, pp. 27–42, Mar. 2018.

[33] S. Benedict, "Energy-aware performance analysis methodologies for HPC architectures—an exploratory study," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1709–1719, Nov. 2012.

[34] A. Noureddine, R. Rouvoy, and L. Seinturier, "A review of energy measurement approaches," *ACM SIGOPS Operating Systems Review*, vol. 47, no. 3, pp. 42–49, Nov. 2013.

[35] S. Luccioni, V. Schmidt, A. Lacoste, and T. Dandres, "Quantifying the carbon emissions of machine learning," in *NeurIPS 2019 Workshop on Tackling Climate Change with Machine Learning*, 2019.

[36] L. Lannelongue, J. Grealey, and M. Inouye, "Green Algorithms: quantifying the carbon footprint of computation," *Advanced Science*, vol. 8, no. 12, p. 2100707, 2021.

[37] L. F. W. Anthony, B. Kanding, and R. Selvan, "Carbontracker: tracking and predicting the carbon footprint of training deep learning models," *CoRR*, vol. abs/2007.03051, Jul. 2020.

[38] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, and S. Luccioni, "CodeCarbon: estimate and track carbon emissions from machine learning computing," *Zenodo*, 2021.

[39] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, and J. Pineau, "Towards the systematic reporting of the energy and carbon footprints of machine learning," *Journal of Machine Learning Research*, vol. 21, no. 248, pp. 1–43, 2020.

[40] I. SPIRALS, "PowerAPI," 2019. [Online]. Available: http://powerapi.org/
[41] B. Petit, "Scaphandre," 2020. [Online]. Available: https://hubblo-org.github.io/scaphandre-documentation/
[42] EnergyScope, "The EnergyScope general information site," 2022, Accessed: 2022-12-08. [Online]. Available: https://www.linkedin.com/company/energy-scope/
[43] J. Treibig, G. Hager, and G. Wellein, "Likwid: a lightweight performance-oriented tool suite for x86 multicore environments," in *Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*, San Diego CA, 2010.
[44] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore, "Measuring energy and power with PAPI," in *2012 41st International Conference on Parallel Processing Workshops*. Pittsburgh, PA, USA: IEEE, Sep. 2012, pp. 262–268.
[45] I. SPIRALS, "PyJoules," 2019. [Online]. Available: https://pyjoules.readthedocs.io/en/latest/
[46] IBM, "Perf," 2009. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page
[47] D. Balouek, A. C. Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Perez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Cham: Springer International Publishing, 2013, pp. 3–20.
[48] OmegaWatt, "Omegawatt," 2018. [Online]. Available: http://www.omegawatt.fr/
[49] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga, "The NAS parallel benchmarks," *RNR Technical Report*, p. 15, Apr. 1994.
[50] G. Araujo, D. Griebler, D. A. Rockenbach, M. Danelutto, and L. G. Fernandes, "NAS parallel benchmarks with CUDA and beyond," *Software: Practice and Experience*, vol. n/a, no. n/a, Nov. 2021.
[51] J. Reback, Jbrockmendel, W. McKinney, J. Van Den Bossche, M. Roeschke, T. Augspurger, S. Hawkins, P. Cloud, Gfyoung, Sinhrks, P. Hoefler, A. Klein, T. Petersen, J. Tratner, C. She, W. Ayd, S. Naveh, J. Darbyshire, R. Shadrach, M. Garcia, J. Schendel, A. Hayden, D. Saxton, M. E. Gorelli, F. Li, T. Wörtwein, M. Zeitlin, V. Jancauskas, A. McMaster, and T. Li, "pandas-dev/pandas: Pandas 1.4.3," Jun. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.3509134
[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: machine learning in Python," *Machine learning in python*, p. 6, Oct. 2011.
[53] "Manual page of perf-stat," 2022, Accessed: 2022-01-31. [Online]. Available: https://perf.wiki.kernel.org/index.php/Latest_Manual_Page_of_perf-stat.1

APPENDIX A
ARTIFACT DESCRIPTION

This section describes the framework that automates measuring benchmarks' power and energy consumption with various software-based power meters. It also automatically retrieves and processes the results and proposes Jupyter notebooks to visualize them and study metrics. The repository also contains the result data used in this work.

A. Framework description

The experiments supporting this work were conducted thanks to the code available in the following repositories:

- Persistent identifier: https://hal.inria.fr/hal-03974900
- Github repository: https://github.com/vladostp/an-experimental-comparison-of-software-based-power-meters

The artifact is divided between the CPU benchmarks and the GPU benchmarks. Mathilde Jay developed the framework for the GPU benchmarks while Vladimir Ostapenco was in charge of the CPU benchmark experiments. More details on the functionality and instructions of both parts can be found in the corresponding folders.

The project is developed mainly in Python, for the collection of environment information, the management of the databases, and the process of the results. Scripts in Bash are used to compile and execute the benchmarks. Jupyter notebooks with Python code are used to visualize results and create figures.

B. Replicate experiments

1) Request access to the Grid'5000 testbed: To get access to the testbed, one needs to ask for an account using the OpenAccess program. To do so, please fill out the following form: https://www.grid5000.fr/w/Special:G5KRequestAccountUMS?openaccess.

Please specify in the "Intended usage" section that this demand is for a conference review. The authors of this work are not maintainers of the testbed and will not have access to your information.

Once you have access to the testbed, please read the Getting started tutorial to have a basic understanding of the plateform.

2) Compilation and execution environment: Our framework was fully tested on the Gemini cluster of the Grid'5000 testbed, as described in the Methodology section.

To replicate experiments for both GPU and CPU benchmarks, please follow the steps described in the corresponding README.md files.

C. Outputs

For each experiment, we gather environment description (duration, number of GPU/CPU, etc.) and energy measurement metrics as reported by selected tools and the power meters (BMC, OmegaWatt). The output of all experiments can be found in the result folders. Its structure is described in the respective README.md files of the provided artifact.

The artifact contains the result data from ten experiments for each tool and benchmark we selected. Experiments were also conducted without a software-based power meter to study their overhead.

To compare the software-based power meters on the various benchmarks, we study the following metrics:

- Power profiles reported: We visualize the relative evolution of power as reported by the tools and the external power meters.
- Total energy consumed per benchmark: We study the differences between the tool reports.
- Correlation with the external power meter: We study the correlation between the values given by each tool and the values reported by the external power meter.
- Offset: We investigate the relationship between the power time series as reported by the external power meters and the tools.
- Energy overhead: We compare the total energy consumed by the benchmarks (as reported by the external power meter) with and without the tools monitoring the executions.

- CPU overhead: We study the CPU overhead of PowerAPI and Scaphandre by running each solution with different sampling frequency configurations on an isolated CPU core and by tracking the utilization rate of that core.
- Consumption reported at the process level: We study how PowerAPI and Scaphandre estimate consumption at the application process level.
- Maximum sampling frequency supported by each solution.
- Maximum simultaneous processes supported by each solution.
- Total energy consumed by the project: We computed the total energy consumed by our experiments (test and production), from the power reported by the external power meter.

Each metric is associated with a Jupyter notebook, as described in the respective README.md files for CPU and GPU benchmarks.

D. Estimated time of all the compilation and run steps

Opening a Grid'5000 account can take a few hours as a manager of the testbed needs to verify the demand and grant access. Reserving the node and building the environment can take up to ten minutes. The compilation steps last less than one second per benchmark application. The duration of the experiments corresponds to the duration of the given benchmark, with a slight time overhead due to the installation and configuration of the given measurement tool. Processing the outputs and visualizing the results requires a few minutes.

In total, testing the code takes less than an hour. Replicating all results requires up to twenty hours.

E. Disclaimer

Energy Scope is not open source and not publicly available. Thus it is not provided and the experiments for Energy Scope are not reproducible at the moment.

F. License