RAPL in Action : Experiences in Using RAPL for Power Measurements

Khan, Kashif Nizam

ASSOC COMPUTING MACHINERY
2018-04

Khan , K N , Hirki , M , Niemi , T , Nurminen , J K & Ou , Z 2018 , ' RAPL in Action : Experiences in Using RAPL for Power Measurements ' , ACM Transactions on Modeling and Performance Evaluation of Computing Systems , vol. 3 , no. 2 , 9 . https://doi.org/10.1145/3177754

http://hdl.handle.net/10138/321707
10.1145/3177754

acceptedVersion

RAPL in Action: Experiences in Using RAPL for Power Measurements

KASHIF NIZAM KHAN, Helsinki Institute of Physics and Aalto University, Finland
MIKAEL HIRKI, Helsinki Institute of Physics and Aalto University, Finland
TAPIO NIEMI, Helsinki Institute of Physics and University of Lausanne, Switzerland
JUKKA K. NURMINEN, Helsinki Institute of Physics, Aalto University and VTT Technical Research Centre of Finland
ZHONGHONG OU, Beijing University of Posts and Telecommunications, China

To improve energy efficiency and comply with the power budgets, it is important to be able to measure the power consumption of cloud computing servers. Intel's Running Average Power Limit (RAPL) interface is a powerful tool for this purpose. RAPL provides power limiting features and accurate energy readings for CPUs and DRAM which are easily accessible through different interfaces on large distributed computing systems. Since its introduction, RAPL has been used extensively in power measurement and modeling. However, the advantages and disadvantages of RAPL have not been well investigated yet. To fill this gap, we conduct a series of experiments to disclose the underlying strengths and weaknesses of the RAPL interface by using both customized microbenchmarks and three well-known application level benchmarks:*Stream*, *Stress-ng* and *ParFullCMS*. Moreover, to make the analysis as realistic as possible, we leverage two production-level power measurement datasets from the *Taito*, a supercomputing cluster of the Finnish Center of Scientific Computing (CSC) and also replicate our experiments on Amazon EC2. Our results illustrate different aspects of RAPL and document the findings through comprehensive analysis. Our observations reveal that RAPL readings are highly correlated with plug power, promisingly accurate enough and have negligible performance overhead. Experimental results suggest RAPL can be a very useful tool to measure and monitor the energy consumption of servers without deploying any complex power meters. We also show that there are still some open issues such as driver support, non-atomicity of register updates and unpredictable timings that might weaken the usability of RAPL in certain scenarios. For such scenarios, we pinpoint solutions and workarounds.

CCS Concepts: • **Hardware → Energy metering**; **Enterprise level and data centers power issues**; **Emerging architectures**; • **Computer systems organization →** *Cloud computing*;

Additional Key Words and Phrases: RAPL, Power Modeling, RAPL accuracy, RAPL validation, DRAM power

ACM Reference format:
Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst.* 00, 00, Article 00 (January 2018), 26 pages.
https://doi.org/0000001.0000001

Author Kashif Nizam Khan would like to thank Nokia Foundation for a grant which helped to carry out this work. Author Zhonghong Ou would like to thank the Fundamental Research Funds for the Central Universities and National Natural Science Foundation of China (Grant No. 61702046) for the support. The authors would also like to thank CSC, IT Center for Science, Finland for the datasets.
Corresponding Authors: Kashif Nizam Khan (kashifnizam.khan@gmail.com,), Zhonghong Ou (zhonghong.ou@bupt.edu.cn).

2376-3639/2018/1-ART00 $15.00
https://doi.org/0000001.0000001

1 INTRODUCTION

Energy efficiency in data centers has become one of the major concerns in the last decade not only because of the monetary cost but also for environmental sustainability. The electricity consumption of data center is constantly increasing and there is an urge to apply optimizations in hardware and software to achieve the best performance per watt.

As such, a number of studies have been directed towards predicting and/or modeling energy consumption in large-scale data centers [24, 31, 43]. Many such models or prediction techniques require an accurate measure of the energy consumption of the data center on the go. Until recently measuring the power consumption of a computer system required separate metering hardware [3]. Besides the difficulties of purchasing, deploying, and using external power meters, their measuring accuracy and granularity are usually inadequate for detailed analysis. Moreover, dividing the power to different parts of the computing system inside the chip is not possible. There are also tools like the Intelligent Platform Management Interface (IPMI) which reports the power measurement readings through sensors mounted with the system. Existing studies [21] find that the accuracy of such sensors is not promising and therefore these sensors cannot be practically used as a substitute for more accurate watt meters on a per machine basis.

The other option is model based power estimation, which uses a set of performance counters and a computational model to turn the performance readings into estimates of electricity consumption. The accuracy of this approach strongly depends on the quality of the model and typically is not able to give good results especially with highly fluctuating workloads [30]. McCullough et al. [30] performed a comprehensive evaluation of the power modeling of computing systems and concluded that power modeling techniques pose several limitations caused by the increased complexity and variability of software and hardware. Their results motivate more towards low cost, direct, and instantaneous energy measurement tools.

Intel's Running Average Power Limit (RAPL) [40] is one such hardware feature which allows to monitor energy consumption across different domains of the CPU chip, attached DRAM and on-chip GPU with promising accuracy. This feature was introduced in Intel's Sandy Bridge architecture and has evolved in the later versions of Intel's processing architecture. With RAPL it is possible to programmatically get real time data on the power consumption of the CPU package and its components, as well as of the DRAM memory that the CPU is managing. RAPL is thus a good tool to measure, monitor, and react to the power consumption of computing. It has potential for new and innovative ideas to better deal with the electricity consumption of computing [24, 38, 43]. Skrenes et al. [36] use RAPL as an 'always on' energy metering tool in their proposed speed scaling simulator, *Profilo*. *Profilo* measures the energy consumption of the CPU with a goal to quantify the differences between different speed scaling strategies and uses RAPL as an effective measurement tool in their automated simulation process. Kelley et al. [22] use RAPL as a power tracing tool to profile peak power under core scaling which reduces the profiling time significantly. Their approach shows that RAPL measurements are reliable enough to be used as a power measurement tool as they use RAPL to collect the power trace for a sampling period and then this trace is used to predict peak power under different settings and across different architectures.

Despite the merits and potentials of RAPL, it is not clear whether RAPL also has weaknesses in measuring and monitoring the energy consumption of various CPU components. Thus, an in-depth study of the RAPL interface itself is still needed to reveal its underlying principles. In this paper, we conduct a thorough study of RAPL by utilizing a series of customized benchmarks, and two well-known application level benchmarks, that is *Stream* and *ParFullCMS*. To make the analysis as realistic as possible, we leverage a production-level power measurement dataset from the *Taito*

supercluster of the Finnish Center of Scientific Computing(CSC) and also use five different instance types from Amazon EC2 as testbeds.

This paper substantially enhances the work presented in [17, 23, 24]. In addition to the prior works, in this paper we analyze RAPL's capability to predict the full system power consumption with different modeling techniques using the large CSC dataset. The in-depth analysis of RAPL as an energy measurement tool in terms of different aspects like performance overhead or high sampling rate is also a significant addition. Lastly, the RAPL measurement experiences with Amazon EC2 is a considerable enhancement. In summary, we make the following major contributions in this work:

(1) With detailed measurements, we investigate the accuracy, granularity and performance of the RAPL measurements. Accuracy of RAPL measurements is measured by determining how closely the RAPL measurements predict the overall power consumption of the system. The 'closeness' or 'predictability' of RAPL's measurement is obtained through modeling and fitting the RAPL measurements with full system power consumption obtained from high accuracy external power meters.
(2) We also highlight a few aspects which need to be considered for the proper use of RAPL, as well as issues which would require further development.
(3) We pinpoint a few major differences in the performance, granularity and accuracy of RAPL in different Intel architectures: Sandy Bridge, Haswell and Skylake.
(4) We present a study of RAPL's performance in virtualized cloud computing environment using Amazon EC2.

Key findings from our study are listed below:

(1) The promising accuracy of RAPL measurements to predict full system power consumption is established by modeling the full system power consumption from RAPL readings using a large production dataset from CSC.
(2) The performance overhead of RAPL measurements is considerably low and negligible.
(3) There is a measurable correlation between RAPL's package power and temperature, at least in the Haswell architecture.
(4) In the case of Skylake, RAPL updates the PP0 domain in the order of µs (≈50-70 µs).
(5) RAPL's support in Amazon EC2 can be useful but it needs more careful considerations.

The rest of the paper is organized as follows: Section 2 presents detailed information about RAPL in general. Section 3 highlights the related literature. Section 4 discusses the experimental setup and benchmark specifications. In Section 5 we discuss the good aspects of RAPL such as its accuracy, granularity etc. Section 6 discusses the weak aspects of RAPL. We also discuss possible solutions to overcome a few of the weak aspects of RAPL in Section 6. In Section 7 we present RAPL measurement results and experiences with Amazon EC2 instances. Section 8 summarizes our findings and Section 9 presents the conclusion with possible future directions.

2 RUNNING AVERAGE POWER LIMIT

The RAPL interface is a feature introduced in the Intel Sandy Bridge architecture. Generally speaking, RAPL provides two different functionalities. First, it allows energy consumption to be measured at very fine granularity and a high sampling rate. Second, it allows limiting (or capping) the average power consumption of different components inside the processor, which also limits the thermal output of the processor [40]. We will focus on the energy measuring functionality in this paper.

RAPL supports multiple power domains. The RAPL power domain is a physically meaningful domain (e.g., Processor Package, DRAM etc) for power management. Each power domain informs

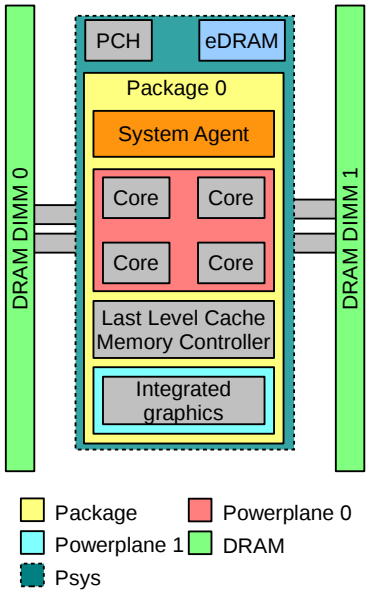

Fig. 1. Power domains supported by RAPL.

the energy consumption of the domain, allows to limit the power consumption of that domain over a specified time window, monitors the performance impact of the power limit and provides other useful information, that is, energy measurement units, minimum or maximum power supported by the domain [20].

Figure 1 shows the hierarchy of the power domains graphically. RAPL provides the following power domains for both measuring and limiting energy consumption:

- **Package**: Package (PKG) domain measures the energy consumption of the entire socket. It includes the consumption of all the cores, integrated graphics and also the uncore components (last level caches, memory controller).
- **Power Plane 0**: Power Plane 0 (PP0) domain measures the energy consumption of all processor cores on the socket.
- **Power Plane 1**: Power Plane 1 (PP1) domain measures the energy consumption of processor graphics (GPU) on the socket (desktop models only).
- **DRAM**: DRAM domain measures the energy consumption of random access memory (RAM) attached to the integrated memory controller.
- **PSys**: Intel Skylake has introduced a new RAPL Domain named PSys. It monitors and controls the thermal and power specifications of the entire SoC and it is useful especially when the source of the power consumption is neither the CPU nor the GPU. As Figure 1 suggests, PSys includes the power consumption of the package domain, System Agent, PCH , eDRAM and a few more domains on a single socket SoC.

For multi-socket server systems, each socket reports its own RAPL values (for example a 2-socket computing system has two separate PKG readings for both the packages, two separate PP0 readings, etc).

Table 1. Comparison of RAPL power domains supported by different Intel processor models. The -EP models are server-grade processors.

Model	Power domain supported?				
	PKG	**PP0**	**PP1**	**DRAM**	**PSys**
Sandy Bridge	Yes	Yes	Yes	No	No
Sandy Bridge-EP	Yes	Yes	No	Yes	No
Haswell	Yes	Yes	Yes	Yes	No
Haswell-EP [12]	Yes	No	No	Yes	No
Skylake	Yes	Yes	Yes	Yes	Yes*

*Not All Skylake versions support PSys

The support for different power domains varies according to the processor model. Table 1 presents a comparison of RAPL domains supported by different models. Power Plane 1 is present only in desktop models. In Haswell, Intel added support for the DRAM domain to desktop models. Both Power Planes 0 and 1 are absent in the Haswell server models. Thus, the only universally supported power domain is the Package domain. In case of Skylake, unlike PKG, PSys requires an additional system level implementation and so it is not supported in all Skylake versions.

The RAPL energy counters can be accessed through model-specific registers (MSRs). The counters are 32-bit registers that indicate the energy consumed since the processor was booted up. The counters are updated approximately once every millisecond. The energy is counted in multiples of model-specific energy units. Sandy Bridge uses energy units of 15.3 microjoules, whereas Haswell and Skylake uses units of 61 microjoules. In some CPU architectures such as Haswell-EP, DRAM units differ from CPU energy units. The units can be read from specific MSRs before doing energy calculations. There is no specific implication of different energy units in different architectures.

The MSRs can be accessed directly on Linux using the *msr* driver in the kernel. Listing 1 shows an example of this. For direct MSR access the MSR driver must be enabled and the read access permission must be set for the driver [40]. Reading RAPL domain values directly from MSRs requires detecting the CPU model and reading the RAPL energy units before reading the RAPL domain (i.e., PKG, PP0, PP1, etc.) consumption values.

Listing 1. Reading RAPL package energy using the MSR method on Haswell.

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */

double energy_units = pow(0.5, 14);
int fd = open("/dev/cpu/0/msr", O_RDONLY);
if (fd < 0) {
        perror("open");
        return -1;
}
if (pread(fd, &msr_value, 8, 0x611) < 0) {
        perror("pread");
        return -1;
}
double energy = msr_value * energy_units;
printf("%f\n", energy);
```

Once the CPU model is detected, the RAPL domains can be read per package of the CPU by reading the corresponding 'MSR status' register. For example, *MSR_PKG_ENERGY_STATUS* holds the energy readings for package domain (PKG). There are basically two types of events that RAPL events

Table 2. System Specifications

Processor (Intel)	Architecture	Type	Sockets	Cores	L3 Cache	Memory
Core i7-4770 @ 3.40 GHz	Haswell	Workstation	1	4	8MB	16GB
i5-6500 @ 3.20GHz	Skylake	Desktop	1	4	6MB	8GB

report: static and dynamic events. Static events reported by RAPL events are thermal specifications, maximum and minimum power caps, and time windows. The RAPL domain energy readings from the chip such as PKG, PP0, PP1 or DRAM are the dynamic events reported by RAPL.

Apart from directly reading MSRs, RAPL readings can also be read from *sysfs* interface, *perf* events or through the PAPI library. RAPL support for the *sysfs powercap* interface is enabled from Linux Kernel version 3.13 and the *perf_event_open* support requires Linux Kernel version 3.14. PAPI library is used for gathering performance-related data. It is platform independent and it has a RAPL interface which uses the MSR driver to report RAPL values.

3 RELATED WORKS

Hähnel et al. [14] evaluated whether RAPL can be used to measure short code paths. They showed that RAPL updates do not occur precisely every millisecond but instead they have some jitter. They also compared the RAPL measurements (Sandy Bridge) with external measurements with a manually instrumented board and showed that RAPL measurements do correlate nicely with external measurement with a fixed offset.

Hackenberg et al. [11] provided a comparison of power measurement techniques. They pointed out that the RAPL updates have no timestamps, which can lead to significant inaccuracy when sampling the RAPL counters. They also showed that the Sandy Bridge-EP implementation of RAPL suffers from systematic errors.

In 2015, Hackenberg et al. [13] studied RAPL on the Intel Haswell-EP platform. They compared the accuracy of RAPL between Sandy Bridge-EP and Haswell-EP and showed that Haswell has improved RAPL measurements. They also showed that RAPL measurements correlate very well with external power measurements.

Ilsche et al. [19] compared different power measurement techniques, including RAPL and showed that the key advantages of RAPL are: low cost and the ability to isolate the power consumption of the processor package from the rest of the system.

Huang et al. [18] evaluated RAPL for Haswell-EP processors and compared RAPL with traditional power monitoring tools. They showed that monitoring with RAPL using the Performance Application Programming Interface (PAPI) can consume 28.6% more power than an idle system. This is however when RAPL is monitored with all its 28 attributes and not all of these attributes are related to power or energy monitoring. They also claimed that if RAPL is monitored with selected attributes (PKG, PP1, PP0 etc), this power overhead can be reduced by 90%. These measurements however do not account for the PAPI library's power consumption and different granularities of RAPL measurements will also affect the energy overhead.

Spencer et al. [6] validated the RAPL DRAM values. We discuss their findings and contributions in Section 6.4 in more detail. Zhang et al. [44] validate RAPL's power limiting features based on stability, accuracy, settling time, overshoot, and efficiency. They show that RAPL power limiting performs well in terms of accuracy, settling time, overshoot and stability. They however argue that RAPL power limiting can underperform at low power limits and with high power limits it can achieve performance which is within 90% of optimal.

Apart from these studies, there are a significant body of literature [7, 8] which independently verifies the accuracy of RAPL readings with different workloads, architectures, systems and settings. RAPL has also been quite extensively used in energy profiling [23, 28, 41], full system power modeling [24, 26], application level power modeling [43] and power limiting under different scenarios [33, 37, 45].

Some of the above mentioned studies have performed the analysis of RAPL energy measurements in specific scenarios. In this work, we present a detailed analysis and technical know-how of RAPL with respect to accuracy, granularity, performance overhead and identify the essential issues regarding the usage of RAPL (we term them as 'bad' in this paper). Our contributions in this work are twofold. Firstly, there are unique discoveries regarding RAPL such as the relation between RAPL power readings and temperature, the non-atomic nature of RAPL updates and the detailed study of RAPL with the CSC data. Secondly we also complement and confirm previous observations such as RAPL performance overhead and RAPL DRAM energy accuracy. We also include our first hand experience of RAPL in Skylake which adds to the analysis of previous literature.

4 EXPERIMENTAL SETUP

We used both Haswell and Skylake based systems in our experiments. The latest Skylake microarchitecture also supports RAPL [20]. Skylake features higher performance than Haswell and added energy efficiency features. These factors make Skylake an attractive platform for energy measurement.

Table 2 shows more details of the systems. Haswell was a workstation-grade system while Skylake was a desktop-grade system. For Haswell we performed experiments with two software configurations. We first disabled turbo boost, huge-pages, and some background processes to create a simple configuration. Then we compared this with the default configuration, which had those optimizations enabled. No major differences were found in the RAPL analysis for these two cases. With optimizations (especially turbo boost) enabled, the software ran slightly faster: 2-3% speed difference with the experiments in Table 4. For Skylake experiments we used the default configuration without disabling any such configurations.

For the analysis of RAPL behavior we used multiple sources:

Microbenchmarks We wrote a set of microbenchmarks in our previous paper [17]. These benchmarks[1] are designed to exercise specific parts inside the CPU (execution units, caches and the instruction decoders). Our benchmarks perform simple mathematical operations like *Stream* [29] however unlike *Stream* our microbenchmarks do not write anything to memory essentially. The motivation behind writing such microbenchmarks is to examine the power expenditure of a system when different components inside the CPU are exercised with different types of loads and operations. While we previously were specifically interested in the instruction decoder, this time we are using the code as a more generic benchmark. Our benchmark code is published in a Git repository [16].

Application level benchmarks We also used a few well-known benchmarks namely: *Stream* and *ParFullCMS*. **Stream** [29] is a well-known benchmark designed to measure sustainable memory bandwidth. Using *Stream* helps us understand the characteristics of different systems in terms of power consumption when running a memory intensive task. **Stress-ng** is originally designed to stress different subsystems of a computer. In our work, we use Stress-ng to stress the CPU cores at 100% load. **ParFullCMS** is a Geant4 [10] benchmark, which is a multi-threaded high energy physics workload. This benchmark employs complex geometry for simulation and essentially exhibits similar properties like Compact Muon Solenoid (CMS)

[1] https://github.com/mhirki/rapl-tools

experiments in CERN. Using *ParFullCMS* helps us to understand whether RAPL is able to measure the power consumption of complex scientific applications with acceptable accuracy and granularity.

Traces from data center We used two data sets from the Finnish Center of Scientific Computing. The datasets are from the production site of CSC data center *Taito*[4]. The first data set contains 0.5 Hz sample rate data over a period of 48 hours and the other one 0.2 Hz sample rate data over 10 days. The datasets include the output of the Linux *vmstat* (Virtual memory statistics) tool, RAPL readings and the plug power consumption for almost 1000 machines from the *Taito* super cluster (Around 460 Haswell computing nodes, around 400 Sandy Bridge computing nodes and a few other special purpose nodes). *vmstat* vms [1] is a Linux tool, which reports the usage summary of memory, interrupts, processes, CPU usage and block I/O. We used this dataset to validate the accuracy of RAPL readings.

Measurement Methodology The RAPL power readings are collected using our *trace-energy* tool. This tool captures the power consumption of RAPL package, PP0, PP1 and DRAM domain by reading the corresponding *msr* registers. The frequency of energy tracing is supplied as a command line argument to the tool. The benchmark program/application program for which we want to measure the power consumption is attached to the *trace-enegy* tool as a command line argument. For our in-house experiments, we used the *Plugwise* [34] power meter to capture the plug power consumption of the system from the wall socket. This device has the highest frequency of 0.1 Hz and we used the same frequency to capture the power consumption data. The CSC datasets captured the vmstat, RAPL and plug power from the IMPI tool at a frequency of approximately 0.5Hz and 0.2Hz. The metrics collection for these datasets was done manually. For the continuous collection and analysis of such data, we need high-resolution automatic energy measurement tools which should ideally work on a cross-platform basis across different hardware and batch job schedulers.

5 GOOD FEATURES OF RAPL

In this section we discuss desirable features of RAPL such as high accuracy and small performance overhead.

5.1 Accuracy of RAPL energy readings

The accuracy of RAPL energy readings is high even though it is based on activity counters [11, 14] and the accuracy varies across different processing architectures. We performed experiments with

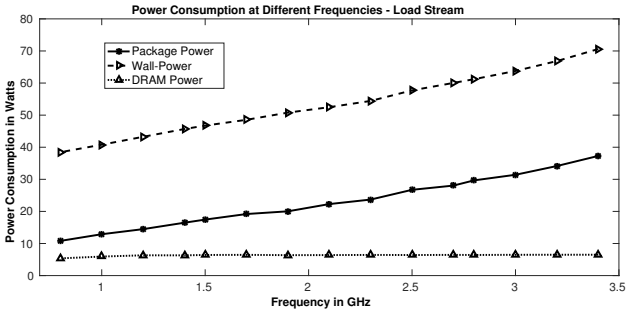

Fig. 2. Wall and RAPL package power consumption with time - *Stream*

the *Stream* benchmark on a Haswell machine which show a strong correlation between RAPL

package power and full system power consumption from the wall socket. The results are presented in Figure 2. The results show that RAPL power values are promisingly accurate and these values can be used to predict or model full system power consumption. From Figure 2 we can see that when we run the *Stream* benchmark at different processor frequencies on a Haswell machine, the full system power consumption from the wall strongly correlates with RAPL package power. The correlation coefficient between the two values is 0.99.

We also compared the accuracy of RAPL using two large datasets collected from a supercomputing cluster containing almost 1000 computing nodes. The RAPL and IPMI power (plug) readings were collected in the first dataset every two seconds for a period of two days and in the second data set every five seconds for a period of ten days. We tested cross correlation among RAPL values and plug power measurements and noticed that there is a significant time difference between these. Therefore, we computed several lagged plug variables. Since the measurements are strongly autocorrelated, we took a random sample of 30,000 measurements for both SandyBridge and Haswell nodes and divided the sample into 80% training and 20% testing sets. The distribution of plug measurements does not follow any standard distribution very well but the normal distribution gives the best results in the prediction models.

We fitted both linear regression and Generalized Additive Models (GAM) with the data to estimate the plug measurements using the RAPL values. Generalized additive models (GAMs) [42] are regression models in which the expected value of the response variable Y depends on unknown smooth functions of some predictors $x_{1,2,\ldots,m}$ as follows:

$$E(Y) = \beta_0 + f_1(x_1) + f_2(x_2) + \ldots + f_m(x_m),$$

where β_0 denotes the intercept, and $f_{1,2,\ldots,m}$ are smooth functions of covariates.

The benefit of GAM over linear regression is that it can also model non-linear relationships. For the Haswell architecture nodes, we used the following linear regression model:

```
Call: lm(formula = plug.l10 ~ CPU1 + CPU2 + DRAM1 + DRAM2, data = training)
Residuals:
     Min       1Q   Median       3Q      Max
-215.384   -4.177   -0.155    3.822  214.368
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 54.366131   0.530629  102.46   <2e-16 ***
CPU1         1.044694   0.005318  196.46   <2e-16 ***
CPU2         1.078358   0.004627  233.04   <2e-16 ***
DRAM1        0.837673   0.040788   20.54   <2e-16 ***
DRAM2        1.088337   0.040950   26.58   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.2 on 22483 degrees of freedom
  (1512 observations deleted due to missingness)
Multiple R-squared:  0.9281,Adjusted R-squared:  0.928
F-statistic: 7.25e+04 on 4 and 22483 DF,  p-value: < 2.2e-16
```

And the following GAM model:

```
Family: gaussian
Link function: identity
Formula: plug.l10 ~ s(CPU1) + s(CPU2) + s(DRAM1) + s(DRAM2)
Parametric coefficients:
             Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) 322.95820    0.07159    4511   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Approximate significance of smooth terms:
           edf Ref.df       F p-value
s(CPU1)  8.755  8.981 2995.61  <2e-16 ***
s(CPU2)  8.905  8.997 5929.64  <2e-16 ***
s(DRAM1) 8.269  8.860   60.09  <2e-16 ***
s(DRAM2) 8.625  8.959   90.40  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
R-sq.(adj) =  0.934   Deviance explained = 93.4%
GCV = 115.43  Scale est. = 115.25    n = 22488
```

For SandyBridge, the models omitted DRAM parameters, since the RAPL does not support them in that architecture.

The best results gave a lag of 10 seconds in the case of Haswell and 24 seconds in SandyBridge. However, we cannot be sure whether this difference is caused by different versions of the IPMI plug power measurement or RAPL itself. The results for the 0.5 Hz data set are shown in Table 3. The linear model performs well but GAM still gives slightly better results. This may indicate that the relationship (RAPL and plug) has a small non-linear effect. The large ten day data set gave similar but slightly better results (e.g., MAE 1.0 and RMSE 5.7 for Haswell). However, these are not directly comparable with the 48 hours data set because of different sampling rates.

Table 3. Regression models for the plug power (mean absolute error, root mean square error and mean absolute percentage error)

Model	Architecture	MAE	RMSE	MAPE
Linear	Haswell	5.6	9.8	1.8%
GAM	Haswell	5.2	9.8	1.7%
Linear	Sandy Bridge	10.5	17.2	4.3%
GAM	Sandy Bridge	9.8	16.3	4.0%

The regression models for Haswell include DRAM values of RAPL but this feature is not supported in SandyBridge. Therefore, we also tested whether the better accuracy in Haswell is due to the DRAM measurements. Results indicate that DRAM does improve the accuracy but even without DRAM the RAPL seems to perform better in Haswell (mean absolute percentage error 3.1%) than SandyBridge (mean absolute percentage error 4.0%).

When trying to estimate power consumption using only OS counters (vmstat outputs), the results are clearly worse than using RAPL. For SandyBridge, the Mean Absolute Percentage Error (MAPE) using the linear model was 11% and 6% when using a GAM model. The same errors for Haswell were 15% and 5%. These results confirm that RAPL readings are accurate enough to predict full system power consumption and can provide better estimation models when compared to the models that are based on OS counters.

Studies have also shown that RAPL power readings show promising accuracy. Hackenberg et al. [12] compared the RAPL implementation in Sandy Bridge-EP and Haswell-EP platforms. They compared power consumption predicted by RAPL with calibrated external readings. The results show that RAPL in Sandy Bridge-EP suffers from systematic errors while the Haswell-EP has a nearly perfect correlation with external measurements across all workloads. Therefore, Haswell-EP

is clearly a better choice for accurate power estimates. This applies to the entire family of models based on the Haswell microarchitecture as we have shown here. Similar to our approach, Khan et al. [24] used RAPL package power reading to propose an empirical power model to predict the full system power.

5.2 Performance overhead

The RAPL energy calculations are implemented in the hardware. There is no need to do any complex calculations, such as numerical integration in the software. Energy consumption can be measured on the same machine without significant overhead. No additional equipment is needed, which makes RAPL a very low-cost option for energy measurements. We performed experiments to measure the performance overhead caused by RAPL measurements with different applications and measurement frequencies. The results are presented in Table 4.

Table 4. RAPL Performance Overhead

Application	100 Hz	200 Hz	500 Hz	1000 Hz	1100 Hz
Idq-bench-float-array-l1-schoenauer	0.07%	0.15%	0.35%	0.70%	0.75%
Idq-bench-float-array-l2-schoenauer	0.15%	0.23%	0.42%	0.78%	0.86%
Idq-bench-float-array-l3-schoenauer	0.15%	0.17%	0.40%	0.75%	0.84%
STREAM-NTIMES-2000	0.46%	0.35%	0.89%	1.20%	0.70%
Idq-bench-int-algo-prng	0.07%	0.14%	0.34%	0.66%	0.70%
Idq-bench-int-algo-prng-multi4	0.07%	0.14%	0.34%	0.67%	0.72%

To measure the RAPL measurement overhead on performance, we used our instruction decoder benchmarks for different caches and data types. In addition, we also used the well-known *Stream* [29] benchmark. For the overhead measurement, we pinned the application and the RAPL measurement program on the same CPU core.In such a manner, the application program will be slowed down depending on the sampling rate of the measurement program. Our *schoenauer* benchmarks are based around the following loop:

```
D += A[j] + B[j] * C[j];
```

By changing the sizes of these arrays, we can make them fit either in the L1, L2 or L3 cache. The *Idq-bench-int-algo-prng* is simply a linear congruential generator used to generate pseudo-random numbers. This code has no dependency on any cache. The *multi4* version has 4 generators in a single loop, creating a bigger loop. Since switching to another process involves flushing the L1 cache, we expect that the benchmarks with a heavier cache dependency will suffer from a bigger overhead.

Table 4 depicts the percentage performance overhead (wall clock time) for RAPL measurements on frequencies 100, 200, 500, 1000 and 1100 Hz. As we can see, for *Idq-bench-float-array-l1-schoenauer* the highest performance overhead is 0.75% at 1100 Hz over a normal run when no RAPL measurement is performed. The results follow the same trend for other applications: for *Idq-bench-float-array-l2-schoenauer* the highest overhead is 0.86% at 1100Hz, for *Idq-bench-float-array-l3-schoenauer* the same value is 0.84% and so on. Interestingly, for *Stream*, the highest performance overhead is 1.20% which occurs at 1000Hz and for 1100Hz the overhead is smaller: 0.70%. We tested this 10 times and for every run the result had no significant difference. This might be due to the nature of *Stream*, however we cannot verify this in the scope of this paper. Nevertheless, from Table 4 it is clear that even for a sampling rate of 1100Hz, the performance overhead will be less than 2%

in most cases. Since the overhead is small, it is possible to take advantage of high sampling rates without disturbing the system too much.[2]

5.3 Correlation with temperature

We performed experiments to analyze how CPU package temperature relates to package power for both Haswell and Skylake architectures. To do this we used our own microbenchmark named *idq-bench-float-array-l3-schoenauer*. This benchmark was originally written to stress specific parts inside the CPU such as execution units, caches and the instruction decoders. This benchmark is used here as it exhibits stable and steady power consumption profiles and it also consumes quite a lot of power which makes the CPU heat up. The results are presented in Figure 3. The temperature and package power data are collected using 250Hz sampling rate. For the sake of simplicity in visualizing data, we take the average of every 50 data points in time so that the trend in data can be visible in the figure.

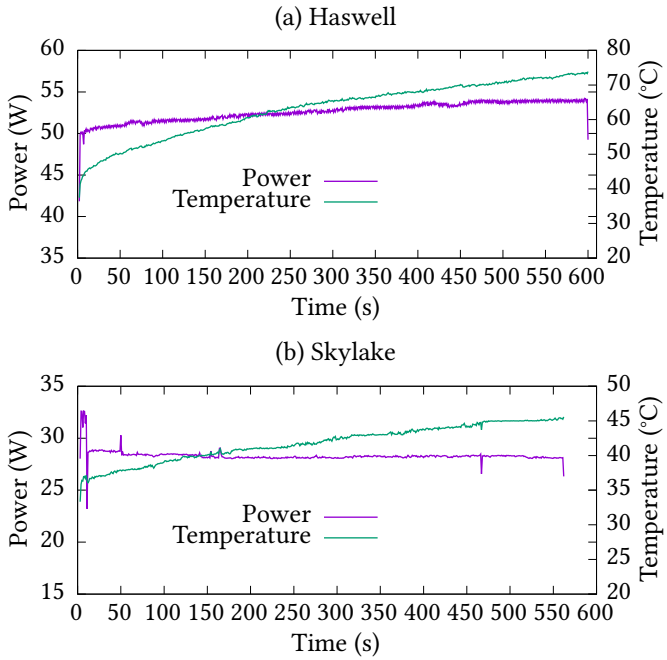

Fig. 3. Package Temperature and Package Power over time with a constant load on CPU

As Figure 3 indicates, it is important to control the temperature of the CPU to obtain accurate power readings through RAPL. The initial data for Haswell suggests that the package power grows by approximately 10-12% between 37°C to 74°C. In the case of Skylake the package power grows by approximately 8-10% between 23°C to 32°C. We can see that the package power drifts 5-10 watts for Haswell, however the Skylake package power readings are quite stable. The correlation coefficient between Haswell's package reading and temperature is *0.93* and the same for Skylake

[2]Note that for calculating the overhead we used 'real' time taken for a program to complete (as measured by the *time* command). The RAPL MSR registers are read from user space and the average latency of reading the MSR (*RDMSR* instruction execution + a system call) file is 393 nanoseconds. So if RAPL is read from kernel space the system call time would be eliminated.

is *0.34*. The high correlation coefficient for Haswell suggests that in case of Haswell temperature can have a measurable impact on the package power consumption and thus it is important to take the temperature of the system into account while measuring power using RAPL. Skylake have improved this phenomenon and there is a smaller correlation between the package power and temperature. The standard deviation for the Haswell package power reading is *1.29* and for Skylake it is *0.28*. This also confirms that Skylake power draw is fairly stable while the temperature of a system grows. One possible reason for this might be the optimized use of the fan in case of Skylake.

The Linux *coretemp* driver only provides a time resolution of 1 Hz. However, the MSR registers for the temperature sensors are actually updated every millisecond. This coincides with the RAPL update interval. There is a good correlation with the RAPL package power and temperature. Nevertheless, Figure 3 shows that CPU temperature can influence the CPU package power draw. To obtain accurate RAPL power readings it would be good to warm up the CPU before obtaining any RAPL measurements. A two-minute warm up period should be enough for any suitable program which keeps the CPU package busy.

5.4 High sampling rate

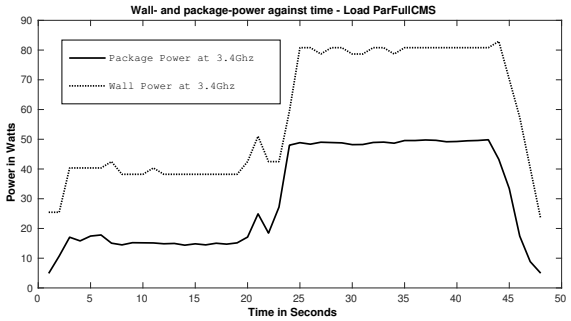

Fig. 4. Wall and RAPL package power consumption with time - *Stream*

RAPL updates the energy counters approximately once every millisecond, that is, 1000 Hz. This frequency is much higher compared to external power meters that typically measure power only once a second. Servat et al. [35] demonstrated that many HPC benchmarks have phases with different package and DRAM power consumption characteristics. We found that the RAPL sample rate is sufficient to distinguish different execution phases in applications.

We tested this phenomenon with a scientific computing application ParFullCMS benchmark on a Haswell based system. Figure 4 demonstrates that the multi-threaded ParFullCMS has two broad distinct phases: the first phase is the initialization phase, which sets up the events to be processed and consumes relatively less power; the second phase is the compute-intensive phase, which performs the bulk of simulation tasks and consumes relatively more power. It is evident from the figure that RAPL package power sampling is able to capture distinct phases in ParFullCMS like High Energy Physics (HEP) benchmarks. We performed another experiment with *Stress-ng* to compare RAPL's sampling rate with a plug power capturing device. *Stress-ng* was invoked as CPU intensive workload with fluctuating loads. The RAPL PKG consumption value was captured every 5 ms and the plug value was captured every 100 ms (10Hz is the highest frequency of sampling supported by the Plugwise device we used). The results are presented in Figure 5. The plug power metering device was unable to capture the distinct phases as the phases changed faster than the

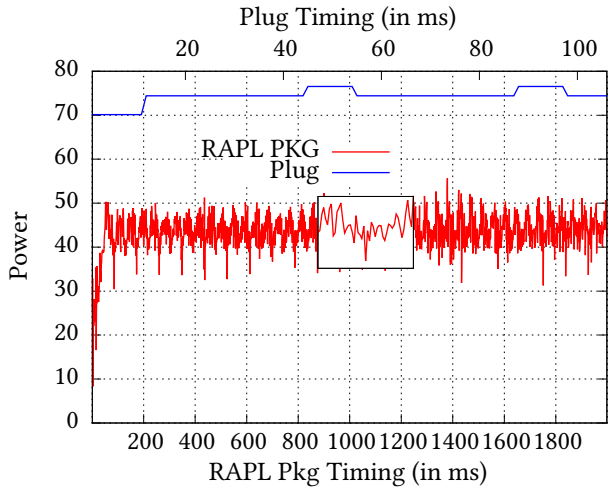

Fig. 5. RAPL Sampling Rate Experiment with Stress-ng

metering device's frequency. RAPL could actually capture the phase changes at a 5 ms sampling frequency (RAPL can be sampled in 1ms frequency). The zoomed area of RAPL package power in Figure 5 shows the distinct phase changes in computation (and thus in power consumption) in finer detail.

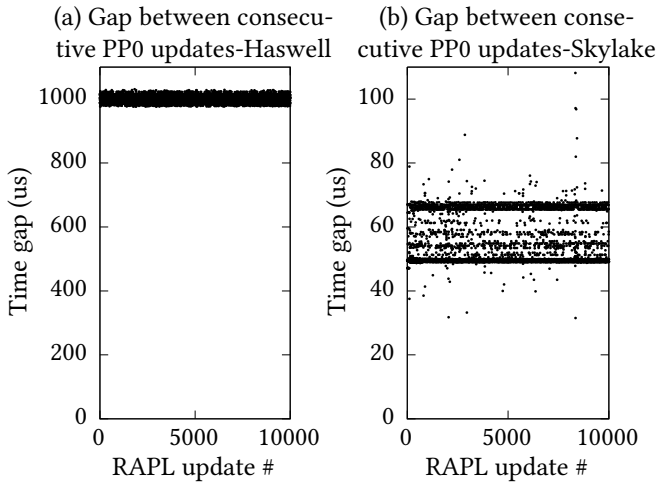

Fig. 6. PP0 sampled every five milliseconds with a constant load on one CPU core.

Our Skylake experiments also reveal that Intel has improved the temporal resolution of PP0 updates in Skylake in comparison to Haswell and previous architectures. PP0 in Skylake updates 20 times on an average in between a single update for other domains (PKG, PP1 and DRAM). Figure 6 shows the time gap between two consecutive PP0 updates in case of Haswell and Skylake for about 10,000 updates. This figure shows that for Haswell most of the PP0 update happen at a time gap of nearly 1 ms or 1000 µs whereas for Skylake the bulk of the updates happen between 50 and 70 µs

with a few outliers. This new improvement for Skylake allows a more granular temporal resolution and as a result it improves the possibility to determine the energy consumption of short code paths.

5.5 Always running

RAPL starts running as soon as the processor boots up. There is no need to configure it if one wants to measure energy consumption, which makes it very easy to use. Since RAPL is always running, there is very little additional overhead introduced by reading the energy counters.

6 RAPL LIMITATIONS

This sections discusses the weaker aspects of RAPL which should be properly addressed in specific scenarios. We also discuss solutions to overcome these issues in this section.

6.1 Poor driver support.

Sandy Bridge processors supporting RAPL were released in 2011. Full RAPL driver support was introduced as late as 2013 to the Linux kernel. Currently, there are three ways to read the energy counters on Linux.

As mentioned in Section 2, the *msr* driver supports raw access to any MSR including the RAPL energy counters. It is nearly universally supported regardless of the kernel version. The main drawback is that root access is required to use the *msr* driver. The earlier Listing 1 shows how to access the counters using the *msr* driver. Popular libraries, such as PAPI, rely on the *msr* driver interface. Thus, root access is also required when using these libraries.

The *powercap* driver allows reading the energy counters and configuring the power consumption limits. The counters can be found under /sys/devices/power/events/. They can be read without root access. The *powercap* driver was introduced in the kernel version 3.13.

The performance events (also called perf_events) subsystem in the Linux kernel supports RAPL energy counters since the kernel version 3.14. This allows the Perf tool to read the counters. Weaver [39] provided an example of how to programmatically read the counters using the perf_events API.

6.2 Register overflows.

The energy counters are limited to 32 bits even though the MSRs are 64-bit wide. Therefore, they will eventually overflow. Fortunately, these overflows are quite rare. We can calculate the time to overflow t_{overflow} using the following equation:

$$t_{\text{overflow}} = \frac{2^{32} \times E_u}{P} \tag{1}$$

Here E_u is the energy units used (61 microjoules for Haswell) and P is the power consumption. For example, a Haswell processor consuming 84 watts would cause an overflow every 52 minutes:

$$\frac{2^{32} \times 61\mu J}{84W} = 3120s$$

The overflows can be mitigated easily by sampling the energy counters more frequently than t_{overflow}. Frequent-enough sampling allows detecting every overflow that occurs. As a general rule, sampling the counters every five minutes should be sufficient for any CPU model.

6.3 Non-atomic register updates

Our measurements show a time delay between updates to different energy counters, which means that the RAPL updates are not atomic. The non-atomicity introduces errors when sampling multiple counters at high sampling rates. It is possible to read both fresh and stale values of different

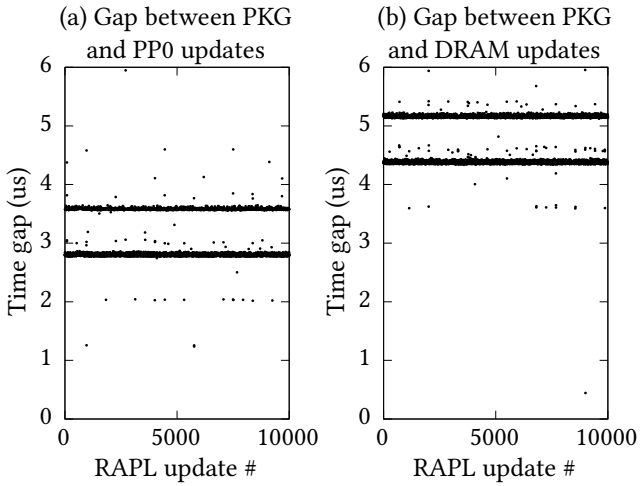

Fig. 7. The time gap between updates to different RAPL registers (Haswell). The timing is measured in microseconds.

counters. Figure 7 shows the timing difference between updates to different RAPL registers for our Haswell machine. In this experiment, we perform busy polling to two RAPL registers alternately. For example, Figure 7 (a) shows the timing gap between the RAPL Package (PKG) and PP0 registers. We record the timestamp of every update for both the registers and calculate the difference between the two timestamps. In the case of Haswell, we observe that RAPL registers are always updated in a specific order. The PP0 register is updated either 2.8 or 3.6 microseconds after the PKG register. The DRAM register is updated either 4.4 microseconds or 5.2 microseconds after the PKG register. The 0.8 microsecond difference between these numbers is due to our limited polling frequency. The average latency of reading the MSR (*RDMSR* instruction execution + a system call) file is 393 nanoseconds [15]. Therefore, polling two registers takes approximately 0.8 microseconds.

There is a measurable delay between updates of different RAPL registers. We observe similar behavior in case of Skylake. Nevertheless, Figure 6 has already shown us that there are measurable gaps between two Skylake RAPL register updates. It is quite obvious that RAPL registers are not updated atomically. This means that it is possible to accidentally read both fresh and stale energy values when sampling multiple RAPL registers. Specific precautions should be taken to avoid this. For example, we can first identify the correct sequence of RAPL register updates for a specific processor architecture and then if busy polling is used to detect RAPL updates, one should poll the last register updated (in our Haswell experiment DRAM register appears to be the last updated).

6.4 DRAM energy accuracy

A few papers [5, 25, 32] have focused on validating the RAPL values specifically for the RAPL DRAM domain. The initial findings suggest that RAPL DRAM values were unstable and unreliable for earlier versions of processors which included RAPL. DRAM values were only available to server-grade systems prior to Haswell. Prior to Haswell, the RAPL DRAM values followed different trends depending on the benchmark, and the values also differed substantially at times with reference values. Since the introduction of Haswell, RAPL DRAM values are now more reliable and follow a strong correlation with AC reference measurements [13, 24].

In our own analysis, DRAM measurements clearly improve accuracy. Modeling the plug power using only CPU measurements gives 3.1% error, including DRAM measurements reduces this error to 1.7%. However using only DRAM measurements alone for estimating the plug power yields quite a large error, 7.7%.

A more recent work on the validation of RAPL DRAM power measurements [6] also confirms these results. According to Spencer et al. [6] RAPL DRAM values now closely match the overall energy power and energy trends by a constant power offset for Haswell processors. They also show that there are differences in RAPL DRAM energy measurements across different architectures although the results with newer architectures are more accurate. Thus it is important to pay attention to the underlying architecture before relying on RAPL DRAM energy measurements.

6.5 Unpredictable timings

The update interval of the RAPL energy counters is approximately one millisecond. Intel documentation states that the time unit used by RAPL is 0.976 milliseconds. Our experiments show that the actual interval is much closer to one millisecond. Unfortunately, the updates do not have any timestamps associated with them [11]. In addition, there seems to be no way to predict the exact timing of future RAPL updates. Nevertheless, there are several solutions for coping with this problem.

First solution: busy polling. It is possible to determine the exact timing of the RAPL updates by busy polling the counters for updates. This technique allows for exact synchronization with the RAPL updates. Hähnel et al. [14] demonstrated that this method can be used to measure the energy consumption of short functions. They use busy polling to detect the start of a RAPL measurement period. They then execute a function they want to measure. Subsequently, they use busy polling to wait until the end of the RAPL period. They obtain the energy used by the function itself by deducting the energy used for busy polling. The downside of this method is the overhead in terms of time and energy.

Second solution: supersampling. Every update can be observed without the timing by sampling more often than every millisecond, for example, every 0.9 milliseconds. We call this method supersampling. The method will occasionally produce duplicate values but they can be filtered out easily. The main advantage is the reduced overhead compared to busy polling. However, the relative overhead is still quite large.

Third solution: High frequency sampling. Sampling frequencies in the range $50 - 1000$ Hz fall into this category. The advantage is lower overhead due to lower sampling rate. A 50 Hz sampling rate is still sufficient to distinguish different execution phases, as Servat et al. [35] demonstrated. A drawback of this solution is that the number of updates in one sampling period is not predictable. For example, when sampling at 200 Hz, most samples correspond to five RAPL updates. However, occasionally one sample contains four or six updates. This will create spikes in the data. The spikes can be filtered out but this leads to a small loss of data.

Figure 8 shows an example of the spikes when sampling at 200 Hz. In this case, the spikes come in two flavors: 1) a spike that is 20% below the mean, and 2) a spike 20% above the mean combined with a second spike 20% below the mean. The first flavor is caused by the fact that the mean time between RAPL updates is 1.00229 milliseconds. Thus, in a ten second time period, there will be only $10,000 \div 1.00229 = 9977$ updates. The second flavor is caused by the random spread in the timings, which allows six updates to be observed inside a five millisecond period. We have drawn circles around the points that are 20% above the mean.

Fourth solution: Low frequency sampling. At sampling rates slower than 50 Hz, the relative error due to the spikes is less than 0.5%, which is small enough to be ignored. The energy values should

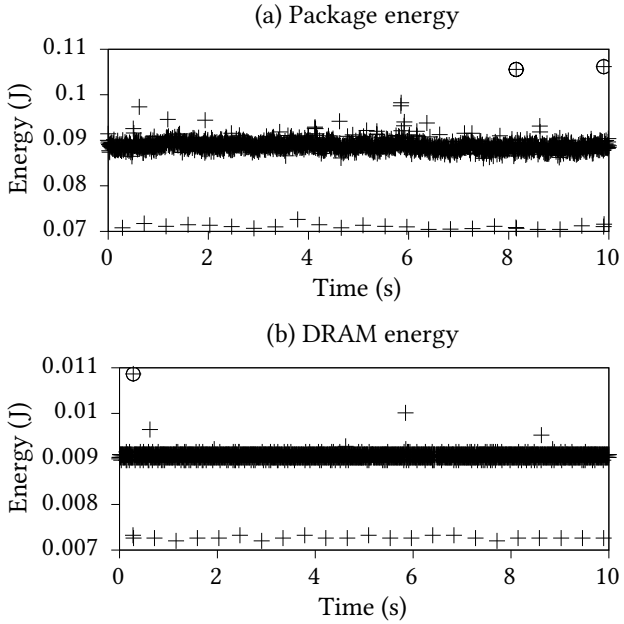

Fig. 8. Package and DRAM energy sampled every five milliseconds with a constant load on one CPU core.

remain accurate even with long sampling periods since they are calculated by the hardware. Thus, the only drawback with slow sampling is the loss of temporal precision.

6.6 Individual core measurement not supported

RAPL does not support measuring the power consumption of individual CPU cores. The Power Plane 0 domain only gives the total energy consumed by all cores in a single package. Each core has its own MSRs but we found that all cores show the same values for the RAPL counters. This prevents us from separating the power consumption of different processes or threads running concurrently on the same processor.

There are still alternatives that allow examining the power consumption of individual threads. The first alternative is to run an application in single-threaded mode. We measured the Power Plane 0 power consumption, which is nearly zero when all the cores are sleeping. Thus, Power Plane 0 is equivalent to the power consumption of a single CPU core when only one core is active.

The second alternative is to use multiple cores and divide the power consumption by the number of active cores. This method works best if all threads are performing the same task. For example, microbenchmarks can execute identical workloads across multiple threads.

Individual core measurement is further complicated by hyperthreading. Hyperthreading allows one core to simultaneously execute instructions from two threads. There is no way to separate the power consumption of one thread from the other. However, there are techniques which use hardware counters to model and attribute the power consumption to individual threads such as HaPPy [43]. Such models tend to have large errors ($\approx$ 10%) and therefore, it is advisable to either explicitly enable or disable hyperthreading depending on what needs to be measured.

Table 5. Amazon EC2 Instance Specifications

Instance Type	Processor	Architecture	Cores	Memory
t2.micro	Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz	Haswell-EP	1	1 GB
t2.xlarge	Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz	Haswell-EP	4	16 GB
t2.2xlarge	Intel(R) Xeon(R) CPU E5-2676 v3 @ 2.40GHz	Haswell-EP	8	32 GB
c3.xlarge	Intel(R) Xeon(R) CPU E5-2680 v2 @ 2.80GHz	IvyBridge-EP	4	7.5 GB
c4.xlarge	Intel(R) Xeon(R) CPU E5-2666 v3 @ 2.90GHz	Haswell-EP	4	7.5 GB

6.7 Fixed sampling rate and energy units

It is not possible to increase the sampling rate from the default of 1000 samples per second, which limits our ability to measure the energy consumption of individual functions using for example, IgProf [23]. One solution is to force the application to sleep before and after the execution of a single function. In this manner only a single function is executed during one RAPL update period. Hähnel et al. [14] have shown that this method works. The size of the RAPL energy units limits how accurately a single function can be measured. There are differences in the energy units between different models, which means that some models are better for measuring small variations in energy consumption. For example, Haswell models use units of 61 microjoules while Sandy Bridge models use units of 15.3 microjoules.

7 RAPL MEASUREMENTS IN AMAZON EC2

In order to further quantify and qualify RAPL measurements we performed several experiments with Amazon Elastic Compute Cloud (Amazon EC2). The motivation behind choosing a cloud environment is to examine how RAPL performs in popular cloud based services like Amazon EC2. Energy consumption in such a cloud environment is very crucial and tools like RAPL can simplify the energy consumption monitoring of individual users and instances and trigger new energy based pricing mechanisms such as $/joule instead of $/core/hour. In this section, we present our experiences while using RAPL in Amazon EC2. With respect to RAPL we performed the following experiments: performance overhead, correlation with temperature, and sampling rates. The methodologies used in these experiments are the same as those mentioned in Section 4 and 5.

Amazon EC2 allows to read RAPL's MSR registers. Currently, with all of the processor family available in Amazon EC2, it is possible to read the package and DRAM power consumptions. We experimented with five different Amazon EC2 instances: t2.micro, t2.xlarge, t2.2xlarge, c3.xlarge and c4.xlarge [9]. The specifications of these instances are presented in Table 5. t2 instances are general purpose computing instances with mostly Intel's *Haswell-EP* architecture whereas c3 and c4 instances are compute optimized instances with a mix of Intel's *Haswell-EP*, *Broadwell-EP*, *IvyBridge-EP* and *Sandybridge-EP* architectures. We could not obtain the plug power consumption in EC2 instances from sources like IPMI and as a result we could not establish or verify the accuracy of the RAPL readings in virutalized cloud computing platforms like EC2. Our first hand experience of using RAPL in Amazon is discussed in the following.

Performance overhead We used the same benchmarks and same frequency settings as mentioned in Table 4 for measuring performance overhead in Amazon EC2. The impact of RAPL measurement on performance overhead in a cloud environment is particularly interesting to examine because the same physical resources are shared between different users in a virtualized cloud environment and there is an expected measurable impact of other co-running user instances on the overall power consumption and load of the system. In our experiments, we mainly focused

Table 6. Amazon EC2- RAPL Performance Overhead

t2.micro					
Application	100 Hz	200 Hz	500 Hz	1000 Hz	1100 Hz
Idq-bench-float-array-l1-schoenauer	0.38%	0.60%	1.00%	1.63%	1.75%
Idq-bench-float-array-l2-schoenauer	0.04%	0.05%	0.64%	1.50%	1.64%
Idq-bench-float-array-l3-schoenauer	0.09%	0.63%	0.77%	1.84%	2.05%
STREAM-NTIMES-2000	0.08%	0.54%	0.98%	1.36%	1.28%
Idq-bench-int-algo-prng	0.19%	0.26%	0.69%	1.40%	1.64%
Idq-bench-int-algo-prng-multi4	0.91%	1.10%	1.39%	1.97%	2.30%
t2.xlarge					
Application	100 Hz	200 Hz	500 Hz	1000 Hz	1100 Hz
Idq-bench-float-array-l1-schoenauer	0.18%	0.53%	1.08%	1.61%	2.24%
Idq-bench-float-array-l2-schoenauer	0.09%	0.24%	0.43%	1.23%	1.42%
Idq-bench-float-array-l3-schoenauer	0.25%	0.48%	0.73%	1.33%	1.25%
STREAM-NTIMES-2000	0.08%	0.54%	0.98%	1.36%	1.28%
Idq-bench-int-algo-prng	0.27%	0.43%	0.84%	1.49%	1.59%
Idq-bench-int-algo-prng-multi4	0.10%	0.31%	0.79%	1.37%	1.56%
t2.2xlarge					
Application	100 Hz	200 Hz	500 Hz	1000 Hz	1100 Hz
Idq-bench-float-array-l1-schoenauer	0.41%	0.65%	1.45%	1.59%	1.79%
Idq-bench-float-array-l2-schoenauer	0.14%	0.10%	0.61%	1.70%	1.70%
Idq-bench-float-array-l3-schoenauer	0.19%	0.43%	0.92%	1.95%	1.81%
STREAM-NTIMES-2000	0.11%	0.41%	0.46%	1.48%	1.52%
Idq-bench-int-algo-prng	0.17%	0.27%	0.64%	1.43%	1.48%
Idq-bench-int-algo-prng-multi4	0.47%	0.62%	1.02%	1.51%	1.59%

on t2 instances for examining performance overhead. The results are presented in Table 6. The performance overheads of RAPL measurements in the case of Amazon t2 instances are quite similar to our observations with *Haswell* based standalone systems as presented in Table 4. In t2.micro the maximum overhead was observed in *Idq-bench-int-algo-prng-multi4* while reading the RAPL MSRs at a frequency of 1100 Hz and the overhead was *2.30%*. In t2.micro we observed a lot of performance variations due to variable load on the system. In such cases, the benchmark itself took unusually longer time to completion compared to cases where sampling frequency was even higher. We carefully discarded the unstable results. We present an example of such a scenario in a later discussion. For the t2.xlarge instance, the maximum overhead was *2.24%* which was reported in *Idq-bench-float-array-l1-schoenauer* at 1100 Hz and in t2.2xlarge the maximum overhead was *1.95%* which was measured at 1000 Hz for the *Idq-bench-float-array-l3-schoenauer* benchmark. For t2.2xlarge, the overhead measurements were less than *2.0%* which was also the case for the standalone *Haswell* based workstation. Since t2.2xlarge was an 8 core *Haswell-EP* based system, the possibility of load variation due to CPU or resource sharing was quite low since the number of virtual CPUs in the instances and number of physical CPUs of the architecture are the same and the observations were more stable. In brief, the performance overhead observed in Amazon EC2 was less than *2.5%*.

Correlation with temperature The correlation between the RAPL package power and the core temperature could not be verified in the case of Amazon EC2. Although, it was possible to read

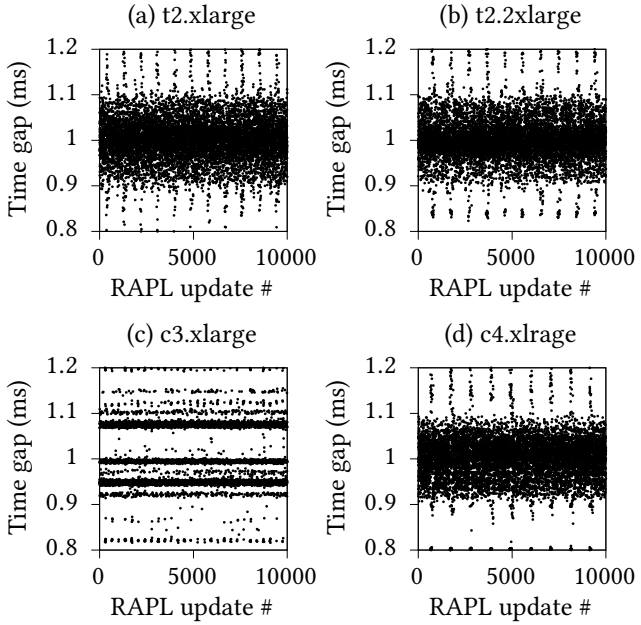

Fig. 9. PKG power timing gap between consecutive updates

the MSRs like *IA32_THERM_STATUS* and *IA32_PACKAGE_THERM_STATUS*, we obtained a constant value of 25°C as the PKG temperature. We verified this with different benchmarks like *Stress-ng*, *Stream* and our *Instruction decoder* benchmark (presented in Table 4) and the package temperature remained static at 25°C. We assume that Amazon EC2 filters out the temperature sensor value and presents a dummy constant value for package temperature. Apparently, Liu et al. reported in [27] that it was possible to read the CPU temperature readings but things have changed since then.

Timing gap in consecutive RAPL updates We performed experiments to identify the timing gaps in consecutive RAPL updates and to verify the high resolution of RAPL updates similar to experiments presented in Figure 6. The experiments were performed using t2.xlarge, t2.2xlarge, c3.xlarge and c4.xlarge instances. The timing gaps (in ms) for 10,000 consecutive RAPL PKG power updates is plotted in Figure 9. As indicated in this figure, the timing gaps are very sporadic for t2.xlarge, t2.2xlarge and c4.xlarge ((a), (b) and (d) respectively in Figure 9). The timing gaps in c3.xlarge ((c) in Figure 9) show a certain pattern but it is still inconclusive and does not show a uniform update interval (which should be ideally 1 ms for Haswell) if we compare it with Figure 6. Interestingly, among the four types of instances examined here, only c3.xlarge has the Ivybridge-EP processor architecture and the remaining three have Haswell-EP processor architecture. The results show that there is a difference in RAPL implementation between the two architectures. We found a measurable difference in polling delay between standalone workstations and Amazon EC2 instances. The hypervisor in EC2 instances traps the MSR reads which can add to the polling delay. The CPU in EC2 also runs at a lower clock rate, which might also add to the delay. Nevertheless, it is hard to pinpoint whether the timing gaps are produced by the hardware or interference from the hypervisor. Further investigation is required to make such claims.

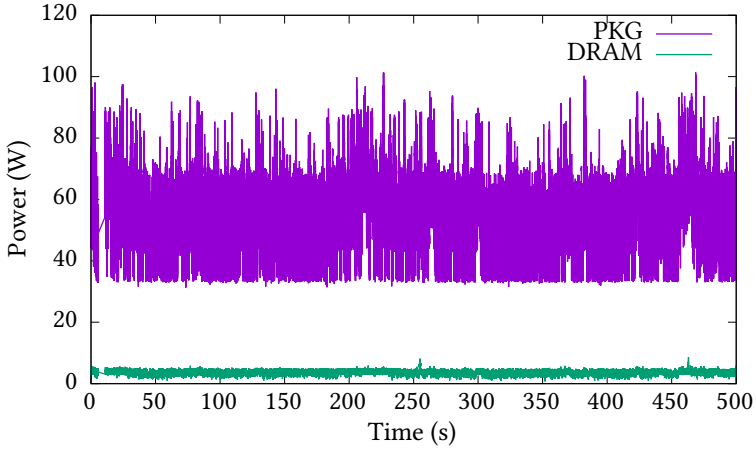

Fig. 10. Amazon EC2 Power profiling

Reading RAPL values with Amazon EC2 is challenging since the current implementation of RAPL does not give the per-core energy consumption measurements. Additionally, it is not necessary for a virtual CPU to be mapped to a physical CPU in a cloud environment. The RAPL readings in such a scenario are also affected by other virtual instances running on the physical server. We profiled the c4.xlarge instance for 500 seconds where we simply captured the RAPL readings at a 250Hz sampling rate without doing anything. Figure 10 shows the power consumption during that 500 seconds when we executed no workload but just listened to the RAPL updates. This figure shows that during the experiment period the PKG power consumption of the instance highly fluctuated between the mid 30W to close to 100W. Therefore it is very hard to do the energy accounting in such a virtualized environment and RAPL needs to address this issue to make it more usable in cloud environments like EC2. A few straightforward enhancements that could make RAPL more usable in virtual environment are listed below:

- Reducing the delay of reading RAPL MSRs through hypervisors.
- Adding per core energy accounting instead of per package. This would also make the RAPL implementation more secure since it would not be possible to see the activities of other instances sharing the same package through power consumption patterns.
- Adding timestamps to RAPL updates to allow per job or per process energy profiling.

Table 7. Summary

Aspects	Findings
Accuracy	Correlation ≈ 0.99 with AC power
Plug Power Estimation	MAPE 1.7%
Correlation with temperature	High Correlation with package power in Haswell: 0.93 Less correlation with package power in Skylake: 0.34
Sampling rate	High sampling rate ≈ 1 ms Skylake PP0 sampling rate ≈ 50 µs
Performance overhead	Negligible: < 1% overhead for standalone systems < 2.5% for Amazon EC2
DRAM energy	≤ 20% deviation from actual measurements [6]

8 DISCUSSION

Table 7 summarizes the important aspects of RAPL that we have discussed in detail in the previous sections. We have pinpointed our findings about Intel RAPL's accuracy, plug power estimation capability, relation with temperature, sampling rate, performance overhead and some other important aspects that we need to quantify about RAPL through our own experiments and a thorough review of the literature. Our study shows that Intel RAPL can provide accurate enough results for the power consumption of a CPU or attached DRAMs without manually instrumenting the system. The sampling rate is high enough and the performance overhead for reading RAPL counters at a higher rate is low enough for most of the general cases.

On the contrary, there are some aspects of RAPL which might not make it a suitable tool for cases where we need to determine the power consumption of short code paths [14] or cases where we need to know the exact timestamps attached with each RAPL update. We have discussed all of these aspects to highlight the advantages as well as weaker aspects of RAPL. For future references, we pinpoint a few of our proposed workarounds in brief:

- Register overflow problem can be mitigated by sampling more frequently than the time to overflow.
- Non-atomicity of register updates can be dealt with correctly by identifying the sequence of RAPL register updates and then busy polling the last register in the sequence.
- In case of DRAM energy measurements, Haswell and later processor architectures should be used.
- To mitigate the lack of timestamps in RAPL updates, one can use busy polling, supersampling, high frequency sampling or low frequency sampling depending on the requirements. (Please refer to Section 6.5 for more details)
- Individual core measurements can be obtained either by running a single threaded mode of a workload or by dividing the power consumption by the number of active cores given that all the threads perform the same task.

RAPL measurements in Amazon EC2 instances suggest that although it is possible to obtain power consumption readings in Amazon EC2, the power consumption of such a virtualized cloud computing environment itself varies depending on the load on the actual systems and therefore the current implementation of RAPL may not be so useful in such scenarios compared to standalone systems. RAPL implementation should properly address this in the future architectures to make it useful and efficient in cloud computing systems.

The promising aspect of RAPL is that it has evolved through generations of architecture. Sandybridge RAPL energy consumption values are based on modeling approaches. On the contrary, Haswell is able to achieve higher levels of accuracy because of the introduction of on-chip voltage regulators and more fine grained control of frequencies [13]. We have also shown that Skylake has improved from Haswell in granularity for measuring the PP0 domain and it introduces a new, PSys domain to monitor and control the power consumption of the platform domain. Our limited experiments with Skylake already show improvements in RAPL implementation and in the future we plan to perform a detailed study on Skylake's RAPL validation.

We do not particularly focus on GPU workloads in this paper since our focus here is traditional CPU or memory intensive workloads. In the future we plan to extend our work with such enhancements. There are alternatives for RAPL in case of AMD, ARM processors and NVIDIA GPUs. Prior works suggest AMD processor can report "Current Power In Watts" using MSRs like RAPL [14], ARM has cross platform chip monitor integrated with recent versions of processors [2] and recent NVIDIA GPUs can also report power usage via the NVIDIA Management Library (NVML) [14].

Despite a few differences in RAPL and reference measurements, Intel's RAPL can be a useful tool to track a program's power consumption behavior in large-scale data centers and it can be a good alternative to metering devices specifically as it does not require a complex manual instrumentation of every system of interest. It will be more useful to use RAPL if it addresses some of the concerns that we discuss in this paper: adding timestamps to individual updates, atomic register updates, individual core measurement features, handling register overflows more gracefully, and perhaps adding more features such as handling even the power consumption of the other system components rather than only the CPU chip.

9 CONCLUSION

The study of energy-efficiency in computing has been a hot topic for a long time. The introduction of RAPL has enabled a number of experiment scenarios that were not possible before. Nevertheless, there are also weaknesses that are associated with such an approach. In this paper, we conducted a comprehensive study to demonstrate how Intel RAPL performs in terms of accuracy, performance, granularity, usability and other important aspects of user experiences. We also pinpointed several directions towards the improved handling of some of the issues that might arise in certain scenarios. Our overall study suggests that RAPL has evolved towards a better energy measurement tool since its introduction in Sandybridge and it has appeared to be a useful and efficient alternative for manually instrumented complex power monitors. With the Haswell architecture, RAPL has improved considerably, its power readings now closely match plug power readings and it has now introduced the new measurement domain PSys and improved the power performance in Skylake. In the future we plan to continue our work with a comparative analysis of RAPL performance in Haswell and Skylake.

REFERENCES

[1] Accessed: February 11, 2017. VMSTAT. (Accessed: February 11, 2017). http://www.linuxcommand.org/man_pages/vmstat8.html.

[2] David Abdurachmanov, Peter Elmer, Giulio Eulisse, Robert Knight, Tapio Niemi, Jukka K. Nurminen, Filip Nyback, Goncalo Pestana, Zhonghong Ou, and Kashif Nizam Khan. 2014. Techniques and tools for measuring energy efficiency of scientific software applications. *CoRR* abs/1410.3440 (2014). arXiv:1410.3440 http://arxiv.org/abs/1410.3440

[3] Xi Chen, Chi Xu, Robert P. Dick, and Zhuoqing Morley Mao. 2010. Performance and Power Modeling in a Multi-programmed Multi-core Environment. In *Proceedings of the 47th Design Automation Conference (DAC '10)*. ACM, New York, NY, USA, 813–818. https://doi.org/10.1145/1837274.1837479

[4] CSC. 2017. Taito supercluster. https://research.csc.fi/taito-supercluster. (2017). Accessed 10/12/2017.

[5] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory Power Estimation and Capping. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '10)*. ACM, New York, NY, USA, 189–194. https://doi.org/10.1145/1840845.1840883

[6] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems (MEMSYS '16)*. ACM, New York, NY, USA, 455–470. https://doi.org/10.1145/2989081.2989088

[7] Mohammed El Mehdi Diouri, Manuel F Dolz, Olivier Glück, Laurent Lefèvre, Pedro Alonso, Sandra Catalán, Rafael Mayo, and Enrique S Quintana-Ortí. 2014. Assessing power monitoring approaches for energy and power analysis of computers. *Sustainable Computing: Informatics and Systems* 4, 2 (2014), 68–82.

[8] Jack Dongarra, Hatem Ltaief, Piotr Luszczek, and Vincent M Weaver. 2012. Energy footprint of advanced dense numerical linear algebra using tile algorithms on multicore architectures. In *Cloud and Green Computing (CGC), 2012 Second International Conference on*. IEEE, 274–281.

[9] Amazon EC2. Instance Types. https://aws.amazon.com/ec2/instance-types/. (????).

[10] S. Agostinelli et al. 2003. Geant4- a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506, 3 (2003), 250 – 303. http://www.sciencedirect.com/science/article/pii/S0168900203013688

[11] Daniel Hackenberg, Thomas Ilsche, Robert Schone, Daniel Molka, Maik Schmidt, and Wolfgang E Nagel. 2013. Power measurement techniques on standard compute nodes: A quantitative comparison. In *Performance Analysis of Systems*

and Software (ISPASS), 2013 IEEE International Symposium on. IEEE, 194–204.

[12] Daniel Hackenberg, Robert Schöne, Thomas Ilsche, Daniel Molka, Joseph Schuchart, and Robin Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *International Parallel and Distributed Processing Symposium (IPDPS)*. (accepted for publication).

[13] D. Hackenberg, R. Schöne, T. Ilsche, D. Molka, J. Schuchart, and R. Geyer. 2015. An Energy Efficiency Feature Survey of the Intel Haswell Processor. In *2015 IEEE International Parallel and Distributed Processing Symposium Workshop*. 896–904. https://doi.org/10.1109/IPDPSW.2015.70

[14] Marcus Hähnel, Björn Döbel, Marcus Völp, and Hermann Härtig. 2012. Measuring energy consumption for short code paths using RAPL. *ACM SIGMETRICS Performance Evaluation Review* 40, 3 (2012), 13–17.

[15] Mikael Hirki. 2015. *Energy and performance profiling of scientific computing*. Master's thesis. Aalto University.

[16] Mikael Hirki. 2017. RAPL Testing and Instruction Decoder Benchmarks. https://github.com/mhirki/idq-bench2. (2017).

[17] Mikael Hirki, Zhonghong Ou, Kashif Nizam Khan, Jukka K Nurminen, and Tapio Niemi. 2016. Empirical Study of the Power Consumption of the x86-64 Instruction Decoder. In *USENIX Workshop on Cool Topics on Sustainable Data Centers (CoolDC 16)*. USENIX Association.

[18] Song Huang, Michael Lang, Scott Pakin, and Song Fu. 2015. Measurement and Characterization of Haswell Power and Energy Consumption. In *Proceedings of the 3rd International Workshop on Energy Efficient Supercomputing (E2SC '15)*. ACM, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/2834800.2834807

[19] T. Ilsche, D. Hackenberg, S. Graul, R. Schöne, and J. Schuchart. 2015. Power measurements for compute nodes: Improving sampling rates, granularity and accuracy. In *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*. 1–8. https://doi.org/10.1109/IGCC.2015.7393710

[20] Intel Corporation 2015. *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3, System Programming Guide*. Intel Corporation.

[21] R. Kavanagh, D. Armstrong, and K. Djemame. 2016. Accuracy of Energy Model Calibration with IPMI. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*. 648–655. https://doi.org/10.1109/CLOUD.2016.0091

[22] Jaimie Kelley, Christopher Stewart, Devesh Tiwari, and Saurabh Gupta. 2016. Adaptive Power Profiling for Many-Core HPC Architectures. In *International Conference on Autonomic Computing*.

[23] K. N. Khan, F. Nybäck, Z. Ou, J. K. Nurminen, T. Niemi, G. Eulisse, P. Elmer, and D. Abdurachmanov. 2015. Energy Profiling Using IgProf. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. 1115–1118. https://doi.org/10.1109/CCGrid.2015.118

[24] Kashif Nizam Khan, Zhonghong Ou, Mikael Hirki, Jukka K. Nurminen, and Tapio Niemi. 2016. How much power does your server consume? Estimating wall socket power using RAPL measurements. *Computer Science - Research and Development* 31, 4 (2016), 207–214. https://doi.org/10.1007/s00450-016-0325-4

[25] R. Khanna, F. Zuhayri, M. Nachimuthu, C. Le, and M. J. Kumar. 2011. Unified extensible firmware interface: An innovative approach to DRAM power control. In *2011 International Conference on Energy Aware Computing*. 1–6. https://doi.org/10.1109/ICEAC.2011.6136703

[26] Gary Lawson, Masha Sosonkina, and Yuzhong Shen. 2015. Towards Modeling Energy Consumption of Xeon Phi. *arXiv preprint arXiv:1505.06539* (2015).

[27] H. Liu. 2011. A Measurement Study of Server Utilization in Public Clouds. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*. 435–442. https://doi.org/10.1109/DASC.2011.87

[28] Ioannis Manousakis, Foivos S Zakkak, Polyvios Pratikakis, and Dimitrios S Nikolopoulos. 2014. TProf: An energy profiler for task-parallel programs. *Sustainable Computing: Informatics and Systems* (2014).

[29] John D. McCalpin. 1995. Memory Bandwidth and Machine Balance in Current High Performance Computers. *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter* (Dec. 1995), 19–25.

[30] John C. McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C. Snoeren, and Rajesh K. Gupta. 2011. Evaluating the Effectiveness of Model-based Power Characterization. In *Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference (USENIXATC'11)*. USENIX Association, Berkeley, CA, USA, 12–12. http://dl.acm.org/citation.cfm?id=2002181.2002193

[31] C. Möbius, W. Dargie, and A. Schill. 2014. Power Consumption Estimation Models for Processors, Virtual Machines, and Servers. *IEEE Transactions on Parallel and Distributed Systems* 25, 6 (June 2014), 1600–1614. https://doi.org/10.1109/TPDS.2013.183

[32] Chad M Paradis. 2015. *Detailed Low-cost Energy and Power Monitoring of Computing Systems*. Master's thesis. The University of Maine.

[33] Tapasya Patki, David K. Lowenthal, Barry Rountree, Martin Schulz, and Bronis R. de Supinski. 2013. Exploring Hardware Overprovisioning in Power-constrained, High Performance Computing. In *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing (ICS '13)*. ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/2464996.2465009

[34] Plugwise:. 2017. Energy management systems of the 21st century. https://www.plugwise.com/. (2017). Accessed on 09/14/2017.
[35] Harald Servat, Germán Llort, Judit Giménez, and Jesús Labarta. 2013. Detailed and simultaneous power and performance analysis. *Concurrency and Computation: Practice and Experience* (2013).
[36] A. Skrenes and C. Williamson. 2016. Experimental Calibration and Validation of a Speed Scaling Simulator. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*. 105–114. https://doi.org/10.1109/MASCOTS.2016.49
[37] Balaji Subramaniam and Wu-chun Feng. 2015. On the Energy Proportionality of Scale-Out Workloads. *CoRR* abs/1501.02729 (2015). http://arxiv.org/abs/1501.02729
[38] Balaji Subramaniam and Wu-chun Feng. 2013. Towards Energy-proportional Computing for Enterprise-class Server Workloads. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13)*. ACM, New York, NY, USA, 15–26. https://doi.org/10.1145/2479871.2479878
[39] Vincent Weaver. 2015. rapl-read.c. (2015). http://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c.
[40] V. M. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. 2012. Measuring Energy and Power with PAPI. In *2012 41st International Conference on Parallel Processing Workshops*. 262–268. https://doi.org/10.1109/ICPPW.2012.39
[41] Vincent M. Weaver, Matt Johnson, Kiran Kasichayanula, James Ralph, Piotr Luszczek, Dan Terpstra, and Shirley Moore. 2012. Measuring Energy and Power with PAPI. In *Proceedings of the 2012 41st International Conference on Parallel Processing Workshops (ICPPW '12)*. IEEE Computer Society, Washington, DC, USA, 262–268. https://doi.org/10.1109/ICPPW.2012.39
[42] Simon N Wood. 2017. *Generalized additive models: an introduction with R*. CRC press.
[43] Yan Zhai, Xiao Zhang, Stephane Eranian, Lingjia Tang, and Jason Mars. 2014. HaPPy: Hyperthread-aware Power Profiling Dynamically. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*. USENIX Association, Philadelphia, PA, 211–217. https://www.usenix.org/conference/atc14/technical-sessions/presentation/zhai
[44] Huazhe Zhang and Henry Hoffmann. 2015. A Quantitative Evaluation of the RAPL Power Control System. In *Proceedings of the 10th International Workshop on Feedback Computing*.
[45] Huazhe Zhang and Henry Hoffmann. 2016. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid Techniques. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*. ACM, New York, NY, USA, 545–559. https://doi.org/10.1145/2872362.2872375

Received June 2017; revised October 2017; accepted January 2018