Comput Sci Res Dev (2016) 31:207–214
DOI 10.1007/s00450-016-0325-4

SPECIAL ISSUE PAPER

How much power does your server consume? Estimating wall socket power using RAPL measurements

Kashif Nizam Khan[1,2] · **Zhonghong Ou**[3] · **Mikael Hirki**[1,2] · **Jukka K. Nurminen**[2,4] · **Tapio Niemi**[2]

Published online: 8 August 2016

Abstract Full system electricity intake from the wall socket is important for understanding and budgeting the power consumption of large scale data centers. Measuring full system power, however, requires extra instrumentation with external physical devices, which is not only cumbersome, but also expensive and time consuming. To tackle this problem, in this paper, we propose to model wall socket power from processor package power obtained from the running average power limit (RAPL) interface, which is available on the latest Intel processors. Our experimental results demonstrate a strong correlation between RAPL package power and wall socket power consumption. Based on the observations, we propose an empirical power model to predict the full system power. We verify the model using multiple synthetic benchmarks (Stress-ng, STREAM), high energy physics benchmark (ParFullCMS), and non-trivial application benchmarks (Parsec). Experimental results show that the prediction model achieves good accuracy, which is maximum 5.6 % error rate.

Keywords Power modeling · RAPL · Energy efficiency · HPC

✉ Kashif Nizam Khan
kashif.khan@aalto.fi

1 Department of Computer Science, Aalto University, Espoo, Finland

2 Helsinki Institute of Physics, Helsinki, Finland

3 Beijing University of Posts and Telecommunications, Beijing, China

4 VTT Technical Research Centre of Finland, Helsinki, Finland

1 Introduction

Power consumption of servers can be acquired by different approaches, e.g., using external devices like energy sensors and energy meters, and modeling the power consumption with the help of performance counters and external devices like *Mantis* [7]. Mounting energy sensors, watt meters, or instrumenting the systems with energy meters, however, is not only expensive, but also hinders the normal operation of the data center [16].

The introduction of running average power limit (RAPL) [12] to the latest Intel processors has mitigated this problem. RAPL provides model specific registers (MSRs) to read the processor energy consumption values in real time. It provides energy readings from four domains, including `package` (both cores and uncores, i.e., last-level cache), `pp0` (all cores in one package), `pp1` [specific device in the uncore, i.e., on-chip graphics processing unit (GPU)], and dynamic random access memory (`DRAM`) plane. With RAPL, we can get fine-grained and reliable energy measurements without needing to custom-instrument the hardware [9].

The introduction of RAPL has enabled multiple new opportunities, e.g., measuring energy consumption of short-code paths [10], power limiting and capping for main memory [6]. Furthermore, RAPL has been extensively studied and incorporated in different tools for fine-grained energy measurements of computing systems [9,11]. Unlike those works, in this paper, we take a different angle to utilize RAPL, i.e., leveraging RAPL readings to model power consumption drawn from the wall socket.

Knowing the wall socket power draw is beneficial. It helps to determine the exact energy spending, and thus allocate proper energy budget for the system. In addition, it is helpful to set power limit properly to best utilize pricing variations (e.g., setting a high power limit when the hourly electricity price is low, and vice versa) [1]. Utilizing RAPL readings to provide estimates for wall socket power brings several advantages: (1) it promises to minimally interrupt the regular operations of data centers; (2) it is easily executable because it does not require any external sensors or energy meters to be mounted with the system.

We make the following major contributions in this paper:

1. Through a set of experiments performed on three different Intel-based systems, we demonstrate that there exists a strong linear relationship between wall power and package power.
2. Based on the observation mentioned above, we apply machine learning techniques to formulate a power model, which can predict the wall power from RAPL package power when arbitrary workloads are running in the system.
3. Experimental results demonstrate good accuracy of the model, i.e., it can achieve maximally 5.6 % error rate at different processor frequencies.

The rest of the paper is structured as follows. Section 2 discusses the related work, while Sect. 3 describes the experimental setup and benchmark specifications. Section 4 presents the experimental results from our empirical study, and Sect. 5 describes the model formulation. Section 6 discusses the use case of our model and Sect. 7 concludes the paper.

2 Related work

Prior to the introduction of RAPL, power modeling techniques usually focused on carefully designed software monitors and hardware measurement tools, or leveraging machine learning techniques such as stochastic power models. McCullough et al. [15] presented an extensive evaluation of such techniques. They observed that such models perform well for single- and multi-core scenarios but perform poorly for subsystem power modeling due to increased system complexity and hidden power states that are not exposed to OS. The introduction of RAPL mitigates the hidden power states problem because they expose the energy readings and power states to the OS now. Hackenberg et al. [8] presented a comprehensive overview of different power consumption measurement methodologies using RAPL. Venkatesh et al. [21] proposed a new shared-memory window-based solution to model the energy consumed by processes engaged in message passing interface (MPI) operations using RAPL. Balaji et al. [20] investigated the efficacy of RAPL in achieving energy proportionality for *SPECpower* benchmark. Similar to our work, Castano et al. [5] presented a model for full system instantaneous power dissipation using energy consumption information from a subset of advanced technology extended (ATX) lines. In addition, a lot of work has been directed towards modeling the full system power consumption of server based systems [4,7,10,17].

There are several differences between these works and our approach. Firstly, most of the power modeling techniques formulate the power models using bottom up approach, i.e., modelling the power consumption of the individual components and then adding up to the full system power. Such approaches tend to accumulate the modeling errors to the full system, and usually the error rate can reach up to 10 %, while our approach can achieve better accuracy. Secondly, the approaches that achieve low error rates usually perform instrumentation of the system board with external metering tools, while our approach avoids the usage of such tools during the regular operation of the system.

3 Experimental setup and benchmark specifications

We choose three different Intel-based systems to conduct the experiments. Table 1 lists the specifications of the three systems. For brevity, in the rest of the paper, we refer to the machines as Machine 1, Machine 2, and Machine 3, respectively. Machine 1 and 2 are workstation-grade and Machine 3 is a server-grade machine. To cover different aspects of the systems, we use in total 16 different workloads (cf. Table 2) which cover CPU, memory, and network-intensive tasks, HEP workloads, and non-trivial applications.

Stress-ng [19] is originally designed to stress different subsystems of a computer. In our work, we use Stress-ng to stress the CPU cores with 100 % workload running a number of *sqrt()* operations on pseudo-random values.

Stream McCalpin [14] is a well known benchmark designed to measure the sustainable memory bandwidth. Using *Stream* helps us understand the characteristics of different systems in terms of power consumption when running a memory intensive task.

ParFullCMS is a Geant4 [2] 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) experiments in CERN.

Parsec is not a synthetic benchmark, hence it provides opportunities to test power models for diverse instruction mix, memory access and network operations [3]. It includes emerging applications from different application domains, e.g., financial, computer vision, deduplication. In Table 2 the 13 benchmarks starting from *Black-scholes* are all from Parsec benchmark suite.

To measure the power consumption from the wall, we deploy Plugwise Smart Plug [18]. The RAPL measurements are obtained using the latest stable version of *Likwid* tool set [13].

Table 1 System specifications

Processor (Intel)	Sockets	Cores	Hyperthreads	Frequency range (GHz)	L3 cache (MB)	Memory (GB)	Tag
Core i7-4770	1	4	4	0.8–3.4	8	16	Machine 1
Xeon E3-1230	1	4	4	0.8–3.3	8	16	Machine 2
Xeon E5-2650	2	16	16	1.2–2.6	40	64	Machine 3

Table 2 Benchmark specifications

Benchmark	Description
Stress-ng	Stresses the CPUs
Stream	Calculates memory bandwidth
ParFullCMS	Simulates HEP events
Black-Scholes	Computes Black-Scholes PDE
Body-track	Tracks human-computer vision
Canneal	Cache-aware simulated annealing
Dedup	Compresses data stream
Facesim	Simulates human face motions
Ferret	Content based similarity search
Fluidanimate	Fluid simulation for animation
Freqmine	Frequent itemset mining
Raytrace	Real-time raytracing
Streamcluster	Performs online stream cluster
Netdedup	High-bandwidth single connection
Netferret	Latency intensive
Netstreamcluster	High-bandwidth multi-connections

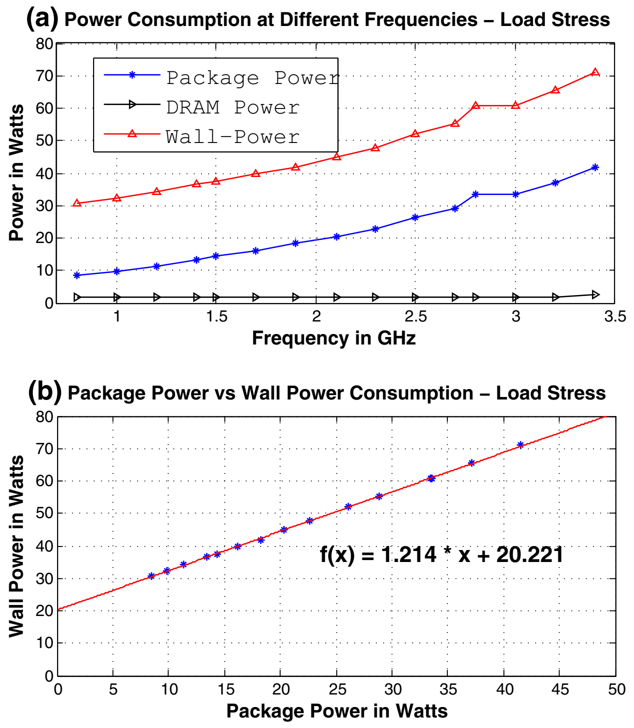

Fig. 1 Experimental results of Machine 1-*Stress-ng*

4 Modeling wall power consumption from RAPL

In this section, we present the experimental results and discuss the empirical analysis that helps understand the modeling technique.

4.1 Stress-ng and Stream benchmarks observations

Figure 1a presents the power consumption in processor package, DRAM interface and wall power consumption when the CPU frequency varies for the CPU-intensive Stress-ng workload. All the three systems presented in Table 1 offer 15 distinct frequencies that are chosen by the operating system in dynamic voltage and frequency scaling (DVFS). For these experiments, we pin the frequency at a specific value to acquire the exact power consumption of the system for that frequency. We plot the wall power consumption against the package power in Fig. 1b. The linear curve fit in this figure suggests that there is a near exact linear relationship between the package power and wall power. In fact the correlation between these two values is *0.999*. For the *Stress-ng* benchmark, the linear equation obtained from the linear fitting is:

$$P_{wall} = 1.214 * P_{package} + 20.221 \tag{1}$$

wherein, $P_{package}$ represents the package power as measured with RAPL, and P_{wall} represents the wall power. For Machine 1, we perform similar experiments with *Stream* benchmark. The observations are similar. In this case, the correlation between the package power consumption and wall power consumption is again *0.999* and the linear equation is presented in Eq. 2.

$$P_{wall} = 1.212 * P_{package} + 25.580 \tag{2}$$

For Machine 1 and 2, the idle wall power consumption are *24.65* and *33.97* watts respectively. We observe that the constant part of the linear Eqs. 1 and 2 are dominated by the idle wall power consumption. In general, the observations for Machine 2 follow the same trends as Machine 1. Thus, for brevity, we do not present the results from Machine 2 in this paper.

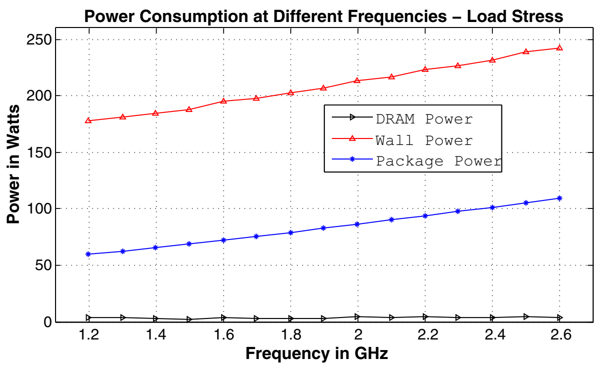

Fig. 2 Experimental results of Machine 3-*Stress-ng*

For the server-grade Machine 3, the same set of experiments also exhibit a linear relationship between package power and wall power consumption. Observations for *Stress-ng* benchmark are presented in Fig. 2. In case of Machine 3, the correlation between package power and wall power is *0.999* for both *Stress* and *Stream* benchmarks. The linear equations for Machine 3 are presented in Eqs. 3 and 4, for *Stress-ng* and *Stream* workload, respectively.

$$P_{wall} = 1.327 * P_{package} + 97.732 \tag{3}$$

$$P_{wall} = 1.330 * P_{package} + 116.70 \tag{4}$$

Although Machine 3 is a server-grade machine with two processor sockets, the relationship between package power and wall power remains linear.

4.2 ParFullCMS and Parsec benchmarks observations

Figure 3 presents a detailed view of the wall power and package power consumption over time (in seconds) as we run ParFullCMS on different processor frequencies. Figure 3 demonstrates that the multithreaded ParFullCMS has two distinct phases: the initialization phase and the compute-intensive phase. The initialization phase sets up the events to be processed and consumes relatively less power, whereas the compute intensive phase performs bulk of the simulation tasks and consumes relatively more power. It is evident from the figure that irrespective of processor frequencies, package power and wall power correlate strongly with each other and follow the same pattern. In this case the correlation coefficient is *0.997*, and the linear equation is presented in Eq. 5.

$$P_{wall} = 1.140 * P_{package} + 21.40 \tag{5}$$

Figure 4 presents the package power vs. wall power consumption for a subset of the Parsec benchmarks running on Machine 1. Because of space limit, Fig. 4 only includes the observations from *canneal*, *fluidanimate*, *ferret*, *facesim*, *netferret* and *netstreamcluster*. The observations from the rest of the Parsec benchmark are also similar. We run Parsec benchmark with *native* input size (the largest input set) and make sure that the number of threads are enough to keep all the physical cores active. Parsec results (Fig. 4) show that the strong correlation between wall power and package power holds for non-trivial applications.

The observations obtained from running ParFullCMS and Parsec benchmarks confirm that irrespective of the type of workload, careful formulation of power models can yield the full system power consumption from RAPL package power. Our experiments show that the package power consumption almost always remains strongly correlated with full system power consumption, even when the non-cpu power draw of a machine is not constant, and when the different components of the system exhibit dynamic behaviour with multiple phases.

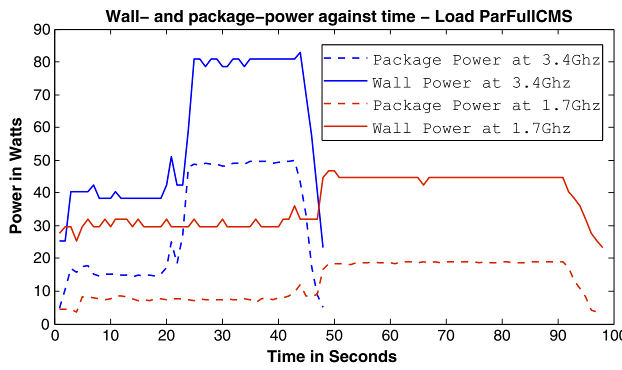

Fig. 3 Wall and package power consumption with time-*ParFullCMS* - Machine 1

5 Wall power modeling

As shown in previous sections, the linear model has an excellent fit but the co-efficients vary for different workloads. Thus, we need an approach to calibrate the model for any arbitrary workload. In this section, we develop a generic power model for wall power consumption using machine learning techniques. Note that for brevity, in the rest of the paper, we use Machine 1 only to present the results. The other machines demonstrate similar trends.

5.1 Model calibration

We use *Stream*, *Stress-ng* and *Parsec* benchmarks data for training, and validating the wall power consumption model. We then use the *ParFullCMS* data to test the accuracy of the model.

We formulate the model using the general least square solution. Assuming that we have a training set of N obser-

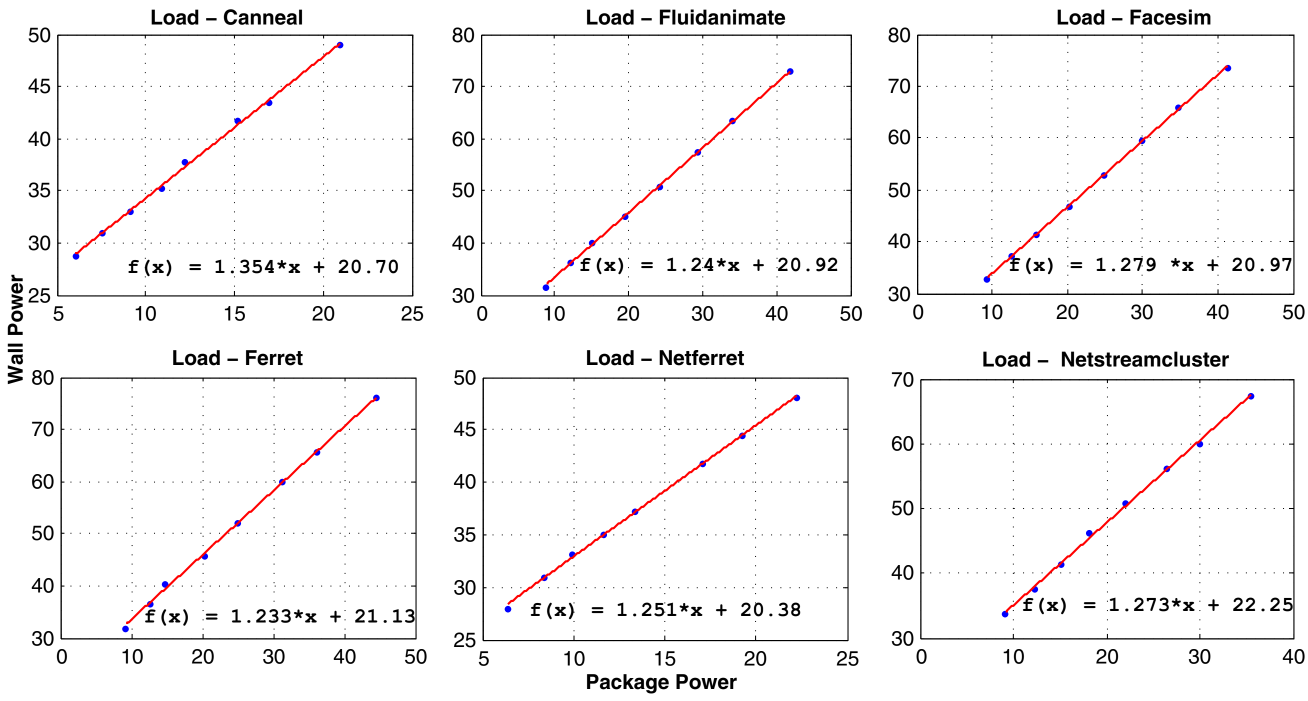

Fig. 4 Package power vs. wall power-*Parsec*

vations of package power and wall power pairs (*pkg*,*wall*), our aim is to obtain a function $f : \mathbb{R} \rightarrow \mathbb{R}$ (which calculates wall power from package power) so that the average squared error E is minimized. E can be defined as

$$E = \frac{1}{N}\sum_{t=1}^{N}(wall_t - f(pkg_t))^2. \tag{6}$$

Let us consider basis functions $y_i : \mathbb{R} \rightarrow \mathbb{R}$. In general terms, $f(pkg)$ can be defined as

$$f(pkg) = \sum_{i=0}^{k-1} a_i \cdot y_i(pkg) \tag{7}$$

where $A = (a_0, \ldots, a_{k-1})$ is a $1 \times k$ vector. Our aim is to find an optimum $f(x)$ that minimizes E. E is minimum when its partial derivatives become zero. We can then acquire

$$\frac{\delta E}{\delta a_j} = \frac{1}{N}\sum_{t=1}^{N} 2\left((wall_t) - \sum_{i=0}^{k-1} a_i \cdot y_i(pkg_t)\right) y_j(pkg_t). \tag{8}$$

If we simplify Eq. 8, we can get

$$\sum_{t=1}^{N} wall_t y_j(pkg_t) = \sum_{t=1}^{N}\sum_{i=0}^{k-1} a_i \cdot y_i(pkg_t) y_j(pkg_t) \tag{9}$$

where j varies from 0 to $k-1$. In simplified matrix notation, Eq. 9 can be written as

$$A = WY^T(YY^T)^{-1}. \tag{10}$$

In Eq. 10, Y is a $k \times N$ matrix where $Y_{i,t} = y_i(pkg_t)$ and $W = (wall_0, \ldots, wall_t)$. We define the basis function as $y_i(pkg) = pkg^i$, which means function f is defined by a polynomial with k terms and A becomes the vector of coefficients obtained from polynomial regression.

We solve Eq. 10 for different orders of polynomials k, where k varies from 1 to 4. From previous sections we get the inductive bias of our learning algorithm that there exist a linear relation between RAPL package and wall power. However, to test the generalization of our assumptions, we test different orders of polynomials on our data set to see whether the linear relationship holds also for diverse workload mix in comparison to higher order polynomials. Once we formulate the model, we calculate E_T, E_V, E_{T+V} and E_{Test}, which represents average squared training error, validation error, training+validation error, and test error, respectively. The values are presented in Table 3. E_V and E_{T+V} (Table 3) give us the accuracy of the validation set and the total training set. The lower the values of E_V and E_{T+V} the better the model performs.

As Table 3 suggests, the training data and test data performs the best for linear regression. The obtained power model for Machine 1 is

$$P_{wall} = 1.227 * P_{package} + 22.084 \tag{11}$$

Table 3 Polynomial regression results

k	E_T	E_V	E_{T+V}	E_{Test}
1	4.87	6.42	5.66	5.59
2	4.38	7.89	6.16	7.08
3	4.37	8.34	6.37	7.13
4	4.37	8.17	6.30	7.18

Equation 11 predicts the wall power consumption for *ParFullCMS* benchmark with an average square error rate of *5.59 %* for different processor frequencies on Machine 1.

Our model requires a one time run of the benchmarks with the external AC-power measurement equipment connected for a single machine. Once we acquire the wall power and the corresponding package power consumption data for a specific machine, we perform the training and validation stage that results in a power model. The power model is then tested for any arbitrary application to measure the accuracy. The model can predict the wall power consumption of any workload running on the machine without the external power meters with promising accuracy. If the machine has access to external power meter at a later point of time, the evaluation stage can recalibrate the predicted data with the real data measured. The feedback information can be fed into the training and validation stage to fine tune and recalibrate the model for better prediction.

6 Discussion

The proposed model can predict wall power from RAPL package power for any work-load where CPU performs the bulk of the system operations. There are cases when RAPL measurements are not enough to measure the wall power consumption, specifically when there are components other than the CPU conducting bulk of the system operations. For example, a file server with multiple disks is performing a disk intensive task, or a server with a separate (non-integrated) GPU processor where the processing is executed by the GPU rather than the CPU. For the former example, the wall power consumption can be estimated using the following equation.

$$P_t = P_i + P_{RAPL} + P_{disk} \tag{12}$$

where P_t is the wall power consumption at time t, P_i is the wall power consumption when the system is idle, P_{RAPL} is the difference of power consumption between operating mode and idle mode in RAPL domain, and P_{disk} is the difference of power consumption between operating mode and idle mode of the disk drive that can be obtained from its specification datasheet.

Be noted that although we only focus on Intel processors supporting RAPL feature, our method itself is not limited to RAPL, because it only needs power consumption data of different components of a computing system (specifically CPU package and DRAM power). As such, similar models can be developed for AMD or ARM processors as well.

7 Conclusion

In this paper, we present an empirical study on wall socket power consumption and propose a power model to predict wall power from RAPL package power. The proposed power model can predict full system power for any workload with only one time calibration with external power meter. Experimental results demonstrate that our model can predict wall power consumption with an accuracy of 5.6 % error rate in the worst case. Our findings suggest that a careful formulation of the linear coefficients can present a useful and efficient model to predict wall socket power consumption from RAPL package. We plan to extend our work by evaluating the power draw characteristics of different processor architecture (AMD, ARM etc.) and fine-tune the power model with diverse workloads.

References

1. Agostinelli S et al (2003) GEANT4: A simulation toolkit. Nucl Instrum Methods A506:250–303
2. Bienia C, Kumar S, Singh JP, Li K (2008) The parsec benchmark suite: characterization and architectural implications. In: Proceedings of the 17th international conference on parallel architectures and compilation techniques
3. Bircher W, John L (2012) Complete system power estimation using processor performance events. Comput IEEE Trans 61(4):563–577
4. Castano M, Catalan S, Mayo R, Quintana-Orti E (2015) Reducing the cost of power monitoring with DC wattmeters. Comput Sci Res Dev 30(2):107–114
5. David H, Gorbatov E, Hanebutte UR, Khanna R, Le C (2010) Rapl: memory power estimation and capping. In: Low-power electronics and design (ISLPED), 2010 ACM/IEEE international symposium on, pp 189–194
6. Economou D, Rivoire S, Kozyrakis C (2006) Full-system power analysis and modeling for server environments. In: Workshop on Modeling Benchmarking and Simulation (MOBS)
7. Hackenberg D, Ilsche T, Schone R, Molka D, Schmidt M, Nagel W (2013) Power measurement techniques on standard compute nodes: a quantitative comparison. In: Performance analysis of systems and software (ISPASS), 2013 IEEE international symposium on, pp 194–204
8. Hackenberg D, Schöne R, Ilsche T, Molka D, Schuchart J, Geyer R (2015) An energy efficiency feature survey of the intel haswell processor. In: 2015 IEEE international parallel and distributed processing symposium workshop, IPDPS 2015, Hyderabad, India, May 25–29, 2015, pp 896–904
9. Hähnel M, Döbel B, Völp M, Härtig H (2012) Measuring energy consumption for short code paths using RAPL. ACM SIGMETRICS Perform Eval Rev 40(3):13–17
10. Huang S, Lang M, Pakin S, Fu S (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, pp 7:1–7:10
11. Intel: Intel 64 and IA-32 Architectures Software Developer's Manual Volume 3 (3A, 3B & 3C): System Programming Guide (2014). http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-system-programming-manual-325384.pdf. Accessed 6 Aug 2015
12. Lightweight performance tools, Likwid. https://code.google.com/p/likwid/. Accessed 6 Aug 2015
13. McCalpin JD. STREAM: sustainable memory bandwidth in high performance computers. http://www.cs.virginia.edu/stream/. Accessed 21 Sep 2015
14. McCullough JC, Agarwal Y, Chandrashekar J, Kuppuswamy S, Snoeren AC, Gupta RK (2011) Evaluating the effectiveness of model-based power characterization. Proceedings of the 2011 USENIX conference on USENIX annual technical conference., USENIXATC'11USENIX Association, Berkeley, pp 12–12
15. Orgerie AC, Assuncao MDd, Lefevre L (2014) A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Comput Surv 46(4) 47:1–47:31
16. Piga L, Bergamaschi R, Rigo S (2014) Empirical and analytical approaches for web server power modeling. Clust Comput 17(4):1279–1293
17. Plugwise. https://www.plugwise.com/. Accessed 6 Aug 2015
18. Qureshi A, Weber R, Balakrishnan H, Guttag J, Maggs B (2009) Cutting the electric bill for internet-scale systems. SIGCOMM Comput Commun Rev 39(4):123–134
19. Stress-ng. http://kernel.ubuntu.com/~cking/stress-ng/. Accessed 21 Sep 2015
20. Subramaniam B, Feng W (2013) Towards energy-proportional computing for enterprise-class server workloads. In: Proceedings of the 4th ACM/SPEC international conference on performance engineering., ICPE '13ACM, New York, pp 15–26
21. Venkatesh A, Kandalla K, Panda D (2013) Evaluation of energy characteristics of MPI communication primitives with RAPL. In: Parallel and distributed processing symposium workshops PhD Forum (IPDPSW), 2013 IEEE 27th International, pp 938–945

Kashif Nizam Khan is currently working with Helsinki Institute of Physics and Department of Computer Science, Aalto University towards his doctoral degree. He obtained his Masters in Security and Mobile Computing from Aalto University, Finland and NTNU, Norway (2008–2010). His current research focus is on energy efficiency in datacenters and big-data processing frameworks.

Zhonghong Ou is an associate professor at the School of Computer Science, Beijing University of Posts and Telecommunications, China, since January 2016. He obtained his Ph.D. degree from University of Oulu, Finland. From 2010 to 2015, he was a post-doc researcher at Aalto University, Finland. From December 2009 to April 2010, he was a visiting scholar at Internet Real-Time (IRT) Lab at Columbia University. From March 2013 through August 2013, he was a visiting scholar at Intel Labs, Portland, United States. Zhonghong has a wide spectrum of research interests, including cloud computing platforms, cloud storage, big data analytics.

Mikael Hirki is currently a software developer at AC Electric Vehicles, Ltd. Previously he worked as a research assistant at Helsinki Institute of Physics. Mikael received a B.Sc degree in 2013 and an M.Sc degree in 2015 from Aalto University. He is interested in high performance software and energy-efficient computing.

Jukka K. Nurminen is a principal scientist at VTT and an adjunct professor of computer science at Aalto University. In 2011-15 he spent five years as a professor at Aalto working on mobile cloud computing and energy-efficiency. He has a strong industry background with almost 25 years experience of software research at Nokia Research Center. Jukkas experience ranges from mathematical modeling to expert systems, from network planning tools to solutions for mobile phones, and from R&D project management to tens of patented inventions. Jukka received his M.Sc degree in 1986 and Ph.D. degree in 2003 from Helsinki University of Technology. His research interest are focused on systems and solution that are energy-efficient, distributed, or mobile.

Tapio Niemi holds a PhD in Computer Science from the University of Tampere (2001). He worked at the University of Tampere as a scientist (1997–2001) and as an assistant professor (2002–2004). Since 2004 he has worked in the Technology Programme of the Helsinki Institute of Physics (HIP) at CERN, and since 2012 also as a senior scientist in the Department of Operations at HEC-Lausanne, Switzerland, focusing on green computing, data analytics, and business intelligence.