Journal of Parallel and Distributed Computing 151 (2021) 38–51

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Improving the accuracy of energy predictive models for multicore CPUs by combining utilization and performance events model variables☆

Arsalan Shahid, Muhammad Fahad, Ravi Reddy Manumachu, Alexey Lastovetsky *

School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland

ARTICLE INFO

Article history:
Received 9 November 2020
Accepted 28 January 2021
Available online 11 February 2021

Keywords:
Performance monitoring counters
CPU utilization
Energy modeling
Energy predictive models
Multicore CPU

ABSTRACT

Energy predictive modeling is the leading method for determining the energy consumption of an application. Performance monitoring counters (PMCs) and resource utilizations have been the principal source of model variables primarily due to their high positive correlation with energy consumption. Performance events, however, have come to dominate the landscape due to their better prediction accuracy compared to utilization variables. Recently, the theory of energy of computing has been proposed whose practical implications for constructing accurate and reliable linear energy predictive models are unified in a consistency test that includes a selection criterion of additivity for model variables. In this work, we analyze the prediction accuracy of models employing utilization variables only, PMCs only, and combination of both utilization variables and PMCs, through the lens of this theory for modern multicore CPU platforms. We discover that employing utilization variables only in linear energy predictive models does not capture all the energy-consuming activities during an application execution. However, combination of utilization variables with PMCs that are highly additive and highly correlated with energy consumption, gives the most accurate linear energy predictive model. Our experimental results show that application-specific and platform-level models using both utilization variables and PMCs exhibit up to 3.6× and 2.6× better average prediction accuracy respectively when compared with models employing utilization variables only and highly additive PMCs only.

1. Introduction

Accurate measurement of energy consumption during an application execution is key to energy minimization at application level. There are three mainstream measurement approaches: (a) System-level physical power measurements using external power meters, (b) Using on-chip power sensors, and (c) Energy predictive models. System-level physical measurements using external power meters are considered the ground truth. Fahad, Shahid, Reddy, Lastovetsky [23] present a comparative study of on-chip sensors and energy predictive models against the ground truth. Briefly, they show that the profiles obtained for dynamic energy consumption of the applications using on-chip sensors deviate significantly from the ground truth suggesting that on-chip power sensors do not capture the dynamic energy consumption during an application run holistically. We will present a synopsis of the development of the energy predictive modeling landscape.

Energy predictive models emerged as a dominant energy measurement approach because of their ability to provide a fine-grained component-level breakdown of energy consumption. Resource utilizations and performance monitoring counters (PMCs) have been the principal source of model variables primarily due to their high positive correlation with energy consumption. There are three prominent kinds of models based on them. The first kind [25,27,32,36,58,64] is based on utilizations of resources (CPU, memory, disk, and network). The second kind [7–10,31,34,39,40,51] employs PMCs. PMCs are special-purpose hardware registers provided in modern processor architectures to record the counts of software events, that represent the kernel-level activities such as *page-faults*, *context-switches*, etc., and hardware events arising from the micro-architecture core and the performance monitoring unit such as *CPU-cycles*, *branch-instructions*, *cache-misses*, etc. While utilizations are high-level metrics, PMCs are pure counters that contain activity or access counts. The third kind of models is based on both utilizations and PMCs [22,37,49]. All the proposed models (with the exception of [37,51]) predict power consumption. The energy consumption during an application execution is then determined through a creative application of the power model. One approach is to obtain the area under the discrete function of the power measurements provided by

☆ This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

* Corresponding author.
E-mail addresses: arsalan.shahid@ucd.ie (A. Shahid), muhammad.fahad@ucdconnect.ie (M. Fahad), ravi.manumachu@ucd.ie (R.R. Manumachu), alexey.lastovetsky@ucd.ie (A. Lastovetsky).

https://doi.org/10.1016/j.jpdc.2021.01.007

Table 1
Specification of the Intel Haswell (HCLServer1) and Intel Skylake (HCLServer2) multicore CPU Server.

Hardware specifications	Intel Haswell server	Intel Skylake server
Processor	Intel E5-2670 v3 @2.30 GHz	Intel(R) Xeon(R) Gold 6152
Micro-architecture	Haswell	Skylake
Cores per socket	12	N/A
Socket(s)	2	1
Thread(s) per core	2	2
Main memory	64 GB DDR4	96 GB
CPU TDP	240 W	140 W
Memory TDP	12 W	9.2 W
NUMA node(s)	2	1
Idle power	58 W	32 W
L11 cache	32 kB	32 kB
L1d cache	32 kB	32 kB
L2 cache	256 kB	1024 kB
L3 cache	30720 kB	30976 kB
Software specifications		
OS release	CentOS 7	Ubuntu 16.04 LTS
Linux kernel	3.10	3.10
OpenMP version	3.1	3.1
MPI version	3.2.1	N/A
Compiler	gcc 4.8.5	gcc 4.8.5
Python version	3.4.3	3.6.8
Likwid version	4.1	4.3.2
Intel MKL version	2017.0.2	2017.0.2

the model versus the time intervals between the measurements. Well-known numerical approaches such as the trapezoidal rule can be used to calculate this area approximately.

Utilization models were shown to exhibit poor prediction accuracy than PMC-based models [48,49] on multicore CPU platforms. Research works [23,30,42,45] demonstrate the poor prediction accuracy of PMC-based models and report that the linear regression models yield prediction errors as high as 150%. This can be explained as follows. First, modern multicore CPU platforms have several inherent complexities which are: (a) Severe resource contention due to tight integration of tens of cores organized in multiple sockets with multi-level cache hierarchy and contending for shared on-chip resources such as the last-level cache, interconnect, and DRAM controllers; (b) Non-uniform memory access (NUMA) where the time for memory access between a core and main memory is not uniform and where main memory is distributed between locality domains or groups called NUMA nodes; and (c) Dynamic power management of multiple power domains (CPU sockets, DRAM). Due to these complexities, the energy consumption of a computing resource demonstrates a non-linear and non-smooth functional relationship with the utilization of the resource. Second, a sound theoretical framework to understand the fundamental significance of the model variables with respect to the energy consumption and the causes of inaccuracy or the reported wide variance of accuracy of the models has been lacking.

The theory of energy of computing has progressively matured over the past three years starting with a proposal of a criterion for selection of PMCs in the research work [52] followed by a formal description of the theory and its practical implications in [53]. Shahid et al. [53] propose a novel theory of energy of computing and unified its practical implications to increase the prediction accuracy of linear energy predictive models in a *consistency* test, which contains a suite of properties that include determinism, reproducibility, and additivity to select model variables and constraints for model coefficients. The authors show that the average prediction accuracy of linear regression models can be improved from 31% to 18% by selecting PMCs that pass the consistency test of the theory of energy of computing [51].

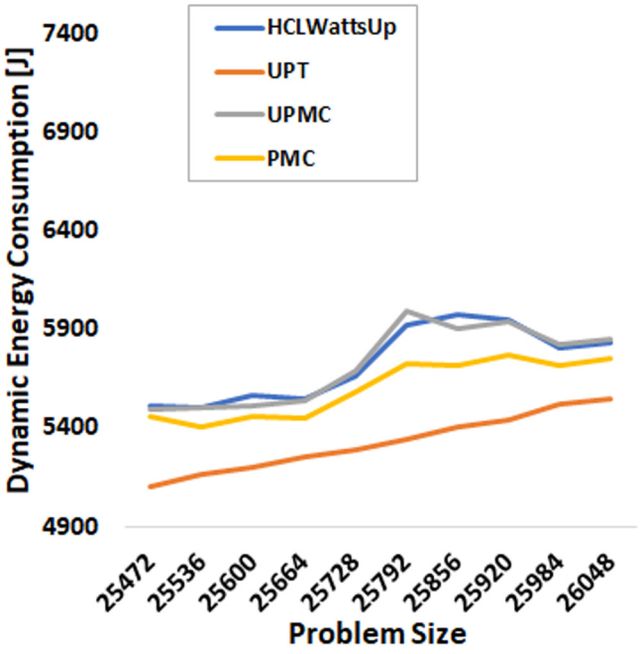


Fig. 1. Comparison of predictions of models employing utilization variables only (UPT), PMCs only (PMC), and combining both utilization variables and PMCs (UMPC). HCLWattsUp represents the ground truth, which is system-level physical measurements using power meters. Application employed is dense matrix multiplication (DGEMM) executing on HCLServer2.

In this work, we aim to improve the prediction accuracy of linear energy predictive models further by combining utilization variables and PMCs. We perform a comparative study of the prediction accuracy of models employing utilization variables only, PMCs only, and combining both utilization variables and PMCs using the consistency test of the theory of energy of computing on modern multicore CPU platforms. We first check the reliability of CPU and memory utilizations as model variables in energy predictive models using the consistency test. We study the additivity of average CPU and memory utilization for the execution of applications on two modern multicore servers, HCLServer1 and HCLServer2 (specifications are shown in Table 1). Our results show that utilization variables are highly additive (satisfying the input tolerance of 5%) for our application suite and pass the consistency test.

We then employ the utilization variables in linear energy predictive models at two levels, platform and application, on both the servers. We demonstrate that the models exhibit poor accuracy with an average prediction error of up to 50%. Models that employ only PMCs have an average prediction error of up to 36%. Combining utilization variables along with PMCs that are highly additive and highly correlated with energy consumption, however, yields the most accurate linear energy predictive model. Application-specific and platform-level models using both utilization variables and PMCs perform up to 3.6× and 2.6× better in terms of average prediction accuracy when compared with models employing utilization variables only and PMCs only.

We illustrate our findings using results from one of our experiments. Fig. 1 shows the predictions of application-specific models constructed for dense matrix multiplication (DGEMM) employing utilization variables only (UPT), PMCs only (PMC), and combining both utilization variables and PMCs (UPMC). HCLWattsUp represents the ground truth, which is system-level physical power measurements using power meters. The ground truth profile exhibits drastic variations due to the inherent complexities in

modern multicore CPU platforms. Due to these complexities, the profiles of dynamic energy and workload size for real-life data-parallel applications have complex and non-smooth functional relationship [38,46,47]). Utilization variables capture the overall energy consumption trend (or the average energy consumption) of the profiles of the application executions. However, they do not capture the variations in the profiles. Models based on highly additive and highly energy-correlated PMCs accurately capture these variations that account for most of the energy-consuming activities during the execution of an application. They do not, however, account for some energy-consuming activities that are captured by high-level utilization variables. Models that employ both utilization variables and highly additive and highly energy-correlated PMCs are able to account for all the energy-consuming activities during the execution of an application and hence are found to provide the best accuracy.

The original contributions of this work are summarized as follows:

- A first experimental study analyzing the prediction accuracy of linear energy predictive models employing utilization variables only, PMCs only, and both utilization variables and PMCs and which are selected based on the theory of energy of computing on modern multicore CPU platforms.
- We show that models employing both utilization variables and highly additive and energy correlated PMCs are able to better account for the energy-consuming activities during the execution of an application and hence are found to provide significant improvements in prediction accuracy compared to models that are based on utilization variables only and PMCs only.

We organize the rest of this paper as follows. Section 2 presents the terminology. Section 3 contains the literature review. Section 4 overviews the theory of energy of computing and contains the expressions for various energy predictive models. Section 5 contains the experimental setup and details the selection and measurement of model variables. Section 6 presents the experimental results and analysis. In Section 7, we present the discussions containing the learned lessons for improving the prediction accuracies of energy predictive models and future work. Finally, Section 8 concludes the paper.

2. Terminology: Energy consumption and energy predictive models

There are two types of energy consumptions, static energy, and dynamic energy. The total energy consumption is the sum of dynamic and static energy consumptions. The static energy consumption is calculated by multiplying the idle power of the platform (without application execution) with the execution time of the application. The dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumed by the platform during the application execution. That is, if P_S is the static power consumption of the platform, E_T is the total energy consumption of the platform during the execution of an application, which takes T_E seconds, then the dynamic energy E_D is equal to $E_D = E_T - (P_S \times T_E)$. We present the rationale behind using dynamic energy consumption instead of total energy consumption in section 2 of the supplemental [54]. In this work, we consider only the dynamic energy consumption.

The dynamic energy predictive models are built using specialized linear regression. The mathematical form of a model is shown below:

$$E_D = \beta_1 \times x_1 + \cdots + \beta_n \times x_n \quad (1)$$

where E_D is the dynamic energy consumption which is the dependent variable, $\{x_1, \ldots, x_n\}$ are the independent variables, and $\{\beta_1, \ldots, \beta_n\}$ are the regression coefficients or the model parameters. In real life, there usually is stochastic noise (measurement errors). Therefore, the measured energy is typically expressed as

$$\tilde{E_D} = E_D + \epsilon \quad (2)$$

where the error term or noise ϵ is a Gaussian random variable with expectation zero and variance σ^2, written $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

3. Related work

Our literature survey is organized as follows: (a) Survey of the mainstream methods for power and energy measurement, (b) Survey of power and energy predictive models based on utilization variables and PMCs. (c) Review of notable literature surveys on energy predictive models, and (d) Finally, recent advancements in this field.

3.1. Mainstream methods for energy measurements

There are three mainstream methods for energy measurement: (1) System-level power measurements using physical power meters, (2) On-chip power sensors, and (3) Energy predictive models. The first method is considered the ground truth. Fahad et al. [24] present the first methodology to measure the component-level energy consumption of a hybrid application on a heterogeneous computing platform using this method.

The second method is now supported by popular processor vendors who provide vendor-specific software libraries to acquire the power data from the on-chip sensors. Intel CPUs offer Running Average Power Limit (RAPL) [50] to monitor power and control frequency (and voltage). AMD starting from Bulldozer micro-architecture equip their processors with an estimation of average power over a certain interval through the Application Power Management (APM) [20] capability. Hackenberg et al. [30] report that APM provides highly inaccurate data, particularly during the processor sleep states. Intel Xeon Phi co-processors are equipped with on-board Intel System Management Controller chip (SMC) [17] providing energy consumption that can be programmatically obtained using Intel manycore platform software stack (Intel MPSS) [16]. The accuracy of Intel MPSS is not available. Nvidia Management Library *NVML* [4] provides programmatic interfaces to obtain the energy consumption of an Nvidia GPU from its on-chip power sensors. There are, however, some issues with the energy measurements provided by Nvidia on-chip sensors [13]. Fahad et al. [23] present the first detailed study on the accuracy of on-chip power sensors and show that deviations of the energy measurements provided by on-chip sensors including Intel RAPL from the ground truth does not motivate their use in the optimization of applications for dynamic energy.

The third method using energy predictive models emerged as a popular alternative to determine the energy consumption of an application. Performance monitoring counters (PMCs) and resource utilizations have been the principal source of model variables primarily due to their high positive correlation with energy consumption. PMCs, however, have come to dominate the landscape due to their better prediction accuracy compared to utilization variables.

3.2. Power and energy predictive models

3.2.1. Utilization based models

The early power models using the resource utilization parameters (such as CPU, memory, network, and I/O utilization statistics) as predictor variables include [18,22,25,26,32,35,36,40,48,49,60].

The general utilization based model for total power consumption can be described as follows:

$$P = c_{base} + c_1 \times U_{CPU} + c_2 \times U_{Mem} + c_3 \times U_{Disk} + c_4 \times U_{Net} \quad (3)$$

where c_{base} is the base power consumption of a processor and $\{c_1, c_2, c_3, c_4\}$ are the regression coefficients or the model parameters. U_{CPU}, U_{Mem}, U_{Disk}, and U_{Net} are the CPU, memory, disk, and network utilizations respectively.

3.2.2. PMC based models

Tools to Determine PMCs: PAPI [2] provides a standard API for accessing PMCs available on most modern microprocessors. It provides two types of events, *native events* and *present events*. *Native events* correspond to PMCs native to a platform. Likwid [59] provides command-line tools and an API to obtain PMCs for both Intel, POWER8, and AMD processors on the Linux OS. For Nvidia GPUs, CUDA Profiling Tools Interface (*CUPTI*) [3] can be used for obtaining the PMCs for CUDA applications. *Intel PCM* [1] is used for reading PMCs of core and uncore (which includes the QPI) components of an Intel processor. It supports the statistical analysis of core frequencies, QPI power, and DRAM activities. *Linux Perf* [61] also called *perf_events* can be used to gather the PMCs for CPUs in Linux. It also comes as a profiling tool suite including *perf stat*, *perf record*, *perf report*, *perf annotate*, *perf top* and *perf bench*.

PMC Based Power and Energy Predictive Model for CPUs and Accelerators: A vast majority of PMC based models are linear. Some works using PMCs as model variables in linear power and energy predictive models include [28,29,31,38,41,56,57,62,63], and [15]. Few notable research works that have proposed energy predictive models for accelerators such as GPUs, Xeon Phis, and FPGAs include [33,44,55,57], and [6]. Research works for power and energy modeling of scientific applications include [12,21,29,34,62,63], and [14].

From the literature, we can divide the approaches to select the PMCs into the following categories:

1. Approaches that consider all the PMCs to capture all possible contributors to energy consumption. To the best of our knowledge, we found no research works that adopt this approach. This could be due to several reasons:
   - Gathering all PMCs requires huge programming effort and time.
   - Interpretation (for example, visual) of the relationship between energy consumption and PMCs is difficult especially when there is a large number of PMCs.
   - Dynamic or runtime models must choose PMCs that can be gathered in just one application run.
   - Typically, simple models (those with fewer parameters) are preferred over complex models not because they are accurate but because simplicity is considered a desirable virtue.
2. Approaches that are based on a statistical methodology such a correlation and principal component analysis for the selection of PMCs.
3. Approaches that use expert advice or intuition to pick a subset (that may not necessarily be determined in one application run) and that, in experts' opinion, is a dominant contributor to energy consumption.
4. Approaches that select PMCs using a theoretical model of the energy of computing, which is the manifestation of the fundamental physical law of energy conservation [52,53].

Critiques of PMCs for Energy Predictive Modeling: There are research works that have critically examined and highlighted the poor prediction accuracy of PMCs for energy predictive modeling. Economou et al. [22] highlight the fundamental limitation, which is the inability to obtain all the PMCs simultaneously or in one application run. They also mention the lack of PMCs to model the energy consumption of disk I/O and network I/O. McCullough et al. [42] evaluate the competence of predictive power models for modern node architectures and show that linear regression models show prediction errors as high as 150%. They suggest that direct physical measurement of power consumption should be the preferred approach to tackle the inherent complexities posed by modern node architectures. O'Brien et al. [45] survey predictive power and energy models focusing on the highly heterogeneous and hierarchical node architecture in modern HPC computing platforms. They report that the prediction errors of linear PMC-based energy predictive models as high as 60%.

3.2.3. Models employing utilization variables and PMCs

Economou et al. [22] propose a linear power predictive model that employs CPU, disk, and network utilizations and a PMC containing the off-chip memory access count. Rivoire et al. [49] compare five full-system real-time power models. Four of these models are utilization-based whereas the fifth includes the model proposed by [22]. They report that the PMC-based model is the best overall in terms of accuracy since it is able to account for majority of the contributors to system's dynamic power.

Khokhriakhov et al. [37] propose a qualitative linear dynamic energy model employing CPU utilization and PMCs to explain the discovered energy nonproportionality on their multicore CPU platforms. The model is shown below:

$$E_{dynamic} = \beta_1 \times u_{cpu} \times t + \beta_2 \times p_l + \beta_3 \times p_s \quad (4)$$

where u_{cpu} is the CPU utilization, p_l is the PMC containing the time of page walk caused by load miss, p_s is the PMC for the time of page walk caused by store miss in dTLB, and t is the execution time of the application. The model is used to show that the energy nonproportionality is due to the activity of the data translation lookaside buffer (dTLB), which is disproportionately energy expensive.

3.2.4. Important surveys on energy predictive models

Mobius et al. [43] present a survey of power consumption models for single-core and multicore processors, virtual machines, and servers. They conclude that linear regression-based approaches dominate and that one prominent shortcoming of these models is that they use static instead of variable workloads for training the models. Dayarathna et al. [19] present an in-depth survey on data center power modeling. Bridges et al. [11] present a survey of techniques to monitor and model the energy consumption of GPUs. They cover in-depth PMC-based modeling of GPUs. They also state that the accuracy of results from internal power meters must be thoroughly verified using external power meters. O'Brien et al. [45] survey predictive power and energy models focusing on the highly heterogeneous and hierarchical node architecture in modern HPC computing platforms.

3.2.5. Recent advancements in the energy predictive models employing PMCs

In all aforementioned works, a sound theoretical framework to understand the fundamental significance of the model variables with respect to the energy consumption and the causes of inaccuracy or the reported wide variance of accuracy of the models has been lacking.

The theory of energy of computing has progressively matured over the past three years starting with a proposal of a criterion for selection of PMCs in the research work [52] followed by a formal description of the theory and its practical implications

in [53]. Shahid et al. [52] propose a novel property of PMCs called *additivity*, which is based on an intuitive and simple rule that if a PMC is intended to be employed as a model variable in a linear energy predictive model, then its count for a compound application should be equal to the sum of its counts for the executions of the base applications forming the compound application. A compound application is defined as the serial execution of two applications. It is based on the experimental observation that the dynamic energy consumption of a serial run of two applications is the sum of dynamic energy consumption observed for the sole executions of each application. The authors study the additivity of PMCs provided by the two mainstream frameworks, Likwid [59] and PAPI [2] on a modern Intel Haswell multicore server. They demonstrate that many PMCs available on modern processors obtained using Likwid and PAPI and that are employed in state-of-the-art models are non-additive.

Shahid et al. [53] proposed a novel theory of energy of computing and unified its practical implications to increase the prediction accuracy of linear energy predictive models in a *consistency* test, which contains a suite of properties that include determinism, reproducibility, and additivity to select model variables and constraints for model coefficients. By applying the consistency test, the authors improve the average prediction accuracy of state-of-the-art linear regression models from 31% to 18%. Shahid et al. [51] demonstrate that the accuracy of energy predictive models based on three popular mainstream techniques (linear regression, random forests, and neural networks) can be improved by following the properties of the consistency test, which includes selecting PMCs based on the property of additivity. They show that the removal of *non-additive* PMCs improves the average prediction accuracy of linear regression models from 31% to 18%, random forest models from 38% to 24%, and neural network models from 30% to 24%.

4. Theory of energy of computing: Practical implications for linear energy predictive models

In this section, we present a brief overview of the theory of energy of computing proposed in [53]. The theory of energy of computing is a formalism containing properties of PMC-based energy predictive models that are manifestations of the fundamental physical law of energy conservation. The properties capture the essence of single application runs and characterize the behavior of serial execution of two applications. They are intuitive and experimentally validated and are formulated based on the following observations:

- In a fully dedicated and stable environment, with each execution of a single application being represented by the same PMC vector, for any two applications, the PMC vector of their serial execution will always be the same.
- An application run that does not perform any work does not consume or generate energy. It is represented by a null PMC vector (where all the PMC values are zeros).
- An application with a PMC vector that is not null must consume some energy. Since PMCs account for energy-consuming activities of applications, an application with any energy-consuming activity higher than zero activity must consume more energy than zero.
- Finally, the consumed energy of compound application is always equal to the sum of energies consumed by the individual applications. The serial execution of two applications, say the base applications, forms a compound application.

The practical implications of the theory for constructing accurate and reliable linear energy predictive models are unified in a *consistency* test. The test includes the following selection criteria for model variables, model intercept, and model coefficients:

- Each model variable must be deterministic and reproducible. In the case of PMC-based energy predictive models, the multiple runs of an application keeping the operating environment constant must return the same PMC count.
- Each model variable must be additive. The property of additivity is further summarized in the following section.
- The model intercept must be zero.
- Each model coefficient must be positive.

The first two properties are combined into an *additivity* test for the selection of PMCs. A linear energy predictive model employing PMCs and which violates the properties of the consistency test will have poor prediction accuracy. By definition and intuition, PMCs are all pure counters of energy-consuming activities in modern processor architectures and as such must be additive. Therefore, according to the theory of energy of computing, any consistent, and hence accurate, energy model, which only employs PMCs, must be linear. This also means that any non-linear energy model employing PMCs only, will be inconsistent and hence inherently inaccurate. A non-linear energy model, in order to be accurate, must employ non-additive model variables in addition to PMCs.

While the theory is proposed with PMC-based energy predictive models as focal point, it is applicable for any model variables that are pure counters of energy-consuming activities. In this work, we show how model variables based on resource utilizations can be designed to leverage the theory.

4.1. Linear energy predictive models employing utilization variables and PMCs

We will now look at the mathematical expressions for the linear energy predictive models employing utilization variables and PMCs. The model parameters (or the regression coefficients) are constrained to be positive to meet the requirements of the consistency test.

For the dynamic energy predictive models that employ PMCs, the mathematical form is shown below:

$$E_{pmc} = \alpha_1 \times p_1 + \cdots + \alpha_n \times p_n \tag{5}$$

where E_{pmc} is the dynamic energy consumption, $\{p_1, \ldots, p_n\}$ are the PMCs, and $\{\alpha_1, \ldots, \alpha_n\}$ are the regression coefficients or the model parameters. We ignore the stochastic noise term.

We consider models employing the utilizations of CPU and memory as model variables. Since utilizations are high-level variables and not pure counters, we have to design model variables that are based on utilizations and that represent energy consumption. We provide expressions for model variables corresponding to CPU and memory only, since they are the most stressed by the applications used in our experimental study and therefore are the dominant consumers of dynamic energy. Similar model variable expressions, however, can be derived for disk and network utilizations for platforms where these two make non-trivial contributions to dynamic energy consumption.

CPU utilization during a time period represents the proportion of time the CPU is busy doing work divided by the total amount of time in the time period. The product of CPU utilization and the maximum power (thermal design power) gives an estimate of the power consumption of the CPU, that is, the average power consumption during the time period. The model variable, which is this product multiplied by the execution time of the application, represents the energy consumption. The CPU utilization model variable, u_{cpu}, therefore is determined using the equation below:

$$u_{cpu} = (\tilde{U}_{cpu}/100) \times TDP_{cpu} \times t \tag{6}$$

Similarly, the memory utilization model variable, u_{mem}, is determined using the equation below:

$$u_{mem} = (\tilde{U}_{mem}/100) \times TDP_{mem} \times t \quad (7)$$

$\tilde{U}_{cpu}$ and $\tilde{U}_{mem}$ are the average CPU and memory utilizations, TDP_{cpu} and TDP_{mem} are the thermal design powers of CPU and memory, and t is the execution time of the application.

The mathematical form of the dynamic energy predictive models employing the utilization variables is shown below:

$$E_u = \beta_1 \times u_{cpu} + \beta_2 \times u_{mem} \quad (8)$$

where E_u is the dynamic energy consumption, $\{u_{cpu}, u_{mem}\}$ are the utilization variables, and $\{\beta_1, \beta_2\}$ are the regression coefficients or the model parameters.

The mathematical form of the dynamic energy predictive models employing both utilization variables and PMCs is shown below:

$$E_{upmc} = \gamma_1 \times u_{cpu} + \gamma_2 \times u_{mem} + \theta_1 \times p_1 + \cdots + \theta_n \times p_n \quad (9)$$

where E_{upmc} is the dynamic energy consumption and $\{\gamma_1, \gamma_2, \theta_1, \ldots, \theta_n\}$ are the regression coefficients or the model parameters.

4.2. Additivity of model variables

The property of *additivity* (first proposed in [52]) is based on an intuitive and simple rule that if a model variable is employed in a linear energy predictive model, its count for a *compound* application should be equal to the sum of its counts for the executions of the base applications forming the compound application.

The additivity of a model variable is determined as follows. We first obtain the counts of the model variable for the separate executions of the base applications. Then, we run the *compound* application and record the count for the model variable. Typically, the main computations for the compound application consist of the main computations of the base applications executed one after the other. If the model variable of the *compound* application is equal to the sum of the model variables obtained for the base applications (within a tolerance of 5%), the model variable is categorized as potentially *additive*. Otherwise, it is labeled as *non-additive*.

For each model variable, we determine the maximum percentage error. For a *compound* application, the percentage error is calculated as follows:

$$Error(\%) = |\frac{(e_{b1} + e_{b2}) - e_c}{(e_{b1} + e_{b2} + e_c)/2}| \times 100 \quad (10)$$

where e_c, e_{b1}, e_{b2} are the model variable values for the compound application and the constituent base applications respectively. The additivity test error for a model variable is the maximum of percentage errors for all the *compound* applications in the experimental test-suite.

The most additive model variables are employed in a model for better prediction accuracy. We will go into the details of this selection process in our experiments section.

5. Experimental setup

We start with selection of model variables using the consistency test followed by comparison of prediction accuracy of application-specific and platform-level linear energy predictive models.

Our experimental platforms include the Intel Haswell and Intel Skylake multicore CPU servers, whose specifications are given in Table 1. Our application test suite is composed of highly optimized scientific applications such as DGEMM and FFT from the Intel math kernel library (MKL), NASA benchmarking suite (NAS Parallel), Intel optimized HPCG, and *stress*. Table 2 lists the applications along with their description.

Table 2
List of applications for experimental analysis.

Application	Description
MKL FFT	Intel optimized 2-dimensional Fast Fourier Transform
HPCG	Intel optimized High Performance Conjugate Gradient.
MKL DGEMM	Intel optimized 3-dimensional Dense Matrix Multiplication
NPB FT	Discrete 3-dimensional fast Fourier Transform
NPB CG	Conjugate Gradient
NPB SP	Scalar Penta-diagonal solver
NPB DT	Data traffic
NPB EP	Embarrassingly Parallel random number generator
NPB BT	Solve synthetic system of nonlinear partial differential equations using Block Tri-diagonal solver
NPB IS	Integer Sort, Kernel for random memory access that sort small integers using the bucket sort technique
NPB LU	Lower–Upper Gauss–Seidel solver
NPB DC	Data Cube
NPB UA	Unstructured Adaptive mesh solving heat equation with convection and diffusion from moving ball.
NPB MG	Approximate 3-dimensional discrete Poisson equation using the V-cycle Multi Grid on a sequence of meshes
stress	CPU, disk and I/O stress
Naive MM	Naive Matrix–matrix multiplication
Naive MV	Naive Matrix–vector multiplication

For each application run, we measure the following: (a) Dynamic energy consumption, (b) Execution time, (c) PMCs, and (d) Utilization variables. The dynamic energy consumption during the application execution is measured using a *WattsUp Pro* power meter and is obtained programmatically via the HCLWattsUp API [5] (section 4 of supplemental [54]). The power meter is periodically calibrated using an ANSI C12.20 revenue-grade power meter, Yokogawa WT210. The calibration methodology is explained in section 6 of the supplemental [54]. PMCs are obtained using the Likwid tool [59] and Linux Perf [61].

To ensure the reliability of our results, we follow a statistical methodology where a sample mean for a response variable (energy, time, PMC, utilization variables) is obtained from multiple experimental runs. The sample mean is calculated by executing the application repeatedly until it lies in the 95% confidence interval and a precision of 0.05 (5%) has been achieved. For this purpose, Student's t-test is used assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions by plotting the distributions of observations. The experimental methodology to determine the sample mean is described in section 3 of the supplemental [54]. The prediction error of a model is calculated as follows: $error = |(E_M - E_G)|/E_G \times 100$, where E_M is the prediction by a model and E_G is the ground truth value. The average prediction error for n data points is calculated as $(\sum_{i=1}^{n} error_i)/n$, where $error_i$ is the prediction error for data point i.

We now apply the consistency test to select PMCs and utilization parameters.

5.1. Measurement and selection of Performance Monitoring Counters (PMCs)

Likwid tool [59] offers 164 and 323 PMCs on HCLServer1 and HCLServer2, respectively. To collect all the PMCs, each application must be executed about 53 and 99 times on HCLServer1 and HCLServer2, respectively. This is due to the availability of a limited number of hardware registers (3–4) to store the PMCs. We apply the first stage of consistency test where we check if the PMCs are deterministic and reproducible as follows:

1. We eliminate PMCs with counts less than or equal to 10. The eliminated PMCs have no significance on modeling energy consumption of our platform because we found them to be non-reproducible. We also remove several PMCs that count equal to zero. The reduced set contains 151 and 298 PMCs on HCLServer1 and HCLServer2, respectively.
2. We compare the PMCs using three different tools, Likwid, PAPI, and Linux Perf. We eliminate the PMCs that show differences. After this elimination, the total number of PMCs reduces to 115 and 224 on HCLServer1 and HCLServer2.

The total work performed during the execution of an application in our test suite is entirely due to CPU and main memory activities. To find their contributions towards the dynamic energy, we use two synthetic applications, A and B, performing floating-point operations and *memcpy()* operations on all the memory blocks, respectively. We execute A employing all processor cores for 10 s and measure its dynamic energy consumption, which is equal to 1337 joules. We then execute B for the same amount of time and discover that the dynamic energy consumption is insignificant and cannot be captured within the statistical confidence of 95%. We increase the execution time of A and B to 20 and 30 s and find their dynamic energy consumptions to be 2596 joules and 3821 joules, and, 0 joules and 4 joules, respectively. We conclude that the major contribution to dynamic energy consumption is due to CPU activities. Therefore, we remove the PMCs that belong to Likwid main memory group for any further analysis due to two reasons. First, the memory activities make negligible contribution to the dynamic energy consumption on our platforms. Second, low counts for memory PMCs add noise that affects the training of models and unduly worsen the prediction accuracy of the models. The main CPU activities can be grouped as PMCs belonging to cache, branch instructions, micro-operations (uops), floating-point instructions, instruction decode queue, and cycles. We denote them as *prime PMCs*.

The second stage of the consistency test involves application of the additivity property. We automate the determination of a PMC's additivity using a tool called *AdditivityChecker* (section 9 of the supplemental [54]). We discover that all the prime PMCs fail the *additivity* test for a vast set of applications with a specified tolerance of 5% on our platforms.

5.2. Measurement and selection of utilization variables

We follow the steps below to determine the utilization variables (u_{cpu}, u_{mem}) on HCLServers:

- Using an automated shell script, we collect the average CPU and memory utilization in percentage for the platform using Linux *ps* tool. The script reads the CPU and memory utilization every 0.25 s during the application execution.
- The CPU utilization for an application is the average utilization of the individual cores employed in the execution of that application.
- For an application, the trapezoidal rule is used to determine the average utilization using the utilization profile for the application.
- Finally, the average CPU and memory utilizations are multiplied with the corresponding TDPs and the execution time of the application.

We now apply the consistency test to each utilization variable on HCLServer1 and HCLServer2. We found both variables deterministic and reproducible by executing applications (Table 2) using different problem sizes on HCLServer1 and HCLServer2.

To study the additivity of the utilization variables, we take the same application set and compose 60 and 40 compound applications from the base applications on both the servers. The additivity test reveals that both variables are highly additive (with errors less than 5%) for all the applications. Therefore, we conclude that both utilization variables can be employed as model variables in any application-specific and platform-level linear energy predictive model.

Since the contribution of memory activities towards dynamic energy consumption of the applications is insignificant on our platforms, we analyze the impact of u_{mem} as a model variable. With no constraints on the model coefficients, we find that the model coefficient of u_{mem} to be negative for all the models constructed in our experiments with the exception of the application-specific model for FFT. For the models with the negative coefficient, we remove the memory utilization variable since it violates the properties of the consistency test and re-construct the models. We also find that the removal of memory utilization variable from the model for FFT reduces the prediction power of the model by $0.02\times$ only. Therefore, for the consistency of the experimental results, we remove u_{mem} as a model variable from our energy predictive models.

6. Experimental results

We divide the experiments into the following two groups:

1. **Group 1:** We study the accuracy of application-specific energy predictive models on HCLServer1 and HCLServer2 using utilization variables only, prime PMCs only, and both utilization variables and PMCs.
2. **Group 2:** We study the accuracy of platform-level energy predictive models on HCLServer1 and HCLServer2 using utilization variables only, prime PMCs only, and both utilization variables and PMCs. We divide the experiments in this section into two classes, class A and class B. In class A, we explore the prediction accuracies of models for a limited set of applications (DGEMM and FFT). In class B, we analyze models that employ data-sets composed using all the applications in our testsuite for a wide range of problem sizes.

6.1. Study of accuracy of application-specific energy predictive models

We select two highly optimized scientific applications: 2-dimensional Fast Fourier Transform (FFT) and Dense Matrix-Multiplication application (DGEMM), from Intel Math Kernel Library (MKL). The experimental steps are below:

- We build two data-sets to study the additivity of PMCs for FFT and DGEMM containing the compound and base applications. Using the additivity test errors, we select the most additive PMCs that are common for both applications.
- By executing FFT and DGEMM for a range of problem sizes, we build a vast data-set containing dynamic energy consumptions, PMCs, and both utilization variables and PMCs to build energy predictive models.
- We employ the utilization variables only, PMCs only, and both utilization variables and PMCs in LR models as predictor variables.
- Finally, we analyze the prediction accuracy of the LR models.

Table 3
Selected PMCs and their correlations with dynamic energy consumption on, (a) HCLServer1 and (b) HCLServer2. 0 to 1 represents correlations of 0% to 100%, respectively.

(a)

	HCLServer1 PMCs (SA)	Correlation
$a1$	IDQ_MITE_UOPS	0.993
$a2$	UOPS_EXECUTED_PORT_PORT_6	0.992
$a3$	L2_RQSTS_MISS	0.990
$a4$	UOPS_ISSUED_TOTAL_CYCLES	0.962
$a5$	UOPS_EXECUTED_PORT_PORT_0	0.932
$a6$	OFFCORE_REQUESTS_ALL_DATA_RD	0.921
$a7$	UOPS_RETIRED_CORE_TOTAL_CYCLES	0.917
$a8$	CPU_CLOCK_UNHALTED_REF_XCLK	0.801

(b)

	HCLServer2 PMCs (SB)	Correlation
$b1$	FP_ARITH_INST_RETIRED_DOUBLE	0.993
$b2$	UOPS_EXECUTED_CORE	0.993
$b3$	IDQ_ALL_CYCLES_6_UOPS	0.993
$b4$	UOPS_RETIRED_CYCLES_GE_4_UOPS_EXEC	0.992
$b5$	IDQ_DSB_CYCLES_6_UOPS	0.981
$b6$	IDQ_ALL_DSB_CYCLES_5_UOPS	0.972
$b7$	UOPS_DISPATCHED_PORT_PORT_4	0.870
$b8$	BR_INST_RETIRED_ALL_BRANCHES	0.860

6.1.1. Experiments to select PMCs and utilization variables

We present the methodology to select the utilization variables and PMCs to be employed in the models in the following steps:

- We build a dataset of 50 base applications using different problem sizes for DGEMM and FFT to apply the additivity test. The range of problem sizes for DGEMM is 6500×6500 to $20\,000 \times 20\,000$, and for FFT is $22\,400 \times 22\,400$ to $29\,000 \times 29\,000$. We select this range because of reasonable execution time (>3 s) of the applications on our platforms.
- For each application in a dataset, we measure the following: PMCs, utilization variables, dynamic energy consumption, and the execution time. We also build a dataset of 30 *compound* applications from the serial execution of base applications. The additivity test based on the two datasets reveals that several PMCs are highly additive and are common for both applications. The utilization variables for both applications are highly additive and highly positively correlated with energy with errors less than 0.5%.
- From the additivity test results on HCLServer1 and HCLServer2, we select PMCs that are commonly additive with additivity test errors of less than 1%. In total there are eight PMCs for both servers with an error equal or less than 1% represented as set $SA = \{a1, a2, a3, a4, a5, a6, a7, a8\}$ and set $SB = \{b1, b2, b3, b4, b5, b6, b7, b8\}$ for HCLServer1 and HCLServer2, respectively. All these PMCs belong to the dominant PMC groups from the CPU that represent the energy consuming activities of our platform.
- We calculate the correlation for all PMCs in *SA* and *SB* with the dynamic energy consumption. The PMCs and their correlations with dynamic energy consumption are given in Table 3.
- We also build two subsets with four most energy correlated PMCs from SA and SB and label them as *SA-Corr* and *SB-Corr*. SA-Corr and SB-Corr are $\{a1, a2, a3, a4\}$ and $\{b1, b2, b3, b4\}$, respectively.

6.1.2. Energy predictive models for DGEMM and FFT

We build a dataset containing dynamic energy consumption, execution time, PMCs (Table 3) and utilization variables for MKL-DGEMM and MKL-FFT for a range of problem sizes on our platforms (Table 1). The number of data points in the data-set and range of problem sizes are given in Table 4. The dataset is split into two subsets for training and testing the models.

We build models for MKL-FFT and MKL-DGEMM using LR by employing the predictor variables from the training sets given in Table 4 for HCLServer1 and HCLServer2. The models are evaluated using the test dataset and are divided into two categories (S1 and S2) as given below:

S1: Linear energy predictive models for FFT and DGEMM executing on HCLServer1.

- S1-UPT-FFT and S1-UPT-DGEMM employ u_{cpu} as model variable.
- S1-PMC-FFT and S1-PMC-DGEMM have $\{a1, a2, a3, a4, a5, a6, a7, a8\}$ as model variables.
- S1-PMC-Corr-FFT and S1-PMC-Corr-DGEMM employ the top four high positively correlated PMCs, $\{a1, a2, a3, a4\}$.
- S1-UPMC-FFT and S1-UPMC-DGEMM employ $\{u_{cpu}, a1, a2, a3, a4\}$.

S2: Linear energy predictive models for FFT and DGEMM executing on HCLServer2.

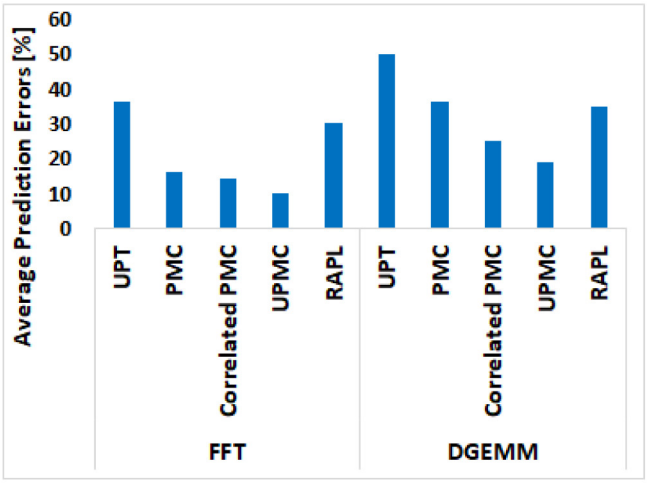
- S2-UPT-FFT and S2-UPT-DGEMM employ u_{cpu} as model variable.
- S2-PMC-FFT and S2-PMC-DGEMM have $\{b1, b2, b3, b4, b5, b6, b7, b8\}$ as model variables.
- S2-PMC-Corr-FFT and S2-PMC-Corr-DGEMM employ the top four high positively correlated PMCs, $\{b1, b2, b3, b4\}$.
- S2-UPMC-FFT and S2-UPMC-DGEMM employ $\{u_{cpu}, b1, b2, b3, b4\}$.

Tables 5 and 6 show the minimum, average, and maximum prediction errors for the models in category S1 and S2, respectively. Fig. 2 compares the average prediction accuracies of models and Intel Running Average Power Limit (RAPL) [50] in both categories. On both our servers, RAPL is an on-chip power sensor that employs voltage regulator current monitor (VR IMON) for both CPU and DRAM. VR IMON is an analog circuit within a voltage regulator (VR), which keeps track of an estimate of the power as the VRs supply current to the CPU. RAPL samples this reading periodically (100 μs to 1 ms).

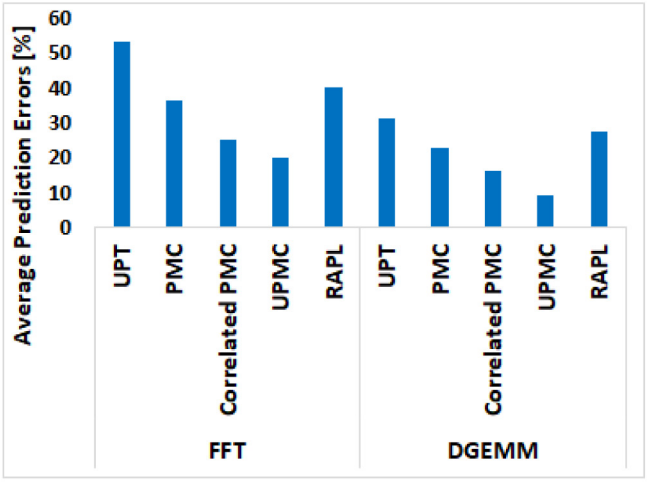
S1-UPMC-FFT and S1-UPMC-DGEMM yield minimum average prediction errors of 9.2% and 11% for models in category A, respectively. Similarly, S2-UPMC-FFT and S2-UPMC-DGEMM have minimum average prediction errors of 19% and 9.4%, respectively. Category S1 models show that DGEMM and FFT models based on

Table 4
Data-set for application-specific models on HCLServer1 and HCLServer2.

Application	Range of problem sizes	Step size	Total data points	Training set	Testing set
HCLServer1					
DGEMM	$12\,000 \times 12\,000$ to $24\,736 \times 24\,736$	64	200	150	50
FFT	$40\,000 \times 40\,000$ to $44\,992 \times 44\,992$	64	79	59	20
HCLServer2					
DGEMM	6400×6400 to $38\,400 \times 38\,400$	64	401	300	101
FFT	$22\,400 \times 22\,400$ to $41\,536 \times 41\,536$	64	300	225	75



(a) S1 Models



(b) S2 Models

Fig. 2. Comparison of average prediction errors for application-specific energy predictive models and Intel RAPL for the applications, FFT and DGEMM.

both utilization variables and PMCs perform (3.4×, 1.83×, 3.2×) and (3.8×, 1.7×, 3.3×) better in terms of average prediction accuracies when compared with models employing utilization variables only, PMCs only, and Intel RAPL, respectively. Similarly, Category S2 models show that DGEMM and FFT models based on both utilization variables and PMCs perform (3.3×, 2.4×, 2.9×) and (2.6×, 1.8×, 1.4×) better in terms of average prediction accuracies when compared with models employing utilization variables only, PMCs only, and Intel RAPL, respectively.

6.1.3. Discussion

Following are the salient observations from the results:

- The models employing only utilization variables have poor prediction accuracy for all model categories.
- Intel RAPL has better average prediction accuracy than the utilization models.
- The average prediction accuracy for models employing additive PMCs (in the sets, SA and SB) is better than models using only utilization variables and Intel RAPL. The accuracy further improves for the models, which employ the top four most positively correlated PMCs (in the sets, SA-Corr and SB-Corr) along with the additive PMCs.

Table 5
Prediction accuracies for application-specific models and Intel RAPL in Category S1.

Model	Model variables	Prediction errors (%) [min, avg, max]
S1-UPT-FFT	u_{cpu}	(2.92, 36.32, 92.15)
S1-PMC-FFT	$\{a1, a2, a3, a4, a5, a6, a7, a8\}$	(1.71, 16.22, 44.91)
S1-PMC-Corr-FFT	$\{a1, a2, a3, a4\}$	(2.15, 14.23, 42.15)
S1-UPMC-FFT	$\{u_{cpu}, a1, a2, a3, a4\}$	(2.04, 10.41, 34.52)
S1-RAPL-FFT		(0.35, 30.51, 155.03)
S1-UPT-DGEMM	u_{cpu}	(2.21, 37.71, 53.23)
S1-PMC-DGEMM	$\{a1, a2, a3, a4, a5, a6, a7, a8\}$	(1.21, 20.11, 89.01)
S1A-PMC-Corr-DGEMM	$\{a1, a2, a3, a4\}$	(0.01, 15.21, 82.27)
S1-UPMC-DGEMM	$\{u_{cpu}, a1, a2, a3, a4\}$	(0.18, 10.98, 51.77)
S1-RAPL-DGEMM		(0.38, 35.23, 160.7)

Table 6
Prediction accuracies for application-specific models and Intel RAPL in Category S2.

Model	Model variables	Prediction errors (%) [min, avg, max]
S2-UPT-FFT	u_{cpu}	(1.53, 53.21, 89.84)
S2-PMC-FFT	$\{b1, b2, b3, b4, b5, b6, b7, b8\}$	(0.447, 36.31, 182.2)
S2-PMC-Corr-FFT	$\{b1, b2, b3, b4\}$	(0.042, 25.12, 190.15)
S2-UPMC-FFT	$\{u_{cpu}, b1, b2, b3, b4\}$	(1.23, 19.98, 122.9)
S2-RAPL-FFT		(0.51, 40.21, 173.25)
S2-UPT-DGEMM	u_{cpu}	(2.12, 31.33, 53.02)
S2-PMC-DGEMM	$\{b1, b2, b3, b4, b5, b6, b7, b8\}$	(0.094, 22.62, 125.48)
S2-PMC-Corr-DGEMM	$\{b1, b2, b3, b4\}$	(0.004, 16.12, 87.25)
S2-UPMC-DGEMM	$\{u_{cpu}, b1, b2, b3, b4\}$	(0.45, 9.40, 63.72)
S2-RAPL-DGEMM		(0.29, 27.41, 142.72)

- The most accurate models for DGEMM and FFT applications employ five and six PMCs. Therefore, at least six hardware registers must be dedicated to storing the PMCs so that these models can be used for online. Currently, only 3–4 hardware registers are dedicated to storing PMCs during an application run on our experimental platforms.
- Fig. 2 shows that the patterns for prediction accuracy for the two applications are different for the two platforms. In HCLServer1, FFT models have better prediction accuracy than DGEMM models. It is, however, the reverse on HCLServer2. The best set of PMCs employed as model variables for the applications is different for the two platforms. This illustrates that the same set of model variables may not represent the energy-consuming activities of an application on all platforms, even if they share the same set of PMCs. The differences in the employed model variables translate into differences in average prediction accuracies for dynamic energy consumption for the same application executing on different platforms. Therefore, we conclude that the prediction accuracy of PMC based models is not just sensitive to an application but also to the platform.
- The best prediction accuracy is achieved for models that employ both utilization variables and PMCs. This is because they capture most of the energy-consuming activities during an application execution on our platforms.

6.2. Study of accuracy of platform-level energy predictive models

In this section, we study the accuracy of platform-level energy predictive models using the testsuite (Table 2). We divide our experiments into two classes:

1. Class A: Comparison of prediction accuracy of models employing utilization variables, PMCs, and both utilization variables and PMCs for two applications, DGEMM and FFT.

Table 7
Prediction accuracies for energy predictive models and Intel RAPL in the sets, S1-MMFT and S2-MMFT.

Model	Model variables	Prediction errors (%) [min, avg, max]
	S1-MMFT models	
S1-UPT-MMFT	u_{cpu}	(2.32, 22.39, 121.31)
S1-PMC-MMFT	{a1, a2, a3, a4, a5, a6, a7, a8}	(0.014, 18.62, 125.48)
S1-PMC-Corr-MMFT	{a1, a2, a3, a4}	(0.014, 16.12, 87.25)
S1-UPMC-MMFT	{u_{cpu}, a1, a2, a3, a4}	(0.25, 10.40, 63.72)
S1-RAPL-MMFT		(0.35, 32.52, 160.7)
	S2-MMFT models	
S2-UPT-MMFT	u_{cpu}	(1.23, 39.21, 132.23)
S2-PMC-MMFT	{b1, b2, b3, b4, b5, b6, b7, b8}	(0.005, 35.32, 225.5)
S2-PMC-Corr-MMFT	{b1, b2, b3, b4}	(0.024, 25.12, 87.25)
S2-UPMC-MMFT	{u_{cpu}, b1, b2, b3, b4}	(0.20, 17.27, 112.90)
S2-RAPL-MMFT		(0.29, 34.61, 173.25)

2. Class B: Comparison of prediction accuracy of models employing utilization variables, PMCs, and both utilization variables and PMCs for a dataset obtained using a diverse set of applications executing a wide range of problem sizes on HCLServer1 and HCLServer2.

6.2.1. Class A: Analysis of prediction accuracy of energy predictive models for DGEMM and FFT

The experiments in this class are run on HCLServer1 and HCLServer2 (Table 1). Since we choose commonly *additive* PMCs and utilization variables for MKL-DGEMM and MKL-FFT for application-specific models (Section 6.1), we combine the dataset for both applications. We build the following two categories of models using the extended dataset:

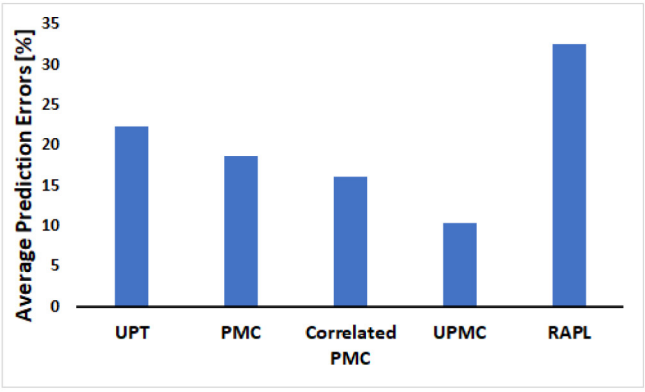
- *S1-MMFT*: S1-UPT-MMFT, S1-PMC-MMFT, S1-PMC-Corr-MMFT, and S1-UPMC-MMFT are LR-based models employing utilization variable only, PMCs in the set SA, PMCs in the set SA-Corr, and utilization variable and PMCs (u_{cpu}, and the set of PMCs, SA-Corr) on HCLServer1, respectively.
- *S2-MMFT*: S2-UPT-MMFT, S2-PMC-MMFT, S2-PMC-Corr-MMFT, and S2-UPMC-MMFT are LR-based models employing utilization variable only, PMCs in the set SB, PMCs in the set SB-Corr, and both utilization variables and PMCs (u_{cpu}, and the set of PMCs, SB-Corr) on HCLServer2, respectively.

The number of data points in the training and test data sets for the models on HCLServer1 and HCLServer2 are 153 and 66, and, 490 and 211, respectively. Table 7 shows the minimum, average, and maximum prediction errors for the models built in class A. Fig. 3 shows the comparison of average prediction accuracies for models in category S1-MMFT and S2-MMFT. The results for S1-MMFT and S2-MMFT to show that models employing both utilization variables and PMCs perform (2.1×, 1.8×, 3.3×) and (2.3×, 2×, 2×) better in terms of average prediction accuracies when compared with models employing only utilization variables only, PMCs only, and Intel RAPL, respectively.

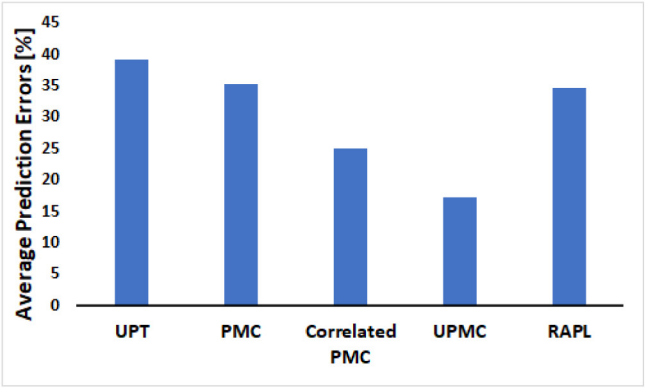
Discussion

Following are the salient observations from the results:

- Intel RAPL performs the worst in terms of average prediction accuracy.
- The average prediction accuracy improves for models employing both utilization variables and PMCs. S1-UPMC-MMFT and S2-UPMC-MMFT result in minimum average prediction errors of 10.4% and 17% for HCLServer1 and HCLServer2, respectively.



(a) S1-MMFT Models



(b) S2-MMFT Models

Fig. 3. Comparison of average prediction errors for energy predictive models with Intel RAPL for combined datasets of DGEMM and FFT on (a) HCLServer1, and (b) HCLServer2.

- The average prediction accuracy of the best performing models employing both utilization variables and PMCs for an application-specific model is better than the models constructed with the combined dataset for the two applications. As you increase the number of applications, the average prediction accuracy would approach the accuracy for that of platform-level models.

6.2.2. Class B: Analysis of prediction accuracy of energy predictive models for a broad set of applications

We choose HCLServer1 for the experiments in this section. We select six PMCs (x_1 to x_6 in Table 8), which are widely used in energy predictive models (Section 3) and belong to the dominant energy-consuming PMC groups. We build a dataset of 277 points as base applications by executing the applications from our test suite with different problem sizes. This dataset is used to train the models. We build a test dataset containing points for 50 compound applications which are composed of serial executions of base applications. Each point contains the dynamic energy consumption and PMCs for the execution of an application. We apply additivity test and find no PMC to be additive within tolerance of 5%. We list the PMCs and their additivity error percentages in Table 8.

We build six LR models, {LR1, LR2, LR3, LR4, LR5, LR6}. Each model contains a decreasing number of non-additive PMCs. Model LR1 employs all the selected PMCs as predictor variables. Model LR2 is based on five most additive PMCs. PMC x_4 is removed because it has the highest non-additivity. Model LR3 uses four most additive PMCs and so on until Model LR6 containing the top additive PMC, which is x_6.

Table 8
List of selected PMCs for modeling with their *additivity* test errors (%).

Selected PMCs	*Additivity* test error (%)
x_1: IDQ_MITE_UOPS	13
x_2: IDQ_MS_UOPS	37
x_3: ICACHE_64B_IFTAG_MISS	36
x_4: ARITH_DIVIDER_COUNT	80
x_5: L2_RQSTS_MISS	14
x_6: UOPS_EXECUTED_PORT_PORT_6	10

Table 9
Linear predictive models (LR1–LR6) using zero intercepts and positive coefficients with their minimum, average, and maximum prediction errors.

Model	PMCs	Prediction errors (%) [min, avg, max]
LR1	$x_1, x_2, x_3, x_4, x_5, x_6$	(6.6, 31.2, 61.9)
LR2	x_1, x_2, x_3, x_5, x_6	(6.6, 31.2, 61.9)
LR3	x_1, x_3, x_5, x_6	(2.5, 25.3, 62.1)
LR4	x_1, x_5, x_6	(2.5, 23.86, 100.3)
LR5	x_1, x_6	(2.5, 18.01, 89.45)
LR6	x_6	(2.5, 68.5, 90.5)

We compare the predictions of the models with system-level physical power measurements using HCLWattsUp, which we consider to be the ground truth. The minimum, average, and maximum prediction errors for the models are given in Table 9. It can be seen that LR5 employing two most additive PMCs yields the most accurate PMC based energy predictive model.

We then expand our dataset with 586 points on HCLServer1 using the applications in our testsuite (Table 2). The input parameters for the applications used to build the dataset are as follows:

- MKL FFT: Problem Size = 40 000 × 40 000 to 44 992×44 992 with step size of 64, verbosity = 0, Iteration = 1.
- MKL DGEMM: Problem Size = 12 000 × 12 000 to 24 736 × 24 736 with step size of 64, verbosity = 0, Iteration = 1.
- Intel HPCG: Problem Size = 40 × 40 × 40 to 240 × 240 × 240 with step size of 8, Iterations = 1, Threads = 48.
- NAS OMP 3D FT: Problem Size = 256 × 256 × 128, Iterations 95 to 395 with step size of 5, Threads 48.
- NAS OMP 3D LU: Problem Size = 70 × 70 × 70 to 139 × 139 × 139 with step size of 1, Iterations = 250, Threads = 48.
- NAS OMP 3D SP: Problem Size = 84 × 84 × 84 to 139 × 139 × 139 with step size of 1, Iterations = 100, dt = 0.0015000, Threads = 48.
- NAS OMP 3D BT: Problem Size = 128 × 128 × 128 to 190 × 190 × 190 with step size of 1, Iterations = 200, dt = 0.0008000, Threads = 48.
- stress: Problem Size = 4 s to 45 s with a step size of 1.

For each application configuration, we measure the dynamic energy consumption, execution time, PMCs, and utilization variables. 410 and 176 points have been used to train and test the models respectively.

We build three platform-level models PL-UPT, PL-PMC, and PL-UPMC, employing utilization variables only, PMCs only, and both utilization variables and PMCs on HCLServer1. Table 10 shows the prediction accuracies (also shown in Fig. 4 as bar charts). The results show that models employing both utilization variables and PMCs have 2.60×, 1.42×, and 1.96× better average prediction accuracies than models employing utilization variables only, PMCs only, and Intel RAPL, respectively.

Discussion

Following are the salient observations from the results:

- The minimum average prediction error of 14.36% is obtained for the model employing both utilization variables and the most additive PMCs.

Table 10
Prediction accuracies for platform-level energy predictive models and Intel RAPL.

Model	Predictor variables	Prediction errors (%) [min, avg, max]
PL-UPT	u_{cpu}	(0.11, 37.35, 140.05)
PL-PMC	x_1, x_6	(2.4, 20.41, 93.01)
PL-UPMC	u_{cpu}, x_1, x_6	(0.06, 14.36, 50.75)
PL-RAPL		(0.14, 36.62, 190.41)

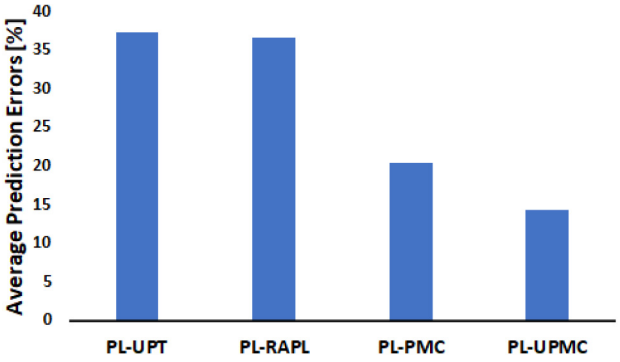


Fig. 4. Comparison of average prediction errors for platform-level models employing utilization variables only (UPT), PMCs only (PMC), both utilization variables and PMCs (UPMC), and Intel RAPL.

- While utilization variables capture the overall energy consumption trend of the application executions, they do not capture holistically and completely all the energy-consuming activities during the execution of an application. Models that employ both utilization variables and PMCs that are highly additive and highly energy-correlated are able to account for most of the energy-consuming activities during the execution of an application and hence are found to provide the best accuracy.

7. Discussion and future work

We now summarize the most important findings from our experiments:

- We analyzed the prediction accuracy of linear energy predictive models employing utilization variables only, PMCs only, and both utilization variables and PMCs. The utilization variables capture the overall energy consumption trend (or the average energy consumption) of the profiles of the application executions. But they fail to capture the variations in the profiles. Models based on highly additive and highly energy correlated PMCs accurately capture these variations. They do not, however, account for some energy-consuming activities that are captured by high-level utilization variables.
- The best models employing both utilization variables and PMCs exhibit 3.6× and 2.6× better average prediction accuracy than models employing utilization variables only and PMCs only. The average prediction accuracies of the application-specific models employing both utilization variables and PMCs for FFT and DGEMM are {10.41%, 10.98%}, and {19.98%, 9.40%} on HCLServer1 and HCLServer2 respectively. The average prediction accuracy of the platform-level model employing both utilization variables and PMCs is 14.36% on HCLServer1.
- The memory activities do not contribute towards dynamic energy consumption of the applications on our platforms. Therefore, we remove the PMCs related to memory activities from the models in our analysis. With no constraints on

the model coefficients, we find that the model coefficient of u_{mem} to be negative for all the models constructed in our experiments with the exception of the application-specific model for FFT. For the models with the negative coefficient, we remove the memory utilization variable since it violates the properties of the consistency test and re-construct the models. We also find that the removal of memory utilization variable from the model for FFT reduces the prediction power of the model by 0.02× only. Therefore, for the consistency of the experimental results, we remove u_{mem} as a model variable from our energy predictive models.

- We observe that the patterns for prediction accuracy for application-specific models for FFT and DGEMM are different for the two experimental platforms. In HCLServer1, FFT models have better prediction accuracy than DGEMM models. It is, however, the reverse on HCLServer2. The best set of PMCs employed as model variables for the applications is different for the two platforms. This illustrates that the same set of model variables may not represent the energy-consuming activities of an application on all platforms, even if they share the same set of PMCs. Therefore, we conclude that the prediction accuracy of PMC based models is not just sensitive to an application but also to the platform.
- The most accurate application-specific models for DGEMM and FFT applications employ five and six PMCs. Therefore, at least six hardware registers must be dedicated to storing the PMCs so that these models can be used for online. Currently, only 3–4 hardware registers are dedicated to storing PMCs during an application run on our experimental platforms.

In our future work, we will pursue two related lines of research. First, we will continue to find improvements to the prediction accuracy of linear energy predictive models by adding influential model variables using the theory of energy of computing. Second, we will analyze how energy predictive models employing both utilization variables and PMCs can be combined with system-level measurements using power meters for accurately and efficiently determining application component level decomposition of energy consumption and energy optimization of parallel applications on heterogeneous hybrid computing platforms.

8. Conclusion

In this work, we performed a comparative study of the prediction accuracy of models employing utilization variables only, PMCs only, and combining both utilization variables and PMCs through the lens of the theory of energy of computing on modern multicore CPU platforms. We discovered that employing utilization variables only in linear energy predictive models does not capture all the energy-consuming activities during application execution. However, combining utilization variables with PMCs that are highly additive and highly correlated with energy consumption gave the most accurate linear energy predictive model. Our experimental results showed that application-specific and platform-level models using both utilization variables and PMCs perform up to 3.6× and 2.6× better in terms of average prediction accuracy when compared with models employing utilization variables only and highly additive PMCs only, respectively.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 14/IA/2474.

References

[1] IntelPCM, Intel performance counter monitor - a better way to measure CPU utilization, 2012, URL https://software.intel.com/en-us/articles/intel-performance-counter-monitor.
[2] PAPI, Performance application programming interface 5.4.1, 2015, URL http://icl.cs.utk.edu/papi/.
[3] CUPTI, CUDA profiling tools interface, 2017, URL https://developer.nvidia.com/cuda-profiling-tools-interface.
[4] Nvidia, Nvidia management library: NVML reference manual, 2018, URL https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf.
[5] HCL, HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter, 2020, URL https://csgitlab.ucd.ie/manumachu/hclwattsup.
[6] Z. Al-Khatib, S. Abdi, Operand-value-based modeling of dynamic energy consumption of soft processors in FPGA, in: International Symposium on Applied Reconfigurable Computing, Springer, 2015, pp. 65–76.
[7] R. Basmadjian, H. de Meer, Evaluating and modeling power consumption of multi-core processors, in: Future Energy Systems: Where Energy, Computing and Communication Meet (E-Energy), 2012 Third International Conference on, 2012, pp. 1–10.
[8] F. Bellosa, The benefits of event: driven energy accounting in power-sensitive systems, in: Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System, ACM, 2000.
[9] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, E. Ayguade, Decomposable and responsive power models for multicore processors using performance counters, in: Proceedings of the 24th ACM International Conference on Supercomputing, ACM, 2010, pp. 147–158.
[10] W.L. Bircher, L.K. John, Complete system power estimation using processor performance events, IEEE Trans. Comput. 61 (4) (2012) 563–577.
[11] R.A. Bridges, N. Imam, T.M. Mintz, Understanding gpu power: A survey of profiling, modeling, and simulation methods, ACM Comput. Surv. 49 (3).
[12] V. Bui, B. Norris, K. Huck, L.C. McInnes, L. Li, O. Hernandez, B. and Chapman, A component infrastructure for performance and power modeling of parallel scientific applications, in: Proceedings of the 2008 CompFrame/HPC-GECO Workshop on Component Based High Performance, CBHPC '08, ACM, 2008, pp. 6:1–6:11.
[13] M. Burtscher, I. Zecena, Z. Zong, Measuring gpu power with the k20 built-in sensor, in: Proceedings of Workshop on General Purpose Processing using GPUs, GPGPU-7, ACM, New York, NY, USA, 2014, pp. 28:28–28:36, http://dx.doi.org/10.1145/2576779.2576783.
[14] A. Cabrera, F. Almeida, V. Blanco, D. Gimenez, Analytical modeling of the energy consumption for the high performance linpack, in: 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing, IEEE, 2013, pp. 343–350.
[15] M. Chadha, T. Ilsche, M. Bielert, W.E. Nagel, A statistical approach to power estimation for x86 processors, in: Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International, IEEE, 2017, pp. 1012–1019, http://dx.doi.org/10.1109/IPDPSW.2017.98.
[16] I. Corporation, Intel® manycore platform software stack (intel MPSS), 2014, URL https://software.intel.com/en-us/articles/intel-manycore-platform-software-stack-mpss.
[17] I. Corporation, Intel xeon phi coprocessor system software developers guide, 2014.
[18] W. Dargie, A stochastic model for estimating the power consumption of a processor, IEEE Trans. Comput. 64 (5).
[19] M. Dayarathna, Y. Wen, R. Fan, Data center energy consumption modeling: A survey, IEEE Commun. Surv. Tutor. 18 (1) (2016) 732–794.
[20] A.M. Devices, Bios and kernel developer's guide (bkdg) for amd family 15h models 00h-0fh processors, 2012, URL https://www.amd.com/system/files/TechDocs/42301_15h_Mod_00h-0Fh_BKDG.pdf.
[21] J. Dongarra, H. Ltaief, P. Luszczek, V. Weaver, Energy footprint of advanced dense numerical linear algebra using tile algorithms on multicore architecture, in: The 2nd International Conference on Cloud and Green Computing, 2012.
[22] D. Economou, S. Rivoire, C. Kozyrakis, P. and Ranganathan, Full-system power analysis and modeling for server environments, in: In Proceedings of Workshop on Modeling, Benchmarking, and Simulation, 2006, pp. 70–77.

[23] M. Fahad, A. Shahid, R.R. Manumachu, A. Lastovetsky, A comparative study of methods for measurement of energy of computing, Energies 12 (11) http://dx.doi.org/10.3390/en12112204. URL https://www.mdpi.com/1996-1073/12/11/2204.

[24] M. Fahad, A. Shahid, R.R. Manumachu, A. Lastovetsky, Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements, IEEE Access 8 (2020) 93793–93829.

[25] X. Fan, W.-D. Weber, L.A. Barroso, Power provisioning for a warehouse-sized computer, in: 34th Annual International Symposium on Computer Architecture, ACM, 2007, pp. 13–23.

[26] X. Feng, R. Ge, K.W. Cameron, Power and energy profiling of scientific applications on distributed systems, in: Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, IEEE, 2005, p. 34.

[27] J. Flinn, M. Satyanarayanan, Powerscope: A tool for profiling the energy usage of mobile applications, in: Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications, IEEE, 1999, pp. 2–10.

[28] B. Goel, S.A. McKee, R. Gioiosa, K. Singh, M. Bhadauria, M. Cesati, Portable scalable per-core power estimation for intelligent resource management, in: Portable, Scalable, Per-Core Power Estimation for Intelligent Resource Management, Green Computing Conference, 2010 International, 2010.

[29] P. Gschwandtner, M. Knobloch, B. Mohr, D. Pleiter, T. Fahringer, G CPU energy consumption of hpc applications on the IBM POWER7, in: Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on, IEEE, 2014, pp. 536–543.

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

[31] J. Haj-Yihia, A. Yasin, Y.B. Asher, A. Mendelson, Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems, ACM Trans. Archit. Code Optim. (TACO) 13 (4) (2016) 56.

[32] T. Heath, B. Diniz, B. Horizonte, E.V. Carrera, R. Bianchini, Energy conservation in heterogeneous server clusters, in: 10th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), ACM, 2005, pp. 186–195.

[33] H. Hong, Sunpyand Kim, An integrated GPU power and performance model, SIGARCH Comput. Archit. News 38 (3).

[34] C. Isci, M. Martonosi, Runtime power monitoring in high-end processors: Methodology and empirical data, in: 36th Annual IEEE/ACM International Symposium on Microarchitecture, IEEE Computer Society, 2003, p. 93.

[35] G. Jung, M.A. Hiltunen, K.R. Joshi, R.D. Schlichting, C. Pu, Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures, in: Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on, IEEE, 2010, pp. 62–73.

[36] A. Kansal, F. Zhao, Fine-grained energy profiling for power-aware application design, ACM SIGMETRICS Perform. Eval. Rev. 36 (2) (2008) 26.

[37] S. Khokhriakov, R.R. Manumachu, A. Lastovetsky, Multicore processor computing is not energy proportional: An opportunity for bi-objective optimization for energy and performance, Appl. Energy 268 (2020) 114957, http://dx.doi.org/10.1016/j.apenergy.2020.114957, URL http://www.sciencedirect.com/science/article/pii/S0306261920304694.

[38] A. Lastovetsky, R. Reddy, New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters, IEEE Trans. Parallel Distrib. Syst. 28 (4) (2017) 1119–1133.

[39] B.C. Lee, D.M. Brooks, Accurate and efficient regression modeling for microarchitectural performance and power prediction, SIGARCH Comput. Archit. News 34 (5) (2006) 185–194.

[40] T. Li, L.K. John, Run-time modeling and estimation of operating system power consumption, SIGMETRICS Perform. Eval. Rev. 31 (1) (2003) 160–171.

[41] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, K. Cameron, Power-aware predictive models of hybrid (mpi/openmp) scientific applications on multicore systems, Comput. Sci.-Res. Dev. 27 (4) (2012) 245–253.

[42] J.C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A.C. Snoeren, R.K. Gupta, Evaluating the effectiveness of model-based power characterization, in: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'11, USENIX Association, 2011.

[43] C. Mobius, W. Dargie, A. Schill, Power consumption estimation models for processors, virtual machines, and servers, IEEE Trans.Parallel Distrib. Syst. 25 (6).

[44] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, S. Matsuoka, Statistical power modeling of GPU kernels using performance counters, in: International Green Computing Conference and Workshops (IGCC), IEEE, 2010.

[45] K. O'Brien, I. Pietri, R. Reddy, A. Lastovetsky, R. Sakellariou, A survey of power and energy predictive models in HPC systems and applications, ACM Comput. Surv. 50 (3).

[46] R. Reddy, A. Lastovetsky, Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy, IEEE Trans. Comput. 64 (2) (2018) 160–177.

[47] R. Reddy Manumachu, A.L. Lastovetsky, Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution, Concurr. Comput.: Pract. Exper. 31 (4) (2019) e4958, http://dx.doi.org/10.1002/cpe.4958.

[48] S. Rivoire, Models and Metrics for Energy-Efficient Computer Systems, (Ph.D. thesis), Stanford University, Stanford, California, 2008.

[49] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models, in: Proceedings of the 2008 Conference on Power Aware Computing and Systems, HotPower'08, USENIX Association, 2008.

[50] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, D. Rajwan, Power-management architecture of the intel microarchitecture code-named sandy bridge, IEEE Micro. 32 (2) (2012) 20–27.

[51] A. Shahid, M. Fahad, R.R. Manumachu, A. Lastovetsky, Improving the accuracy of energy predictive models for multicore CPUs using additivity of performance monitoring counters, in: V. Malyshkin (Ed.), Parallel Computing Technologies, Springer International Publishing, Cham, 2019, pp. 51–66.

[52] A. Shahid, M. Fahad, R. Reddy, A. Lastovetsky, Additivity: A selection criterion for performance events for reliable energy predictive modeling, Supercomput. Front. Innov.: Int. J. 4 (4) (2017) 50–65.

[53] A. Shahid, M. Fahad, R. Reddy Manumachu, A. Lastovetsky, Energy of computing on multicore cpus: Predictive models and energy conservation law, arXiv. URL arXiv:1907.02805.

[54] A. Shahid, M. Fahad, R. Reddy Manumachu, A. Lastovetsky, Supplemental: Improving the accuracy of energy predictive models using the utilization variables and performance events for multicore cpus, 2019, URL https://github.com/ArsalanShahid116/SLOPE-PMC/blob/master/supplementalJPDC2020.pdf.

[55] Y.S. Shao, D. Brooks, Energy characterization and instruction-level energy model of Intel's Xeon Phi processor, in: Proceedings of the 2013 International Symposium on Low Power Electronics and Design, ISLPED '13, IEEE Press, 2013.

[56] K. Singh, M. Bhadauria, S.A. McKee, Real time power estimation and thread scheduling via performance counters, SIGARCH Comput. Archit. News 37 (2) (2009) 46–55.

[57] S. Song, C. Su, B. Rountree, K.W. Cameron, A simplified and accurate model of power-performance efficiency on emergent GPU architectures, in: 27th IEEE International Parallel & Distributed Processing Symposium (IPDPS), IEEE Computer Society, 2013, pp. 673–686.

[58] M.B. Srivastava, A.P. Chandrakasan, R.W. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 4 (1) (1996) 42–55.

[59] J. Treibig, G. Hager, G. Wellein, Likwid: A lightweight performance-oriented tool suite for x86 multicore environments, in: Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, IEEE, 2010, pp. 207–216.

[60] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, L. Zhou, Distributed systems meet economics: pricing in the cloud, in: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, USENIX Association, 2010.

[61] P. Wiki, Perf: Linux profiling with performance counters, 2017, URL https://perf.wiki.kernel.org/index.php/Main_Page.

[62] M. Witkowski, A. Oleksiak, T. Piontek, J. Weglarz, Practical power consumption estimation for real life HPC applications, Future Gener. Comput. Syst. 29 (1).

[63] X. Wu, V. Taylor, J. Cook, P.J. Mucci, Using performance-power modeling to improve energy efficiency of HPC applications, Computer 49 (10) (2016) 20–29.

[64] W. Ye, N. Vijaykrishnan, M. Kandemir, M.J. Irwin, The design and use of simplepower: a cycle-accurate energy estimation tool, in: Proceedings of the 37th Annual Design Automation Conference, ACM, 2000, pp. 340–345.

Arsalan Shahid received his PhD degree from the University College Dublin. He graduated as a gold medalist for the best final year project in BS Electrical Engineering from HITEC University Pakistan, in 2016. His research interests are energy-aware high-performance heterogeneous computing and intelligent cloud computing.

Muhammad Fahad is a Ph.D. researcher in Heterogeneous Computing Lab (HCL) at University College Dublin, Ireland. He received his MS degree from KTH - Royal Institute of Technology, Sweden in 2012, and BS degree from International Islamic University Islamabad, Pakistan in 2008. His main research interests include high-performance heterogeneous computing, energy-efficient computing, parallel/distributed and peer-to-peer computing.

Ravi Reddy Manumachu received a B.Tech degree from I.I.T, Madras in 1997 and a Ph.D. degree from the School of Computer Science, University College Dublin in 2005. His main research interests include high performance heterogeneous computing, distributed computing, energy-efficient computing, and sparse matrix computations.

Alexey Lastovetsky received a Ph.D. degree from the Moscow Aviation Institute in 1986, and a Doctor of Science degree from the Russian Academy of Sciences in 1997. His main research interests include high-performance heterogeneous computing and energy-efficient computing. He has published over 150 technical papers in refereed journals, edited books, and international conferences. He authored the monographs Parallel computing on heterogeneous networks (Wiley, 2003) and High-performance heterogeneous computing (Wiley, 2009).