IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 16, 2021, accepted April 19, 2021, date of publication April 22, 2021, date of current version May 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3075139

Energy Predictive Models of Computing: Theory, Practical Implications and Experimental Analysis on Multicore Processors

ARSALAN SHAHID, MUHAMMAD FAHAD, RAVI REDDY MANUMACHU, AND ALEXEY LASTOVETSKY

School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland

Corresponding author: Ravi Reddy Manumachu (ravi.manumachu@ucd.ie)

This work was supported by the Science Foundation Ireland (SFI) under Grant 14/IA/2474.

ABSTRACT The energy efficiency in ICT is becoming a grand technological challenge and is now a first-class design constraint in all computing settings. Energy predictive modelling based on performance monitoring counters (PMCs) is the leading method for application-level energy optimization. However, a sound theoretical framework to understand the fundamental significance of the PMCs to the energy consumption and the causes of the inaccuracy of the models is lacking. In this work, we propose a small but insightful theory of energy predictive models of computing, which formalizes both the assumptions behind the existing PMC-based energy predictive models and properties, heretofore unconsidered, that are basic implications of the universal energy conservation law. The theory's basic practical implications include selection criteria for model variables, model intercept, and model coefficients. The experiments on two modern Intel multicore servers show that applying the proposed selection criteria improves the prediction accuracy of state-of-the-art linear regression models from 31.2% to 18%. Finally, we demonstrate that employing energy models constructed using the proposed theory for energy optimization can save a significant amount of energy (up to 80% for applications used in experiments) compared to state-of-the-art energy measurement tools.

INDEX TERMS Multicore processor, energy predictive modeling, performance monitoring counters, energy conservation, energy optimization, linear regression.

I. INTRODUCTION

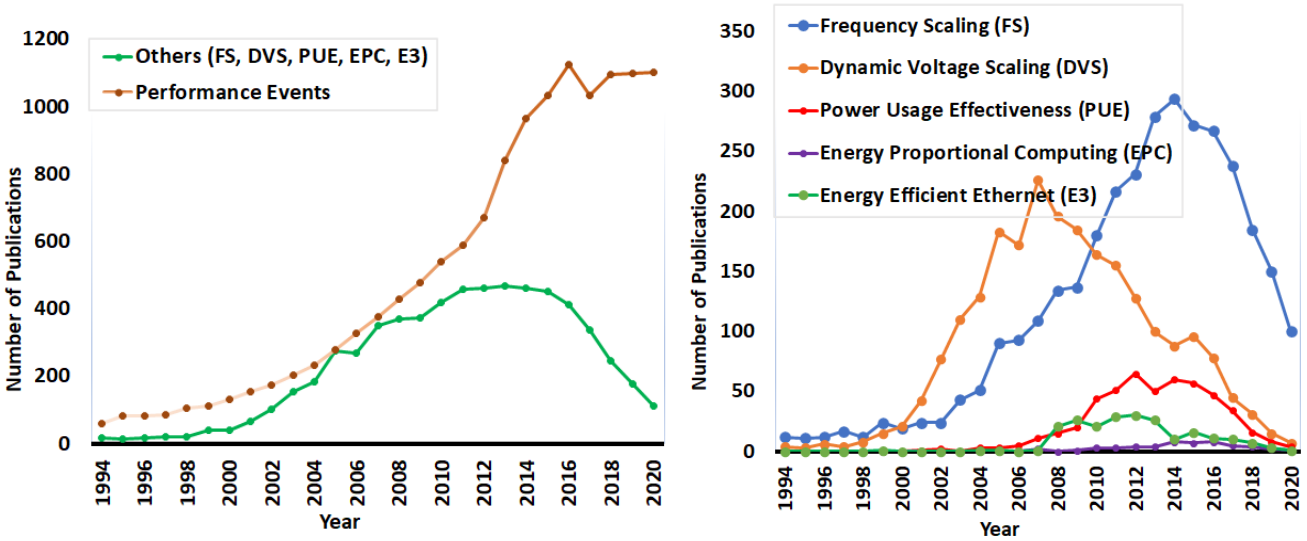
According to the study [1], the energy consumption of Information and Communications Technology (ICT) is 7% of the global electricity usage in 2020 and is forecast to be around the average of the best-case and expected scenarios (7% and 21%) by 2030.

Multicore processors are at the heart of modern computing platforms. Architects of modern multicore processors follow a key design goal called energy proportionality (EP) (Barroso and Hölzle [2]), which means designing microprocessors that consume energy proportional to the amount of work performed. One of EP's implications is that optimizing an application for performance will also optimize it for energy, signified by a monotonically increasing relationship between energy and execution time. Therefore, the mainstream thinking is that optimizing an application for performance optimizes it automatically for energy.

However, Khokhriakhov *et al.* [3] demonstrate that EP does not hold for modern multicore processors using a novel application-level bi-objective optimization method for energy and performance on a single multicore processor. They experiment with four popular and highly optimized multithreaded data-parallel applications on four modern multicore processors. They show that optimizing for performance alone may result in a significant increase in dynamic energy consumption and optimizing for dynamic energy alone – in considerable performance degradation. Their optimization method determined a good number of Pareto-optimal solutions.

Therefore, energy efficiency in ICT is becoming a grand technological challenge and is now a first-class design constraint in all computing settings

The associate editor coordinating the review of this manuscript and approving it for publication was Jie Tang.



(a) Research publications in performance events and other areas of energy of computing (1994-2020)

(b) Research publications in energy of computing excluding performance events (1994-2020)

FIGURE 1. A comparison of number of research publications in the energy of computing using the performance monitoring counters (PMCs) and research publications in other areas of energy of computing. These statistics are collected from Google Scholar and Microsoft Academic. The number of publications in the area of PMC based energy modelling and prediction has increased by 4.5×.

(Barroso and Hölzle [2], and U.S Department of Energy report [4]).

The mainstream approach to improving energy efficiency in computing is hardware-level or system-level, driven mainly by hardware innovations in manufacturing energy-efficient devices. Energy-efficient hardware design techniques such as the clock and power gating, dynamic voltage and frequency scaling, and dynamic power management are present in all modern processors (Ayoub *et al.* [5], Zhuravlev *et al.* [6], Petrucci *et al.* [7]). The second approach consists of solution methods that optimize applications rather than the operating environment. The methods use energy predictive models of applications and application-level decision variables such as the number of threads, loop tile size, and workload distribution (Lastovetsky *et al.* [8], Chakrabarti *et al.* [9], Khaleghzadeh *et al.* [10]). An overview of the system-level and application-level energy-efficiency techniques are given in the Appendix B.

Accurate measurement of energy consumption during an application execution is essential to energy minimization at the application level. There are three mainstream measurement approaches: (a). System-level physical power measurements using external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models. System-level physical measurements using external power meters is considered the *ground truth*.

Fahad *et al.* [11] present a comparative study of on-chip sensors and energy predictive models against the ground truth. An overview of the study is given in the Appendix C. To summarize, system-level physical power measurements provided by power meters is an expensive approach. Energy measurements provided by state-of-the-art on-chip sensors are not recommended for energy optimization of applications since their reported accuracy is low and does not capture the holistic picture of the energy consumption during application execution.

Software energy predictive models emerged as a popular alternative to determine the energy consumption of an application. Figures 1a, 1b illustrate that this approach has been most dominant in recent years. A vast majority of the models employ performance monitoring counters (PMCs) as the model variables. A software energy predictive model allows determination of fine-grained decomposition of energy consumption during an application execution at less cost than the ground truth. However, this approach suffers from serious drawbacks such as the complexity of model construction and lack of consensus among the research works, which report prediction accuracies ranging from poor to excellent (Economou *et al.* [12], McCullough *et al.* [13], Witkowski *et al.* [14], Jarus *et al.* [15], Gschwandtner *et al.* [16], Wu *et al.* [17], Haj-Yihia *et al.* [18], O'Brien *et al.* [19]). Moreover, PMCs are selected solely based on their high positive correlation with energy consumption without understanding their physical significance. In summary, a sound theoretical framework to understand the fundamental significance of the model variables with energy consumption and the causes of inaccuracy in models is lacking. We bridge the gap in this work.

In this work, we summarize and generalize the assumptions behind the existing research on PMC-based energy predictive modelling. We use a model-theoretic approach to formulate the assumed properties of the existing models in a mathematical form. We extend the formalism by adding properties, heretofore unconsidered, that are basic implications of the universal energy conservation law. The new properties are intuitive and have been experimentally validated.

The extended formalism defines our theory of energy predictive models of computing. An energy predictive model satisfying the extended model's properties is termed a *consistent* energy model. Using the theory, we prove that a consistent energy predictive model is linear if and only if each PMC variable is *additive* in the sense that the PMC for a serial execution of two applications is the sum of PMCs for the individual execution of each application. The theory's basic practical implications for improving the prediction accuracy of linear energy predictive models are unified in a *consistency* test. The test includes selection criteria for model variables, model intercept, and model coefficients.

We employ the consistency test in the state-of-the-art models and study their prediction accuracy using a strict experimental methodology on a modern Intel multicore processor. We improve the prediction accuracy of state-of-the-art linear regression models from 31.2% to 18%. We explore the accuracy limits of application-specific energy predictive models using a comprehensive experimental study based on the consistency test. The results show that the most accurate models employ five and six performance monitoring counters as predictor variables. Therefore, at least six hardware registers are needed to storing the performance monitoring counters so that the models can be employed online.

We study optimization of a parallel matrix-matrix application for dynamic energy using two measurement tools, IntelRAPL [20], which is a popular mainstream tool, and physical measurements using power meters, considered the ground truth. Finally, we demonstrate that employing energy models constructed using the proposed theory in the energy optimization loop of an application can save a significant amount of energy (up to 80% for applications used in experiments) compared to state-of-the-art energy measurement methods.

In summary, the main contributions of this work are:

- A novel theory of energy predictive models of computing and its practical implications to improve the prediction accuracy of linear energy predictive models.
- Improvements to the prediction accuracy of state-of-the-art linear energy predictive models by applying the practical implications of our proposed theory.
- A detailed experimental study establishing that the high positive correlation of the model variables with dynamic energy consumption alone is insufficient to provide good prediction accuracy for a model. However, the model variables must also satisfy the properties of the consistency test.
- A study demonstrating that employing energy models constructed using the proposed theory for energy optimization can save a significant amount of energy compared to state-of-the-art energy measurement tools.

We organize the rest of this paper as follows. We start with literature survey in section II. Our proposed theory is introduced in III. Then, we present our experimental results and discussion IV. Finally, we conclude the paper in section V.

II. RELATED WORK

This section presents a literature review of the following related topics on energy predictive models of computing:

- Tools widely used to obtain PMCs;
- Energy predictive models, and
- Critical reviews of PMCs.

A. TOOLS TO OBTAIN PMCs

Perf [21] can be used to gather software events such as *context-switches*, *minor-faults*, etc., and hardware events such as *instructions retired*, *L1 cache misses*, etc., for Linux-based systems. PAPI [22] is a well-known portable API for reading PMCs found in majority of the microprocessors. Intel PCM [23] is used for reading PMCs of core and uncore (which includes the QuickPath Interconnect) components of an Intel processor. Likwid [24] can be used to obtain PMCs for both Intel and AMD processors in Linux. For Nvidia GPUs, CUDA Profiling Tools Interface (CUPTI) [25] can be used for obtaining the PMCs.

B. NOTABLE ENERGY PREDICTIVE MODELS FOR CPUs

One of the first models correlating PMCs to energy values was developed by Bellosa [26]. Their model is based on integer operations, floating-point operations, and memory requests due to cache misses, which they believed to be strongly correlated with energy consumption. Icsi and Martonosi [27] propose an elaborate methodology to determine component-level power estimates from the access rates of the components, which are based on PMCs. Li and John [28] propose power models for the operating system (OS) based on their observations of a strong correlation between instructions per cycle (IPC) and OS routine power. Lee and Brooks [29] adopt a statistically rigorous approach to derive regression models using performance events to predict power.

A linear model based on the utilization of CPU, disk, and the network is proposed by Heath, Diniz, Horizonte, Carrera, and Bianchini [30]. A more complex power model (Mantis) studied by Economou, Rivoire, Kozyrakis, and Ranganathan [12] employs utilization metrics of CPU, disk, and network components and hardware performance counters for memory as predictor variables. Fan, Weber, and Barroso [31] propose a simple linear model that correlates the power consumption of a single-core processor with its utilization. Rivoire, Ranganathan, and Kozyrakis [32] study and compare five full-system real-time power models using various machines and benchmarks. Four of these models are utilization-based, whereas the fifth includes CPU PMCs in the model variable set along with the utilization of CPU and disk. They report that the PMC-based model is the best overall in terms of accuracy since it accounted for most of the contributors to the system's dynamic power (especially the memory activity).

Singh, Bhadauria, and McKee [33] develop per-core power models based on multiple linear regression using PMCs.

Powell, Biswas, Emer, Mukherjee, Sheikh and Yardi [34] use a linear regression model to estimate activity factors and power for many micro-architectural structures using a small number of PMCs. Goel, McKee, Gioiosa, Singh, Bhadauria, and Cesati [35] derive per-core power models using PMC values and temperature readings. A linear model that takes into account CPU utilization and I/O bandwidth is described in [36] to predict the power consumption of a server. Basmadjian, Robert and Ali, Nasir and Niedermeier, Florian and de Meer, Hermann and Giuliani, Giovanni [37] construct a power model of a server as a summation of power models of its components, the processor (CPU), memory (RAM), fans, and disk (HDD). Bertran, Gonzalez, Martorell, Navarro, and Ayguade [38] present a power model that provides a per-component power breakdown of a multicore CPU. Their model is based on activity factors obtained from PMCs for various components in a multicore CPU.

Bircher and John [39] propose an iterative modeling procedure to predict power using PMCs. They use PMCs that trickle down from the processor to other subsystems such as CPU, disk, and GPU, and PMCs that flow inward into the processor such as Direct Memory Access (DMA) and I/O interrupts. Basmadjian and de Meer [40] report that the summation of power consumption of all active cores to derive the total power consumption is inaccurate and take into account resource sharing in their power prediction model for multicore processors.

Rotem, Naveh, Ananthakrishnan, Weissmann, and Rajwan [20] present a software power model, which eventually became *RAPL*, in Intel Sandybridge. This model predicts the energy consumption of core and uncore components (QPI, LLC) based on some PMCs (which are not disclosed). McPAT [41] is an integrated power, area, and timing modeling framework for multithreaded, multicore, and manycore architectures. It supports the estimation of power consumption for various components in a multiprocessor including shared caches, integrated memory controllers, in-order and out-of-order processor cores, and networks-on-chip. However, McPAT has known limitations in power estimation, which were reported by Xi, Jacobson, Bose, Wei, and Brooks [42]. Dargie *et al.* [43] use the statistics of CPU utilization to model the relationship between the power consumption of the multicore processor and workload quantitatively. They demonstrate that the relationship is quadratic for a single-core processor and linear for multicore processors. Haj-Yihia, Yasin, Asher, and Mendelson [18] present a linear regression model for Intel Skylake processors based on PMCs. They selected the PMCs, which are popular in well-known energy and power models.

Lastovetsky and Reddy [8] present an application-level energy model where a function of problem size represents the dynamic energy consumption of a processor. Unlike PMC-based models containing hardware-related PMCs and not considering problem size as a variable, this model considers the highly non-linear and non-convex nature of the relationship between energy consumption and problem size for solving optimization problems of data-parallel applications on homogeneous multicore clusters for energy.

C. NOTABLE ENERGY PREDICTIVE MODELS FOR SOFTWARE AND HARDWARE ACCELERATORS

Hong and Kim [44] propose an energy prediction model for an Nvidia GPU similar to the PMC-based unit power prediction approach of Icsi and Martonosi [27]. Nagasaka, Maruyama, Nukada, Endo, and Matsuoka [45] present a statistical approach that uses GPU performance counters exposed for CUDA applications to predict the power consumption of GPU kernels. Song, Su, Rountree, and Cameron [46] propose power and energy prediction models that employ a configurable, back-propagation, artificial neural network (BP-ANN). The variables of the BP-ANN model are ten carefully selected PMCs of a GPU. Shao and Brooks [47] construct an instruction-level energy model of a Xeon Phi processor. Al-Khatib and Abdi [48] propose a linear instruction-level model to predict dynamic energy consumption for soft processors in FPGA. The model considers both inter-instruction effects and the operand values of the instructions.

D. NOTABLE ENERGY PREDICTIVE MODELS FOR HPC APPLICATIONS

Witkowski, Oleksiak, Piontek, and Weglarz [14], Jarus, Oleksiak, Piontek, and Weglarz [15] propose system-wide power prediction models for HPC servers based on performance counters. They cluster real-life HPC applications into groups and create specialized power models for them. Gschwandtner, Knobloch, Mohr, Pleiter, and Fahringer [16] present linear regression models based on hardware counters for prediction of energy consumption of HPC applications executing on IBM POWER7 processor. They pick a small subset from 500 different hardware counters offered by the POWER7 processor. Wu, Taylor, Cook, and Mucci [17] present a PMC-based energy predictive model for HPC application workloads.

E. CRITICAL REVIEWS OF PMCs FOR ENERGY PREDICTIVE MODELLING

Economou, Rivoire, Kozyrakis, and Ranganathan [12] highlight the fundamental limitation: 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, Agarwal, Chandrasekhar, Kuppuswamy, Snoeren, and Gupta [13] 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. Hackenberg, Ilsche, Schöne, Molka, Schmidt, and Nagel [49] present a study of various power measurement strategies, which includes *Intel RAPL* [20]. They report that the accuracy of RAPL depends

on the type of workload and is quite poor for workloads that use the hyper-threading feature.

O'Brien, Pietri, Reddy, Lastovetsky, and Sakellariou [19] survey predictive power and energy models focusing on the highly heterogeneous and hierarchical node architecture in modern HPC computing platforms. Using a case study of PMCs, they highlight the poor prediction accuracy and ineffectiveness of models to accurately predict the dynamic power consumption of modern nodes due to the inherent complexities (contention for shared resources such as Last Level Cache (LLC), NUMA, and dynamic power management). Shahid, Fahad, Reddy, and Lastovetsky [50] propose a novel selection criterion for PMCs called additivity to determine the subset of PMCs that can potentially be considered for reliable energy predictive modelling. They study the additivity of PMCs offered by two popular tools, Likwid [24], and PAPI [22], using a detailed statistical experimental methodology on a modern Intel Haswell multicore server CPU. They show that many PMCs in Likwid and PAPI are non-additive and that some of these PMCs are key predictor variables in energy predictive models.

F. SUMMARY

Energy predictive models using performance monitoring counters emerged as a dominant measurement method. Its main advantage compared to the ground truth (system-level physical measurements using power meters) is the determination of fine-grained decomposition of energy consumption of an application's execution. This approach, however, has several shortcomings:

- The high complexity of model construction and lack of consensus among the research works, reporting prediction accuracy ranging from poor to excellent.
- A vast majority of research works select PMCs solely based on their high positive correlation with energy consumption without any deep understanding of the model variables' physical significance.
- The lack of a sound theoretical framework to understand the model variables' physical significance to the energy consumption and the causes of the inaccuracy of the models.

In this work, we address the shortcomings by proposing a novel theory of energy predictive models of computing and its practical implications to improve the prediction accuracy of linear energy predictive models. We also demonstrate that the employment of inaccurate energy measurement tools for energy optimization can cause significant energy losses because they do not consider the proposed theory's properties. We further show that employing an accurate energy predictive model constructed using the theory can lead to significant energy savings.

III. ENERGY PREDICTIVE MODELS OF COMPUTING: INTUITION, MOTIVATION, AND THEORY

We summarize and generalize the assumptions behind the current work on PMC-based power/energy modelling. We use a model-theoretic approach to formulate the assumed properties of these models in a mathematical form. Then we extend the formalism by adding properties, which are intuitive and which we have experimentally validated but have never been considered previously. The properties are manifestations of the fundamental physical law of energy conservation. We introduce two definitions based on the properties of the extended model, called *weak composability* and *strong composability*. An energy predictive model satisfying all the properties of the extended model is termed a *consistent* energy model. The extended model and the two definitions define our theory of energy predictive models of computing.

Finally, we mathematically derive properties of *linear* consistent energy predictive models. We prove that a consistent PMC-based energy model is linear if and only if it is strongly composable with each PMC variable being additive. The practical implication of this theoretical result is that each PMC variable of a linear energy predictive model must be *additive*. The significance of this property is that it can be efficiently tested and hence used in practice to identify PMC variables that must not be included in the model. The notation and the terminology used in the proposed theory is given in Table 1.

A. INTUITION AND MOTIVATION

The essence of PMC-based energy predictive models is that an application run can be accurately characterized by a n-vector of PMCs over $\mathbb{R}_{\geq 0}$. Any two application runs characterized by the same PMC vector are supposed to consume the same amount of energy. The applications in these runs may be different, but the same computing environment is always assumed. Thus, PMC-based models are computer system-specific.

Based on these assumptions, any PMC-based energy model is formalized by a set of PMC vectors over $\mathbb{R}_{\geq 0}$, and a function, $f_E : \mathbb{R}^n_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, mapping these vectors in the set to energy values. *No other properties of the set and the function are assumed.*

In this work, we extend this model by adding properties that characterize the serial execution of two applications. To aid the exposition, we follow some notation and terminology. A *compound application* is defined as the serial execution of two applications, which we call the *base* applications. If the base applications are A and B, we denote their compound application by $A \oplus B$. We will refer solely to energy predictive models hereafter since there exists a linear functional mapping from PMC-based power predictive models to them. When we say energy consumption, we mean dynamic energy consumption. The energy consumption that is experimentally observed during the execution of an application A is denoted by $E(A)$. The energy consumption of the compound application $A \oplus B$, $E(A \oplus B)$, is the energy consumption that is experimentally observed during the execution of the compound application.

First, we aim to reflect in the model the observation that in a stable and dedicated environment, where each run of the same

TABLE 1. Notation and terminology used in the theory of energy predictive models of computing.

Notation	Description
$A, B, ...$	Base applications
$A \oplus B$	Compound application of the base applications A and B
$\mathcal{A}$	Set of applications
$E(A)$	Energy consumption of application A
$E(A \oplus B)$	Energy consumption of compound application $A \oplus B$
$p = \{p_k\}_{k=1}^n, q = \{q_k\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n$	PMC vectors p and q
$\mathcal{NULL} = \{0\}_{k=1}^n$	A null vector of PMCs
$f_E : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$	A PMC-based energy predictive model
$f_E(a)$	Energy value for the input PMC vector a
$\mathcal{O}$	Set of binary operators
$a_k \circ_{AB,k} b_k$	Binary operator $\circ_{AB,k}$ combining the k-th PMCs a_k and b_k in the PMC vectors a and b for the applications $A, B \in \mathcal{A}$, respectively
$\{\circ_{AB,1}, \cdots, \circ_{AB,n}\}$	Set of binary operators combining the PMC vectors for the applications $A, B \in \mathcal{A}$

application is characterized by the same PMC vector, for any two applications, the PMC vector of their serial execution will always be the same. To introduce this property, we add to the model a (infinite) set of applications denoted by $\mathcal{A}$. We postulate the existence of binary operators, $\mathcal{O} = \{\circ_{AB,k} : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}, A, B \in \mathcal{A}, k \in [1, n]\}$ so that for each $A, B \in \mathcal{A}$ and their PMC vectors $a = \{a_k\}_{k=1}^n, b = \{b_k\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n$ respectively, the PMC vector of the compound application $A \oplus B$ will be equal to $\{a_k \circ_{AB,k} b_k\}_{k=1}^n$.

Next, we introduce properties, which are manifestations of the universal energy conservation law. The following property essentially states that doing nothing (signified by a null vector of PMCs, $\mathcal{NULL} = \{0\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n$) does not consume or generate energy,

$$f_E(\mathcal{NULL}) = 0$$

The following property postulates that an application with a PMC vector that is not $\mathcal{NULL}$ must consume some energy. The intuition behind this property is that since PMCs account for energy consuming activities of applications, an application with any energy consuming activity higher than zero activity (a $\mathcal{NULL}$ PMC vector), must consume more energy than zero.

$$\forall a \in \mathbb{R}_{\geq 0}^n \wedge a \neq \mathcal{NULL}, \; f_E(a) > 0$$

Finally, we aim to reflect the observation that the consumed energy of compound application $A \oplus B$ is always equal to the sum of energies consumed by the individual applications A and B respectively,

$$E(A \oplus B) = E(A) + E(B) \tag{1}$$

To introduce this property in the extended model, we postulate the following,

$$\forall A, B \in \mathcal{A}, a = \{a_k\}_{k=1}^n, b = \{b_k\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n, \; \circ_{AB,k} \in \mathcal{O}, \\ f_E(\{a_k \circ_{AB,k} b_k\}_{k=1}^n) = f_E(a) + f_E(b)$$

To summarize, while existing models are focused on abstract application runs and lack any notion of applications, we introduce this notion in the extended model. The additional structure introduced in the extended model allows one to prove the mathematical properties of energy predictive models.

B. FORMAL SUMMARY OF PROPERTIES OF EXTENDED MODEL

The formal summary of the properties of the extended model follows:

Property 1 (Inherited From Basic Model): An abstract application run is accurately characterized by a set of n-vector of PMCs over $\mathbb{R}_{\geq 0}$. A null vector of PMCs is represented by $\mathcal{NULL} = \{0\}_{k=1}^n$. There exists a function, $f_E : \mathbb{R}_{\geq 0}^n \rightarrow \mathbb{R}_{\geq 0}$, mapping the vectors to energy values and $\forall p, q \in \mathbb{R}_{\geq 0}^n, p = q \implies f_E(p) = f_E(q)$.

Property 2 Weak Composability, Applications and Operators: There exists an application space, $(\mathcal{A}, \oplus)$, where $\mathcal{A}$ is a (infinite) set of applications and $\oplus$ is a binary function on $\mathcal{A}$, $\oplus : \mathcal{A} \times \mathcal{A} \rightarrow \mathcal{A}$. There exists a (infinite) set of binary operators, $\mathcal{O} = \{\circ_{PQ,k} : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}, P, Q \in \mathcal{A}, k \in [1, n]\}$ so that for each $P, Q \in \mathcal{A}$ and their PMC vectors $p = \{p_k\}_{k=1}^n, q = \{q_k\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n$ respectively, the PMC vector of the compound application $P \oplus Q$ will be equal to $\{p_k \circ_{PQ,k} q_k\}_{k=1}^n$.

Property 3 (Zero Energy, Energy Conservation): $f_E(\mathcal{NULL}) = 0$.

Property 4 (Positive-Definiteness, Energy Conservation): $\forall p \in \mathbb{R}_{\geq 0}^n \wedge p \neq \mathcal{NULL}, \; f_E(p) > 0$.

Property 5 (Weak Composability, Energy Conservation): $\forall P, Q \in \mathcal{A}, \; p = \{p_k\}_{k=1}^n, q = \{q_k\}_{k=1}^n \in \mathbb{R}_{\geq 0}^n, \; \circ_{PQ,k} \in \mathcal{O}, f_E(\{p_k \circ_{PQ,k} q_k\}_{k=1}^n) = f_E(p) + f_E(q)$.

We term an energy predictive model satisfying all the above properties of the extended model a *consistent* energy model.

C. STRONG COMPOSABILITY: DEFINITION

The definition of *strong composability* of models follows:

Definition 1 (Strong Composability): A consistent energy model is strongly composable *if* $\forall P, Q, R, S \in \mathcal{A}, p = \{p_k\}_{k=1}^n, q = \{q_k\}_{k=1}^n, r = \{r_k\}_{k=1}^n, s = \{s_k\}_{k=1}^n \in \mathbb{R}^n_{\geq 0}$, $k \in [1, n]$, $\circ_{PQ,k} = \circ_{RS,k}$.

The *strong composability* property of a model essentially states that binary operators used in the model to compute PMC vectors of compound applications are not application specific. In other words, the set $\mathcal{O}$ consists of only n binary operators, one for each PMC parameter, $\mathcal{O} = \{\circ_k\}_{k=1}^n$, so that for any $P, Q \in \mathcal{A}$ and their PMC vectors $p = \{p_k\}_{k=1}^n, q = \{q_k\}_{k=1}^n \in \mathbb{R}^n_{\geq 0}$, the PMC vector of the compound application $P \oplus Q$ will be equal to $\{p_k \circ_k q_k\}_{k=1}^n$.

D. MATHEMATICAL ANALYSIS OF LINEAR ENERGY PREDICTIVE MODELS BASED ON THE THEORY OF ENERGY OF COMPUTING

In this section, we mathematically derive properties of *linear* consistent energy predictive models, that is, linear energy models satisfying properties (1 to 5).

By definition, a model is *linear* iff $f_E(x)$ is a linear function.

To the best of our knowledge, all the state-of-the-art energy predictive models for multicore CPUs are based on linear regression. While they model total energy consumption, we consider dynamic energy consumption for reasons described in the Appendix D. The mathematical form of these models can be stated as follows: $\forall p = (p_k)_{k=1}^n, p_k \in \mathbb{R}_{\geq 0}$,

$$f_E(p) = \beta_0 + \beta \times p = \beta_0 + \sum_{k=1}^{n} \beta_k \times p_k \tag{2}$$

where β_0 is called the model intercept, the $\beta = \{\beta_1, \ldots, \beta_n\}$ is the vector of regression coefficients or the model parameters. In real life, there usually is stochastic noise (measurement errors). Therefore, the measured energy is typically expressed as

$$\tilde{f}_E(p) = f_E(p) + \epsilon \tag{3}$$

where the error term or noise ϵ is a Gaussian random variable with expectation zero and variance σ^2, written $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We will ignore the noise term in our mathematical proofs to follow.

Theorem 1: If a linear energy predictive model (2) is consistent, the model intercept must be zero and the model coefficients must be positive.

Proof: From the energy conservation property 3,

$$\begin{aligned} &\mathcal{NULL} = \{0\}_{k=1}^n \in \mathbb{R}^n_{\geq 0},\ f_E(\mathcal{NULL}) = 0 \\ &\implies \beta_0 + \sum_{k=1}^{n} \beta_k \times 0 = 0 \\ &\implies \beta_0 = 0 \end{aligned}$$

From the energy conservation property 4,

$$\begin{aligned} &\forall k \in [1, n], p = \{0, \ldots, 0, p_k, 0, \ldots, 0\} \wedge p \neq \mathcal{NULL}, \\ &f_E(p) > 0 \\ &\implies \sum_{i=1}^{n} \beta_i \times p_i > 0 \\ &\implies \beta_k \times p_k > 0 \\ &\implies \beta_k > 0 \text{ since } p_k > 0 \end{aligned}$$

□

To summarize, a linear energy predictive model satisfying energy conservation properties (3 and 4) has a zero model intercept and positive model coefficients. Also as we only consider models satisfying property 3, then the linearity of function $f_E(x)$ can be equivalently defined as follows: for any $\alpha \in \mathbb{R}_{\geq 0}$ and $p, q \in \mathbb{R}^n_{\geq 0}$

$$f_E(p + q) = f_E(p) + f_E(q) \tag{4}$$

and

$$f_E(\alpha \times p) = \alpha \times f_E(p) \tag{5}$$

Theorem 2: If a consistent energy model is linear, then it is strongly composable with $O = \{+\}$.

Proof: From properties 2 and 5 of *weak composability*, we have

$$\begin{aligned} &\forall P, Q \in \mathcal{A}, \forall k \in [1, n], p \\ &\quad = \{0, \ldots, 0, p_k, 0, \ldots, 0\}, \\ &\quad q = \{0, \ldots, 0, q_k, 0, \ldots, 0\} : \\ &f_E(\{0, \ldots, 0, p_k \circ_{PQ,k} q_k, 0, \ldots, 0\}) \\ &\quad = f_E(p) + f_E(q) \end{aligned}$$

Using the property (4) of a linear predictive model,

$$\begin{aligned} &f_E(p + q) \\ &= f_E(p) + f_E(q) \\ &\quad \implies f_E(p + q) = f_E(\{0, \ldots, 0, p_k \circ_{PQ,k} q_k, 0, \ldots, 0\}) \\ &\quad \implies f_E(\{0, \ldots, p_k + q_k, 0, \ldots, 0\}) \\ &= f_E(\{0, \ldots, 0, p_k \circ_{PQ,k} q_k, 0, \ldots, 0\}) \\ &\quad \implies p_k + q_k = p_k \circ_{PQ,k} q_k \quad \text{(from linearity of} f_E(x)) \\ &\quad \implies \circ_{PQ,k} = + \end{aligned}$$

Therefore, if a consistent energy model is linear, then it is strongly composable with $O = \{+\}$. □

Theorem 3: If a consistent energy model is strongly composable with $O = \{+\}$ *and function* $f_E(x)$ *is continuous, then it is linear.*

Proof: First, we prove the first defining linearity property (4),

$$f_E(p + q) = f_E(p) + f_E(q)$$

for any $p, q \in \mathbb{R}^n_{\geq 0}$.

As the model is *strongly composable* with $O = \{+\}$, then

$$\forall P, Q \in \mathcal{A}, \forall k \in [1, n] : \circ_{PQ,k} = +$$

From property 5 of *weak composability*,

$$f_E(\{p_k \circ_{PQ,k} q_k\}_{k=1}^n)$$

$$= f_E(p) + f_E(q)$$
$$\implies f_E(p) + f_E(q) = f_E(\{p_k \circ_{PQ,k} q_k\}_{k=1}^{n})$$
$$\implies f_E(p) + f_E(q) = f_E(\{p_k + q_k\}_{k=1}^{n}) = f_E(p+q)$$

This proves the first property of linearity.

We now prove the second defining property of linearity (5),

$$f_E(\alpha \times p) = \alpha \times f_E(p)$$

for any $p \in \mathbb{R}^n_{\geq 0}$ and $\alpha \in \mathbb{R}_{\geq 0}$.

For any integer $m > 0$,

$$\begin{aligned} f_E(m \times p) &= f_E(p + p + \ldots + p) \\ &= f_E(p) + f_E(p) + \ldots + f_E(p) \\ &= m \times f_E(p) \end{aligned}$$

For any integer $n > 0$,

$$\begin{aligned} f_E(p) &= f_E(\frac{p}{n}) + f_E(\frac{p}{n}) + \ldots + f_E(\frac{p}{n}) \\ &= n \times f_E(\frac{p}{n}) \\ &\implies \frac{1}{n} f_E(p) = f_E(\frac{q}{n}) \end{aligned}$$

Thus, for any rational $\frac{m}{n} > 0$,

$$\begin{aligned} \frac{m}{n} f_E(q) &= \frac{1}{n} f_E(q) + \frac{1}{n} f_E(q) + \ldots + \frac{1}{n} f_E(q) \\ &= f_E(\frac{q}{n}) + f_E(\frac{q}{n}) + \ldots + f_E(\frac{q}{n}) \\ &= f_E(m \times \frac{q}{n}) = f_E(\frac{m}{n} q) \end{aligned}$$

By definition, any real number α is a limit of an infinite sequence of rational numbers. Consider a sequence $\{\alpha_k\}$ of positive rational numbers such that $\lim_{k \to +\infty} \alpha_k = \alpha$. Then,

$$\begin{aligned} f_E(\alpha \times p) &= f_E((\lim_{k \to +\infty} \alpha_k) \times p) \\ &= f_E(\lim_{k \to +\infty} (\alpha_k \times p)) \\ &= \lim_{k \to +\infty} f_E(\alpha_k \times p) \quad \text{(from continuity of} f_E(x)) \end{aligned}$$

As α_k are positive rational numbers, $f_E(\alpha_k \times p) = \alpha_k \times f_E(p)$. Therefore,

$$\begin{aligned} f_E(\alpha \times p) &= \lim_{k \to +\infty} (\alpha_k \times f_E(p)) \\ &= f_E(p) \times \lim_{k \to +\infty} \alpha_k \\ &= f_E(p) \times \alpha \end{aligned}$$

□

Therefore, we prove using theorem 2 and theorem 3 that a consistent energy model is linear if and only if it is strongly composable with $O = +$. A consistent PMC-based energy model is linear if and only if it is strongly composable, with each PMC variable being additive. The practical implication of this theoretical result is that each PMC variable of a linear energy predictive model must be additive.

E. DISCUSSION

In this section, we present practical implications of our proposed theory that can be employed to construct accurate and reliable linear energy predictive models and some guidelines for the design of reliable energy predictive models.

- The basic practical implications of the theory for improving the prediction accuracy of 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 reproducible.
  - Each model variable must be additive in the sense that the value of the model variable for a serial execution of two applications be equal to the sum of its values obtained for the individual execution of each application.
  - 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 our theory, any consistent, and hence accurate, energy model, which only employs PMCs, must be linear. It also means that any non-linear energy model only employing PMCs will be inconsistent and hence inherently inaccurate.
- Thus, a non-linear energy model must employ non-additive parameters in addition to PMCs in order to be accurate.
- If the prediction accuracy of the best linear energy predictive model satisfying the properties of the *consistency* test is still low, then one must explore the use of non-linear modelling techniques, which employ non-additive model variables that are highly positively correlated with dynamic energy consumption. However, non-linear models must still be consistent (i.e., must satisfy all the properties of the extended model) to be reliable and accurate.

Our experiments apply the practical implications of the proposed theory to the state-of-the-art energy predictive models to study their prediction accuracy.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

We employ two modern Intel multicore servers, HCLServer1 (Table 2) and HCLServer2 (Table 3). Our experimental test suite comprises seventeen optimized and unoptimized applications summarized in the Table 4.

Total energy consumption, dynamic power consumption, execution time, and PMCs are obtained for each application run. The dynamic energy consumption during the

TABLE 2. Specification of the Intel Skylake multicore CPU (HCLServer2).

Hardware Specifications	**Intel Haswell Server**
Processor	Intel E5-2670 v3 @2.30GHz
Micro-architecture	Haswell
Cores per socket	12
Socket(s)	2
L1d cache, L11 cache, L2 cache, L3 cache	32 KB, 32 KB, 256 KB, 30720 KB
Main memory	64 GB DDR4
TDP, Idle Power	240 W, 58 W
Software Specifications	
OS release	CentOS 7
Linux kernel	3.10
OpenMP version	3.1
Compiler	gcc 4.8.5
Likwid version	4.1
Intel MKL Version	2017.0.2

TABLE 3. Specification of the Intel Skylake multicore CPU (HCLServer2).

Hardware Specifications	**Intel Skylake Server**
Processor	Intel(R) Xeon(R) Gold 6152
Micro-architecture	Skylake
Socket(s)	1
Cores per socket	22
L1d cache, L11 cache, L2 cache, L3 cache	32 KB, 32 KB, 1024 KB, 30976 KB
Main memory	96 GB
TDP, Idle Power	140 W, 32 W
Software Specifications	
OS release	Ubuntu 16.04 LTS
Linux kernel	3.10
OpenMP version	3.1
Compiler	gcc 4.8.5
Likwid version	4.3.2
Intel MKL Version	2017.0.2

application execution is measured using a *WattsUp Pro* power meter and is obtained programmatically via the HCLWattsUp interface [51]. The power meter is periodically calibrated using an ANSI C12.20 revenue-grade power meter, Yokogawa WT210. The application programming interface for HCLWattsUp is further explained in Appendix F.

We divide our experimental study into four categories:

1) Study of additivity of PMCs.
2) Improving the accuracy of platform-level linear energy predictive models using the consistency test.
3) Exploration of the accuracy limits of application-specific linear energy predictive models.
4) A comparative study of optimization of a parallel matrix multiplication application for dynamic energy using two mainstream measurement tools and the most accurate linear energy predictive model constructed using the consistency test.

We start with the experimental methodology employed to ensure the reliability of our results.

A. EXPERIMENTAL METHODOLOGY

To ensure the reliability of our results, we follow a statistical methodology where a sample mean for a response variable 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 0.025 (2.5%) is 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 using Pearson's chi-squared test. The procedure to determine the sample mean using the Student's t-test is described in Appendix E.

The essential steps of the methodology are described below:

1) The server is fully reserved and dedicated to these experiments during their execution. We also made certain that there are no drastic fluctuations in the load due to abnormal events in the server by continuously monitoring its load for a week using the tool *sar*. Insignificant variation in the load was observed during this monitoring period suggesting normal and clean behaviour of the server.
2) HCLWattsUp API [51] gives the total energy consumption of the server during the execution of an application using system-level physical power measurements from the external power meters. The API is described in Appendix F. The total energy consumption includes the contribution of components such as NIC, SSDs, and fans. To ensure that the value of dynamic energy consumption is purely due to CPUs and DRAM, we verify that all the components other than CPUs and DRAM are idle using the following steps:
   - Monitoring the disk consumption before and during the application run. We ensure no I/O is performed by the application using tools such as sar and iotop.
   - Ensuring that the problem size used in the execution of an application does not exceed the main memory and that swapping (paging) does not occur.
   - Ensuring that the network is not used by the application using monitoring tools such as sar and atop.
   - Binding an application during its execution to resources using cores-pinning and memory-pinning.

TABLE 4. List of benchmarks in the application suite.

Application	Description
MKL FFT	Intel optimized 2-dimensional fast Fourier transform
MKL DGEMM	Intel optimized dense matrix multiplication of two square matrices
HPCG	Intel optimized High Performance Conjugate Gradient. 3-dimensional regular 27-point discretization of an elliptic partial differential equation
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 EP	Embarrassingly Parallel random number generator
NPB BT	Solve synthetic system of nonlinear partial differential equations using Block Tri-diagonal solver
NPB MG	Approximate 3-dimensional discrete Poisson equation using the V-cycle Multi Grid on a sequence of meshes
NPB FT	A 3D fast Fourier Transform partial differential equation benchmark
NPB DC	Arithmetic Data Cube, a data intensive grid benchmark representing data mining operations
NPB UA	Unstructured Adaptive mesh solving heat equation with convection and diffusion from moving ball
NPB CG	Solving an unstructured sparse linear system using Conjugate Gradient method
NPB SP	Solve synthetic system of nonlinear partial differential equations using Scalar Penta-diagonal solver
NPB DT	A graph benchmark evaluating communication throughput (Data Traffic)
stress	CPU, disk and I/O stress
Naive MM	Naive Matrix-matrix multiplication
Naive MV	Naive Matrix-vector multiplication

3) Our platform supports three modes to set the speeds of the fans: *minimum*, *optimal*, and *full*. The speeds of all the fans are set to *optimal* during the execution of the experiments. We make sure there is no contribution to the dynamic energy consumption from fans during an application run by following the steps below:
   - Continuously monitoring the temperature of the server components and the speed of fans, both when the server is idle, and during the application run. We obtain this information by using the Intelligent Platform Management Interface (IPMI) sensors.
   - We observed that both the temperature of the server and the speeds of the fans remained the same whether the given application is running or not.
   - We set the fans at *full* speed before starting the application run. This experiment's results were the same as when the fans were run at *optimal* speed.
   - To ensure that pipelining and cache effects do not happen, the experiments are not executed in a loop, and sufficient time (120 seconds) is allowed to elapse between successive runs. This time is based on observations of the times taken for the memory utilization to revert to base utilization and processor (core) frequencies to come back to the base frequencies.

B. STUDY OF ADDITIVITY OF PMCs

We employ the dual-socket multicore platform (HCLServer1) and seventeen optimized and unoptimized base applications (Table 4) to study the additivity of PMCs. Likwid [24] offers 164 PMCs on HCLServer1.

We compose a dataset of 60 compound applications from the seventeen base applications using different base application configurations. The additivity error of a PMC is determined as follows. For each compound application in the dataset, we obtain the PMC counts for the base applications, which are executed serially. Then, we run the compound application and record its PMC count. The error averaged over several runs for this compound application is calculated as follows:

$$Error(\%) = |\frac{(e_{b1} + e_{b2}) - e_c}{(e_{b1} + e_{b2} + e_c)/2}| \times 100 \quad (6)$$

where e_c, e_{b1}, e_{b2} are the PMC counts for the compound application and the constituent base applications respectively. The additivity error of the PMC is the maximum of errors for all the compound applications in the dataset.

For the dataset comprising all the compound applications, all PMCs have additivity errors exceeding 5%. 50 PMCs, 109 PMCs, and 15 PMCs have additivity errors exceeding 20%, 30%, and 100%, respectively. The highest additivity error is 3075% [50]. However, looking at application-specific additivity, many PMCs have additivity errors less than 5%. There are 38 such PMCs found for Intel MKL DGEMM by employing a dataset comprising compound applications constructed using different runtime configurations of Intel MKL DGEMM. Similarly, there are 18 such PMCs for Intel MKL FFT.

To identify the cause of this non-additivity, we study the additivity of PMCs for three applications employing

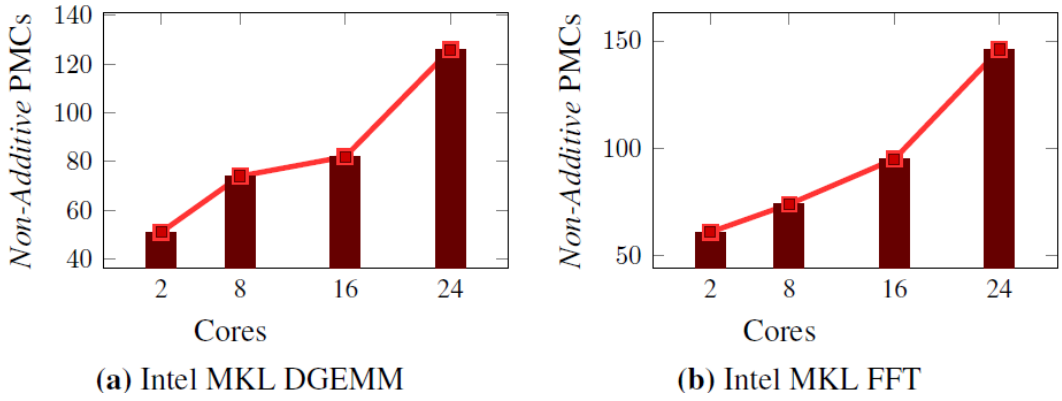


FIGURE 2. Increase in the number of non-additive PMCs with threads/cores used in an application. In the 2-core configuration, the application is pinned to one core of each socket. In the 8-core configuration, the application is pinned to four cores of each socket and so on. (a) and (b) shows non-additive PMCs (with additivity error above 5%) for Intel MKL DGEMM and Intel MKL FFT. For DGEMM, 51 PMCs are non-additive for 2-core configuration. The number increases to 126 for 24-core configuration. For FFT, the number increases from 61 to 146. The minimum number of non-additive PMCs is for the 2-core configuration for both applications.

different configurations of threads/cores. Figure 2 shows that the number of PMCs with additivity error above 5% increases as the number of cores increases. We attribute this to resource sharing and severe resource contention inherent in modern multicore platforms.

C. IMPROVING THE ACCURACY OF PLATFORM-LEVEL LINEAR ENERGY PREDICTIVE MODELS USING THE CONSISTENCY TEST

We employ the dual-socket multicore platform (HCLServer1) and seventeen optimized and unoptimized base applications (Table 4). We select six PMCs common to the state-of-the-art models [18], [27], [28], [33], [52], [53]. The PMCs ($\{X_1, \ldots, X_6\}$) and their additivity errors are shown in the Table 5. They belong to dominant energy-consuming PMC groups on our platform: cache, branch instructions, micro-operations (uops), floating-point instructions, and main memory accesses.

We build three types of linear regression models as follows:

- **Type1**: Models MA_1-MG_1 with no restrictions on intercepts and coefficients.
- **Type2**: Models MA_2-MG_2 whose intercepts are forced to zero.
- **Type3**: Models MA_3-MG_3 whose intercepts are forced to zero and whose coefficients cannot be negative.

Within each type t, MA_t employs all the PMCs as predictor variables. MB_t uses five PMCs with the least additive PMC (X_4) removed. MC_t uses four PMCs with two most non-additive PMCs (X_2, X_4) removed and so on until MF_t containing only the most additive PMC (X_6). MG_t uses three PMCs (X_4, X_5, X_6) with the highest correlation with dynamic energy consumption.

To train the models, we employ data points corresponding to 277 different configurations of base applications. 50 compound applications are used to test the models. This division of training and test datasets follows the best practices and experts' opinions in this domain. The results for Type1 and Type2 models are described in Appendix G-A. Both types of models violate the theory and therefore exhibit low prediction accuracy.

The Type3 models incorporate basic sanity checks that disallow violations of the theory. They have zero intercepts and are built using penalized linear regression that forces the coefficients to be non-negative. The prediction errors of these models are presented in Table 6. The average prediction accuracy of the models improves significantly as we remove highly non-additive PMCs one by one. ME_3 with the two most additive PMCs is the best in terms of average prediction accuracy. Therefore, we conclude that employing non-additive PMCs can significantly impair the prediction accuracy of models and that inclusion of highly additive PMCs improves the prediction accuracy of models drastically. The average prediction accuracy of RAPL is equal to that of MA_3 and MB_3, which contains the highest number of non-additive PMCs. MG_3 fares worse than RAPL and MA_3 even though it contains highly correlated PMCs with dynamic energy consumption. ME_3 with two most additive PMCs has better average prediction accuracy than MG_3, demonstrating that additivity is a more important criterion than correlation. Models employing one PMC exhibit the highest average prediction errors suggesting that a single PMC is unlikely to capture all energy consumption activities on a modern multicore platform.

Figure 3 presents the percentage deviations in predictions by Type3 models (Table 6) from the ground truth for different compound applications. RAPL, MA_3, and MG_3 exhibit higher average percentage deviations than the best model, ME_3. While RAPL distribution is normal, MA_3 and MG_3 demonstrate non-normality suggesting systemic (not fully random) deviations from the average.

D. ACCURACY LIMITS OF APPLICATION-SPECIFIC LINEAR ENERGY PREDICTIVE MODELS

We employ the single-socket Intel Skylake server (HCLServer2) and two applications, MKL FFT and MKL

TABLE 5. Correlation of PMCs with dynamic energy consumption (E_D). (a) PMCs and their additivity errors. (b) Correlation matrix of dynamic energy with PMCs. 100% correlation is denoted by 1. X_4, X_5, and X_6 are highly correlated with E_D.

Selected PMCs	*Additivity* 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

(a)

	E_D	X_1	X_2	X_3	X_4	X_5	X_6
E_D	1	0.53	0.50	0.42	0.58	0.99	0.99
X_1	0.53	1	0.41	0.25	0.39	0.45	0.44
X_2	0.50	0.41	1	0.19	0.99	0.48	0.48
X_3	0.42	0.25	0.19	1	0.21	0.41	0.40
X_4	0.58	0.39	0.99	0.21	1	0.57	0.56
X_5	0.99	0.45	0.48	0.41	0.57	1	0.99
X_6	0.99	0.44	0.48	0.40	0.56	0.99	1

(b)

TABLE 6. The minimum, average, and maximum prediction errors of IntelRAPL and the linear models. Linear models (MA_3-MG_3) have zero intercepts and positive regression coefficients.

Model	PMCs	Coefficients	Percentage prediction errors (min, avg, max)
MA_3	$X_1, X_2, X_3, X_4, X_5, X_6$	3.83E-09, 3.67E-10, 5.30E-07, 0, 5.56E-08, 0	(6.6, 31.2, 61.9)
MB_3	X_1, X_2, X_3, X_5, X_6	3.83E-09, 3.67E-10, 5.30E-07, 0, 5.56E-08	(6.6, 31.2, 61.9)
MC_3	X_1, X_3, X_5, X_6	3.75E-09, 5.34E-07, 5.58E-08, 0	(2.5, 25.3, 62.1)
MD_3	X_1, X_5, X_6	4.00E-09, 5.59E-08, 0	(2.5, 23.86, 100.3)
ME_3	X_1, X_6	4.60E-09, 1.46E-09	(2.5, 18.01, 89.45)
MF_3	X_6	1.60E-09	(2.5, 68.5, 90.5)
MG_3	X_4, X_5, X_6	1.72E-07, 5.86E-08, 0	(2.5, 50, 77.9)
IntelRAPL			(4.1, 30.6, 58.9)

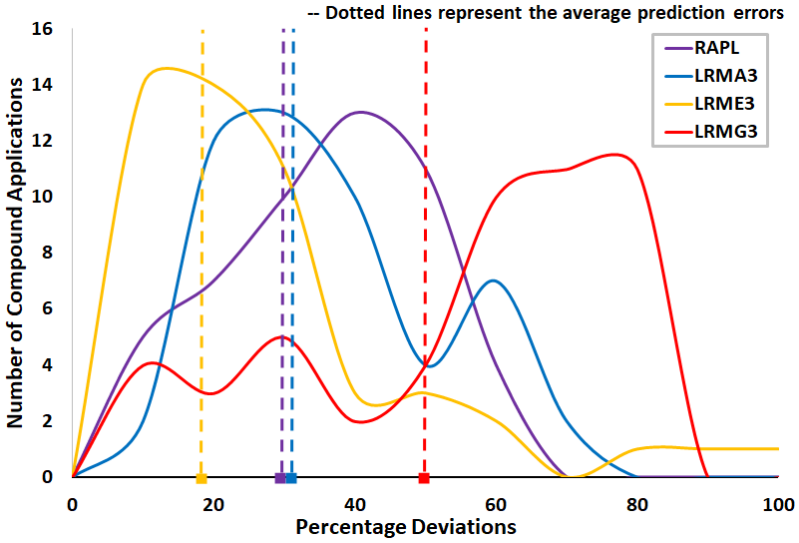


FIGURE 3. Percentage deviations of the Type3 models shown in Table 6 from the ground truth, which is system-level physical power measurements provided by power meters (HCLWattsUp). The dotted lines represent the averages.

DGEMM. Likwid [24] offers 323 PMCs on HCLServer2. The selection procedure of PMCs for the applications is described in Appendix G-B.

We select nine PMCs with additivity errors less than 1% in the set, PA9, and nine PMCs with additivity errors above 1% but are employed in state-of-the-art models in the

TABLE 7. Selected additive and non-additive PMCs and their correlations with dynamic energy consumption. Additive PMCs are denoted by PA9 = {A1, A2, . . ., A9}. Non-additive PMCs are denoted by PNA9 = {NA1, NA2, . . ., NA9}. 0 to 1 represents positive correlation of 0% to 100%.

	Additive PMCs (PA9)	Correlation
A1	FP_ARITH_INST_RETIRED_DOUBLE	0.993
A2	UOPS_EXECUTED_CORE	0.993
A3	IDQ_ALL_CYCLES_6_UOPS	0.993
A4	UOPS_RETIRED_CYCLES_GE_4_UOPS_EXEC	0.992
A5	IDQ_DSB_CYCLES_6_UOPS	0.981
A6	IDQ_ALL_DSB_CYCLES_5_UOPS	0.972
A7	MEM_INST_RETIRED_ALL_STORES	0.870
A8	UOPS_DISPATCHED_PORT_PORT_4	0.870
A9	MEM_LOAD_RETIRED_L3_MISS	-0.112
	Non-additive PMCs (PNA9)	
NA1	BR_MISP_RETIRED_ALL_BRANCHES	0.992
NA2	IDQ_MS_UOPS	0.99
NA3	ARITH_DIVIDER_COUNT	0.986
NA4	ICACHE_64B_IFTAG_MISS	0.960
NA5	L2_TRANS_CODE_RD	0.860
NA6	FRONTEND_RETIRED_L2_MISS	0.806
NA7	CPU_CLOCK_THREAD_UNHALTED	0.600
NA8	ITLB_MISSES_STLB_HIT	0.111
NA9	MEM_LOAD_L3_HIT_RETIRED_XSNP_MISS	-0.020

TABLE 8. Prediction accuracies of application-specific energy predictive models, (a) using nine PMCs and (b) using four highly correlated PMCs. DGEMM-PA, FFT-PA, and M-PA exhibit 1.3×, 2.5×, and 2.4× improvements in average prediction accuracy than DGEMM-PNA, FFT-PNA, and M-PNA, respectively. Models based on PNA4 containing highly correlated but non-additive PMCs do not demonstrate any improvement in average prediction accuracy than models PNA containing nine non-additive PMCs. DGEMM-PA4 has the least average prediction error of 16%.

Model	PMCs	Prediction Errors (%) [Min, Avg, Max]
DGEMM-PA9	PA9	(0.094, 22.62, 125.48)
DGEMM-PNA9	PNA9	(0.218, 31.23, 173.9)
FFT-PA9	PA9	(0.447, 36.31, 182.2)
FFT-PNA9	PNA9	(3.510, 92.68, 397.2)
M-PA9	PA9	(0.005, 35.32, 225.5)
M-PNA9	PNA9	(0.449, 85.61, 4039)

(a)

Model	PMCs	Prediction Errors (%) [Min, Avg, Max]
DGEMM-PA4	PA4	(0.004, 16.12, 87.25)
DGEMM-PNA4	PNA4	(0.091, 33.61, 212.3)
FFT-PA4	PA4	(0.042, 25.12, 190.15)
FFT-PNA4	PNA4	(5.12, 98.01, 450.2)
M-PA4	PA4	(0.024, 25.12, 87.25)
M-PNA4	PNA4	(0.449, 85.61, 4039)

(b)

set, PNA9. The PMCs and their correlations with dynamic energy consumption are shown in Table 7.

The models {DGEMM-PA9,FFT-PA9} are trained using PMCs belonging to PA9, and the models {DGEMM-PNA9,FFT-PNA9} are trained using PMCs belonging to PNA9. Two linear models, {M-PA9,M-PNA9}, employ PMCs belonging to PA9 and PNA9, respectively, but are trained and tested using the extended dataset combined from the datasets for both the applications.

Models based on PA9 exhibit better average prediction accuracy than the models based on PNA9 (Table 8a). The models, {DGEMM-PA9,FFT-PA9}, demonstrate better average prediction accuracy than the models, {M-PA9,M-PNA9}, suggesting that a specific set of carefully selected PMCs may yield a more accurate application-specific model.

1) IMPACT OF ADDITIVITY AND CORRELATION OF PMCs

Fast construction of accurate online energy predictive models is crucial for real-time systems that require quick reading of the application's energy consumption. Since only four PMCs can be collected in a single application run on our platform, selecting such a reliable subset is crucial to the prediction accuracy of online energy models.

We build two subsets of PMCs, PA4 and PNA4. PA4 contains the four most energy correlated PMCs from the set of nine highly additive PMCs, PA9. PNA4 contains four most energy correlated PMCs from the set, PNA9 (Table 7), respectively. The models {DGEMM-PA4,FFT-PA4,M-PA4} are trained using PMCs belonging to PA4, and the models {DGEMM-PNA4,FFT-PNA4,M-PNA4} are trained using PMCs belonging to PNA4.

TABLE 9. Prediction accuracy of models for DGEMM employing 5, 6, 7, and 8 PMCs that are most positively correlated with energy and highly additive.

Model	Prediction Errors (%) [Min, Avg, Max]
DGEMM-PA5	(0.94, 13.41, 119.43)
DGEMM-PA6	(0.32, 16.65, 123.16)
DGEMM-PA7	(0.17, 19.17, 142.32)
DGEMM-PA8	(0.03, 21.18, 126.83)

TABLE 10. Prediction accuracy of models for FFT employing 5, 6, 7, and 8 PMCs that are most positively correlated with energy and highly additive.

Model	Prediction Errors (%) [Min, Avg, Max]
FFT-PA5	(0.42, 22.41, 82.36)
FFT-PA6	(0.62, 19.21, 85.49)
FFT-PA7	(0.23, 27.31, 136.16)
FFT-PA8	(0.48, 29.62, 130.72)

The models based on PA4 demonstrate better average prediction accuracy than models based on PA (Table 8). Therefore, a high positive correlation of PMCs with dynamic energy consumption alone is insufficient to provide good average prediction accuracy. However, the PMCs must also satisfy the properties of the *consistency* test that considers the physical significance of the PMCs originating from the fundamental energy conservation law.

2) STUDY TO EXPLORE MOST ACCURATE APPLICATION-SPECIFIC MODELS

We find that all the application-specific models with less than four PMCs demonstrate a lower prediction accuracy than models using PA4. Therefore, we build four sets ({S1,S2,S3,S4}) of models comprising more than four PMCs for DGEMM and FFT. The PMCs are selected from the set of most additive PMCs (Table 7) and in the increasing order of their correlation with dynamic energy consumption. The sets of models are given below:

- S1 = {DGEMM-PA5,FFT-PA5} employ PMCs, {A1, . . . ,A5}.
- S2 = {DGEMM-PA6,FFT-PA6} employ PMCs, {A1, . . . ,A6}.
- S3 = {DGEMM-PA7,FFT-PA7} employ PMCs, {A1, . . . ,A7}.
- S4 = {DGEMM-PA8,FFT-PA8} employ PMCs, {A1, . . . ,A8}.

Tables 9 and 10 show the minimum, average, and maximum errors of the models. Figures 4a and 4b show the prediction error distributions. The average prediction errors are the least for the model with five PMCs for DGEMM (13.41%) and the model with six PMCs for FFT (19.21%).

Since the most accurate models for DGEMM and FFT employ five and six PMCs, at least six hardware registers must be dedicated to the PMCs so that the models can be employed online. Only 3-4 hardware registers are currently dedicated to PMCs during an application run on our experimental platforms.

E. OPTIMIZATION OF PARALLEL MATRIX MULTIPLICATION FOR DYNAMIC ENERGY

This section demonstrates that employing energy models constructed using the proposed theory for energy optimization can save a significant amount of energy compared to state-of-the-art energy measurement tools.

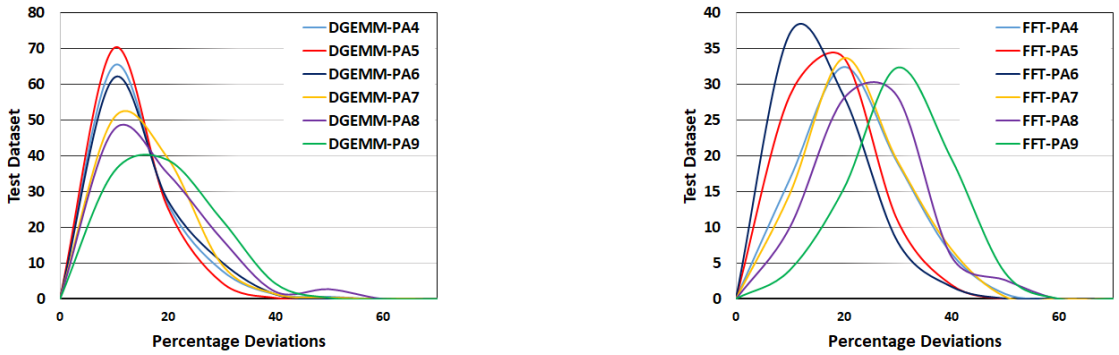
We study optimization of a parallel matrix multiplication application for dynamic energy using a), The most accurate PMC-based linear energy predictive model, LR MM, constructed based on the proposed theory, b). *Intel RAPL* [20], c). System-level physical power measurements using power meters (HCLWattsUp [51]), the ground truth. To ensure the reliability of our experimental results, we follow the experimental methodology described in the section IV-A.

The parallel application multiplies two dense square matrices A and B of size $N \times N$ and is executed on two processors, HCLServer1 and HCLServer2. The matrix A is partitioned using a model-based data partitioning algorithm between the processors as A_1 and A_2 of sizes $M \times N$ and $K \times N$ where $M + K = N$. The algorithm (detailed in the Appendix G-C) takes as input the number of rows of matrix A, N, and the discrete energy functions of the processors, $e_1(x, y)$ and $e_2(x, y)$, where $e_i(x, y)$ represents the dynamic energy consumption of multiplication of matrices of sizes $x \times y$ and $y \times y$. The outputs from the algorithm are M and K. Informally, the algorithm cuts the surfaces of the functions by a plane $y = N$ to produce two curves, and then it determines two points on the curves, $(M, e_1(M, N))$ and $(K, e_2(K, N))$, such that the sum of their energy consumptions, $e_1(M, N) + e_2(K, N)$, is minimal.

Matrix B is replicated at both the processors. HCLServer1 computes the product of matrices A_1 and B, and processor HCLServer2 computes the product of matrices A_2 and B. The local matrix products are computed using Intel MKL DGEMM routine. There are no communications involved.

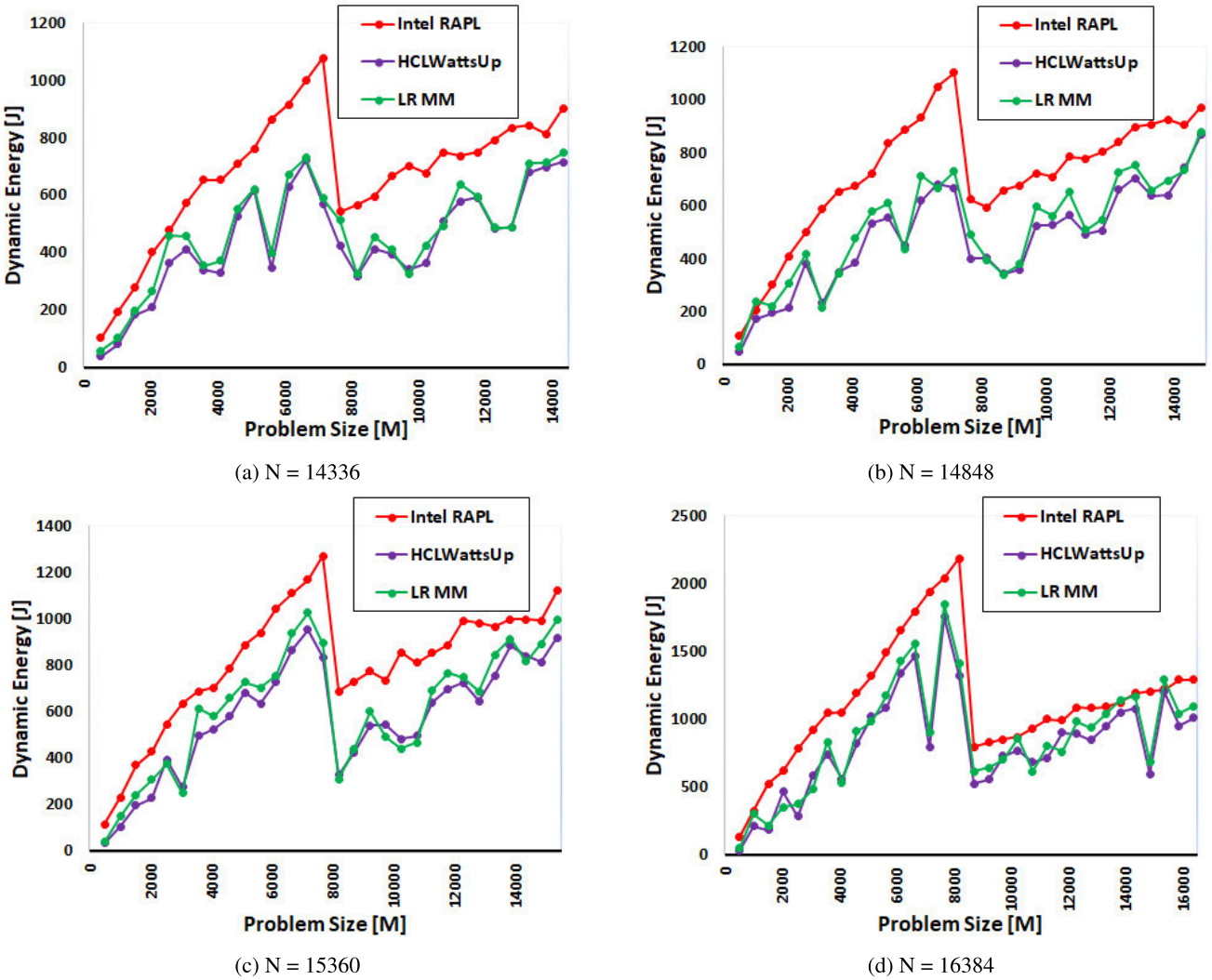
Figure 5 illustrates the dynamic energy profiles for four workload sizes (N). Given the workload size and the corresponding dynamic energy profiles of the two processors, the workload distribution is determined using the data partitioning algorithm. Then, the dynamic energy consumption is obtained by executing the parallel application using the workload distribution. The HCLWattsUp profile is the most accurate, its use yields the minimal energy consumption. The percentage of losses of dynamic energy incurred by employing LR MM and IntelRAPL against HCLWattsUp for the four workload sizes are {1, 3, 10, 16} and {54, 37, 31, 84}, respectively. Therefore, the energy models constructed using the proposed theory save a significant amount of energy (up to 80%) compared to the state-of-the-art energy measurement tools.

RAPL is shown to exhibit good prediction accuracy for applications employing decision variables such as dynamic voltage and frequency scaling (DVFS) [54] and the num-



(a) (b)

FIGURE 4. Percentage deviations of the application-specific models shown in Table 8, 9 and 10 from the system-level physical power measurements provided by power meters (HCLWattsUp) for (a). DGEMM and (b). FFT.



(a) N = 14336 (b) N = 14848

(c) N = 15360 (d) N = 16384

FIGURE 5. Dynamic energy profiles of Intel MKL DGEMM application executed on HCLServer1 and HCLServer2. Here M represents the range for workload sizes for matrix A partitioned on both processors i.e. $M = A_1 + A_2$ where A_1 is executed on HCLServer1 and A_2 on HCLServer2.

ber of application-level threads [49] but keeping the workload size fixed. However, Fahad *et al.* [11] demonstrate that RAPL shows poor correlation with real measurements if the workload size is varied and all the other parameters are fixed. We validate this finding here but, most importantly, show that employing RAPL in energy optimization methods where the

decision variable is workload distribution leads to significant energy losses.

V. CONCLUSION

Energy efficiency in Information Communications Technology is becoming a grand technological challenge and is now a first-class design constraint along with performance in all computing settings. The mainstream approach to improving energy efficiency in computing is hardware-level or system-level, driven mainly by hardware innovations in manufacturing energy-efficient devices. The other approach, comparatively understudied, consists of solution methods that optimize applications rather than the executing environment.

We focused on the application-level approach in this work. Accurate measurement of energy consumption during an application execution is key to energy minimization at the application level. Energy predictive modelling based on performance monitoring counters is now the leading method for predicting energy consumption during application execution. A vast majority of research works propose models that select performance monitoring counters solely based on their high positive correlation with energy consumption and report prediction accuracies ranging from poor to excellent. Therefore, a sound theoretical framework to understand the fundamental significance of the model variables to energy consumption and the causes of the inaccuracy of the models is lacking. We bridged the gap in this work.

In this work, we summarized the assumptions behind the existing models and used a model-theoretic approach to formulate their assumed properties in a mathematical form. We extended the formalism by adding properties, heretofore unconsidered, that are basic implications of the universal energy conservation law. The extended formalism forms our theory of energy predictive models of computing. An energy predictive model satisfying all the properties of the extended model is termed a *consistent* energy model. Using the theory, we proved that a consistent energy predictive model is linear if and only if each PMC variable is *additive* in the sense that the PMC for a serial execution of two applications is the sum of PMCs for the individual execution of each application. The theory's basic practical implications for improving the prediction accuracy of linear energy predictive models are unified in a *consistency* test, which contains a suite of properties that include reproducibility and additivity to select model variables and constraints for model coefficients.

We studied the additivity of performance monitoring counters on a modern Intel platform. We showed that a PMC can be non-additive with an error as high as 3075%, and there are performance monitoring counters where the error is over 100%. We discovered that the number of non-additive performance monitoring counters rises with an increase in the number of cores employed in the application. We consider this an inherent trait of modern multicore computing platforms because of severe resource contention and non-uniform memory access.

We applied the practical implications of our theory to improve the prediction accuracy of the state-of-the-art energy predictive models. We demonstrated how the accuracy of energy predictive models built using linear regression could be improved by selecting performance monitoring counters based on a property of additivity. A high positive correlation of the model variables with dynamic energy consumption alone is insufficient to provide good prediction accuracy for a model. However, the model variables must also satisfy the consistency test's properties that take into account the physical significance of the model variables originating from the conservation of energy of computing, which is the manifestation of the fundamental physical law of energy conservation.

We explored the construction of most accurate application-specific models on our platform. The results show that the most accurate models employ five and six performance monitoring counters as predictor variables. Therefore, at least six hardware registers must be dedicated to storing the performance monitoring counters so that the models can be employed online.

Finally, we studied the optimization of a parallel matrix-matrix multiplication application for dynamic energy using three measurement tools: a). Intel Running Average Power Limit, which is a popular mainstream tool, b). The most accurate linear model constructed using the practical implications of the theory, and c). System-level physical measurements using power meters, which is the ground truth. We demonstrated that employing energy models constructed using the proposed theory for energy optimization can save a significant amount of energy (up to 80% for applications used in experiments) compared to state-of-the-art energy measurement tools.

APPENDIX A
SUPPLEMENTAL MATERIAL

The supporting materials for the main manuscript are:

- Overview of system-level and application-level techniques for energy optimization.
- Comparative analysis of the state-of-the-art energy measurement methods.
- Procedure employed to determine the sample mean using Student's t-test.
- HCLWattsUp [51] API for energy measurements using external power meters.
- Details of experimental results.
  - Improving the accuracy of platform-Level linear energy predictive models Using the consistency test.
  - Accuracy limits of application-specific linear energy predictive models.
  - Optimization of parallel matrix multiplication for dynamic energy.

APPENDIX B
OVERVIEW OF SYSTEM-LEVEL AND APPLICATION-LEVEL TECHNIQUES FOR ENERGY OPTIMIZATION

A. OVERVIEW OF SYSTEM-LEVEL TECHNIQUES

The mainstream approach to improving energy efficiency in computing is hardware-level or system-level, driven mainly by hardware innovations in manufacturing energy-efficient devices. Energy-efficient hardware design techniques such as the clock and power gating, dynamic voltage and frequency scaling (DVFS), and dynamic power management (DPM) are now built into modern processors. Briefly, there are two types of power consumptions in a component executing an application: dynamic power and static power. Dynamic power consumption is caused by the switching activity in the component's circuits. Static power or idle power is the power consumed when the component is not active or doing work. Clock gating reduces dynamic power dissipation by disabling portions of the circuitry so that the flip-flops in them do not switch states. Power gating reduces static power dissipation by switching off inactive portions of circuitry. DPM allows electronic components to be turned off or moved to a low power state when idle to reduce energy consumption. DVFS reduces the dynamic power consumed by a processor by throttling its clock frequency. Modern processors allow programmers to set different clock frequencies to obtain performance/energy trade-offs but it requires deep understanding of their applications.

System-level methods aim to optimize the energy and performance of the environment where the applications are executed. The methods employ application-agnostic models and hardware parameters as decision variables. They are principally deployed at the operating system (OS) level and therefore require changes to the OS. They do not involve any changes to the application. The methods can be further divided into the following prominent groups: I.

1) Thread schedulers that are contention-aware and that exploit cooperative data sharing between threads (Petrucci *et al.* [7], Kim *et al.* [55]). The goal of a scheduler is to find thread-to-core mappings to determine Pareto-optimal solutions for energy and performance.
2) Dynamic private cache (L1 and L2) reconfiguration and shared cache (L3) partitioning strategies (Wang *et al.* [56], Cheng *et al.* [57]). The proposed solutions in this category mitigate contention for shared on-chip resources such as last level cache by physically partitioning it and therefore require substantial changes to the hardware or OS. Zhuravlev *et al.* [6] present a survey of these strategies on multicore processors.
3) Thermal management algorithms that place or migrate threads to not only alleviate thermal hotspots and temperature variations in a chip but also reduce energy consumption during an application execution (Yang *et al.* [58], Ayoub *et al.* [5]).
4) Asymmetry-aware schedulers that exploit the asymmetry between sets of cores in a multicore platform to find thread-to-core mappings that provide Pareto-optimal solutions for energy and performance (Li *et al.* [59], Humenay *et al.* [60]).

B. OVERVIEW OF APPLICATION-LEVEL TECHNIQUES

The second approach consists of solution methods that optimize applications rather than the operating environment. The methods use application-level decision variables and predictive models for the performance and energy consumption of applications. The dominant decision variables include the number of threads, loop tile size, and workload distribution. The application-level approach is comparatively understudied. However, recent breakthroughs in application-level optimization methods address the challenges posed by inherent complexities in modern multicore CPUs such as 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 last level cache (LLC), interconnect (For example, Intel's Quick Path Interconnect, AMD's Hyper Transport), 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 (DPM) of multiple power domains (CPU sockets, DRAM).

Lastovetsky and Reddy [8], Reddy and Lastovetsky [61] propose data partitioning algorithms that solve single-objective optimization problems of data-parallel applications for performance or energy on homogeneous clusters of multicore CPUs. They take as an input, discrete performance and dynamic energy functions with no shape assumptions that accurately and realistically account for resource contention and NUMA inherent in modern multicore CPU platforms. Reddy and Lastovetsky [62] propose a solution method to solve bi-objective optimization problem of an application for performance and energy on homogeneous clusters of modern multicore CPUs. They demonstrate that the method gives a diverse set of Pareto-optimal solutions and that it can be combined with DVFS-based multi-objective optimization methods to give a better set of (Pareto-optimal) solutions. The methods target homogeneous high-performance computing (HPC) platforms.

Modern HPC platforms, cloud computing systems, and data centers are highly heterogeneous, comprising nodes where a multicore processor is tightly integrated with one or more accelerators to address the twin critical concerns of performance and energy efficiency. A crucial requirement to determine the energy-optimal configuration of a parallel/hybrid application executing on such platforms is to determine component-level dynamic energy profiles of the application executing on multiple independent devices of the platform. Fahad *et al.* [63] present the first methodology to measure the component-level energy consumption of a hybrid application on a heterogeneous computing platform. Chakraborti *et al.* [9] consider the effect of heterogeneous

workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters. Khaleghzadeh *et al.* [10] propose a solution method solving the bi-objective optimization problem on heterogeneous processors and comprising of two principal components. The solution method employs the methodology [63] to determine the fine-grained component-level decomposition of an application's energy consumption.

APPENDIX C
COMPARATIVE ANALYSIS OF THE STATE-OF-THE-ART ENERGY MEASUREMENT METHODS

We categorize the dominant approaches for determining the energy consumption by an application as follows: a). System-level physical measurements using external power meters, b). Measurements using on-chip power sensors, and c). Energy predictive models.

Before we present an overview of the advantages and drawbacks of the three approaches, we present terminology related to energy to aid the exposition.

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 given by formula below:

$$E_D = E_T - (P_S \times T_E) \tag{7}$$

Using the system-level physical power measurements provided by external power meters in the first approach, the dynamic energy consumption during an application execution is determined by applying the formula (7). The total energy consumption E_T is the area under the discrete function of the power samples provided by the power meter versus the time intervals between the samples. Well-known numerical approaches such as trapezoidal rule can be used to calculate this area approximately. The trapezoidal rule works by approximating the area under a function using trapezoids rather than rectangles to get better approximations. The execution time T_E of the application execution can be determined accurately using the processor clocks. The accuracy of obtaining the total energy consumption E_T and the static power consumption P_S is equal to the accuracy provided in the specification of the power meter. Therefore, we consider this approach to be the ground truth.

However, the first approach provides the physical power measurement at a computer level only and therefore lacks the ability to furnish fine-grained decomposition of the energy consumption of an application executing on several independent computing devices in a computer. This component-level decomposition of energy consumption is significant in order to optimize the application by distributing the workload. A naïve approach to optimize the application for dynamic energy consumption will have exponential complexity. The approach must explore all possible workload distributions. For each workload distribution, it determines the total dynamic energy consumption during the parallel execution of the workload by applying the formula (7). It, then, returns the workload distribution with the minimum total dynamic energy consumption.

The second approach is based on on-chip power sensors now available in mainstream processors such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Intel Xeon Phis. There are vendor specific libraries to acquire the power data from these sensors. For example, Running Average Power Limit (RAPL) [20] can be used to monitor power and control frequency (and voltage) of Intel CPUs, and Nvidia Management Library (NVML) [64] and Intel System Management Controller chip (SMC) [65] provide the power consumption by Nvidia GPUs and Intel Xeon Phi respectively. The dynamic energy consumption during an application execution on a compute device equipped with on-chip sensors is also calculated using the Formula (7). The execution time T_E of the application execution can be determined accurately using the timers provided in the compute device. The base power consumption P_S is obtained using the on-chip sensors when the component is idle. The total energy consumption E_T is calculated from the power samples using the trapezoidal rule.

While the accuracy of Nvidia GPU on-chip sensors is reported in the NVML manual ($\pm 5\%$) [64], the accuracies of the other sensors are not known. For the GPU and Xeon Phi on-chip sensors, there is no information about how a power reading is determined that would allow one to determine its accuracy. For the CPU on-chip sensors, RAPL uses separate voltage regulators (VR IMON) for both CPU and DRAM. VR IMON is an analog circuit within voltage regulator (VR), which keeps track of an estimate of the current [66]. RAPL is shown to exhibit good prediction accuracy for applications employing decision variables such as dynamic voltage and frequency scaling (DVFS) [54] and the number of application-level threads [49] but keeping the workload size fixed. However, Fahad *et al.* [11] demonstrate that RAPL shows poor correlation with real measurements if the workload size is varied and all the other parameters are fixed. They present the first comprehensive comparative study of the accuracy of state-of-the-art on-chip power sensors against system-level physical power measurements using external power meters (the ground truth). They show that energy measurements provided by the state-of-the-art on-chip sensors significantly deviate from the ground truth. Moreover, owing to the nature of the deviations, calibration can not improve the accuracy of the on-chip sensors to the extent that could favour their use in optimization of applications for dynamic energy. We define calibration as a constant adjustment (positive or negative value) made to the data points in

a dynamic energy profile of an application obtained using a measurement approach (on-chip sensors or energy predictive models) with the aim to reduce its error against the ground truth.

The third approach based on software energy predictive models emerged as a popular alternative to determine the energy consumption of an application. A vast majority of such models are linear and employ performance monitoring counters (PMCs) as predictor variables. PMCs are special-purpose registers provided in modern microprocessors to store the counts of software and hardware activities. A pervasive approach is to determine the energy consumption of a hardware component based on linear regression of the PMC counts in the component during an application run. The total energy consumption is then calculated as the sum of these individual consumptions. While the models allow determination of fine-grained decomposition of energy consumption during the execution of an application, there are research works highlighting their poor accuracy [11]–[13], [19], [50]. Shahid *et al.* [50] propose a novel property of PMCs called additivity, which is true for PMCs, whose value for a serial execution of two applications is equal to the sum of values for the individual execution of each application, and study the additivity of PMCs offered by the popular state-of-the-art tools, Likwid [24] and PAPI [22] on a modern Intel Haswell multicore server CPU. They show that many PMCs in Likwid and PAPI that are widely used in models as key predictor variables are non-additive. Fahad *et al.* [11] study the accuracy of platform-level and application-level energy predictive models employing linear regression. The models employ PMCs that have high positive correlation with dynamic energy consumption. They show that the average error between the energy predictive models and the ground truth ranges from 14% to 32% and the maximum reaches 100%.

APPENDIX D
RATIONALE BEHIND USING DYNAMIC ENERGY CONSUMPTION INSTEAD OF TOTAL ENERGY CONSUMPTION

We consider only dynamic energy consumption modelling in our work for reasons below:

1) Static energy consumption, a major concern in embedded systems, is becoming less compared to the dynamic energy consumption due to advancements in hardware architecture design in HPC systems.
2) We target applications and platforms where dynamic energy consumption is the dominating energy dissipator.
3) Finally, we believe its inclusion can underestimate the true worth of an optimization technique that minimizes the dynamic energy consumption. We elucidate using two examples from published results.
   - In our first example, consider a model that reports predicted and measured the total energy consumption of a system to be 16500J and 18000J. It would report the prediction error to be 8.3%. If it is known that the static energy consumption of the system is 9000J, then the actual prediction error (based on dynamic energy consumption only) would be 16.6% instead.
   - In our second example, consider two different energy prediction models (M_A and M_B) with the same prediction errors of 5% for application execution on two different machines (A and B) with same total energy consumption of 10000J. One would consider both the models to be equally accurate. But supposing it is known that the dynamic energy proportions for the machines are 30% and 60%. Now, the true prediction errors (using dynamic energy consumption only) for the models would be 16.6% and 8.3%. Therefore, the second model M_B should be considered more accurate than the first.

APPENDIX E
PROCEDURE FOR DETERMINING THE SAMPLE MEAN USING STUDENT's T-TesT

The function, *MeanUsingTtest*, shown in Solution Method 1, determines the sample mean for a data point. For each data point, the function repeatedly executes the application *app* until one of the following three conditions is satisfied:

1) The maximum number of repetitions (*maxReps*) is exceeded (Line 3).
2) The sample mean falls in the confidence interval (or the precision of measurement *eps* is achieved) (Lines 13-15).
3) The elapsed time of the repetitions of application execution has exceeded the maximum time allowed (*maxT* in seconds) (Lines 16-18).

So, for each data point, the function *MeanUsingTtest* returns the sample mean *mean*. The function *Measure* measures the execution time using *gettimeofday* function. In our experiments, we set the minimum and maximum number of repetitions, *minReps* and *maxReps*, to 15 and 100000. The values of *maxT*, *cl*, and *eps* are 3600, 0.95, and 0.025. If the precision of measurement is not achieved before the completion of maximum number of repeats, we increase the number of repetitions and also the allowed maximum elapsed time. Therefore, we make sure that statistical confidence is achieved for all the data points that we use in our experiments.

APPENDIX F
APPLICATION PROGRAMMING INTERFACE (API) FOR ENERGY MEASUREMENTS USING EXTERNAL POWER METER INTERFACES

Each research server platform has a dedicated power meter installed between their input power sockets and wall A/C outlets. The power meter captures the total power consumption of the node. It has a data cable connected to the USB port of the node. A perl script collects the data from the power meter

Algorithm 1 Function Determining the Mean of an Experimental Run Using Student's t-Test

1: **procedure** MeanUsingTtest(*app*, *minReps*, *maxReps*, *maxT*, *cl*, *accuracy*, *repsOut*, *clOut*, *etimeOut*, *epsOut*, *mean*)

Input:
The application to execute, *app*
The minimum number of repetitions, $minReps \in \mathbb{Z}_{>0}$
The maximum number of repetitions, $maxReps \in \mathbb{Z}_{>0}$
The maximum time allowed for the application to run, $maxT \in \mathbb{R}_{>0}$
The required confidence level, $cl \in \mathbb{R}_{>0}$
The required accuracy, $eps \in \mathbb{R}_{>0}$

Output:
The number of experimental runs actually made, $repsOut \in \mathbb{Z}_{>0}$
The confidence level achieved, $clOut \in \mathbb{R}_{>0}$
The accuracy achieved, $epsOut \in \mathbb{R}_{>0}$
The elapsed time, $etimeOut \in \mathbb{R}_{>0}$
The mean, $mean \in \mathbb{R}_{>0}$

2: $reps \leftarrow 0$; $stop \leftarrow 0$; $sum \leftarrow 0$; $etime \leftarrow 0$
3: **while** ($reps < maxReps$) and (!$stop$) **do**
4: $st \leftarrow measure(TIME)$
5: $Execute(app)$
6: $et \leftarrow measure(TIME)$
7: $reps \leftarrow reps + 1$
8: $etime \leftarrow etime + et - st$
9: $ObjArray[reps] \leftarrow et - st$
10: $sum \leftarrow sum + ObjArray[reps]$
11: **if** $reps > minReps$ **then**
12: $clOut \leftarrow$ fabs(gsl_cdf_tdist_Pinv(cl, $reps - 1$)) $\times$ gsl_stats_sd($ObjArray$, 1, $reps$) / sqrt($reps$)
13: **if** $clOut \times \frac{reps}{sum} < eps$ **then**
14: $stop \leftarrow 1$
15: **end if**
16: **if** $etime > maxT$ **then**
17: $stop \leftarrow 1$
18: **end if**
19: **end if**
20: **end while**
21: $repsOut \leftarrow reps$; $epsOut \leftarrow clOut \times \frac{reps}{sum}$
22: $etimeOut \leftarrow etime$; $mean \leftarrow \frac{sum}{reps}$
23: **end procedure**

using the serial USB interface. The execution of this script is non-intrusive and consumes insignifcant power.

We use HCLWattsUp API functions to gather the readings from the power meters to determine the average power and energy consumption during the execution of an application on a given platform. The functions provide following four types of measures during the execution of an application:

- *TIME*—The execution time (seconds).
- *DPOWER*—The average dynamic power (watts).
- *TENERGY*—The total energy consumption (joules).
- *DENERGY*—The dynamic energy consumption (joules).

We confirm that the overhead due to the API is very minimal and does not have any noticeable influence on the main measurements. The power meter readings are only processed if the measure is not *hcl* :: *TIME*. Therefore, for each measurement, we have two runs. One run for measuring the execution time. And the other for energy consumption. The main program in the Figure 6 illustrates the use of statistical methods to measure the dynamic energy consumption during the execution of an application.

```
1  #include <wattsup.hpp>
2  int main(int argc, char** argv) {
3      std::string pathsToMeters = "/opt/powertools/bin/
       wattsup1";
4      std::string argsToMeters = "--interval=1";
5      hcl::Wattsup wattsup(1, pathsToMeters, argsToMeters);
6      hcl::Precision pIn = { maxRepeats, cl, maxElapsedTime
       , maxStdError };
7      hcl::Precision pOut;
8      double sampleMean, sd;
9      int rc = wattsup.execute(hcl::DENERGY, executablePath
       , executableArgs,
10                              &pIn, &pOut, &sampleMean, &
       sd);
11     if (rc == 0)
12        std::cerr << "Precision NOT achieved.\n";
13     else
14        std::cout << "Precision achieved.\n";
15     std::cout << "Max repetitions " << pOut.reps_max <<
         ", Elasped time "
16               << pOut.time_max_rep << ", Relative error "
         << pOut.eps
17               << ", Mean energy " << sampleMean << ",
       Standard Deviation "
18               << sd << std::endl;
19     exit(EXIT_SUCCESS);
20 }
```

FIGURE 6. Main program illustrating the use of HCLWattsUp API functions for measuring the dynamic energy consumption.

The API is confined in the *hcl* namespace. Lines 10–12 construct the Wattsup object. The inputs to the constructor are the paths to the scripts and their arguments that read the USB serial devices containing the readings of the power meters.

The principal method of *Wattsup* class is *execute*. The inputs to this method are the type of measure; the path to the executable, *executablePath*; the arguments to the executable, *executableArgs*; the statistical thresholds, *pIn*). The outputs are the achieved statistical confidence *pOut*; the estimators, the sample mean (*sampleMean*) and the standard deviation (*sd*), calculated during the execution of the executable.

The *execute* method repeatedly invokes the executable until one of the following conditions is satisfied:

- The maximum number of repetitions specified in *maxRepeats* is exceeded.
- The sample mean is within *maxStdError* percent of the confidence interval *cl*. The confidence interval of the mean is estimated using Student's t-distribution.
- The maximum allowed time *maxElapsedTime* specified in seconds has elapsed.

TABLE 11. Linear predictive models (MA_1-MG_1) with intercepts and their minimum, average, and maximum prediction errors. Coefficients can be positive or negative.

Model	PMCs	Intercept followed by Coefficients	Percentage prediction errors (min, avg, max)
MA_1	$X_1, X_2, X_3, X_4, X_5, X_6$	10.2, 3.06E-09, 1.95E-08, 3.30E-07, -1.02E-06, 6.18E-08, -9.39E-11	(2.7, 32, 99.9)
MB_1	X_1, X_2, X_3, X_5, X_6	12.8, 3.68E-09, 2.26E-10, 3.43E-07, 7.40E-08, -4.763E-10	(2.5, 23.32, 80.42)
MC_1	X_1, X_3, X_5, X_6	16.4, 3.71E-09, 3.34E-07, 7.45E-08, -4.87E-10	(2.5, 21.86, 76.9)
MD_1	X_1, X_5, X_6	29.9, 3.72E-09, 7.54E-08, -5.076E-10	(2.5, 21.78, 77.33)
ME_1	X_1, X_6	130, 4.21E-09, 1.456E-09	(2.5, 18.01, 89.23)
MF_1	X_6	749, 1.53E-09	(2.5, 14.39, 34.64)
MG_1	X_4, X_5, X_6	492, 6.79E-08, 9.45E-08, -9.60E-10	(2.5, 23.46, 80)

TABLE 12. Linear predictive models (MA_2-MG_2) with zero intercepts and their minimum, average, and maximum prediction errors. Coefficients can be positive or negative.

Model	PMCs	Coefficients	Percentage prediction errors (min, avg, max)
MA_2	$X_1, X_2, X_3, X_4, X_5, X_6$	1.08E-09, 1.96E-08, 3.51E-07, -1.02E-06, 6.19E-08, -9.78E-11	(2.5, 32, 78.7)
MB_2	X_1, X_2, X_3, X_5, X_6	3.71E-09, 2.37E-10, 3.69E-07, 7.42E-08, -4.82E-10	(2.5, 23.32, 80.57)
MC_2	X_1, X_3, X_5, X_6	3.75E-09, 3.66E-07, 7.48E-08, -4.95E-10	(2.5, 22.1, 77.5)
MD_2	X_1, X_5, X_6	3.80E-09, 7.61E-08, -5.27E-10	(2.5, 22.4, 78.5)
ME_2	X_1, X_6	4.60E-09, 1.46E-09	(2.5, 18.01, 89.45)
MF_2	X_6	1.60E-09	(3.0, 68.53, 90.53)
MG_2	X_4, X_5, X_6	1.34E-07, 1.22E-07, -1.65E-09	(2.5, 47.5, 111.22)

If anyone of the conditions is not satisfied, then a return code of 0 is output, suggesting that statistical confidence is not achieved. If statistical confidence is achieved, then the number of repetitions performed, the elapsed time, and the final relative standard error are returned in the output argument *pOut*. At the same time, the sample mean and standard deviation are returned. For our experiments, we use values of (1000, 95%, 2.5%, 3600) for the parameters (maxRepeats, cl, maxStdError, maxElapsedTime) respectively. Since we use Student's t-distribution to calculate the confidence interval of the mean, we confirm specifically that the observations follow normal distribution by plotting the density of the observations using the *R* programming language.

APPENDIX G
EXPERIMENTAL RESULTS

A. IMPROVING THE ACCURACY OF PLATFORM-LEVEL LINEAR ENERGY PREDICTIVE MODELS USING THE CONSISTENCY TEST

Table 11 summarizes the type 1 models. Following are the salient observations:

- The model intercepts are significant. In our proposed theory where we consider modelling of dynamic energy consumption, the intercepts are not present since they have no real physical meaning. Consider the case where no application is executed. The values of the PMCs will be zero and therefore the models must output the dynamic energy consumption to be zero. The models, however, output the values of their intercepts as the dynamic energy consumption. This violates the energy conservation property in the theory.
- MA_1 has negative coefficients for PMCs, X_4 and X_6. Models MB_1-MD_1 have negative coefficients for PMC, X_6. The negative coefficients in these models can give rise to negative predictions for applications where the counts for X_4 and X_6 are higher than the other PMCs. We illustrate this case by designing a microbenchmark that stresses specifically hardware components resulting in large counts for the PMCs with the negative coefficients. Since, in our case, X_4 and X_6 count the division and floating-point instructions, our microbenchmark is a simple *assembly* language program that performs floating-point division operations in a loop. When run for forty seconds, the PMC counts for this application on our platform were: X_1=7022011, X_2=623142, X_3=121489, X_4=5101219180, X_5=33210, and X_6=186971207082. The energy consumption predictions for this application from our four models $\{A_1, B_1, C_1, D_1\}$ are $\{-5210.52, -76.23, -74.59, -64.98\}$ which violate the energy conservation law.
- Since the predictor variables have a high positive correlation with energy consumption, their coefficients should exhibit the same relationship. The coefficients, however, have different signs for different models. Consider, for example, X_4 in MA_1 and MC_1. While it has a positive coefficient of A_1, it has a negative coefficient of MC_1. Similarly, X_6 in A_1 and B_1 has negative coefficient, whereas in MF_1 it has a positive coefficient. We have found that the research works that propose linear models using these PMCs do not contain any sanity check for these coefficients. Therefore, we believe that using them in models without understanding the true meaning or the nature of their relationship with dynamic energy consumption can lead to serious inaccuracy.

The type 2 models are built using specialized linear regression, which forces the intercept to be zero. Table 12 contains their summary. All the models excepting ME_2 and MF_2 contain negative coefficients and therefore present the same issues that violate the energy conservation properties of the theory.

B. ACCURACY LIMITS OF APPLICATION-SPECIFIC LINEAR ENERGY PREDICTIVE MODELS

We choose the single-socket Intel Skylake server (Table 3) for the experiments. A total of 323 PMCs are exposed using Likwid tool on this platform. In order to collect all the PMCs, an application has to be run 99 times.

There is no PMC which is additive within a tolerance of 5% if we consider the large dataset of compound applications composed for the test suite of applications in Table 4. However, we discover that many PMCs are additive (within 5%) for a small subset of applications. We select two such applications, MKL FFT and MKL DGEMM. We now describe below the process of selection of PMCs for the applications:

- From the literature and based on the nature of the applications, we find that the PMCs that mainly reflect the hardware and software activities and that contribute towards the dynamic energy consumption are from the following dominant PMC groups: cache, branch instructions, micro-operations (μops), floating-point instructions, and main memory accesses. We call the PMCs in these groups as *prime PMCs*. For our platform (Table 3), prime PMCs are 122.
- We make sure that all the prime PMCs are reproducible by executing the FFT and DGEMM applications with the same problem sizes for a number of times. We find that the PMC counts for successive runs of applications are within an accuracy of 99.99%.
- We study the additivity of the prime PMCs. We build a dataset of 50 *base* applications using different problem sizes for DGEMM and FFT. The range of problem sizes for DGEMM is 6500 × 6500 to 20000 × 20000, and for FFT is 22400 × 22400 to 29000 × 29000. We select this range because of reasonable execution time (> 3 seconds) of the applications on our platform. For each application in a dataset, we measure the following: PMCs, dynamic energy consumption, and execution time. We then build a dataset of 30 *compound* applications from these *base* applications. The additivity test based on the two datasets reveals that there are a number of PMCs that are additive for both the applications.
- We select nine highly additive PMCs (with additivity test errors of less than 1%) for both the applications. These are labeled as $A1, A2, A3, \ldots, A9$.
- We then select nine non-additive PMCs but which have been employed as predictor variables in notable energy predictive models in the literature. We label them, $NA1, NA2, NA3, \ldots, NA9$. The set of additive PMCs is denoted by *PA9* and non-additive PMCs by *PNA9*.
- Finally, we determine the correlations of the PMCs with dynamic energy consumption.

We build a dataset containing 401 workload sizes for DGEMM ranging from 6400 × 6400 to 38400 × 38400. The dataset is further divided for two subsets, one for training and the other for testing of models. The dataset for FFT contains 300 workload sizes ranging from 22400 × 22400 to 41536 × 41536. Both ranges use a constant step size of 64. 300 and 101 points are used for the training and testing of DGEMM models. The training and testing subsets for FFT contain 225 and 75 points, respectively.

We also build an extended dataset containing 701 points that combine the datasets for DGEMM and FFT. The extended dataset is further divided into training and testing subsets containing 551 and 150 points, respectively.

C. OPTIMIZATION OF PARALLEL MATRIX MULTIPLICATION FOR DYNAMIC ENERGY

This section describes the parallel matrix multiplication application and the model-based data partitioning algorithm used in the optimization of the application for dynamic energy.

The parallel application computes a matrix product of two dense square matrices A and B of sizes $N \times N$ and is executed using two processors, HCLServer1 and HCLServer2. The matrix A is partitioned between the processors as A_1 and A_2 of sizes $M \times N$ and $K \times N$ where $M + K = N$. Matrix B is replicated at both the processors. HCLServer1 computes the product of matrices A_1 and B and processor HCLServer2 computes the product of matrices A_2 and B. The local matrix products are computed using Intel MKL DGEMM routine. There are no communications involved.

The decomposition of the matrix A is computed using a model-based data partitioning algorithm. The inputs to the algorithm are the number of rows (N) of the matrix A and the dynamic energy consumption functions of the processors, $\{E_1, E_2\}$. The output is the partitioning of the rows, (M, K). The discrete dynamic energy function of processor P_i is given by $E_i = \{e_i(x_1, y_1), \ldots, e_i(x_m, y_m)\}$ where $e_i(x, y)$ represents the dynamic energy consumption during the matrix multiplication of two matrices of sizes $x \times y$ and $y \times y$ by the processor i. For HCLServer1, the dimension x ranges from 512 to $y/2$ in increments of 512. For HCLServer2, the dimension x ranges from $y - 512$ to $y/2$ in decrements of 512.

The main steps of the data partitioning algorithm are as follows:

1. Plane intersection of dynamic energy functions: Dynamic energy functions $\{E_1, E_2\}$ are cut by the plane $y = N$ producing two curves that represent the dynamic energy consumption functions against x given y is equal to N.

2. Determine M and K:

$$(M, K) = \underset{\substack{M \in (512, N/2),\\ K \in (N-512, N/2),\\ M+K=N}}{\arg\min} \; (e_1(M, N) + e_2(K, N))$$

For each workload size (N), the workload distribution (M, K) is determined using the data partitioning algorithm given the dynamic energy consumption functions of the two processors based on a model. Then, the dynamic energy consumption is obtained by executing the parallel application using the workload distribution. Let e_{lrmm}, e_{rapl}, and $e_{hclwattsup}$ represent the dynamic energy consumptions determined by employing the models based

on LR MM, IntelRAPL, and HCLWattsUp, respectively. The percentage of losses of dynamic energy consumption incurred by employing LR MM and IntelRAPL are calculated as $(e_{lrmm} - e_{hclwattsup})/e_{hclwattsup} \times 100$ and $(e_{rapl} - e_{hclwattsup})/e_{hclwattsup} \times 100$. The percentage saving of dynamic energy using LR MM against IntelRAPL is calculated as $(e_{rapl} - e_{lrmm})/e_{rapl} \times 100$.

ACKNOWLEDGMENT

For the purpose of Open Access, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

REFERENCES

[1] A. S. G. Andrae, "New perspectives on Internet electricity use in 2030," *Eng. Appl. Sci. Lett.*, vol. 3, pp. 19–31, Jun. 2020.

[2] L. A. Barroso and U. Hölzle, "The case for energy-proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, Dec. 2007.

[3] S. Khokhriakov, R. R. Manumachu, and A. Lastovetsky, "Multicore processor computing is not energy proportional: An opportunity for bi-objective optimization for energy and performance," *Appl. Energy*, vol. 268, Jun. 2020, Art. no. 114957. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0306261920304694

[4] R. Lucas *et al.*, "DOE advanced scientific computing advisory subcommittee (ASCAC) report: Top ten exascale research challenges," USDOE Office Sci., Adv. Sci. Comput. Res., USA, Tech. Rep. 1222713, Feb. 2014.

[5] R. Z. Ayoub and T. S. Rosing, "Predict and act: Dynamic thermal management for multi-core processors," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2009, pp. 99–104.

[6] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of scheduling techniques for addressing shared resources in multicore processors," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 1–28, Dec. 2012.

[7] V. Petrucci, O. Loques, D. Mossé, R. Melhem, N. A. Gazala, and S. Gobriel, "Energy-efficient thread assignment optimization for heterogeneous multicore systems," *ACM Trans. Embedded Comput. Syst.*, vol. 14, no. 1, Jan. 2015, Art. no. 15.

[8] A. Lastovetsky and R. R. Manumachu, "New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1119–1133, Apr. 2017.

[9] A. Chakrabarti, S. Parthasarathy, and C. Stewart, "A Pareto framework for data analytics on heterogeneous systems: Implications for green energy usage and performance," in *Proc. 46th Int. Conf. Parallel Process. (ICPP)*, Aug. 2017, pp. 533–542.

[10] H. Khaleghzadeh, M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on heterogeneous HPC platforms for performance and energy through workload distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 543–560, Mar. 2021.

[11] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, p. 2204, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/11/2204

[12] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Proc. Workshop Modeling, Benchmarking, Simulation*, 2006, pp. 70–77.

[13] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *Proc. USENIX Annu. Tech. Conf.*, 2011, pp. 1–14.

[14] M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz, "Practical power consumption estimation for real life HPC applications," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 208–217, Jan. 2013.

[15] M. Jarus, A. Oleksiak, T. Piontek, and J. Weglarz, "Runtime power usage estimation of HPC servers for various classes of real-life applications," *Future Gener. Comput. Syst.*, vol. 36, pp. 299–310, Jul. 2014.

[16] P. Gschwandtner, M. Knobloch, B. Mohr, D. Pleiter, and T. Fahringer, "Modeling CPU energy consumption of HPC applications on the IBM POWER7," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib., Netw.-Based Process. (PDP)*, Feb. 2014, pp. 536–543.

[17] X. Wu, V. Taylor, J. Cook, and P. J. Mucci, "Using performance-power modeling to improve energy efficiency of HPC applications," *Computer*, vol. 49, no. 10, pp. 20–29, 2016.

[18] J. Haj-Yihia, A. Yasin, Y. B. Asher, and A. Mendelson, "Fine-grain power breakdown of modern out-of-order cores and its implications on skylake-based systems," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 4, p. 56, 2016.

[19] K. O'brien, I. Pietri, R. Reddy, A. Lastovetsky, and R. Sakellariou, "A survey of power and energy predictive models in HPC systems and applications," *ACM Comput. Surv.*, vol. 50, no. 3, pp. 1–38, Oct. 2017.

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

[21] P. Wiki. (2017). *Perf: Linux Profiling With Performance Counters*. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page

[22] PAPI. (2015). *Performance Application Programming Interface 5.4.1*. [Online]. Available: http://icl.cs.utk.edu/papi/

[23] IntelPCM. (2012). *Intel Performance Counter Monitor—A Better Way to Measure CPU Utilization*. [Online]. Available: https://software.intel.com/en-us/articles/intel-performance-counter-monitor

[24] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments," in *Proc. 39th Int. Conf. Parallel Process. Workshops (ICPPW)*, Sep. 2010, pp. 207–216.

[25] CUPTI. (2017). *CUDA Profiling Tools Interface*. [Online]. Available: https://developer.nvidia.com/cuda-profiling-tools-interface

[26] F. Bellosa, "The benefits of event: Driven energy accounting in power-sensitive systems," in *Proc. 9th Workshop ACM SIGOPS Eur. Workshop, Beyond PC, New Challenges Oper. Syst.*, 2000, pp. 37–42.

[27] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2003, p. 93.

[28] T. Li and L. K. John, "Run-time modeling and estimation of operating system power consumption," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 31, no. 1, pp. 160–171, Jun. 2003.

[29] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 5, pp. 185–194, Oct. 2006.

[30] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proc. 10th ACM SIGPLAN Symp. Princ. Pract. Parallel Program. (PPoPP)*, 2005, pp. 186–195.

[31] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *Proc. 34th Annu. Int. Symp. Comput. Archit. (ISCA)*, 2007, pp. 13–23.

[32] S. Rivoire, P. Ranganathan, and C. Kozyrakis, "A comparison of high-level full-system power models," in *Proc. Conf. Power Aware Comput. Syst. (HotPower)*, 2008, pp. 1–5.

[33] K. Singh, M. Bhadauria, and S. A. McKee, "Real time power estimation and thread scheduling via performance counters," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 46–55, May 2009.

[34] M. D. Powell, A. Biswas, J. S. Emer, S. S. Mukherjee, B. R. Sheikh, and S. Yardi, "CAMP: A technique to estimate per-structure power at run-time using a few simple parameters," in *Proc. IEEE 15th Int. Symp. High Perform. Comput. Archit.*, Feb. 2009, pp. 289–300.

[35] B. Goel, S. A. McKee, R. Gioiosa, K. Singh, M. Bhadauria, and M. Cesati, "Portable scalable per-core power estimation for intelligent resource management," in *Proc. Int. Conf. Green Comput.*, 2010, pp. 135–146.

[36] H. Wang, Q. Jing, R. Chen, B. He, Z. Qian, and L. Zhou, "Distributed systems meet economics: Pricing in the cloud," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 1–7.

[37] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, "A methodology to predict the power consumption of servers in data centres," in *Proc. 2nd Int. Conf. Energy-Efficient Comput. Netw. (e-Energy)*, 2011, pp. 1–10.

[38] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in *Proc. 24th ACM Int. Conf. Supercomput. (ICS)*, 2010, pp. 147–158.

[39] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 563–577, Apr. 2012.

[40] R. Basmadjian and H. de Meer, "Evaluating and modeling power consumption of multi-core processors," in *Proc. 3rd Int. Conf. Future Energy Syst. Where Energy, Comput. Commun. Meet (e-Energy)*, 2012, pp. 1–10.
[41] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing," *ACM Trans. Archit. Code Optim.*, vol. 10, no. 1, pp. 1–29, Apr. 2013.
[42] S. L. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 577–589.
[43] W. Dargie, "A stochastic model for estimating the power consumption of a processor," *IEEE Trans. Comput.*, vol. 64, no. 5, pp. 1311–1322, May 2015.
[44] S. Hong and H. Kim, "An integrated GPU power and performance model," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 280–289, Jun. 2010.
[45] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," in *Proc. Int. Green Comput. Conf. Workshops (IGCC)*, 2010, pp. 115–122.
[46] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent GPU architectures," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. (IPDPS)*, May 2013, pp. 673–686.
[47] Y. S. Shao and D. Brooks, "Energy characterization and instruction-level energy model of Intel's Xeon Phi processor," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Sep. 2013, pp. 389–394.
[48] Z. Al-Khatib and S. Abdi, "Operand-value-based modeling of dynamic energy consumption of soft processors in FPGA," in *Proc. Int. Symp. Appl. Reconfigurable Comput.* Cham, Switzerland: Springer, 2015, pp. 65–76.
[49] D. Hackenberg, T. Ilsche, R. Schöne, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2013, pp. 194–204.
[50] A. Shahid, M. Fahad, R. Reddy, and A. Lastovetsky, "Additivity: A selection criterion for performance events for reliable energy predictive modeling," *Supercomput. Frontier Innov.*, vol. 4, no. 4, pp. 50–65, Dec. 2017.
[51] HCL. (2020). *HCLWattsUp: API for Power and Energy Measurements Using Watts Up Pro Meter*. [Online]. Available: https://csgitlab.ucd.ie/manumachu/hclwattsup
[52] S. Wang, *Software Power Analysis and Optimization for Power-Aware Multicore Systems*. Detroit, MI, USA: Wayne State Univ., 2014.
[53] M. F. Dolz, J. Kunkel, K. Chasapis, and S. Catalán, "An analytical methodology to derive power models based on hardware and software metrics," *Comput. Sci.-Res. Develop.*, vol. 31, no. 4, pp. 165–174, Nov. 2016.
[54] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, "RAPL in action: Experiences in using RAPL for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, pp. 9:1–9:26, Mar. 2018.
[55] Y. G. Kim, M. Kim, and S. W. Chung, "Enhancing energy efficiency of multimedia applications in heterogeneous mobile multi-core processors," *IEEE Trans. Comput.*, vol. 66, no. 11, pp. 1878–1889, Nov. 2017.
[56] W. Wang, P. Mishra, and S. Ranka, "Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems," in *Proc. 48th ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2011, pp. 948–953.
[57] G. Chen, K. Huang, J. Huang, and A. Knoll, "Cache partitioning and scheduling for energy optimization of real-time MPSoCs," in *Proc. IEEE 24th Int. Conf. Appl.-Specific Syst., Archit. Processors*, Jun. 2013, pp. 35–41.
[58] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin, "Dynamic thermal management through task scheduling," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2008, pp. 191–201.
[59] T. Li, D. Baumberger, D. A. Koufaty, and S. Hahn, "Efficient operating system scheduling for performance-asymmetric multi-core architectures," in *Proc. ACM/IEEE Conf. Supercomput. (SC)*, Nov. 2007, pp. 1–11.
[60] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Proc. Design, Automat. Test Eur. Conf. Exhib.*, Apr. 2007, pp. 1–6.
[61] R. R. Manumachu and A. L. Lastovetsky, "Design of self-adaptable data parallel applications on multicore clusters automatically optimized for performance and energy through load distribution," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 4, p. e4958, Feb. 2019.
[62] R. R. Manumachu and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 160–177, Feb. 2018.
[63] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements," *IEEE Access*, vol. 8, pp. 93793–93829, 2020.
[64] Nvidia. (Oct. 2018). *Nvidia Management Library: NVML Reference Manual*. [Online]. Available: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf
[65] *Intel Xeon Phi Coprocessor System Software Developers Guide*, Intel Corporation, Santa Clara, CA, USA, Jun. 2014.
[66] C. Gough, I. Steiner, and W. Saunders, *Energy Efficient Servers: Blueprints for Data Center Optimization*. New York, NY, USA: Apress, 2015.

ARSALAN SHAHID received the B.S. degree in electrical engineering from HITEC University Pakistan, in 2016, and the Ph.D. degree from University College Dublin, in 2020. He graduated as a Gold Medalist for the best final year project. His research interests include energy-aware computing, high-performance heterogeneous computing, and intelligent cloud computing.

MUHAMMAD FAHAD received the B.S. degree from International Islamic University Islamabad, Pakistan, in 2008, the M.S. degree from the KTH Royal Institute of Technology, Sweden, in 2012, and the Ph.D. degree from University College Dublin, in 2020. His main research interests include high-performance heterogeneous computing, energy-efficient computing, and parallel/distributed and peer-to-peer computing.

RAVI REDDY MANUMACHU received the B.Tech. degree from IIT Madras, in 1997, and the Ph.D. degree from the School of Computer Science, University College Dublin, in 2005. His main research interests include high performance heterogeneous computing, energy-efficient computing, and distributed computing.

ALEXEY LASTOVETSKY received the Ph.D. degree from Moscow Aviation Institute, in 1986, and the D.Sc. degree from the Russian Academy of Sciences, in 1997. He has published over 175 articles in refereed journals, edited books, and international conferences. He has authored two monographs *Parallel Computing on Heterogeneous Networks* (Wiley, 2003) and *High Performance Heterogeneous Computing* (J. Dongarra, Wiley, 2009). His main research interests include high performance heterogeneous computing and energy efficient computing.