Received July 11, 2020, accepted July 29, 2020, date of publication August 3, 2020, date of current version August 17, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3013812

A Comparative Study of Techniques for Energy Predictive Modeling Using Performance Monitoring Counters on Modern Multicore CPUs

ARSALAN SHAHID, MUHAMMAD FAHAD, RAVI REDDY MANUMACHU, AND ALEXEY LASTOVETSKY
School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland
Corresponding author: Arsalan Shahid (arsalan.shahid@ucdconnect.ie)

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

ABSTRACT Accurate and reliable measurement of energy consumption is essential to energy optimization at an application level. Energy predictive modelling using performance monitoring counters (PMCs) emerged as a promising approach, one of the main drivers being its capacity to provide fine-grained component-level breakdown of energy consumption. In this work, we compare two types of energy predictive models constructed from the same set of experimental data and at two levels, platform and application. The first type contains linear regression (LR) models employing PMCs selected using a theoretical model of energy of computing. The second type contains sophisticated statistical learning models, random forest (RF) and neural network (NN), that are based on PMCs selected using correlation and principal component analysis. Our experimental results performed on two modern Intel multicore processors using a diverse set of applications and a wide range of application configurations, show that the average proportional prediction accuracy of platform-level LR models is 5.09× and 4.37× times better than the platform-level RF and NN models. We also present an experimental methodology to select a reliable subset of four PMCs for constructing accurate application-specific *online* models. Using the methodology, we demonstrate that LR models perform 1.57× and 1.74× times better than RF and NN models. The consistent accuracy of LR models stress the importance of taking into account domain-specific knowledge for model variable selection, in this case, the physical significance of the PMCs originating from the conservation of energy of computing. The results also endorse the guidelines of the theory of energy of computing, which states that any non-linear energy model (in this case, the RF and NN models) employing PMCs only, will be inconsistent and hence inherently inaccurate.

INDEX TERMS Energy predictive modelling, energy additivity, linear regression, neural networks, random forest, matrix multiplication, fast Fourier transform.

I. INTRODUCTION

Energy of computing is a key environmental concern and optimizing it has become a principal technological challenge. Information and Communications Technology (ICT) systems and frameworks are currently utilizing about 2000 terawatt-hours (TWh) per year that represent about 10% of the worldwide electricity demand [1]. Andrae and Edler [2] predict that computing systems and devices will consume up to 50% of global electricity in 2030 with a contribution towards greenhouse gas emissions of 23%.

Energy optimization in computing is driven by innovative developments both at system-level and application-level. System-level optimization strategies [3]–[9] target to improve the energy efficiency of the overall execution environment of applications using methods including Dynamic Voltage and Frequency Scaling (DVFS), Dynamic Power Management (DPM) and energy-aware task scheduling. Application-level optimization strategies [10]–[14] consider

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

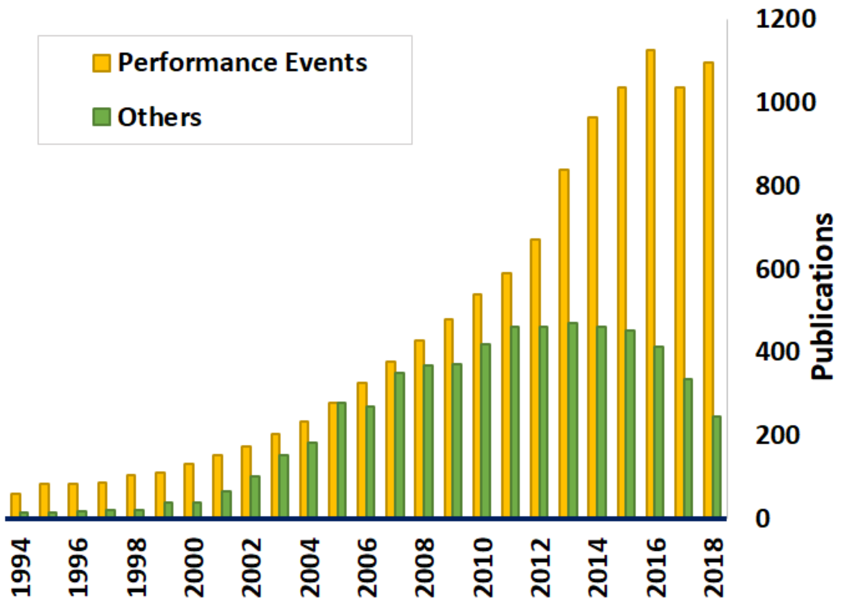


FIGURE 1. Number of research publications on computing energy using PMCs and in other areas of energy in computing including frequency scaling, dynamic voltage scaling, power usage effectiveness, energy proportional computing, and energy-efficient ethernet. The number of publications addressing PMC based energy modelling increase by 4.5×. These statistics have been collected from Google Scholar and Microsoft Academic.

the application-level parameters as model variables in analytic models to improve the energy efficiency of the applications.

Accurately measuring the energy consumption during the execution of an application is vital for energy optimization methods at the software level. The three mainstream approaches for energy consumption measurement include: (a) External power meter based system-level measurements, (b) use of on-chip voltage and current sensors, and (c) predictive models using application and system characteristics as model variables. Fahad *et al.* [15] present a comprehensive study of the three mainstream approaches to providing a measurement of energy consumption during application execution. An overview of the three methods is presented in the related work. Briefly, power meters based system-level physical measurements are reliable, accurate, and is considered the ground truth. The profiles obtained for the dynamic energy consumption of the applications using on-chip sensors exceptionally diverge from the ones obtained using the ground truth. This discovery suggested that the use of on-chip sensors based measurements do not capture the full sketch of the dynamic energy consumption during an application run.

Energy predictive models using performance monitoring counters (PMCs) emerged as a promising energy measurement approach because of its ability to provide a fine-grained component-level breakdown of energy consumption. PMCs are special-purpose hardware registers provided in modern processor architectures to record the counts of software events, that represent the kernel-level activities such as *page-faults*, *context-switches*, etc., and hardware events arising from the micro-architecture core and the performance monitoring unit (PMU) such as *CPU-cycles*, *branch-instructions*, *cache-misses*, etc. Figure 1 graphs the number of academic publications over the years in the field of energy of computing and shows that energy predictive modelling using performance events has become a dominant research topic.

Modern multicore CPUs deliver a vast set of PMCs. However, the system users can obtain a limited number of PMCs (typically 3-4 at a time) for an application execution because of the little number of hardware registers devoted to record them. Let us consider the Intel Haswell server (specification in Table 1). On this platform, the *Likwid* tool [16] exposes 167 PMCs. An application must be run about 53 times to collect the values of all the representative PMCs. Since only 3-4 PMCs can be collected in a single application run, immense programming work and time are needed to automate and record all the PMCs. Therefore, selecting a reliable subset of PMCs is crucial to the construction and prediction accuracy of the energy predictive models. We now summarize the mainstream approaches for selecting the PMCs:

- Approaches that take into account all the PMCs provided by a framework or tool for a platform for a given application with the aim to record all the activities that are viable contributors to energy consumption. Due to the high complexity of this approach, we did not find any research works adopting it.
- Approaches that are based on a statistical methodology for feature selection and feature extraction such as correlation, principal component analysis (PCA), etc. [17], [18].

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

Hardware Specifications	**Intel Haswell Server**	**Intel Skylake Server**
Processor	Intel E5-2670 v3 @2.30GHz	Intel(R) Xeon(R) Gold 6152
Micro-architecture	Haswell	Skylake
L1d cache	32 KB	32 KB
L1l cache	32 KB	32 KB
L2 cache	256 KB	1024 KB
L3 cache	30720 KB	30976 KB
Main memory	64 GB DDR4	96 GB
Thread(s) per core	2	2
Cores per socket	12	N/A
Socket(s)	2	1
NUMA node(s)	2	1
Idle Power	58 W	32 W
TDP	240 W	140 W
	Software Specifications	
OS release	CentOS 7	Ubuntu 16.04 LTS
MPI version	3.2.1	N/A
Linux kernel	3.10	3.10
OpenMP version	3.1	3.1
Python version	3.4.3	3.6.8
Compiler	gcc 4.8.5	gcc 4.8.5
Intel MKL Version	2017.0.2	2017.0.2
Likwid version	4.1	4.3.2

- Approaches that take into account the expert advice or intuition to select a possibly reliable subset (that may not be exploited in one application run), which, in experts' opinion, is a dominant contributor to energy consumption [19].
- Approaches that select PMCs using a theoretical model of the energy of computing, which is the manifestation of the fundamental physical law of energy conservation. [20], [21].

The theory of energy of computing has progressively matured over the past three years starting with a proposal of a criterion for selection of PMCs in the research work [20] followed by a formal description of the theory and its practical implications in [21]. Shahid *et al.* [20] propose a novel property of PMCs called *additivity*, which is true for all PMCs, whose count for the serial executions of two applications is equal to the sum of counts for the sole execution of each application. The authors study the additivity of PMCs provided by the two mainstream frameworks, Likwid [16] and PAPI [22] on a modern Intel Haswell multicore server. Energy additivity is based on an experimental observation that the energy consumption of the serial execution of two or more applications is equal to the sum of the energy consumption of the individual applications. If a PMC is employed as a model variable in an energy predictive model, its must follow the rule of additivity. The authors further demonstrate that many PMCs available on modern processors obtained using Likwid and PAPI that are extensively used in models as crucial model variables are non-additive. Shahid *et al.* [21] proposed a novel theory of energy of computing and unified its practical implications to increase the prediction accuracy of linear energy predictive models in a *consistency* test, which contains a suite of properties that include determinism, reproducibility, and additivity to select model variables and constraints for model coefficients. The authors show that failure to satisfy the requirements of the test worsens the prediction accuracy of linear energy predictive models.

In this work, we compare two types of energy predictive models constructed from the same set of experimental data and at two levels, platform and application. The first type contains linear regression (LR) models employing PMCs selected using the theoretical model of the energy of computing. The second type has sophisticated statistical learning models, random forest (RF), and neural network (NN), that are based on PMCs selected using correlation and principal component analysis.

We divide the experiments in this article into two main groups: Group 1 and Group 2. In Group 1, we experimentally compare the prediction accuracy of platform-level energy predictive models on HCLServer1 (Table 1). The models are analyzed in two configurations. In the first configuration, the models are trained and tested using datasets that contain all the applications. In the second configuration, the dataset of applications is split into two datasets, one for training models and the other for testing models. We demonstrate that LR models exhibit better prediction accuracies than RF and NN models in both the configurations (5.09× and 4.37× times specifically for the first configuration).

In Group 2, we examine the accuracy of application-specific energy predictive models. The experiments in this group are performed on HCLServer2 (Table 1) and contain models of the two types. We choose two well-known and highly optimized scientific kernels offered by the Intel Math Kernel Library (MKL), 2D fast Fourier transform (FFT) and dense matrix multiplication (DGEMM). We select a set of nine most *additive* PMCs (PA) and a set of nine PMCs that are *non-additive* (PNA) that are common for both the applications. PNA belongs to the dominant PMC groups reflecting the energy-consuming activities and has been widely employed in the models found in the literature (Section III). We build LR models employing PA and PNA. We demonstrate that the models based on PA have better prediction accuracy than the models based on PNA. To build online energy predictive models based on four PMCs, we compose two subsets of PMCs, *PA4* and *PNA4* from *PA* and *PNA*, containing four PMCs highly positively correlated with energy. Models that use *PA4* exhibit 3.44× and 1.71× better average prediction accuracy than models using *PNA4*. We conclude, therefore, that a high positive correlation of model variables with dynamic energy consumption alone is not sufficient to provide good prediction accuracy but should be combined with methods such as *additivity* that consider the physical significance of the model variables originating from the theory of energy conservation of computing. For the same two applications, we compare the LR models based on the set of four most additive and highly positively correlated PMCs (PA4) with the RF and NN models based on four PMCs selected using correlation and PCA. The results show that the LR model performs 1.57× and 1.74 × times better than RF and NN models.

Based on our experiments, we conclude that linear regression models based on PMCs selected using the theoretical model of the energy of computing perform better than RF and NN models using the standard statistical approaches.

To summarize, our key contribution in this work is that we present the first comprehensive experimental study comparing linear regression models employing PMCs selected using a theoretical model of the energy of computing with sophisticated statistical learning models, random forest and neural network, that are constructed using PMCs selected based on correlation and principal component analysis. We show that the LR models perform better than the RF and NN models thereby highlighting two important points. First, the consistent accuracy of LR models highlight the importance of taking into account domain-specific knowledge for model variable selection, in this case, the physical significance of the PMCs originating from the conservation of energy of computing. Second, according to the theory of energy of computing, any non-linear energy model (in this case, the RF and NN models) employing PMCs only will be inconsistent and hence inherently inaccurate. A non-linear energy model, in order to be accurate, must employ non-additive model variables in addition to PMCs.

The rest of this article is organized as follows. Section 2 presents the terminology. Section 3 highlights the related work followed by the practical implications of the theory of energy of computing in Section 4. Section 5 presents our experimental setup including the platform and application details, tools, and modelling techniques. In Section 6, we present the experimental results and discussions. Finally, Section 7 concludes the paper.

II. TERMINOLOGY RELATED TO ENERGY, PREDICTION ERROR MEASURES, AND STATISTICAL TECHNIQUES

A. ENERGY CONSUMPTION

Total energy consumption can be represented as a sum of static energy and dynamic energy. We determine the static energy consumption by multiplying the base or idle power of the system (i.e., with no running application) with the application's execution time. However, we calculate the dynamic energy consumption (energy consumption of the application) by subtracting the static energy from the total energy utilized by the system during the application execution. In other words, if P_S represents the base or idle power of the system, E_T is the total energy consumption of the system during an application run for T_E seconds, then the dynamic energy consumption E_D can be determined by using Equation 1.

$$E_D = E_T - (P_S \times T_E) \tag{1}$$

The rationale backing the use of dynamic energy consumption rather than total energy consumption is given in the Appendix I.

B. PREDICTION ERROR MEASURES

We compare the prediction accuracy of models using two measures: a) Relative error, and b) Proportional error. The relative error p of a predicted dynamic energy consumption e with respect to the ground truth dynamic energy consumption r is given below:

$$p = \frac{|r-e|}{r} \times 100 \tag{2}$$

The measure p gives a lower relative error for a model that underestimates than a model that overestimates (for example: when you consider the same proportion for the underestimated and the overestimated values of e with r). This can negatively impact the interpretation of the results. Rico-Gallego *et al.* [23] propose the proportional error μ to correct the anomaly. The proportional error for model prediction e with the ground truth r is a ratio of a maximum of the two values with the minimum of the two values. It is represented by the following equation 3.

$$\mu = \frac{max(r, e)}{min(r, e)} \tag{3}$$

μ is always greater than 1 if there exists an error, and equal to 1 otherwise.

C. MODEL VARIABLE SELECTION TECHNIQUES

We employ two model variable selection methods for random forest and neural network models. They are: 1). Correlation, and 2). Principal Component Analysis (PCA).

Correlation is a statistical metric to understand the relationship between two variables and is calculated using the following Equation 4.

$$C_{ep} = \frac{\sum(e_i - \overline{e})(p_i - \overline{p})}{\sqrt{\sum(e_i - \overline{e})^2 \sum(p_i - \overline{p})^2}} \tag{4}$$

where, C_{ep} is the correlation coefficient between the dynamic energy consumption e and the PMC p_i. e_i represents energy consumption of an application and $\overline{e}$ is the mean of the energy of all the applications in the data-set. p_i is the PMC count and $\overline{p}$ is it's mean for all the applications in the data-set. The value of the correlation coefficient is between -1 to 1. A value of -1 for C_{ep} means perfect negative correlation, 0 signifying no correlation, and $+1$, a perfect correlation between the energy and the PMC.

Principal Component Analysis (PCA) [24] is applied to determine the most statistically influential PMCs. It is a multivariate statistical technique for feature extraction and is used for dimensionality reduction in high-dimensional data. It uses a correlation matrix to ease the analysis by selecting the most valuable features in a data-set. The top principal component captures the maximum variability in the data, and each succeeding component has the highest variability subject to the constraint imposing orthogonality with the previous principal components.

III. RELATED WORK

This section presents a literature survey of mainstream energy measurement methods and highlights their pros and cons, dominant tools used to obtain PMCs, prominent works towards the construction of energy predictive models, research works that provide a critical review of PMCs, and finally, recent developments in the energy predictive models using PMCs.

A. ENERGY MEASUREMENT METHODS

The three mainstream methods for energy measurement during an application execution are: (a). Power meters-based physical measurements at platform-level, (b). on-chip sensors based voltage and current measurements, and (c). Energy predictive models using application and system characteristics as model variables.

The first method using the external power meters is known to be accurate and considered as the ground truth. It has been used for providing the measurements at a system-level. Fahad *et al.* [25] presented the first methodology (that is, AnMoHA) to measure the component-level energy consumption of a platform using external power meters. The authors demonstrate that their approach provides accurate energy consumption decomposition up to socket-level. However, the core-level granularity for energy consumption decomposition has not been achieved.

The second method used on-chip voltage and current sensors to determine power and are now supported by popular processor vendors such as Intel, AMD, and IBM Power 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) [26] is used to monitor power and control frequency (and voltage) of Intel CPUs. Similarly, Nvidia NVIDIA Management Library (NVML) [27] and Intel System Management Controller chip (SMC) [28] provide the power consumption by Nvidia GPUs and Intel Xeon Phi. NVML provides the instant power draw values with nominal accuracy up to $\pm 5\%$ [27]. The accuracy of Intel SMC is not available. Apart from insufficient documentation, there are other issues with the power data values provided by these vendor-specific libraries. For example, it lacks details such as update frequency of power readings and also suffers from potential complications such as sampling interval variability of significant sensor lag as reported by [29]. Fahad *et al.* [15] present the first detailed study on the accuracy of on-chip power sensors and show that deviations of the energy measurements provided by on-chip sensors from the system-level power measurements (considered as ground truth) do not motivate their use in the optimization of applications for dynamic energy.

The third method using energy predictive models (employing application and hardware parameters as model variables) emerged as a popular alternative to determine the energy consumption of an application. It is because of the model's ability to provide fine-grained component-level (such as core-level, cache-level, etc.) energy consumption measurements. A vast number of these models are linear and make use of performance monitoring counters (PMCs) as model variables. This approach forms the focus of this work.

B. TOOLS TO OBTAIN PMCs

Linux Perf [30] also called *perf_events* can be used to gather the PMCs for CPUs in Linux. It also comes as a performance profiling tool suite including *perf stat*, *perf record*, *perf report*, *perf annotate*, *perf top* and *perf bench*.

Intel PCM [31] is used for reading PMCs of core and uncore (which includes the QPI) components of an Intel processor. It is exposed to programmers as a C++ API and is also able to provide energy measurements from Intel on-chip sensors. It can further support the statistical analysis of core frequencies, QPI power, and DRAM activities.

PAPI [22] provides a standard API for accessing PMCs available on most modern microprocessors. It provides two types of events, *native events* and *present events*. *Native events* correspond to PMCs native to a platform. They form the building blocks for *present events*. A *preset event* is mapped onto one or more native events on each hardware platform. While *native events* are specific to each platform, *preset events* obtained on different platforms can not be compared.

Likwid [16] provides command-line tools and an API to obtain PMCs for both Intel, POWER8, and AMD processors on the Linux OS. It contains a variety of performance measurement and application tunning tools such as *likwid-pin* and *likwid-bench*. Furthermore, Likwid is light-weight, which means the performance overheads of Likwid is less than 6000 cycles. The recent stable released version is Likwid 5.0 with support to extract PMCs of accelerators such as GPUs.

For Nvidia GPUs, CUDA Profiling Tools Interface (*CUPTI*) [32] can be used for obtaining the PMCs for CUDA applications. CUPTI provides the following APIs: Activity API, Callback API, Event API, Metric API, and Profiler API. Sample PMCs that can be obtained using these APIs are total instruction count, data rate, memory load, and store counts, cache hits and misses, number of branches instructions, etc.

C. NOTABLE ENERGY PREDICTIVE MODELS FOR CPUs

The concept of exploiting the utilization metrics of active computing resources (such as CPU, memory, disks, and I/O) has been widely adopted in dynamic power management techniques [33]–[35]. Some of the initial efforts towards energy consumption modelling correlating PMCs with power and energy measurements include [36]–[43]. The frequently used parameters include integer operations, floating-point operations, memory requests due to cache misses, component access rates, instructions per cycle (IPC), CPU/disk and network utilization, etc. were trusted to be highly correlated with energy utilization. Simple linear models have been developed using PMCs and correlated features to predict the energy consumption of platforms. Rivoire *et al.* [44] and Rivoire [45] study and compare five full-system, real-time power models, using a variety of machines and benchmarks. They report that the PMC-based model is the best overall in terms of accuracy since it accounted for the majority of the contributors to the system's dynamic power. Other notable PMC-based linear models are [19], [46]–[51].

Rotem *et al.* [26] present *RAPL*, in Intel Sandybridge to predict the energy consumption of core and uncore components (QPI, LLC) based on PMCs (which are not disclosed). Lastovetsky and Reddy [13] present an application-level energy model where the dynamic energy consumption of a processor is represented by a function of problem size. Unlike PMC-based models that contain hardware-related PMCs and do not consider problem size as a variable, this model takes into account 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.

D. NOTABLE ENERGY PREDICTIVE MODELS FOR ACCELERATORS AND HPC APPLICATIONS

Some of the promising research contributions for the energy predictive models on GPU accelerators using PMCs include [52]–[54]. PMCs counts can be recorded during the application run by using the CUDA Profiling Tools Interface (CUPTI) [32]. An instruction-level energy consumption model for Xeon Phi processors forms the basis of work presented by Shao and Brooks [55]. Another linear instruction-level model for predicting the dynamic energy consumption on the soft processors based on FPGA was presented by Khatib *et al.* [56]. Inter-instruction effects and the operand values of the instructions have been considered by the proposed model.

Some notable platform-wide power models for HPC servers based on PMCs include [57]–[59]. Gschwandtner *et al.* [60] presented linear regression models based on hardware counters for prediction of energy consumption of HPC applications executing on the IBM POWER7 processor.

E. CRITIQUES OF PMCs FOR ENERGY PREDICTIVE MODELLING

References [20], [41], [61]–[63] are research works that critically examine the accuracy of PMC based energy predictive models and demonstrate their poor prediction accuracy. The researchers in [61] point out the fundamental limitation to obtain all the PMCs simultaneously or in a single application run and argue that the linear regression models yield prediction errors as high as 150%.

Fahad [15] demonstrated the state-of-the-art linear energy predictive models employing PMCs that are selected using statistical correlations are pruned to errors because they do not consider the physical significance of model variables, coefficients, and intercepts. The authors show average error between platform level PMC models using linear regression and the ground truth (power-meters) ranges from 14% to 32% and the maximum reaches 100%.

F. RECENT DEVELOPMENTS IN THE ENERGY PREDICTIVE MODELS USING PMCs

In all previous works, the causes of the inaccuracy of energy predictive models or the reported wide variance of the accuracy of the models have not been studied. Furthermore, a sound theoretical framework to understand the fundamental significance of the model variables concerning the dynamic energy consumption has been lacking.

The theory of energy of computing has progressively matured over the past three years starting with a proposal of a criterion for selection of PMCs in the research work [20] followed by a formal description of the theory and its practical implications in [21]. Shahid *et al.* [20] proposed a novel selection criterion for PMCs called *additivity*, which can be used to determine the set of PMCs for accurate and reliable energy predictive modelling. They experimentally demonstrate that majority of PMCs exposed by *Likwid* and *PAPI* on a modern Intel Haswell server are *non-additive*. Furthermore, the authors argue that non-additive PMCs have been employed as key model variables for energy predictions thereby questioning the accuracy and reliability of such models.

Shahid *et al.* [64] show that the accuracy of energy predictive models based on three popular mainstream techniques (linear regression, random forests, and neural networks) can be improved by selecting PMCs using the property of additivity. They show that the removal of *non-additive* PMCs improves the average prediction accuracy of linear regression models from 31% to 18%, random forest models from 38% to 24%, and neural network models from 30% to 24%.

Shahid *et al.* [21] further proposed a novel theory of energy of computing and unified its practical implications to increase the prediction accuracy of linear energy predictive models in a *consistency* test, which contains a suite of properties that include determinism, reproducibility, and additivity to select model variables and constraints for model coefficients. The authors conducted a fundamental study showing the rise in the number of non-additive PMCs along with the increase in the number of cores employed during the execution of the applications. The authors attribute the rise in the number of non-additive PMCs to the inherent complexities of modern multicore platforms such as severe resource contention and non-uniform memory access (NUMA).

IV. THEORY OF ENERGY OF COMPUTING: PRACTICAL IMPLICATIONS

A theory of energy of computing has been developed over the past three years starting with the proposal of a criterion for selection of PMCs in the research work [20] followed by a formal description of the theory and its practical implications for improving the prediction accuracy of linear energy predictive models in [21].

The theory of energy of computing is a formalism containing properties of PMC-based energy predictive models that are manifestations of the fundamental physical law of energy conservation. The properties capture the essence of single application runs and characterize the behavior of serial execution of two applications. They are intuitive and experimentally validated and are formulated based on the following observations:

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

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

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

The first two properties are combined into a *additivity* test for the selection of PMCs. A linear energy predictive model employing PMCs and which violates the properties of the consistency test will have poor prediction accuracy.

By definition and intuition, PMCs are all pure counters of energy-consuming activities in modern processor architectures and as such must be additive. Therefore, according to the theory of energy of computing, any consistent, and hence accurate, energy model, which only employs PMCs, must be linear. This also means that any non-linear energy model employing PMCs only, will be inconsistent and hence inherently inaccurate. A non-linear energy model, in order to be accurate, must employ non-additive model variables in addition to PMCs.

A. *Additivity* OF PMCs

The property of *additivity* is based on an intuitive and simple rule that if a PMC is intended to be employed as a model variable in a linear energy predictive model, then its count for a *compound* application should be equal to the sum of its counts for the executions of the base applications forming the compound application. It is based on the experimental observation and a well-known fact that the dynamic energy consumption of a serial run of two applications is the sum of dynamic energy consumption observed for the sole executions of each application.

The additivity of a PMC is determined as follows. We first obtain the counts of the PMC for the sole executions of the base applications. Then, we run the *compound* application and record its count of the PMC. Generally, the main computations for the compound application consist of the main computations of the base applications executed one after the other. If the PMC of the *compound* application is equal to the sum of the PMCs obtained for the base applications (within a tolerance of 5.0%), the PMC is categorized as potentially *additive*. Else, it is labeled as *non-additive*.

For each PMC, we determine the maximum percentage error. Since the ground truth between the sum of base applications' PMCs and the compound application PMCs is not known, we use Equation 5 instead of Equation 2 to calculate the percentage error for a compound application, as follows:

$$Error(\%) = |\frac{(e_{b1} + e_{b2}) - e_c}{(e_{b1} + e_{b2} + e_c)/2}| \times 100 \tag{5}$$

TABLE 2. List of applications employed for studying the prediction accuracy of the models.

Application	Description
NPB EP	Embarrassingly Parallel, Kernel
NPB BT	Block Tri-diagonal solver
HPCG	Intel-optimized high-performance conjugate gradient
NPB IS	Integer Sort, Kernel for random memory access
NPB LU	Lower-Upper Gauss-Seidel solver
NPB UA	Unstructured Adaptive mesh, dynamic memory access
NPB CG	Conjugate Gradient
NPB MG	Multi-Grid on a sequence of meshes
NPB FT	Discrete 3D fast Fourier Transform
NPB SP	Scalar Penta-diagonal solver
MKL FFT	2-D Fast Fourier Transform
MKL DGEMM	Intel-optimized Dense Matrix Multiplication
stress	CPU, disk and I/O stress
Naive MM	Naive Matrix-matrix multiplication
Naive MV	Naive Matrix-vector multiplication

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

V. EXPERIMENTAL SETUP

A. PLATFORM AND APPLICATIONS

The experiments are carried out on two modern multicore platforms: 1). Intel Haswell dual-socket server, and 2). Intel Skylake single-socket server. The specifications for both are given in Table 1.

Our test suite (Table 2) contain a broad set of standard benchmarks with highly memory-bound and compute-bound scientific computing applications such as DGEMM and FFT from Intel math kernel library (MKL), scientific applications from NAS Parallel benchmark suite, Intel HPCG, *stress*, and two unoptimized applications.

B. SOFTWARE TOOLS

We measure the following during an application execution: 1). Dynamic energy consumption, 2). Execution time, and 3). PMCs. The experimental workflow is shown in Figure 2. The dynamic energy consumption is determined using system-level power measurements provided by WattsUp Pro power meter. The readings are recorded programmatically using a detailed statistical methodology employing HCLWattsUp API [65]. We periodically calibrate the power meters by using Yokogawa WT210 that is an ANSI C12.20 revenue-grade power meter. We detail the usage of HCLWattsUp API in Appendix II and experiments for calibration of the power-meters are presented in Appendix V. We follow a strict statistical methodology to confirm the experimental accuracy and reliability where a sample mean for a response variable is obtained from several experimental application runs (Appendix III).

Likwid package [16] has been used to record the PMCs. On an Intel Haswell platform, it offers 164 PMCs, whereas, 385 PMCs are exposed on an Intel Skylake platform. The PMCs with counts less than or equal to 10 have been eliminated considering their insignificance for modelling the dynamic energy consumption since they are non-reproducible over several executions of the same application on our platform.

The resultant set of PMCs contains a total of 151 for Intel Haswell and 323 for Intel Skylake. The process of collection of all the PMCs consumes a lot of time since only four PMCs can be recorded for one execution of an application reason being a small number of available hardware registers storing them. Furthermore, many PMCs need to be recorded separately or in sets of two or three for a single application run. Therefore, to record all the PMCs, one must execute an application about 53 and 99 times on Intel Haswell and Intel Skylake platform, respectively.

We use a tool called *AdditivityChecker* (Appendix VII), that automates the determination of the additivity value of a PMC.

C. TYPES OF ENERGY PREDICTIVE MODELS

The types of energy predictive models that we compare in this work are described below:

- **Linear Regression (*LR*)**: A *LR* based model can be stated as follows:

$$E_D = \sum_{k=1}^{n} \beta_k \times e_k \tag{6}$$

where E_D is the dynamic energy consumption, β_0 is called the model intercept, the $\beta = \{\beta_1, \ldots, \beta_n\}$ is the vector of regression coefficients or the model parameters

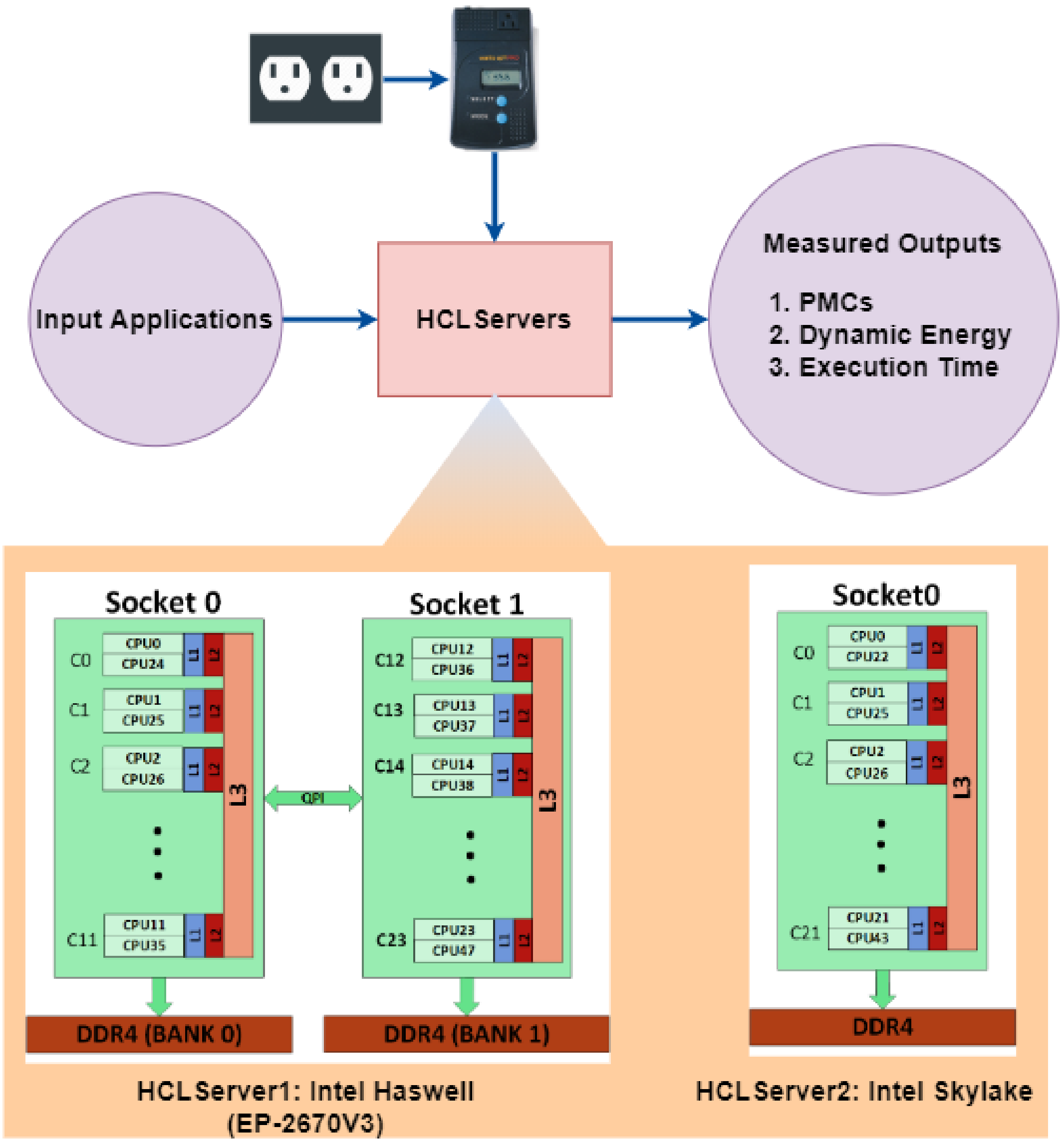


FIGURE 2. Experimental workflow to determine the PMCs for our HCLServer platforms.

and $\{e_1, \ldots, e_n\}$ are the PMCs. In real life, there usually is stochastic noise (measurement errors). Therefore, the measured energy is typically expressed as

$$\tilde{E_D} = \sum_{k=1}^{n} \beta_k \times e_k + \epsilon \tag{7}$$

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 build a specialized linear model using a regression technique that constrains the regression coefficients (β) to be positive.

- **Random Forest (*RF*)**: A RF technique is a supervised learning algorithm using a decision tree-based approach to train on a data-set and output mean prediction from individual trees. It is considered for its accuracy in classification and regression-based tasks [66]. It is a non-linear machine learning model build by constructing many linear boundaries. The overall non-linearity is because a single linear function can not be used to classify and regress on each iteration of the decision tree.
- **Neural Networks (*NN*)**: A *NN* model is inspired by neurons of a human brain and contains an interconnected group of nodes where each node computes weights and biases and give an output prediction. We set the transfer/activation function as linear. The learning function is Bayesian regularization that gives optimal regularization parameters in an automated fashion [67]. Bayesian regularization updates the weight and biases by using

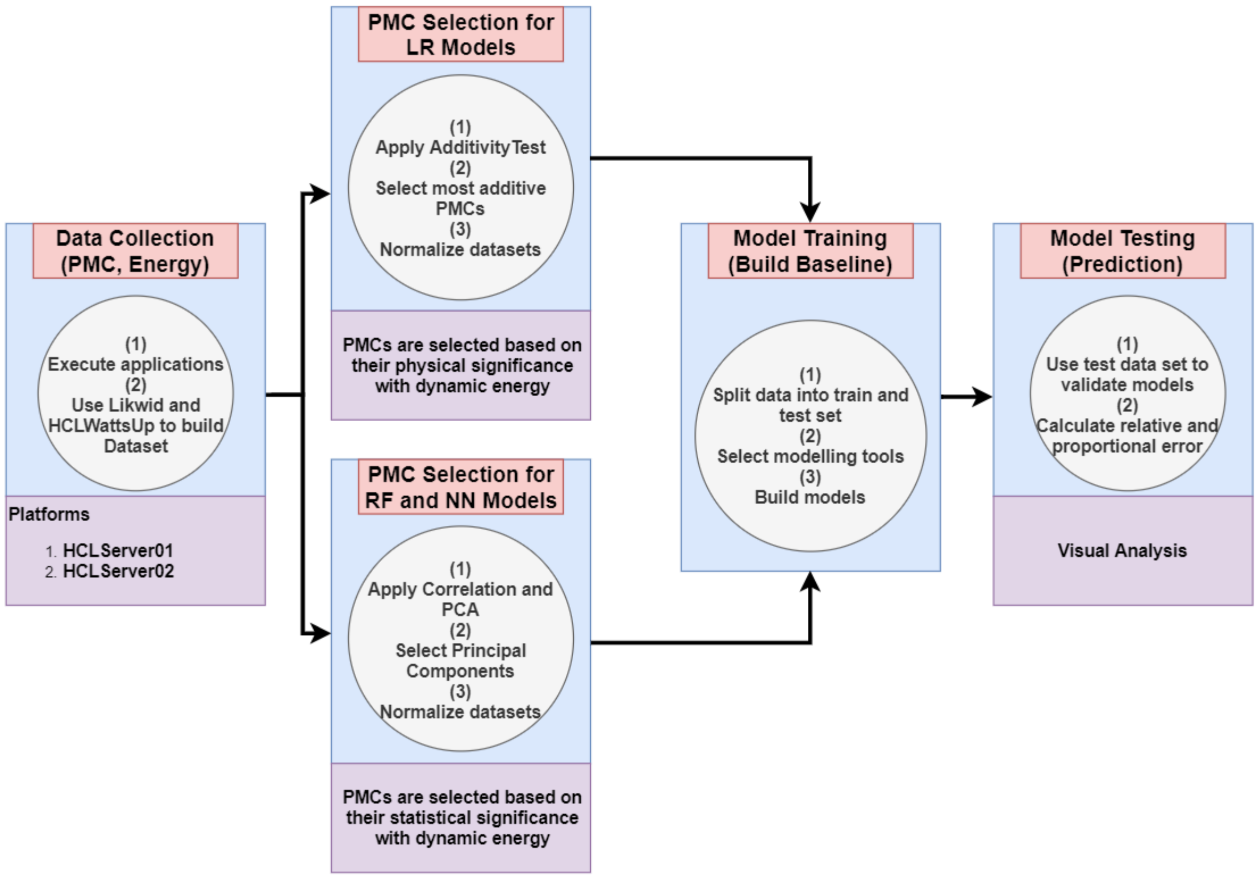

PMC Selection for LR Models
(1) Apply Additivity Test (2) Select most additive PMCs (3) Normalize datasets
PMCs are selected based on their physical significance with dynamic energy
Data Collection (PMC, Energy)
(1) Execute applications (2) Use Likwid and HCLWattsUp to build Dataset
Platforms
1. HCLServer01
2. HCLServer02
PMC Selection for RF and NN Models
(1) Apply Correlation and PCA (2) Select Principal Components (3) Normalize datasets
PMCs are selected based on their statistical significance with dynamic energy
Model Training (Build Baseline)
(1) Split data into train and test set (2) Select modelling tools (3) Build models
Model Testing (Prediction)
(1) Use test data set to validate models (2) Calculate relative and proportional error
Visual Analysis

FIGURE 3. Model training and testing pipelines for LR, RF, and NN models.

the Levenberg-Marquardt algorithm [68], which is used to train the NN up to 100 times quicker in comparison with the commonly used gradient-descent and back-propagation method.

The training parameters employed to build the models are given in Table 3. Figure 3 explains the machine learning model pipeline. It has four main stages: 1). Data collection, 2). PMC selection, 3). Model training, and 4). Model testing or validation. After the collection of the data-set from HCLServers, the data is passed through a PMC selection stage which first normalizes the PMC counts. To construct the LR models, PMCs are first checked for their additivity using the Additivity Test, and the topmost additive PMCs are selected as model variables. To build the RF and NN models, the PMCs are first evaluated based on their statistical correlation. The set of top positively correlated PMCs is then further pruned using PCA. The correlation and PCA methods are explained in detail in the section VI-2. The set of selected PMCs is then split into two subsets. One for training the models and the other for testing their accuracy.

D. SELECTION METHODS FOR PMCs

We now summarize the steps to select model variables or PMCs using two approaches as described below:

TABLE 3. Modelling Parameters.

Linear Regression	
Estimation Method	Least-squares
Model Coefficients	Positive
Intercept	0
Random Forests	
Software Tool	R Programming Studio
Variables in each split	3
Type	Regression
Number of Trees	500
Neural Networks	
Network type	Feed-forward back propagation
Input parameters	PMCs
Output parameters	Dynamic energy
Training algorithm	Bayesian regularization
Performance function	Mean-Squared Error (MSE)
Number of neurons	20
Network layers	2
Transfer function	Linear

1) PMCs are selected based on the consistency test from the theory of energy of computing (Section IV) and employed as model variables in linear regression (LR) models. The steps for the PMC selection include:

- The most additive PMCs for a set of applications are selected.
- During the execution of the applications, the individually powered computing components (memory and CPU) with activities that result in dynamic energy-consuming are identified.
- The most additive PMCs that belong to computing components contributing to dynamic energy consumption are then selected as model variables.

2) PMCs are selected using correlation and PCA based statistical methods and then employed in non-linear models such as random forest (RF) and neural network (NN). The selection method is composed of two stages:

- In the first stage, we list all the PMCs in the increasing order of positive correlation with dynamic energy consumption. We select all the PMCs with a correlation coefficient of over 0.90.
- In the second stage, we apply the principal component analysis (PCA) on the PMCs selected in the first stage to pick the most statistically influential PMCs. Figure 4 illustrates this PMC selection process.

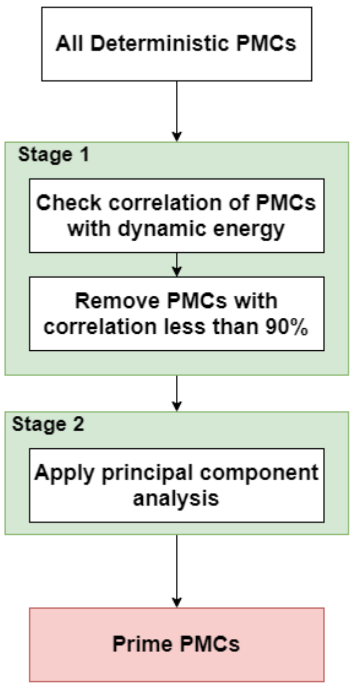

All Deterministic PMCs
Stage 1
Check correlation of PMCs with dynamic energy
Remove PMCs with correlation less than 90%
Stage 2
Apply principal component analysis
Prime PMCs

FIGURE 4. PMC selection process using statistical methods.

VI. EXPERIMENTAL RESULTS

We divide our experiments into two groups, Group 1 and Group 2, as follows:

- **Group 1:** We employ this group to study the prediction accuracy of the platform-level energy predictive models. We use two experimental configurations. In the first configuration, we split the full data-set representing all the applications into two subsets, one for the training and the other for testing. The training and test data-sets contain data points encompassing all the applications. In the second configuration, the models are trained on a data-set for one set of applications and tested against a different set of applications. The experiments are performed on HCLServer1 (Table 1).
- **Group 2:** We employ this group to study the accuracy limits of the application-level energy predictive models. Two highly memory-bound and compute-bound scientific computing applications, DGEMM and FFT from Intel MKL, are used for this purpose. The experiments are performed on HCLServer2 (Table 1).

Group 1*: COMPARISON OF PREDICTION ACCURACY OF PLATFORM-LEVEL ENERGY PREDICTIVE MODELS*

Using a diverse application set (Table 2), we build platform-level energy predictive models employing model variables that are selected using two aforementioned approaches, consistency test, and statistical methods.

1) ENERGY PREDICTIVE MODELS USING CONSISTENCY TEST

The experimental methodology for measuring and selecting the PMCs follows:

- The PMCs are obtained using Likwid tool, which classifies them into performance groups. The list of the performance groups is given in Appendix VIII. We apply the first step of the consistency test, which is to check if the PMCs are deterministic and reproducible using the following two steps:
  - PMCs with counts less than or equal to 10 are removed. These PMCs have no statistical significance on modelling energy consumption of our platform because we found them to be non-reproducible. Several PMCs with counts equal to zero are also removed. The reduced set contains 151 and 298 PMCs on Intel Haswell and Intel Skylake, respectively.
  - We broadly compare the PMCs obtained using Likwid, PAPI, and Linux Perf. We remove PMCs that show different counts for different tools. The final set contains 115 and 224 PMCs on Intel Haswell and Intel Skylake platform.
- We discover that all the work performed during the execution of the applications in our test suite is due to CPU and memory activities. We run a set of experiments to evaluate the contribution of both of these components towards the dynamic energy consumption. We summarize them below:
  - We execute a synthetic application (app-cpu) performing floating-point operations on all the processor cores for 10 seconds and measure its dynamic

energy consumption. HCLWattsUp reports the dynamic energy consumption to be 1337 joules.
  - We then execute another synthetic application (app-mem) performing *memcpy()* operations on all the memory blocks for 10 seconds and measure the dynamic energy consumption. We find the energy consumption to be insignificant and can not be measured within the statistical confidence of 95%.
  - We further execute app-cpu for 20 seconds and 30 seconds and found the dynamic energy consumption to be equal to 2596 joules and 3821 joules, respectively. However, the execution of app-mem for 20 and 30 seconds results in dynamic energy consumption less than 5 joules.

- Based on the above experiments, we remove the PMCs that belong to Likwid main memory group for any further analysis due to two reasons. First, the memory activities do not reflect any contributions to the dynamic energy consumption on our platforms. Second, low counts for memory PMCs add noise that affects the training of models and unduly worsen the prediction accuracy of the models.
- The CPU activities during the application run are represented by PMCs that belong to the following dominant groups: cache, branch instructions, micro-operations (uops), floating-point instructions, instruction decode queue, and cycles.
- We then study the additivity of PMCs belonging to the dominant groups.
  - We build a data-set of 277 points as base applications by executing the applications from our test suite with different problem sizes. Each point contains the dynamic energy consumption and PMCs corresponding to the base applications.
  - We execute another set of 50 compound applications from the serial combination of base applications and record their dynamic energy consumption and PMCs.
  - For all PMCs, we calculate the percentage errors of each compound application with the sum of base applications. The additivity test error for each PMC is the maximum of the percentage errors for all compound applications.
- We found no PMC to be absolutely additive (with the additivity test error of less than 5%), in general, for all applications in the test suite (Table 2). Therefore, we select one top additive PMC for each dominant PMC group.

Table 4 list the selected PMCs (PL1,$\cdots$,PL6) in the order of increasing additivity test error. The PMC PL1 is highly additive compared to the rest.

We construct a data-set of 448 points for different configurations for applications in our test suite (Table 2). We split the data-set into two subsets, 335 points for training the models, and 113 points for testing their prediction accuracy. We used

TABLE 4. List of selected PMCs and their *additivity* test errors (%).

Selected PMCs	***Additivity* Test Error(%)**
PL1: AVX_INSTS_ALL	7
PL2: UOPS_EXECUTED_PORT_PORT_6	8
PL3: IDQ_MITE_UOPS	11
PL4: CPU_CLOCK_THREAD_UNHALTED	13
PL5: L2_RQSTS_MISS	17
PL6: BR_INST_RETIRED_ALL_BRANCHES	18

this division based on best practices and expert opinions in this domain.

We build six LR models, {LR1, LR2, LR3, LR4, LR5, LR6}. To impose the constraints of the consistency test, the linear models are built using penalized linear regression using R programming interface that forces the coefficients to be non-negative and to have zero intercepts. The models contain decreasing number of *non-additive* PMCs. Model LR1 employs all the selected PMCs as predictor variables. Model LR2 is based on five most *additive* PMCs. PMC PL6 is removed because it has the highest *non-additivity*. Model LR3 uses four most *additive* PMCs and so on until Model LR6 containing the highest *additive* PMC, which is PL1.

We compare the predictions of the models with system-level physical measurements using HCLWattsUp, *which we consider to be the ground truth* [15]. The minimum, average, and maximum prediction errors for the models are given in Table 5. One can see that the accuracy of the models improves as we remove the highest non-additive PMCs one by one until Model LR5, which exhibits the least average (p, μ) of (21.8%, 1.382), respectively. LR5 employs two most additive PMCs, PL1 and PL2. PL1 accounts for the floating-point operations and PL2 accounts for a portion of micro-operations executing inside the CPU cores during the execution of an application. We observe that LR1 has the worst average (p, μ) of (27.9%, 1.703) due to the poor linear fit.

Figures 5a and 5b show the plots for ground truth and predicted dynamic energy consumptions obtained using HCLWattsUp and LR6 against the top additive PMC (that is, PL1) for the train and test data-sets, respectively. Similarly, Figure 5c and 5d shows the plots for ground truth and predicted dynamic energy consumptions obtained using HCLWattsUp and LR5 against the top two additive PMCs (that are, PL1 and PL2) for train and test data-sets, respectively. It can be seen that the training dataset and the test dataset include all the applications. Furthermore, the combined use of PL1 and PL2 as model variables in LR5 increases its prediction power and makes it the most accurate and consistent model. This is because only one PMC (despite being most additive) is not able to track all the dynamic energy-consuming activities for applications in a modern multicore CPU.

TABLE 5. Linear regression models with their minimum, average, and maximum prediction errors.

Model	PMCs	Relative Errors p in [%] (Min, Avg, Max)	Proportional Errors μ (Min, Avg, Max)
LR1	PL1,PL2,PL3,PL4,PL5,PL6	(2.1, 27.9, 85.1)	(1.04, 1.703, 5.190)
LR2	PL1,PL2,PL3,PL4,PL5	(1.9, 26.01, 82.91)	(1.02, 1.61, 5.021)
LR3	PL1,PL2,PL3,PL4	(1.9, 26.01, 82.91)	(1.02, 1.61, 5.021)
LR4	PL1,PL2,PL3	(1.21, 25.15, 80.2)	(1.01, 1.56, 4.91)
LR5	PL1,PL2	(0.66, 21.80, 71)	(1.003, 1.382, 4.65)
LR6	PL1	(0.96, 24.6, 79)	(1.004, 1.50, 4.85)

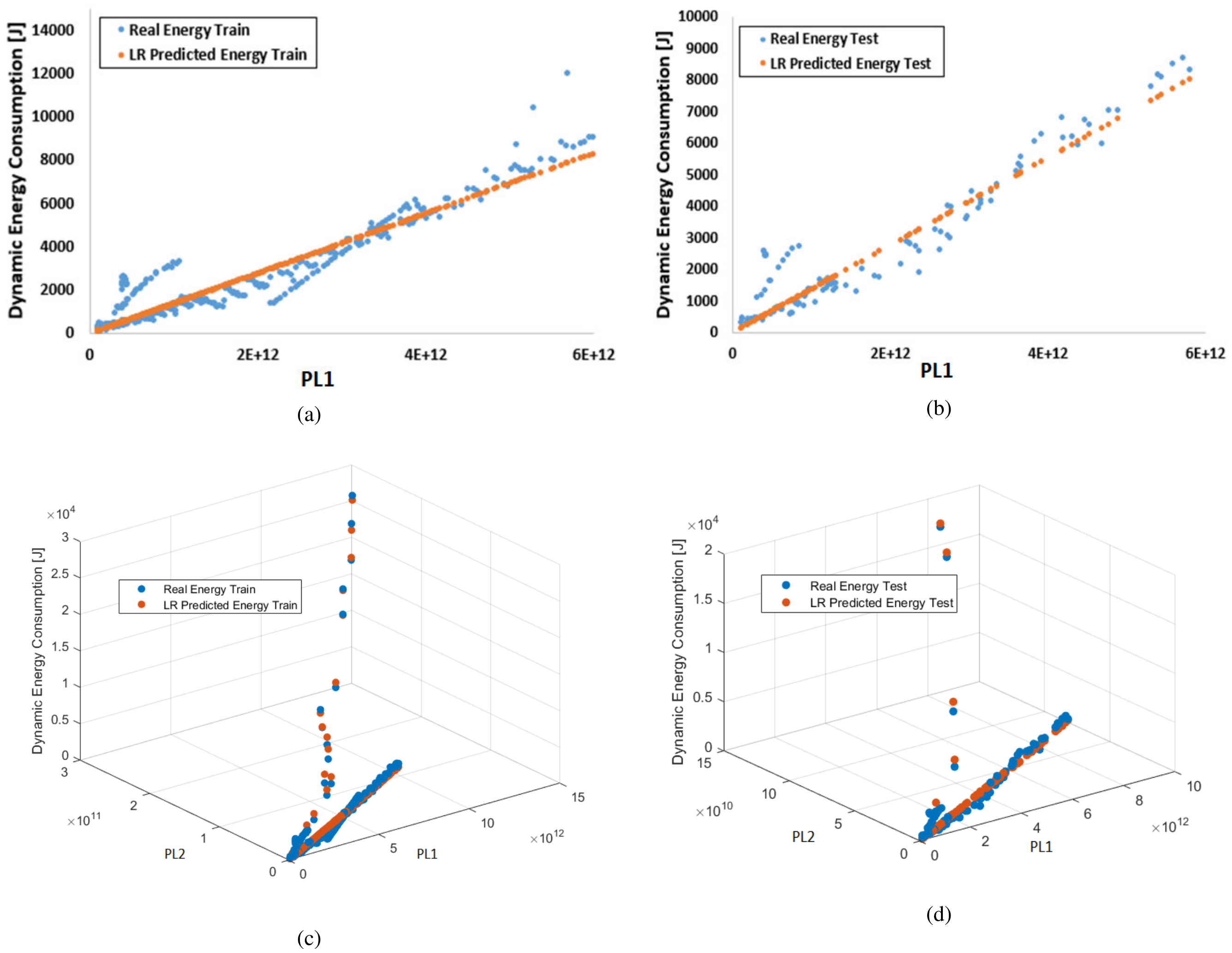


FIGURE 5. Real and predicted dynamic energy consumptions using HCLWattsUp and linear regression models versus (a). PMC PL1 for train set applications, (b). PMC PL1 for test set applications, (c). PMCs, PL1 and PL2, for train set applications, and (d). PMCs, PL2 and PL2, for test set applications.

The model LR5, employing PL1 and PL2 as model variables, is built using a training data set that contains all the applications and is tested against the test dataset that also contains all the applications. Let us denote this split configuration of training and test datasets as A1. We consider a different split configuration, A2. In A2, the test set applications do not include the training set applications. The training set contains 335 points and the test set contains 113 points.

We then build a model LR5-A2 using the training data-set from configuration A2. The minimum, average, and maximum p and μ for train and test set using LR5-A2 are given in Table 6. Comparison of the average p and μ for both test

TABLE 6. Prediction accuracies for linear regression models for configuration A2.

Model	Data-set	Relative Errors p in [%] (Min, Avg, Max)	Proportional Errors μ (Min, Avg, Max)
LR5-A2	Training	(1.19, 25, 136)	(1.002, 1.527, 4.77)
LR5-A2	Testing	(1.03, 24, 101)	(1.009, 1.416, 4.71)

TABLE 7. List of PMCs selected in stage 1 of approach B where the PMCs are listed in the increasing order of positive correlation with dynamic energy consumption .

UOPS	L2
UOPS_EXECUTED_TOTAL_CYCLES	L2_RQSTS_ALL_DEMAND_DATA_RD_HIT
UOPS_EXECUTED_CORE	L2_RQSTS_ALL_DEMAND_DATA_RD
UOPS_EXECUTED_CORE_STALL_CYCLES	L2_RQSTS_CODE_RD_HIT
UOPS_RETIRED_CORE_USED_CYCLES	L2_RQSTS_CODE_RD_MISS
BRANCHES	**ICACHE**
BR_INST_RETIRED_ALL_BRANCHES	ICACHE_ACCESSES
BR_MISP_RETIRED_ALL_BRANCHES	**CPU CLOCK**
IDQ	CPU_CLK_UNHALTED_ANY
IDQ_MITE_UOPS	CPU_CLOCK_THREAD_UNHALTED_ONE_THREAD
IDQ_DSB_UOPS	CPU_CLOCK_UNHALTED_TOTAL_CYCLES
IDQ_ALL_DSB_CYCLES_4_UOPS	**AVX**
MEM	AVX_INSTS_ALL
MEM_LOAD_UOPS_RETIRED_ALL_ALL	
MEM_UOPS_RETIRED_ALL	

sets using LR5 (or LR5-A1) and LR5-A2 shows only a minor increase from 21.8% to 24% and 1.382 to 1.416, respectively. Therefore, we conclude that the accuracy and consistency of the LR model employing the two most additive PMCs are generic and therefore the LR model generalizes well.

2) ENERGY PREDICTIVE MODELS USING STATISTICAL METHODS

We first present the experimental methodology to select the model variables. The data-set used for this purpose includes 277 base applications. Each application is represented by a data point that contains the dynamic energy consumption and 151 PMCs. We apply the first stage of PMC selection using statistical methods shown in Figure 4. We list all the PMCs in the increasing order of positive correlation with dynamic energy consumption. All the PMCs with a correlation coefficient of over 0.90 are then selected. The selected PMCs are listed in Table 7 based on their groups. In the second stage, we apply the principal component analysis (PCA) on the selected PMCs from the first stage. The most statistically influential PMCs obtained after stage 2 are termed as prime PMCs, which are shown in Table 8.

We employ prime PMCs in RF and NN models using a data-set of 448 points. Each point in a data-set contains dynamic energy consumption for an application with particular input and the prime PMCs. We divide the data-set into a training dataset (335 points) and a test dataset (113 points). The splitting is done using two aforementioned configurations, A1 and A2.

TABLE 8. List of prime PMCs obtained after applying principal component analysis.

PL7: UOPS_EXECUTED_TOTAL_CYCLES
PL8: UOPS_EXECUTED_CORE
PL9: UOPS_EXECUTED_CORE_STALL_CYCLES
PL10: CPU_CLOCK_THREAD_UNHALTED_ONE_THREAD
PL11: CPU_CLOCK_UNHALTED_TOTAL_CYCLES
PL12: BR_INST_RETIRED_ALL_BRANCHES

We build two sets of models, RFS = {A1-RF, A2-RF}, and NNS = {A1-NN, A2-NN}. The RR and NN parameters used to build the models are given in Table 3. We compare the predictions of the models with the ground truth. Tables 9 and 10 show the minimum, average, and maximum p and μ from RFS and NNS. Figures 6a and 6b compare the average prediction accuracies of the most accurate LR model with the RF and NN models. The least average p and μ is obtained for the LR model with two model variables, that is, 21.80% and 1.38. RF-A2 and NN-A2 for the test set yields the highest average (p, μ) of (37%, 7.21) and (44%, 6.19), respectively.

3) DISCUSSION

Following are the salient observations from the results:

- The PMCs selected for training the models should represent the dynamic energy-consuming activities during the

TABLE 9. Prediction accuracies for random forest models.

Model	Data-set	Relative Errors p in [%] (Min, Avg, Max)	Proportional Errors μ (Min, Avg, Max)
RF-A1	Training	(1.12, 26.6, 63)	(1.003, 2.13, 8.51)
RF-A1	Testing	(1.24, 28.7, 57)	(1.002, 3.61, 8.22)
RF-A2	Training	(2.37, 25.2, 61)	(1.001, 2.92, 7.76)
RF-A2	Testing	(3.92, 37, 293)	(1.001, 7.21, 21.41)

TABLE 10. Prediction accuracies for neural network models.

Model	Data-set	Relative Errors p in [%] (Min, Avg, Max)	Proportional Errors μ (Min, Avg, Max)
NN-A1	Training	(2.31, 26.6, 65)	(1.002, 3.32, 9.12)
NN-A1	Testing	(2.09, 25.1, 58)	(1.01, 2.21, 6.28)
NN-A2	Training	(2.17, 25.4, 57)	(1.005, 2.56, 5.39)
NN-A2	Testing	(4.96, 44, 492)	(1.033, 6.19, 19.74)

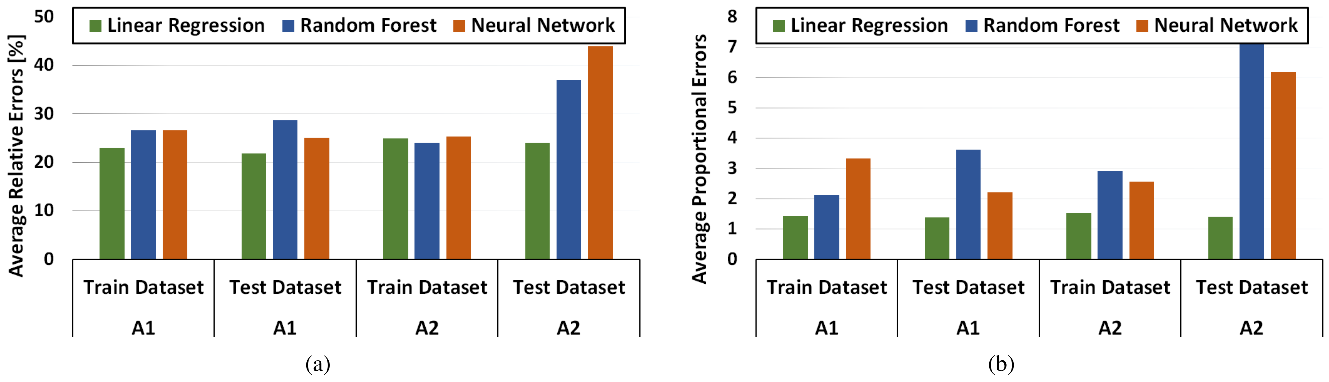


FIGURE 6. Comparison of (a). average relative prediction accuracies, and (b). average proportional prediction accuracies, for LR5, RF, and NN.

application execution. We discover that, in our platform, the contribution of memory-centric operations towards the dynamic energy consumption is insignificant. Therefore, we use CPU-centric PMCs such as floating-point operations and micro-operations originating from processor core for training the energy predictive models.

- The average relative and proportional prediction errors for LR is better than RF and NN for all the models. There is a significant difference in the average prediction errors for test applications in the configuration A2 where the test applications do not include the training set applications. RF and NN models perform poorly for A2. The average relative prediction accuracy of the best LR model only degrades by 2%. We conclude that a machine learning-based platform-level energy predictive model employing PMCs selected using statistical methods provide good prediction accuracy when data points in the train set and test set belong to the same set of applications. This is because of their ability to memorize well the input domain of energy values of the applications. However, their accuracy suffers when the train and test data sets contain different sets of applications suggesting their inability to provide good prediction accuracy for a general set of applications (that is, to generalize well).
- The average prediction accuracies (p) for LR is 1.54× and 1.84× better than RF and NN, respectively, for test applications in A2. The proportional prediction accuracies (μ) for LR is 5.09× and 4.37× better than RF and NN, respectively. This suggests that μ is better than p for the interpretation of results.
- The results highlight two important points. First, the consistent accuracy of LR models stresses the importance of taking into account domain-specific knowledge for model variable selection, in this case, the physical significance of the PMCs originating from the conservation of energy of computing. Second, according to the theory of energy of computing, any non-linear energy model (in this case, the RF and NN models) employing PMCs only, will be inconsistent and hence inherently inaccurate. A non-linear energy model, to be accurate, must employ non-additive model variables in addition to PMCs.

TABLE 11. Additive and non-additive PMCs highly correlated with dynamic energy consumption. 0 to 1 represents positive correlation of 0% to 100%.

	Additive PMCs	Correlation
AL1	UOPS_RETIRED_CYCLES_GE_4_UOPS_EXEC	0.992
AL2	FP_ARITH_INST_RETIRED_DOUBLE	0.993
AL3	BR_INST_RETIRED_ALL_BRANCHES	0.860
AL4	UOPS_EXECUTED_CORE	0.993
AL5	UOPS_DISPATCHED_PORT_PORT_4	0.870
AL6	IDQ_DSB_CYCLES_6_UOPS	0.981
AL7	IDQ_ALL_DSB_CYCLES_5_UOPS	0.972
AL8	IDQ_ALL_CYCLES_6_UOPS	0.993
AL9	L2_RQSTS_CODE_RD_HIT	0.821
	Non-additive PMCs	
AL10	ICACHE_64B_IFTAG_MISS	0.960
AL11	CPU_CLOCK_THREAD_UNHALTED	0.600
AL12	BR_MISP_RETIRED_ALL_BRANCHES	0.992
AL13	UOPS_RETIRED_CORE_USED_CYCLES	0.751
AL14	FRONTEND_RETIRED_L2_MISS	0.806
AL15	ITLB_MISSES_STLB_HIT	0.111
AL16	L2_TRANS_CODE_RD	0.860
AL17	IDQ_MS_UOPS	0.99
AL18	ARITH_DIVIDER_COUNT	0.986

Group 2: *COMPARISION OF PREDICTION ACCURACY OF APPLICATION-LEVEL ENERGY PREDICTIVE MODELS*

In this section, we study the accuracy of application-specific energy predictive models. The experiments are conducted HCLServer2 (Table 1). We compare the models built using PMCs selected via the aforementioned approaches, consistency test and statistical methods. First, we build LR models that satisfy the properties of the consistency test that is based on the theory of energy conservation of computing. We also build LR models employing non-additive PMCs that belong to the dominant PMCs groups reflecting energy-consuming activities for application execution and that have been widely employed by energy models found in the literature (Section III). Finally, we build RF and NN models employing PMCs selected using statistical methods such as correlation and PCA.

We now present the experimental methodology using a consistency test to build LR models:

- Out of the 385 PMCs available for this platform, we found no PMC to be *additive*, in general, within the tolerance of 5% for all applications in our test suite (Table 2). We use this tolerance to inline the theoretical accuracy of models with the accuracy of measurements obtained using power-meters. However, many PMCs are highly additive for each application. We select for two highly optimized scientific kernels: Fast Fourier Transform (FFT) and Dense Matrix-Multiplication application (DGEMM), from Intel Math Kernel Library (MKL).
- We check if the PMCs are reproducible and deterministic by running the same application several times without any change in the operating environment. The PMC is considered to be reproducible and deterministic if its value for multiple executions of the same application lies within the confidence interval of 95%. We discover that 323 PMCs are reproducible and deterministic.
- A data-set of 50 *base* applications with a range of problem sizes for DGEMM and FFT is used to study the additivity of their representative PMCs. For DGEMM, the problem sizes vary from 6500×6500 to 20000×20000, and for FFT, the range of problem sizes is 22400×22400 to 29000×29000. These problem sizes are selected because of the applications' considerable execution time ($>$ 3 seconds) so that HCLWattsUp can accurately capture the dynamic energy consumptions. A data-set of 30 *compound* applications is build using the serial execution of *base* applications. The two data sets are then given as an input to AdditivityChecker that returns the additivity test errors as an output. We found some PMCs that are additive in common for both applications.
- We select nine PMCs that are highly additive with additivity test errors of less than 1%. We also select nine PMCs that are non-additive for both the applications but which have been employed as predictor variables in energy predictive models given in the literature (Section III). The set of *additive* PMCs are denoted by *PA* and *non-additive* PMCs by *PNA*. In both sets, there are no PMCs from the memory group because of the insignificant contribution of memory activities towards the dynamic energy consumption. The selected PMCs with their correlations are given in Table 11.
- By executing DGEMM and FFT, for the problem sizes ranging from 6400×6400 to 38400×38400 and 22400×22400 to 41536×41536, respectively, with a

TABLE 12. Prediction accuracies of LR models using nine PMCs.

Model	**PMCs**	**Relative Errors p in [%] (Min, Avg, Max)**	**Proportional Errors μ (Min, Avg, Max)**
LR-A	*PA*	(0.013, 36.11, 226.1)	(1.006, 1.55, 6.53)
LR-NA	*PNA*	(0.513, 86.11, 4073)	(1.022, 2.41, 8.94)

TABLE 13. Prediction accuracies of LR models using four PMCs.

Model	**PMCs**	**Relative Errors p in [%] (Min, Avg, Max)**	**Proportional Errors μ (Min, Avg, Max)**
LR-A4	PA4	(0.024, 25.12, 87.25)	(1, 1.42, 6.49)
LR-NA4	PNA4	(0.449, 85.61, 4039)	(1.009, 2.43, 9)

constant step sizes of 64, we build a data-set containing 801 points. We record the dynamic energy consumption and the selected PMCs (Table 11) for each application. The data-set is further divided into two subsets, one for training and the other for testing the models. The train data-set contains 651 points and the test data-set contains 150 points.

- We build two linear models, {LR-A, LR-NA}. The model LR-A is trained using PMCs belonging to PA and the model LR-NA is trained using PMCs belonging to PNA. Table 12 show the relative and proportional prediction errors of the models. One can see that the models based on PA have better average prediction accuracies than the models based on PNA.
- Since only four PMCs can be collected in a single application run, the selection of such a reliable subset is crucial to the prediction accuracy of online energy models. We use PA and PNA to build two sets of four most energy correlated PMCs. The first set PA4, {AL1, AL2, AL4, AL8}, is constructed using PA and the second set PNA4, {AL10, AL12, AL17, AL18}, using PNA. We build two linear models, {LR-A4, LR-NA4}. The model LR-A4 is trained using PMCs belonging to PA4 and the models LR-NA4 is trained using PMCs belonging to PNA4. The training and test data-sets are the same as before.
- Table 13 shows the relative and proportional prediction errors of the models. We can see that model LR-NA4 built using highly correlated but *non-additive* PMCs do not demonstrate much improvement in average prediction accuracies when compared to model LR-NA based on nine non-additive PMCs. However, LR-A4 performs 1.43× and 1.09× better in terms of average relative and proportional accuracies, respectively.

The experimental methodology to select PMCs using statistical methods and building RF and NN models is:

- We select the same two highly optimized scientific kernels and remove the PMCs that are not reproducible and deterministic.
- We build a data-set of 50 applications using different problem sizes for DGEMM and FFT. The range of problem sizes used 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. We remove all the PMCs that have less than 90% positive correlation with dynamic energy. We then select the topmost correlated PMC from each Likwid group. The selected prime PMCs are {AL19, AL20, ..., AL27} and listed in Table 14.

TABLE 14. List of PMCs obtained for application-specific modelling using correlation.

AL19: BR_INST_RETIRED_ALL_BRANCHES	0.99
AL20: ICACHE_64B_IFTAG_MISS	0.96
AL21: OFFCORE_REQUESTS_ALL_DATA_RD	0.96
AL22: L2_RQSTS_MISS	0.97
AL23: MEM_LOAD_RETIRED_L3_MISS	0.99
AL24: CYCLE_ACTIVITY_CYCLES_MEM_ANY	0.96
AL25: ITLB_MISSES_WALK_COMPLETED	0.95
AL26: IDQ_MS_MITE_UOPS	0.99
AL27: LSD_UOPS	0.97

- Since only four PMCs can be employed in an online model, we select the top four principal PMCs by applying principal component analysis. The final list of prime PMCs includes {AL19, AL23, AL26, AL27}.
- We build a data-set containing 801 points representing DGEMM and FFT for a range of problem sizes from 6400 × 6400 to 38400 × 38400 and 22400 × 22400 to 41536 × 41536, respectively, with a constant step size of 64. We record the dynamic energy consumption and the prime PMCs (AL19, AL23, AL26, and AL27) for each application. The data-set is further divided into two subsets, one for training and the other for testing the models. The train data-set contains 651 points and the test data-set contains 150 points.
- We build a random forest model (RF-MA) and a neural network model (NN-MA) using the training set. Table 15 shows the relative and proportional prediction errors of the models. The average relative and proportional

TABLE 15. Prediction accuracies of application-specific RF and NN models.

Model	Relative Errors p in [%] (Min, Avg, Max)	Proportional Errors μ (Min, Avg, Max)
RF-MA	(.82, 24.82, 82.7)	(1.004, 2.23, 5.32)
NN-MA	(0.91, 23.95, 91.4)	(1.002, 2.48, 4.17)

prediction accuracies for RF and NN models are 24.8% and 2.23, and, 23.9% and 2.48, respectively.

- If we compare the average prediction accuracies of LR-A4 with RF-MA and NN-MA, we do not see much difference in their relative prediction accuracies. However, average proportional errors show that LR-A4 is 1.57× and 1.74× better than RF-MA and NN-MA, respectively.

4) DISCUSSION

Following are the salient observations from the experimental results:

- For our experiments, the training data set is constructed by executing the DGEMM and FFT with a constant increment of workload sizes using a fixed step size of 64. The generation of training data is a tedious task. However, once a model with the desired accuracy is constructed, it represents the relationship of a specific combination of PMCs and understands the underlying pattern with dynamic energy consumption. Therefore, the model can be used to predict the energy consumption for any workload size for an application or a set of applications.
- The statistical methods lack the ability to check the physical significance of the model variables with energy consumption. The PMCs used to build the models for using this approach contains memory parameters as model variables (for example, MEM_LOAD_RETIRED_L3_MISS in Table 14). The practical implications of the theory of energy of computing incorporate domain knowledge for linear dynamic energy predictive models by introducing properties for the selection of model variables, coefficients, and intercepts. In order to identify the processor components that are the dominating contributors to dynamic energy consumption during the application executions, we conducted experiments by stressing memory and CPU. We found that the dynamic energy consumption because of memory operations is negligible on our platforms. Therefore, linear regression models built using the theory of energy of computing do not employ any memory PMCs as model variables.
- The models based on most additive and highly correlated PMCs have better average prediction accuracy when compared to the models based on non-additive and highly positively correlated PMCs. We conclude, therefore, that correlation with dynamic energy consumption alone is not sufficient to provide good average prediction accuracy but should be combined with methods such as *additivity* that take into account the physical significance of the model variables originating from the theory of energy of computing.
- Online LR models that employ PMCs selected using the theory of energy conservation of computing perform better in terms of average proportional accuracy than RF and NN based models that use purely statistical methods to select PMCs.
- While we do not see much difference in the relative prediction accuracies (p) of LR-A4, RF-MA, and NN-MA, average proportional errors show that LR-A4 is 1.57× and 1.74× better than RF-MA and NN-MA, respectively. This suggests that μ is a better statistic than p for accurate interpretation of results.
- The consistent accuracy of LR models highlight the importance of taking into account domain-specific knowledge for model variable selection, in this case, the physical significance of the PMCs originating from the conservation of energy of computing.
- The results also endorse the guidelines of the theory of energy of computing, which states that any non-linear energy model (in this case, the RF and NN models) employing PMCs only, will be inconsistent and hence inherently inaccurate. A non-linear energy model, in order to be accurate, must employ non-additive model variables in addition to PMCs.

VII. CONCLUSION

Accurate and reliable measurement of energy consumption is essential to energy optimization at an application level. Energy predictive modelling using performance monitoring counters (PMCs) emerged as a promising approach owing to its ability to provide a fine-grained component-level decomposition of energy consumption.

In this work, we compared two types of energy predictive models constructed from the same set of experimental data and at two levels, platform and application. The first type contains linear regression (LR) models employing PMCs selected using a theoretical model of the energy of computing, which is the manifestation of the fundamental physical law of energy conservation. The second type contains sophisticated statistical learning models, random forest (RF), and neural network (NN), that are constructed using PMCs selected based on correlation and principal component analysis.

We demonstrated how the accuracy of LR models can be improved by selecting PMCs based on a theoretical model of the energy of computing. We compared the prediction accuracy of LR models with RF and NN models, which are based on PMCs selected using correlation and principal component analysis. We employed two different configurations of train and test datasets. In the first configuration, the train and test datasets contain all the applications. In the second configuration, the applications are split between the train and test datasets. We showed that the average prediction accuracy of the best LR model is almost the same in both the experimental configurations and its average prediction accuracy is better than RF and NN models. This highlights the consistent accuracy of LR models. The prediction accuracy of RF and NN models is better in the second configuration than the first configuration. It implies that RF and NN models memorize well the domain of inputs but are not able to predict well and hence generalize well for a new set. This is because they use PMC selection techniques that are domain oblivious and that do not take into account the physical significance of the PMCs originating from the conservation of energy of computing.

We also studied application-specific energy predictive models. We experimentally demonstrated that the use of highly additive PMCs results in notable improvements in the average prediction accuracy of LR models when compared to LR models employing non-additive PMCs. We concluded, therefore, that a high positive correlation with dynamic energy consumption alone is not sufficient to provide good prediction accuracy but should be combined with selection criteria that take into account the physical significance of the PMCs originating from fundamental laws such as energy conservation of computing. Finally, we presented an experimental methodology to select a reliable subset of four PMCs for constructing accurate application-specific *online* models. We showed that the LR models perform better than RF and NN models.

Our results highlight two important points. First, the consistent accuracy of LR models stresses the importance of taking into account domain-specific knowledge for model variable selection, in this case, the physical significance of the PMCs originating from the conservation of energy of computing. Second, according to the theory of energy of computing, any non-linear energy model (in this case, the RF and NN models) employing PMCs only, will be inconsistent and hence inherently inaccurate. A non-linear energy model, in order to be accurate, must employ non-additive model variables in addition to PMCs. In our future work, we will explore methods to improve the prediction accuracy of energy predictive models that employ high-level model variables such as the utilization rates of compute devices, unlike PMCs which are pure counts.

APPENDIX I. RATIONALE BEHIND USING DYNAMIC ENERGY CONSUMPTION INSTEAD OF TOTAL ENERGY CONSUMPTION

We consider only the dynamic energy consumption 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 II. APPLICATION PROGRAMMING INTERFACE (API) FOR MEASUREMENTS USING EXTERNAL POWER METER INTERFACES (HCLWattsUp)

HCLServer1 and HCLServer2 have 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 using the serial USB interface. The execution of this script is non-intrusive and consumes insignificant power.

We use HCLWattsUp API function, which gathers the readings from the power meters to determine the average power and energy consumption during the execution of an application on a given platform. HCLWattsUp API can provide the 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. It is important to note that 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 following example illustrates the use of statistical methods to measure the dynamic energy consumption during the execution of an application.

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* and 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 the Student's t-distribution.
- The maximum allowed time *maxElapsedTime* specified in seconds has elapsed.

If any of the conditions are not satisfied, then a return code of 0 is output suggesting that statistical confidence has not been achieved. If statistical confidence has been achieved, then the number of repetitions performed, the time elapsed and the final relative standard error is 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 for the calculation of 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* tool.

APPENDIX III. EXPERIMENTAL METHODOLOGY TO DETERMINE THE SAMPLE MEAN

We followed the methodology described below to make sure the experimental results are reliable:

- 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 monitoring its load continuously for a week using the tool *sar*. Insignificant variation in the load was observed during this monitoring period suggesting normal and clean behavior of the server.
- An application during its execution is bound to the physical cores using the *numactl* tool.

```
1  #include <wattsup.hpp>
2  int main(int argc, char** argv)
3  {
4      std::string pathsToMeters[2] = {
5         "/opt/powertools/bin/wattsup1",
6         "/opt/powertools/bin/wattsup2"};
7      std::string argsToMeters[2] = {
8         "--interval=1",
9         "--interval=1"};
10     hcl::Wattsup wattsup(
11         2, pathsToMeters, argsToMeters
12     );
13     hcl::Precision pIn = {
14         maxRepeats, cl, maxElapsedTime,
       maxStdError
15     };
16     hcl::Precision pOut;
17     double sampleMean, sd;
18     int rc = wattsup.execute(
19                hcl::DENERGY, executablePath,
20                executableArgs, &pIn, &pOut,
21                &sampleMean, &sd
22     );
23     if (rc == 0)
24        std::cerr << "Precision NOT achieved.\n";
25     else
26        std::cout << "Precision achieved.\n";
27     std::cout << "Max repetitions "
28               << pOut.reps_max
29               << ", Elasped time "
30               << pOut.time_max_rep
31               << ", Relative error "
32               << pOut.eps
33               << ", Mean energy "
34               << sampleMean
35               << ", Standard Deviation "
36               << sd
37               << std::endl;
38     exit(EXIT_SUCCESS);
39 }
```

FIGURE 7. Example illustrating the use of HCLWattsUp API for measuring the dynamic energy consumption.

- To obtain a data point, the application is repeatedly executed until the sample mean lies in the 95% confidence interval with a precision of 0.025 (2.5%). For this purpose, we use Student's t-test 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. When we mention a single number such as execution time (seconds) or floating-point performance (in MFLOPs or GFLOPs), we imply the sample mean determined using the Student's t-test.

  The function *MeanUsingTtest*, shown in Algorithm 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).

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
```

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 the 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 the 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 IV. STEPS TO ENSURE RELIABLE EXPERIMENTS

To ensure the reliability of our results, we follow a statistical methodology where a sample mean for a response variable is obtained from several 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%) has been achieved. For this purpose, Student's t-test is used assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions by plotting the distributions of observations.

The server is fully dedicated for the experiments. To ensure reliable energy measurements, we took following precautions:

1) *HCLWattsUp* API [65] gives the total energy consumption of the server during the execution of an application using physical measurements from the external power meters. This includes the contribution from components such as NIC, SSDs, fans, etc. 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 that there is no I/O performed by the application using tools such as *sar*, *iotop*, etc.
   - 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 network is not used by the application using monitoring tools such as *sar*, *atop*, etc.
   - Bind an application during its execution to resources using cores-pinning and memory-pinning.
2) Our platform supports three modes to set the fans speed: *minimum*, *optimal*, and *full*. We set the speed of all the fans to *optimal* during the execution of our experiments. We make sure there is no contribution to the dynamic energy consumption from fans during an application run, by following the steps below:
   - We continuously monitor the temperature of server and the speed of fans, both when the server is idle, and during the application run. We obtain this information by using Intelligent Platform Management Interface (IPMI) sensors.

- We observed that both the temperature of 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. The results from this experiment were the same as when the fans were run at *optimal* speed.
- To make sure that pipelining, cache effects, etc, 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.

APPENDIX V. CALIBRATION OF WattsUp PRO POWER-METERS

The dynamic energy consumption during the application execution is measured using a *WattsUp Pro* power meter on both servers (HCLServer1 and HCLServer2) and obtained programmatically via the HCLWattsUp interface [65]. The power meter is periodically calibrated using an ANSI C12.20 revenue-grade power meter, Yokogawa WT210. In this chapter, we explain our methodology and some results to calibrate our power-meters.

We compare the WattsUp Pro power-meter power measurements with Yokogawa using three methods that are explained as follows:

1) **Naked-eye visual monitoring**
   - We first attach the WattsUp Pro power-meters to both servers.
   - Once the servers are switched on and are in stable condition, we monitor the WattsUp pro LCDs and note the power readings in watts for both servers.
   - we carefully plugged off the WattsUp Pro power meters and plug the servers via Yokogawa power meter.
   - Once the servers are switched on and are in stable condition, we monitor the Yokogawa LCDs and note the power readings in watts for both servers.
   - On comparison, we find a difference of 2 watts and 3 watts for HCLServer1 and HCLServer2, respectively.

2) **Monitoring server base power**
   - We connect both power meters to both servers one by one and once the servers are stable, we programmatically obtain the power readings from both power meters.
   - For WattsUp Pro, a Perl script provides the power readings with a granularity of 1 second. Similarly, Yokogawa comes with its own software and allows us to read power readings at the granularity of 1 second.

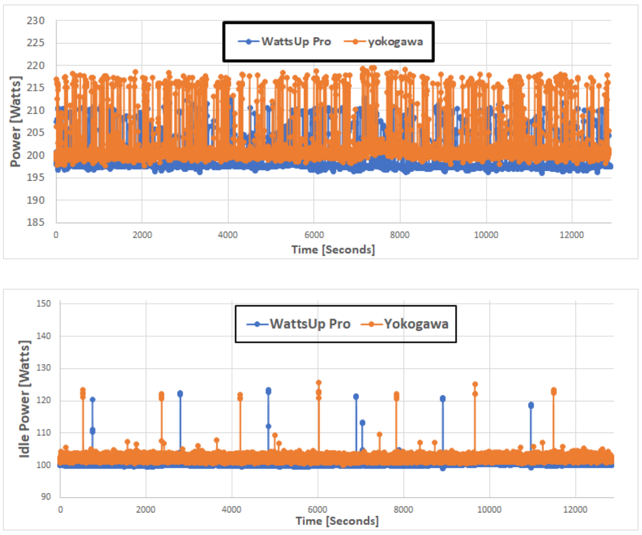


FIGURE 8. Calibration test for idle power using WattsUp Pro and Yokogawa PowerMeter on (a) HCLServer1 and (b) HCLServer2.

   - We measure and record the base powers of both servers for 3.5 hours using both power meters. Figure 8 compare the idle power profiles of HCLServer1 and HCLserver2 using both power meters, respectively. It can be seen that both profiles are almost the same. However, HCLServer1 has more power variations and HCLServer2 is considerably stable in terms of base power. For HCLServer2, there are power spikes after every half an hour for a couple of seconds. This is because of a daemon service being triggered by the OS.
   - Table 16 show the minimum, average, and maximum power consumption for both servers and power meters. If we compare the average, WattsUp Pro gives 1-2 watts less power consumption than Yokogawa.

3) **Measurement of total energy consumption for two scientific applications**
   - We choose two scientific applications: 1) DGEMM and 2) FFT from Intel MKL.
   - We execute DGEMM for problem sizes 4096 × 4096 to 30720 × 30720 with a constant step size of 1024 on HCLServer1. We then execute DGEMM for problem sizes 15360 × 15360 to 30720 × 30720 with a constant step size of 1024 on HCLServer2. We build the total energy consumption profiles using both power meters for these application executions.
   - We execute FFT for problem sizes 4096 × 4096 to 30720 × 30720 with a constant step size of 1024 on HCLServer1. We then execute FFT for problem sizes 8384 × 8384 to 62880 × 62880 with a constant step size of 2096 on HCLServer2. We build

TABLE 16. Minimum, maximum and average of idle power using WattsUp Pro and Yokogawa PowerMeter on HCLServer1 and HCLServer2.

	HCLServer1		HCLServer2	
	WattsUp Pro	Yokogawa	WattsUp Pro	Yokogawa
Min	196	197.72	99.1	99.93
Max	212.8	219.47	123.3	125.6
Average	198.2	201.0	100.03	102.06

TABLE 17. Comparison of minimum, average, and maximum measurement errors for DGEMM and FFT on HCLServer1 and HCLServer2 using WattsUp Pro and Yokogawa.

	HCLServer2 Errors [%] (Min, Avg, Max)	HCLServer1 Errors [%] (Min, Avg, Max)
DGEMM	(2.01, 4.95, 8.16)	(0.58, 9.25, 33.18)
FFT	(0.11, 6.02, 15.84)	(0.20, 7.41, 16.90)

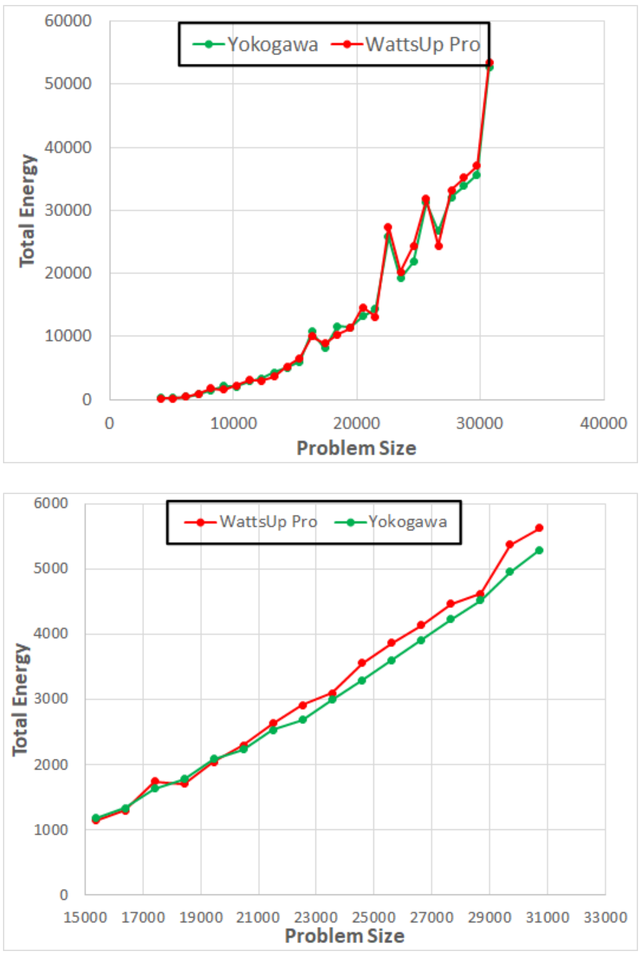


FIGURE 9. Comparison of total power for Intel MKL DGEMM using WattsUp Pro and Yokogawa PowerMeter on (a) HCLServer1 and (b) HCLServer2.

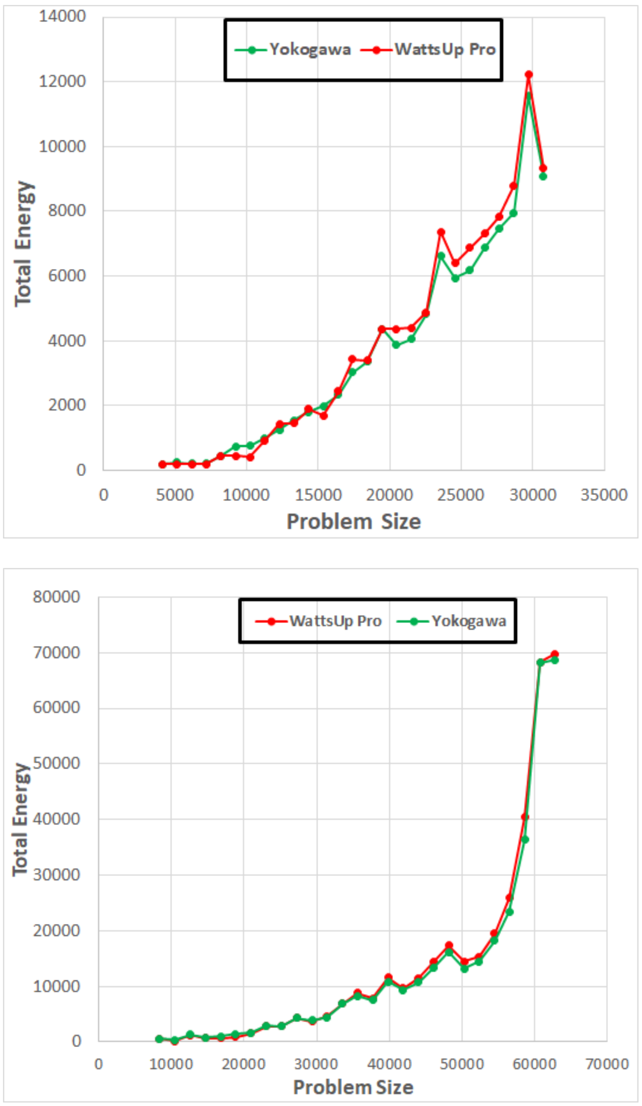


FIGURE 10. Comparison of total power for Intel MKL FFT using WattsUp Pro and Yokogawa PowerMeter on (a) HCLServer1 and (b) HCLServer2.

the total energy consumption profiles using both power meters for these application executions.

- Figure 9 and 10 show the total energy consumption profiles for DGEMM and FFT on both servers, respectively.
- Table 17 show the relative error in percentage for total energy consumptions obtained using HCLServer1 and HCLServer2. It can be seen that the average measurement error for

DGEMM is 4.95% and 9.25% on HCLServer2 and HCLServer1, respectively. For FFT, the average measurement error is 6% and 7.4% on HCLServer2 and HCLServer1, respectively.

APPENDIX VI. METHODOLOGY TO DETERMINE THE COMPONENT-LEVEL ENERGY CONSUMPTION USING HCLWattsUp

We provide here the details of how system-level physical measurements using HCLWattsUp can be used to determine the energy consumption by a component (such as a CPU) during application execution.

We define the group of components running a given application kernel as an *abstract processor*. For example, consider a matrix multiplication application running on a multicore CPU. The abstract processor for this application, which we call *AbsCPU*, comprises of the multicore CPU processor consisting of a certain number of physical cores and DRAM. In this work, we use only such configurations of the application which execute on *AbsCPU* and do not use any other system resources such as solid-state drives (SSDs), network interface cards (NIC) and so forth. Therefore, the change in energy consumption of the system reported by HCLWattsUp reflects solely the contributions from CPU and DRAM. We take several precautions in computing energy measurements to eliminate any potential interference of the computing elements that are not part of the abstract processor *AbsCPU*. To achieve this, we take the following precautions:

1) We ensure the platform is reserved exclusively and fully dedicated to our experiments.
2) We monitor the disk consumption before and during the application run and ensure that there is no I/O performed by the application using tools such as *sar*, *iotop*, and so forth.
3) We ensure that the problem size used in the execution of an application does not exceed the main memory and that swapping (paging) does not occur.
4) We ensure that the network is not used by the application by monitoring using tools such as *sar*, *atop*, etc.
5) We set the application kernel's CPU affinity mask using SCHED API's system call SCHED_SETAFFINITY. Consider for example MKL DGEMM application kernel running on only abstract processor *A*. To bind this application kernel, we set its CPU affinity mask to 12 physical CPU cores of Socket 1 and 12 physical CPU cores of Socket 2.
6) Fans are also a great contributor to energy consumption. On our platform fans are controlled in two zones: (a) zone 0: CPU or System fans, (b) zone 1: Peripheral zone fans. There are 4 levels to control the speed of fans:
   - Standard: BMC control of both fan zones, with CPU zone based on CPU temp (target speed 50%) and Peripheral zone based on PCH temp (target speed 50%)
   - Optimal: BMC control of the CPU zone (target speed 30%), with Peripheral zone fixed at low speed (fixed 30%)
   - Heavy IO: BMC control of CPU zone (target speed 50%), Peripheral zone fixed at 75%
   - Full: all fans running at 100%

   In all speed levels except the full, the speed is subject to be changed with temperature and consequently, their energy consumption also changes with the change of their speed. Higher the temperature of CPU, for example, higher the fans' speed of zone 0 and higher the energy consumption to cool down. This energy consumption to cool the server down, therefore, is not consistent and is dependent on the fans' speed and consequently can affect the dynamic energy consumption of the given application kernel.

   Hence, to rule out the fans' contribution to dynamic energy consumption, we set the fans at full speed before launching the experiments. When set at full speed, the fans run consistently at a fixed speed until we do so to another speed level. Hence, fans consume the same amount of power which is included in the static power of the platform.
7) We monitor the temperature of the platform and speed of the fans (after setting it at full) with help of Intelligent Platform Management Interface (IPMI) sensors, both with and without the application run. We find no considerable difference in temperature and find the speed of fans the same in both scenarios.

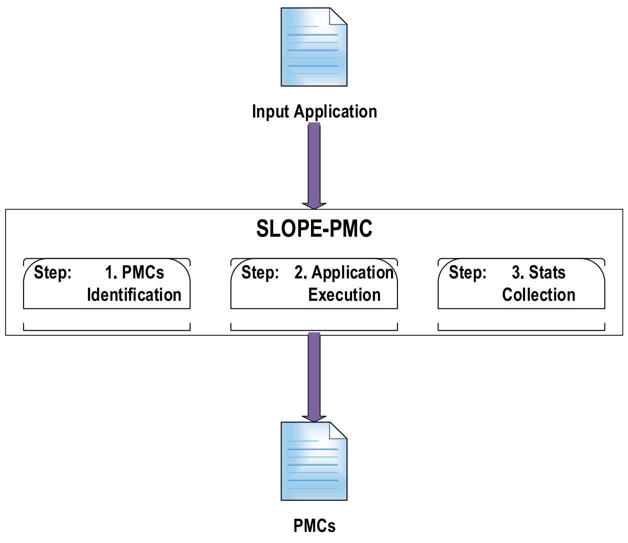

Input Application
SLOPE-PMC
Step: 1. PMCs Identification
Step: 2. Application Execution
Step: 3. Stats Collection
PMCs

FIGURE 11. ***SLOPE-PMC*: Towards the automation of PMC collection on Modern Computing Platforms.**

Thus, we ensure that the dynamic energy consumption obtained using HCLWattsUp reflects the contribution solely by the *abstract processor* executing the given application kernel.

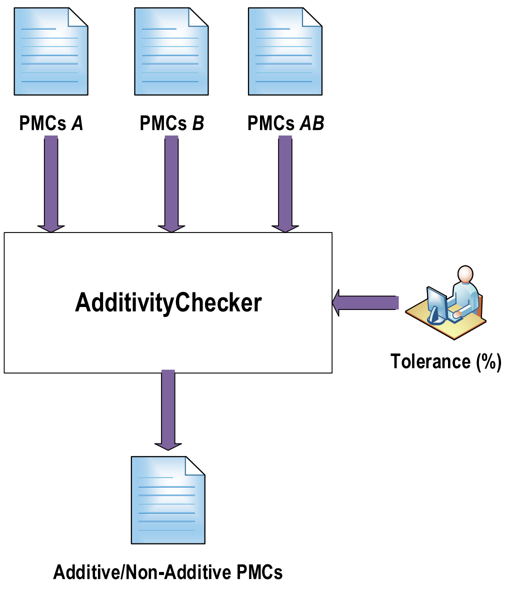


FIGURE 12. ***AdditivityChecker*: Test PMCs for *Additivity*.**

```
$ likwid-perfctr -a

  Group name      Description
-----------
       ----------------------------------------
       BRANCH     Branch prediction miss rate/ratio
       CACHES     Cache bandwidth in MBytes/s
         CBOX     CBOX related data and metrics
        CLOCK     Power and Energy consumption
         DATA     Load to store ratio
       ENERGY     Power and Energy consumption
  FALSE_SHARE     False sharing
    FLOPS_AVX     Packed AVX MFLOP/s
           HA     Main memory bandwidth in MBytes/s
                  seen from Home agent
       ICACHE     Instruction cache miss rate/ratio
           L2     L2 cache bandwidth in MBytes/s
      L2CACHE     L2 cache miss rate/ratio
           L3     L3 cache bandwidth in MBytes/s
      L3CACHE     L3 cache miss rate/ratio
          MEM     Main memory bandwidth in MBytes/s
         NUMA     Local and remote memory accesses
          QPI     QPI Link Layer data
     RECOVERY     Recovery duration
         SBOX     Ring Transfer bandwidth
     TLB_DATA     L2 data TLB miss rate/ratio
    TLB_INSTR     L1 Instruction TLB miss rate/ratio
         UOPS     UOPs execution info
    UOPS_EXEC     UOPs execution
   UOPS_ISSUE     UOPs issueing
  UOPS_RETIRE     UOPs retirement
 CYCLE_ACTIVITY   Cycle Activities
```

FIGURE 13. **List of PMC groups provided by Likwid tool on HCLServer2.**

APPENDIX VII. BRIEF OVERVIEW OF *SLOPE-PMC* AND *AdditivityChecker*

SLOPE-PMC is developed on top of Likwid tool to automate the process of PMC collection. It takes an input application and operates in three steps. First, it identify the available PMCs on a given platform and list them in a file. In second step, the input application is executed several times as in a single invocation of an application only 4 PMCs can be collected. To ensure reliable results, we also take an average of each PMC count using multiple executions (atleast 3) of an application. In the final step, the PMCs are extracted with labels in a stats file. Figure 11 summarizes the work-flow of *SLOPE-PMC*.

Figure 12 describes the *AdditivityChecker* where it takes as an input: 1). PMCs of two base applications *A* and *B*, and a compound application (*AB*) composed of base applications and 2). user-specified tolerance in percentage. It returns a list of *additive* and *non-additive* PMCs along with their percentage errors.

APPENDIX VIII. LIST OF PMC GROUPS PROVIDED BY LIKWID

The list of PMC groups provided by Likwid tool [16] on HCLServer2 is shown in the Figure 13.

REFERENCES

[1] N. Jones, "How to stop data centres from gobbling up the world's electricity," *Nature*, vol. 561, no. 7722, pp. 163–166, 2018.

[2] A. Andrae and T. Edler, "On global electricity usage of communication technology: Trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, Apr. 2015.

[3] 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.

[4] 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*, Aug. 2009, pp. 99–104.

[5] W. Wang, P. Mishra, and S. Ranka, "Dynamic cache reconfiguration and partitioning for energy optimization in real-time multi-core systems," in *Proc. 48th Design Autom. Conf. (DAC)*, Jun. 2011, pp. 948–953.

[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, Nov. 2012.

[7] 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.

[8] 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, pp. 1–26, Jan. 2015.

[9] 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.

[10] J. Demmel, A. Gearhart, B. Lipshitz, and O. Schwartz, "Perfect strong scaling using no additional energy," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, May 2013, pp. 649–660.

[11] J. Lang and G. Rünger, "An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration," *J. Parallel Distrib. Comput.*, vol. 74, no. 9, pp. 2884–2897, Sep. 2014.

[12] 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.

[13] A. Lastovetsky and R. Reddy 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.

[14] 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.

[15] 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, Jun. 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/11/2204

[16] 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*, Sep. 2010, pp. 207–216.

[17] J. Mair, Z. Huang, and D. Eyers, ''Manila: Using a densely populated PMC-space for power modelling within large-scale systems,'' *Parallel Comput.*, vol. 82, pp. 37–56, Feb. 2019.

[18] Z. Zhou, J. H. Abawajy, F. Li, Z. Hu, M. U. Chowdhury, A. Alelaiwi, and K. Li, ''Fine-grained energy consumption model of servers based on task characteristics in cloud data center,'' *IEEE Access*, vol. 6, pp. 27080–27090, 2018.

[19] 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.

[20] A. Shahid, M. Fahad, R. Reddy, and A. Lastovetsky, ''Additivity: A selection criterion for performance events for reliable energy predictive modeling,'' *Supercomput. Frontiers Innov.*, vol. 4, no. 4, pp. 50–65, Dec. 2017.

[21] A. Shahid, M. Fahad, R. Reddy Manumachu, and A. Lastovetsky, ''Energy of computing on multicore CPUs: Predictive models and energy conservation law,'' 2019, *arXiv:1907.02805*. [Online]. Available: http://arxiv.org/abs/1907.02805

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

[23] J.-A. Rico-Gallego, J.-C. Díaz-Martín, and A. L. Lastovetsky, ''Extending τ-Lop to model concurrent MPI communications in multicore clusters,'' *Future Gener. Comput. Syst.*, vol. 61, pp. 66–82, Aug. 2016. [Online]. Available: https://doi.org/10.1016/j.future.2016.02.021

[24] H. Abdi and L. J. Williams, ''Principal component analysis,'' *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.

[25] 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.

[26] 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.

[27] Nvidia. (Oct. 2018). *Nvidia Management Library: NVML Reference Manual*. [Online]. Available: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf

[28] I. Corporation, ''Intel xeon phi coprocessor system software developers guide,'' Tech. Rep., Jun. 2014.

[29] M. Burtscher, I. Zecena, and Z. Zong, ''Measuring GPU power with the K20 built-in sensor,'' in *Proc. Workshop Gen. Purpose Process. Using GPUs*, Mar. 2014, p. 28.

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

[31] 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

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

[33] L. Benini, A. Bogliolo, S. Cavallucci, and B. Riccó, ''Monitoring system activity for OS-directed dynamic power management,'' in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, 1998, pp. 185–190.

[34] Y.-H. Lu, L. Benini, and G. De Micheli, ''Operating-system directed power reduction,'' in *Proc. ISLPED00: Proc. Int. Symp. Low Power Electron. Design*, Jul. 2000, pp. 37–42.

[35] L. Benini, A. Bogliolo, and G. De Micheli, ''A survey of design techniques for system-level dynamic power management,'' *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 3, pp. 299–316, Jun. 2000.

[36] 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 Operating Syst.*, 2000, pp. 37–42.

[37] C. Isci and M. Martonosi, ''Runtime power monitoring in high-end processors: Methodology and empirical data,'' in *Proc. 22nd Digit. Avionics Syst. Conf.*, Dec. 2003, p. 93.

[38] 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, doi: 10.1145/885651.781048.

[39] B. C. Lee and D. M. Brooks, ''Accurate and efficient regression modeling for microarchitectural performance and power prediction,'' *ACM SIGPLAN Notices*, vol. 41, no. 11, pp. 185–194, Nov. 2006.

[40] T. Heath, B. Diniz, E. V. Carrera, W. M. 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.

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

[42] 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.

[43] A. Kansal and F. Zhao, ''Fine-grained energy profiling for power-aware application design,'' *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 2, p. 26, Aug. 2008.

[44] S. Rivoire, P. Ranganathan, and C. Kozyrakis, ''A comparison of high-level full-system power models,'' in *Proc. Conf. Power Aware Comput. Syst.*, 2008, pp. 32–39.

[45] S. Rivoire, ''Models and metrics for energy-efficient computer systems,'' Ph.D. dissertation, Dept. Elect. Eng., Stanford Univ., Stanford, CA, USA, 2008.

[46] 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.

[47] 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.

[48] 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.* Chicago, IL, USA: IEEE Press, 2010, pp. 135–146.

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, ''Statistical power modeling of GPU kernels using performance counters,'' in *Proc. Int. Conf. Green Comput.*, Aug. 2010, pp. 115–122.

[54] 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.*, May 2013, pp. 673–686.

[55] 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.

[56] 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.* Darmstadt, Germany: Springer, 2015, pp. 65–76.

[57] M. Witkowski, A. Oleksiak, T. Piontek, and J. Węglarz, ''Practical power consumption estimation for real life HPC applications,'' *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 208–217, Jan. 2013.

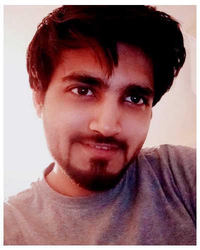
[58] M. Jarus, A. Oleksiak, T. Piontek, and J. Węglarz, ''Runtime power usage estimation of HPC servers for various classes of real-life applications,'' *Future Gener. Comput. Syst.*, vol. 36, pp. 299–310, Jul. 2014.

[59] 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, Oct. 2016.

[60] 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.*, Feb. 2014, pp. 536–543.

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

[62] D. Hackenberg, T. Ilsche, R. Schone, D. Molka, M. Schmidt, and W. E. Nagel, "Power measurement techniques on standard compute nodes: A quantitative comparison," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Apr. 2013, pp. 194–204.
[63] 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. Surveys*, vol. 50, no. 3, pp. 1–38, Oct. 2017.
[64] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "Improving the accuracy of energy predictive models for multicore CPUs using additivity of performance monitoring counters," in *Parallel Computing Technologies*, V. Malyshkin, Ed. Cham, Switzerland: Springer, 2019, pp. 51–66.
[65] H. C. Laboratory and U. C. Dublin. (2020). *HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter*. [Online]. Available: https://csgitlab.ucd.ie/ucd-hcl/hclwattsup
[66] A. Liaw and M. Wiener, "Classification and regression by randomforest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.
[67] F. Dan Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 3, 1997, pp. 1930–1935.
[68] J. J. Moré, "The Levenberg–Marquardt algorithm: Implementation and theory," in *Proc. Conf. Numer. Anal.*, Dundee, U.K. New York, NY, USA: Springer, 1978, pp. 105–116.



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. His research interests include energy-aware high-performance heterogeneous computing and intelligent cloud computing.

MUHAMMAD FAHAD received the B.S. degree from International Islamic University, Islamabad, Pakistan, in 2008, and the M.S. degree from the KTH Royal Institute of Technology, Sweden, in 2012. He is currently a Ph.D. Researcher with the Heterogeneous Computing Laboratory (HCL), University College Dublin, Ireland. 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, distributed computing, energy-efficient computing, and sparse matrix computations.

ALEXEY LASTOVETSKY received the Ph.D. degree from the Moscow Aviation Institute, in 1986, and the Doctor of Science degree from the Russian Academy of Sciences, in 1997. He has published over a 100 technical papers in refereed journals, edited books, and international conferences. He has authored the monographs *Parallel Computing on Heterogeneous Networks* (Wiley, 2003) and *High-Performance Heterogeneous Computing* (Wiley, 2009). His main research interests include algorithms, models, and programming tools for high-performance heterogeneous computing.