PMC-based Power Modelling with Workload Classification on Multicore Systems

Jason Mair, Zhiyi Huang, David Eyers and Haibo Zhang
Department of Computer Science
University of Otago
Dunedin, New Zealand
Email: {jkmair;hzy;dme;haibo}@cs.otago.ac.nz

***Abstract*—In this paper, we propose a PMC-based power modelling methodology that utilizes workload classification. Unlike traditional approaches which use a limited set of benchmarks, our methodology uses a single well-designed micro-benchmark to collect samples of PMCs and power values for training the power estimation model. The micro-benchmark can generate a large variety of representative workloads that are generic in a wide range of applications. Since the micro-benchmark is independent from any applications but includes generic workloads of many applications, our methodology is more widely applicable than the approaches based on a limited set of benchmarks that may have similar workloads. Another novelty of our methodology is that it adopts workload classification. Traditional approaches usually use multi-variable linear regression to correlate PMCs with power for all types of workloads. Since different PMCs may correlate well with power under different workloads, using a single linear multi-variable function to model power is insufficient and ineffective. In our methodology, we classify the workloads and for each workload we use a different, independent linear function to model the relationship between PMCs and power. In this way, the resulting power model is refined and its accuracy of power estimation can be increased. Based on our methodology, we have implemented a power estimation model called W-Classifier. Experimental results show that W-Classifier can estimate power usage well for a larger variety of workload types than the traditional approaches with a single multi-variable linear regression function.**

I. Introduction

Given the increasing global power use of computer technology, there is a continuing drive to develop more energy efficient and sustainable systems [1]–[8]. One proposed software approach is to make the operating system more power-aware. This would allow the operating system to not only allocate system resources based upon performance, but to additionally consider the power use of the system in making decisions. The operating system is required to be capable of evaluating the tradeoff between performance and power use, ensuring the system does not consume excessive power. It is desirable that power caps could be enforced from software, setting a maximum power value for the entire system or for a virtual machine in a cloud, or on a per application basis. However, such developments are dependent upon the availability of fine-grained, run-time power measurements on a per application basis.

Unfortunately, making fine-grained, run-time power measurements is not possible with current hardware. Existing power meters that measure power use from the mains outlet, can only provide coarse-grained power measurements for the whole system. Hardware-based power sensors embedded in the Intel Sandy Bridge micro-architecture provide power measurements limited to the entire processor, not per-core [9], [10]. While hardware-based power measuring techniques have the potential to provide per-core power readings to the operating system, they suffer from two key drawbacks. First, the cost of upgrading or replacing system hardware will be prohibitive for widespread deployment and adoption. Second, hardware-based power measurements currently do not provide any of the required context for the root causes of power use, e.g., which functional units are drawing power within the CPU cores. Depending on the type of the current workload, for the same power reading, the operating system may actually want to use a different power saving policy. For instance, a low CPU power reading could be caused by a memory-bound workload, which is thus experiencing low CPU utilization. In contrast, the same measurement might indicate a CPU-bound workload experiencing a period of thread synchronization, which will only cause low CPU utilization to occur temporarily. For a memory-bound workload, the operating system can often lower the CPU operating frequency to save power without impacting the overall performance, but this option is not suitable for the CPU-bound workload. However, without the context of workload classification, it is hard for a power-aware operating system to reliably evaluate the power and performance tradeoffs and to make the most appropriate power saving decision.

Classifying the type of workload of an application is possible through the use of the low-level performance monitoring counters (PMCs), which are a set of hardware registers designed to monitor and record detailed performance events. Each model of CPU has a specific set of PMCs. Such events include among many others, floating point unit (FPU) instructions, retired micro-operations (retired UOPS), L2 cache misses (l2-cache misses) and dispatch stalls. It is this unique and recent ability to provide low-level performance events that has led to the development of PMC-based power estimation models [8], [11]–[15]. The principle technique of these models is to find the subset of all available PMCs for which a change in performance strongly correlates to an observed change of the measured power. For example, the model in [11] decomposes the set of PMCs into four categories, based on factors assumed to connect to the operation of the processor's main functional units: FP Units, Memory, Stalls, and Instructions Retired. The four PMCs that are found to most strongly correlate to each functional unit are combined into a single multi-variable linear equation that is used for run-time power estimation. While

these approaches use PMCs for deriving power estimation models, they do not use these PMC values in performing any workload classification to add extra context to the power estimation.

In this paper we present an alternative methodology for PMC-based power estimation, designed around the requirement for workload classification to add context to power estimation. The model works by first classifying the workload type for a given PMC sample. It then uses an individual linear function to estimate power for each specific workload. The use of workload classification provides the extra contextual information for the operating system. Based on the workload classification, the operating system can more efficiently make power-aware decisions on resource allocation, e.g. task scheduling. In addition, the workload classification allows our power estimation model to be more general and applicable to a larger variety of applications. Traditional power models based on multi-variable linear regression have to generalize across all possible workloads, whereas, in our approach, we are able to apply a different multi-variable equation to each workload.

The remainder of this paper is organized as follows. We present our power modeling methodology in Section II. The experimental setup and configuration is given in Section III. Section IV evaluates and analyses our power estimation model with the NAS Parallel Benchmark suite [16]. Related work is discussed in Section V, while our conclusions and future work is presented in Section VI.

II. Methodology

The methodology proposed in this paper derives a collection of workload-specific, linear functions for power estimation. The main principle behind this methodology is that classifying a given workload type allows for the use of a specific linear function for power estimation of that workload type, rather than relying on a single generalized linear function for all workload types. This model of multiple linear functions has the distinct advantage of being able to estimate power for a much broader range of applications than would otherwise be possible, because it is not feasible to use a single linear function to characterize the power features of various workload types in a wide range of applications. For example, floating point calculations consume power that is linear to the number of FPU instructions, but these calculations are less related to the number of cache misses. Data movement in RAM consumes power that is linear to the intensity of the memory accesses, but the memory operations are largely independent of the FPU operations. Deriving a general linear function with the number of FPU instructions and the number of memory accesses (or cache misses) is not as effective as using two linear functions for the two workload types respectively. Since each linear function has better linear correlation with its corresponding workload type (as we demonstrate in section IV), it can produce more accurate power estimates than the single general linear function whose linear correlation with both workload types is compromised due to the generalization.

Our power model with workload classification is able to operate over a diverse range of applications. A general classification is applied to detect, e.g., FPU intensive or memory intensive workloads. This information helps our power model meet the more general objective of improving power-aware task schedulers, by providing extra contextual information for power measurements, i.e., what workload the power is used for.

A. Micro-benchmark

For the derived power estimation model to remain applicable to different architectures, the training data must be parameterised so as to produce the widest possible range of workloads. A single benchmark suite is not likely to provide this generality. Therefore, our model is derived using a single micro-benchmark designed to reproduce a selection of synthetic workloads. While some previous work [11] used two sets of benchmarks, one for selection of correlated PMCs and the other for deriving the estimation model with linear regression, we are able to achieve both objectives with the use of a single micro-benchmark with a parameterized design. This makes our methodology more readily deployable on different systems as there is no requirement for designing new micro-benchmarks based upon the results of the previous steps in the modeling process, which thus allows the modeling process to be automated.

To achieve this, our micro-benchmark is designed to ensure enough workloads are reproduced in order to provide a set of representative workload samples within the expected range of applications. Otherwise the model will be incapable of estimating power for a wide range of applications, as a strong bias towards a few reproduced workloads would exist. However, this does not necessarily mean a large number of workloads have to be used. Based on our experimental analysis, the five workload types in Table I provide a representative sample of workloads commonly encountered in a general system. These five key workloads are built into the micro-benchmark shown in Program 1, though our micro-benchmark can be easily extended with new workloads in order to support new, yet unseen, machine architectures.

The micro-benchmark structure consists of a larger outer loop, intended to perform a large number of iterations, and a small `for` loop for short bursts of each workload type. While the loop control will create some minor overhead, it has little impact on the workload type and in any case reflects the structure of most real-world applications. The inner `for` loops execute quickly, ensuring that a mix of workload types will be represented whenever the PMCs are sampled, i.e. once a second. Modest variations in this burst time are provided by the pseudo-random numbers. This is then multiplied by a ratio, which is designed to allow extra weight to be added to a particular workload type during power modeling, and is the key parameterization of our micro-benchmark.

TABLE I. Types of workload

workload type	Description
FPU	Floating point operations
INT	Integer operations. Represents most micro operations
memory	Random memory accesses
NOP	Idle loop with NOPs
cache	Memory accesses with high cache-hit ratio

By default, all ratios are set to one. However, if we are interested in the impact of a particular workload, e.g. FPU

```
for large_number_of_iterations do
    for pseudo_random_number_n× fpu_ratio do
        fpu_microbenchmark();
    end for
    for pseudo_random_number_{n+1}× int_ratio do
        int_microbenchmark();
    end for
    for pseudo_random_number_{n+2}× memory_ratio do
        memory_microbenchmark();
    end for
    for pseudo_random_number_{n+3}× nop_ratio do
        nop_microbenchmark();
    end for
    for pseudo_random_number_{n+4}× cache_ratio do
        cache_microbenchmark();
    end for
end for
```

Program 1: Pseudo-code of the micro-benchmark

operations, we increase the *fpu_ratio*, leaving all other ratios unchanged. This allows each workload type to be analyzed independently. Note that the code is supposed to run simultaneously on multicore systems. Though it does not incur any synchronization overhead, the synchronization workload can be represented with a combination of NOP and memory-access workloads.

B. Deriving our PMC-based Power Estimation Model

The principle behind our PMC-based power estimation model is to find the relationship between key low-level performance events, the type of workload they represent and the corresponding power use. For example, if the PMC for FPU utilization is high, the current workload is FPU intensive, causing a large power draw. Alternatively, if the number of cache misses is high, there may be many memory accesses, resulting in lower utilization of the processor and a lower power draw from the processor. In quantifying the strength of such relations, an analytical model can be formed, allowing run-time power estimation.

To assess the relationship between PMCs and workload classifications, the micro-benchmark is run with multiple iterations under different workload ratios. Adjusting the ratio for a single workload type, while holding all other ratios constant, allows the relative impact for the given workload to be analyzed independently. To further isolate these results from external influences, two additional steps are taken to mitigate the impact of temperature changes during execution [17]. First, each micro-benchmark iteration is configured with a large loop count, resulting in a long execution time (about 25 minutes). This ensures a stable operating temperature is reached and maintained for the majority of the execution. Second, several minutes of data are removed from the start of each iteration to eliminate the effect of a slow processor warm-up. We have written in more detail [17] specifically about the effects that different types of temperature variation have on CPU power consumption.

The PMC-based power model is derived from the periodically collected values for a given PMC. The raw PMC, TSC (Time Stamp Counter) and power-meter measurements are logged to a file once a second during the execution of the micro-benchmark. This enables the intensity values for the event of each PMC to be calculated by taking the difference in two adjacent PMC samples and dividing by the corresponding difference in the TSC. Since the TSC provides a low-level time value, which is recorded as the number of cycle counts since the CPU was reset, the intensity of each PMC can accurately be calculated for every sampling period (i.e., one second).

Since it is not known which PMCs will strongly correlate with the power use of a specific workload, it is important to use a large variety of PMCs during initial data collection. However, some architectures support more than 120 event counters [18], making it prohibitive to perform exhaustive analysis. Nevertheless, a representative selection of counters can be used in any case. For example, different levels of cache and memory accesses, retired instructions, processor pipeline and any specialized functional units like FPU will provide metrics related to potentially orthogonal causes of power consumption within the CPU. The more diverse the range of used PMCs are, the more likely the model is to accurately estimate power for workloads across a range of applications.

Having collected all of the required sample data, the next step is to determine which PMCs best represent each workload. This can readily be achieved by calculating the Spearman rank correlation coefficient for each workload-PMC-power combination. Spearman's rank correlation quantifies how well the relationship between two variables can be described using a monotonic function.

With the PMCs having the strongest correlation with power, for a given workload type, the final power estimation model can be derived using linear least squares fitting. Linear least squares fitting fits a regression line through the data points such that it minimizes the sum of the squared differences between the modeled and observed values. However the model can be impacted by outliers, so the extra step of removing points that are more than three standard deviations away from the mean should be taken first to improve accuracy. Since our methodology has no strict requirement on the form of the model for each workload, different workloads can be modeled with different PMCs and regression functions. While a regression function with a single variable may be sufficient for some workloads, our methodology allows the use of multiple variables for other workloads in their regression functions if it is necessary for them. This flexibility additionally allows our methodology to work on a variety of architectures. For instance, some architectures may provide performance events for monitoring memory accesses, thereby only requiring a single variable in the regression model of a memory workload. Alternatively, an architecture without this event can use multiple counters to achieve the same result.

C. Workload Classification

While decomposing the power model into a collection of workload-specific linear functions will help to improve the accuracy of power estimation across a range of applications, it creates the additional requirement for accurate, run-time workload classification. Though it may seem that significantly more data will be needed for workload classification, the classification can be achieved based on the existing set of PMC sample data.

To illustrate the use of PMC data, Figure 1 plots the least squares fitting functions, each of which models the data points of each workload type independently. The figure shows the relationship between the FPU event (PMC_{FPU}) intensity and the power for each of the five workload types of the micro-benchmark in Program 1. The x-axis presents the PMC_{FPU} intensity, calculated as the difference in two adjacent PMC_{FPU} values, divided by the difference in the corresponding TSC values. The y-axis is the whole system power measurement at the time when the PMC_{FPU} intensity is collected.

It can be seen in Figure 1 that there is a central crossing point for all workloads around the PMC_{FPU} intensity of 1.8. This is due to the fact that all workload types start to execute with the same parameterization, i.e., with a default value of 1 for all ratios, such as *int_ratio*, in Program 1. As the ratio of a specific workload is changed, the results begin to radiate out in different directions, depending on the corresponding workload type. The fitted line for FPU workload is the only one to show a strong correlation between the power and the PMC_{FPU} intensity while the FPU workload is dominant, i.e., $PMC_{FPU} \geq 1.8$. That means, for FPU workload, *FPU_ratio* keeps increasing from 1, while the other ratios such as *int_ratio* are kept to the initial value 1. In this way, the relative weight of the FPU workload is increasing. We carry out the same process for other workloads as well.

Figure 1 also shows the fitted lines when other workload types are dominant. Since those other workloads are dominant, the PMC_{FPU} intensity becomes relatively small, i.e., $PMC_{FPU} < 1.8$.

This observation is true for other workloads, which allows us to set a classification threshold for each workload type. For example, if the PMC_{FPU} intensity is greater than 1.8, we can classify the current workload as FPU workload and use the longest fitted line in Figure 1 to get the power estimation.

With the clear separation of the fitted lines for different workload types in Figure 1, observant readers may spot that we can potentially use only the PMC_{FPU} to get power estimation for all workloads. This is true to some extent. However, it is important to note that the PMC_{FPU} intensity has a very small range for other workload types but with a large span of power variation. This means the accuracy of the power estimation using PMC_{FPU} for other workloads could be a problem because a small error in collecting the PMC_{FPU} intensity will cause large fluctuations in the power estimation. Also the reason that the PMC_{FPU} values are always present in Figure 1 is due to the mixed workloads in the micro-benchmark. This will not be true for real applications as they may not have any FPU operations at all. Therefore, a different workload needs to be modeled by other PMCs that can represent the workload and have stronger correlation with the power when the workload is dominantly present.

The above workload classification could be further improved by considering multiple dominant workloads. In our implemented model, we only collect sample data using a single dominant workload at a time. That means, only five workloads are explicitly considered. However, a greater mix of workloads may occur during application execution than was specifically reproduced by our micro-benchmark. An example of this

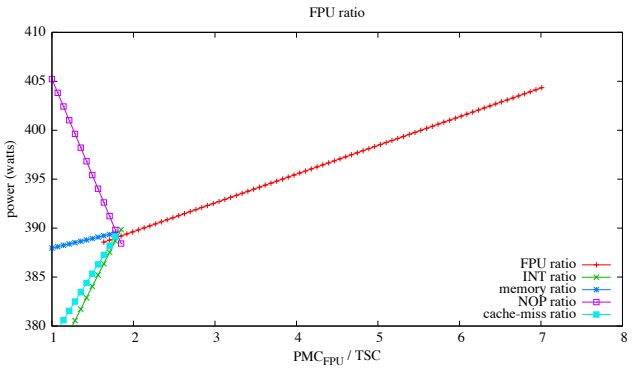

Fig. 1. regression lines for each micro-benchmark ratio and PMC_{FPU}

would be cache misses, which are not likely to be the primary workload type, but would occur jointly with compute tasks, such as INT and FPU workloads. Therefore, we can supplement the classification with these more practical expectations. In our implementation below, we have considered two mixed workloads, FPU/cache and INT/cache, which include the effect of cache on FPU or INT operations using the collected micro-benchmark data.

Furthermore, as a future work, it would most likely be more accurate if we proceed to consider additional combinations of workloads, e.g. a mix of FPU and INT operations that are both dominant. In this way, we can have more classes of workloads and model them separately. For the mixed workload, we will need multiple PMC values for linear regression. For example, for a mixed FPU and INT workload, we will need to use both PMC_{FPU} and the PMCs representing the intensity of INT operations to model the power. Though we have not implemented such fine-grained classification, our proposed methodology is general enough to be useful and applicable when the workload classification is further refined. However, it is worth considering that as more classifications are used, the additional run-time overhead will begin to outweigh the increased accuracy for power estimation.

The use of workload classification enables our power model to accurately estimate power for a large variety of workload types and applications, as will be shown in our experiments (§IV). It additionally helps serve a broader objective of providing execution context for power use, which can be utilized by a power-aware operating system in evaluating power saving policies. This is very important because accurate power estimation is not all that is required in developing a truly energy efficient system.

Since our power model is more general and adaptable, it is more robust to architectural changes than traditional multi-variable models that use a single linear function to generalize the relationship between performance events and power. This architectural independence ensures it will remain usable on different system architectures and will be able to capitalize on any future architectural changes, such as new functional units or an increased selection of PMCs. For example, if a new independent functional unit is added to a system architecture, we could simply model its power with an independent linear function using the performance events related to the unit.

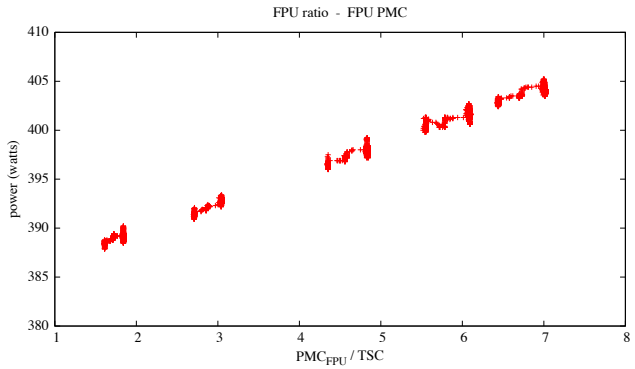

Fig. 2. Scatter plot for the correlation between PMC_{FPU} and power for FPU dominant workload

D. Methodology Implementation

The implementation of the methodology proposed in this paper collected all PMC sample data from 16 concurrent instances of the micro-benchmark given in Program 1, configured to use workload ratios of 1, 2, 4, 6 and 8. In an attempt to mitigate the impact of the temperature changes of the processor during execution, the micro-benchmark was additionally configured with a large iteration count, causing execution times of about 25 minutes for each experiment. Also, the first couple of minutes is trimmed from the sampled data logs in order to remove the effect of processor warm-up [17].

The set of 13 PMCs (shown in Table II) and the power meter measurements were logged to a file once every second during execution. This specific set of PMCs was chosen for evaluation, as they provide a representative sample of system performance events for our AMD multicore machine, ranging from memory accesses to the various levels of the cache hierarchy to processor utilization.

TABLE II. PMCs POLLED WITHIN OUR EXPERIMENTS

1: FPU	2: Data cache miss
3: dispatch stalls	4: Instruction cache miss
5: CPU clocks not halted	6: l2-cache miss
7: Retired Move ops	8: l3-cache miss
9: Prefetch Instructions Dispatched	10: DTLB miss
11: Retired uops	12: DRAM access
13: Retired Branch Instructions	

After collecting all of the sample data, we analyze the strength of relationship between the workload type and the PMC intensity with Spearman's rank correlation, gaining further insight by visualizing the relationship on a scatter plot. For example, Figure 2 plots the PMC_{FPU} intensity values and their corresponding power readings, with the *FPU_ratio* increased from 1 in the micro-benchmark. The x-axis shows the intensity of FPU events, calculated by taking the difference in two adjacent PMC_{FPU} measurements and dividing by the difference in the corresponding TSC values. The y-axis is the corresponding whole system power measurement from the power meter. The modest spread of data points within each cluster is due to the micro-benchmark pseudo random-numbers, while the spread of clusters is a result of the variations in *FPU_ratio*.

The data points in Figure 2 illustrates the existence of a strong linear relationship between the PMC_{FPU} intensity and the power for a FPU dominant workload. It can also be concluded from the strength of this relationship that a single PMC is able to represent the power use for a dominant workload. In analyzing the remaining PMCs and workload types, similar linear relationships were found for each workload type, although not all relationships were as expected. For example, we were surprised to find that memory related PMCs, such as l2-cache-misses, did not show a meaningful relationship with memory-intensive workloads, which we have presented and discussed in our previous work [17]. This anomaly is mentioned here as it helps illustrate the importance of selecting a wide range of PMCs, including PMCs other than those expected to correlate well, when initially evaluating performance events for the power model.

Based on the strength of the determined relationships, the four PMCs to be used in the power model are:[1]

- **FPU**—the number of cycles in which at least one FPU operation is present in the FPU.
- **Dispatch stalls**—the number of processor cycles where the decoder is stalled for any reason (i.e., it has one or more instructions ready but cannot dispatch them due to resource limitations in execution).
- **CPU clocks not halted**—the number of clocks that the CPU is not in a halted state (due to `STPCLK` or a `HLT` instruction).
- **Retired UOPS**—the number of micro-operations retired. This includes all processor activity (instructions, exceptions, interrupts, microcode assists, etc.).

Only four PMCs are chosen as this is the architectural limit for the number of counters which are able to be read simultaneously on our system. Having selected the counters, each workload-specific linear function is derived by applying linear least squares fitting to the appropriate PMC-power data sets for the specific workload.

Since each workload uses a different linear function for power estimation, a decision has to be made regarding which workload the PMC values represent once the PMC values are collected during power estimation. Given that each selected performance event is usually the primary descriptor of a workload type, it can be used as an indicator of the corresponding workload type. For example, the *FPU* PMC can be used to represent FPU-dominant workload, the PMC for *Retired UOPS* can be used to represent INT-dominant workload, and *Dispatch stalls* can suggest cache-dominant workload.

The threshold, unique to a specific workload, can be determined by identifying the boundary point within each data set, indicating the point of separation between the dominant workload and other workloads. For example, the threshold for an FPU-dominant workload can be identified as any $PMC_{FPU} \geq 1.8$, as was previously shown in Figure 1. In this case, a clear point of separation between the dominant workload to be classified and all other workloads can readily be identified with the collected PMC_{FPU}. Alternatively, for a cache-dominant workload, the boundary threshold will be

[1]PMC descriptions are taken from the AMD BIOS and Kernel Developers Guide [**?**]

determined by the maximum PMC_{cache} for the computation-intensive workloads. Any PMC_{cache} values greater than this threshold will indicate a cache/memory-dominant workload. Applying this general procedure to each workload results in a set of thresholds. For example, we have determined the thresholds T_{FPU} (1.8), T_{cache} (10.0), and T_{INT} (1.0) used in Program 2 according to the boundary points within the data sets. They can be used to classify FPU-dominant, cache-dominant, and INT-dominant workloads respectively. They can also be used combinatorially to represent mixed workload such as both INT-dominant and cache-dominant.

The general threshold determination procedure outlined above enables a rudimentary workload classification algorithm to be derived. The pseudo-code for the derived classification algorithm is presented in Program 2.

```
if PMC_cache > T_cache then
    if PMC_FPU > T_FPU then
        mixed FPU and cache workload
    else if PMC_INT > T_INT then
        mixed INT and cache workload
    end if
else if PMC_FPU > T_FPU then
    FPU workload
else if PMC_INT > T_INT then
    INT workload
else
    memory/idle workload
end if
```

Program 2: Pseudo-code for workload classification

As discussed before, considering mixed workloads can help improve accuracy. We have included two mixed workloads, FPU/cache and INT/cache, in our classification algorithm so that power estimation can be more accurate with little extra overhead in the algorithm.

In future implementations, the selection of threshold values may be aided by the use of machine learning, such as support vector machines or decision trees, for the classification algorithm. These techniques will likely be required when processing significantly larger data sets that may be caused by increases in the number of performance events or types of workload. Our current implementation does not suffer from these restrictions, as it uses a limited set of performance events that represent a limited set of workloads. This allows a manual selection of threshold values from the input data sets. It also allows insights to be gained which would not have been possible with a black box approach like a blind multi-variable linear regression. Despite these details being implementation specific, they do not impact on the suitability of the general modeling methodology proposed in this paper.

Our resulting power model and the classification algorithm are evaluated in Section IV.

III. Experimental Setup

Our power model is evaluated on a Dell PowerEdge R905 with four quad-core AMD Opteron 8380 processors. Each of the 16 cores has its own FPU and an architectural limit of four PMCs. The machine has a total of 16GiB RAM organized in a NUMA (Non-Uniform Memory Accesses) architecture, with 4GiB of RAM allocated to each processor. Each core has four alternate operating frequencies through DVFS (Dynamic Voltage and Frequency Scaling). However, we restrict the frequency to the highest (2.5 GHz) for our experiments in this paper, as it is the most commonly used frequency in practice.

The micro-benchmark is compiled using gcc 4.6.3, with no optimizations, so as to ensure that none of the workload operations are optimized away. The power model is evaluated using the OpenMP benchmarks from the NAS Parallel Benchmark suite. Each benchmark is run with 16 threads, utilizing all available cores, configured with the problem sizes shown in Table III. The benchmarks were compiled with gcc 4.6.3, using OpenMP 3.0 [19] and the optimization argument `-O3`. All benchmarks are running on a standard installation of Linux version 2.6.32-25.

The power is measured with the Watts Up? PRO .net power meter, connected via USB to an external monitoring system. The accuracy of the power meter is $\pm$1.5% + 0.3 watts. [2] An iSocket (InSnergy Socket)[3] power meter was additionally used to validate power measurements, which has an accuracy of 1%. The measured base/idle power (i.e., static power) for our server is 249W. The monitoring system was additionally configured to remotely monitor the server's temperatures using IPMI (Intelligent Platform Management Interface), while recording the power consumption.

TABLE III. NAS Parallel Benchmarks[4] used to evaluate the power model

NPB	Class	Description
DC	B	Data Cube
EP	D	Embarrassingly Parallel
FT	D	discrete 3D fast Fourier Transform, all-to-all communication
LU	C	Lower-Upper Gauss-Seidel solver
MG	D	Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
SP	C	Scalar Penta-diagonal solver
UA	C	Unstructured Adaptive mesh, dynamic and irregular memory access

IV. Evaluation and Analysis

We evaluate our power estimation model, W-Classifier, by running each of the OpenMP multi-threaded benchmarks from the NAS Parallel Benchmark (NPB) suite on all 16 cores. When each benchmark is running, W-Classifier is used to estimate power use for the entire system while the power meter is measuring the real power use for comparison. We further compare W-Classifier with a multi-variable regression model, showing the advantages of workload classification. Finally, we discuss some interesting factors that impact power estimation.

For comparison, a multi-variable linear regression model is implemented based on the same sample data collected from the micro-benchmark. The same four PMCs in Section II-D are selected in the multi-variable linear regression. To guarantee fair comparison, the power modeling process is the same as W-Classifier with the same sample data, except that workload classification is not used. Therefore, we can take the multi-variable power model as a special case of W-Classifier where

[2] https://www.wattsupmeters.com/secure/products.php?pn=0

[3] Institute for Information Industry, http://web.iii.org.tw/

[4] Descriptions are taken from http://www.nas.nasa.gov/publications/npb.html

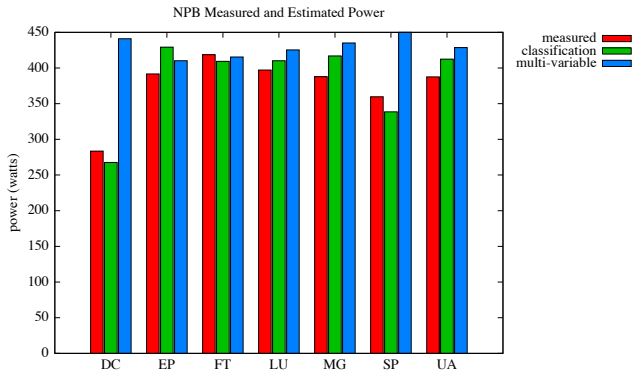

Fig. 3. Power estimation for NPB benchmarks

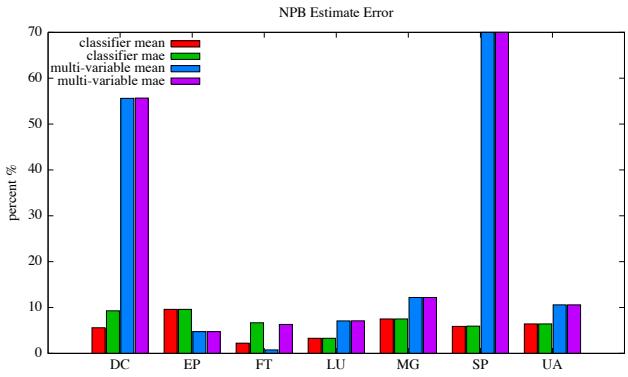

Fig. 4. Mean error and MAE of power estimation in percentage for NPB benchmarks

there is only one workload type. The model is derived by using a multi-variable, linear least squares regression function on the PMC data for the FPU, INT and cache micro-benchmark workloads. This subset of workloads was chosen as it provides a similar workloads to some previous research [12] and gave better regression results than using all workloads. The CPU-clocks-not-halted PMC was removed from the model as it resulted in a coefficient of zero when included. In the derived model, retired-uops is the strongest explanatory PMC, while the FPU and dispatch-stalls PMCs accounted for less of the power variation in the sample data sets for linear regression.

A. *Evaluation of power estimation*

First, in Figure 3, we show the mean power estimations for each benchmark by W-Classifier and the multi-variable model, together with the mean power measured by the power meter. The mean power estimation of each benchmark, taken during the whole execution period, provides a coarse-grained metric of power estimation accuracy. We can see in the figure that W-Classifier is able to reasonably track the real power use. Compared with the multi-variable model, W-Classifier is more accurate in terms of mean power estimation for most benchmarks. Note that the mean power estimation of the multi-variable model for SP is out of the scale of the figure.

Figure 4 shows the mean error and Mean Absolute Error (MAE) of power estimation for both W-Classifier and the multi-variable model. MAE measures the absolute difference in the estimated and measured power for every data point, e.g., every second. It indicates how well the trend of the estimates follows that of the real power use measured by the power meter. An important objective of power estimation is to be able to accurately estimate run-time power, which requires the trend of estimated power to closely match the measured real power use. Therefore, we use MAE for our evaluation as it provides an indication of trend. In contrast, the mean error simply measures the difference in mean power for the entire execution, providing no indication how closely the estimates follow the real power use. It is conceivable that a low mean error can be achieved by a model that wildly over-estimates and under-estimates power use, but the large positive and negative errors cancel out each other in the mean error. In these situations, however, MAE would return a more meaningful result with a large error.

According to Figure 4, W-Classifier has an average mean error of 5.78% for all benchmarks while the multi-variable model has an average mean error of 39.94%. W-Classifier has a much smaller mean error than the multi-variable model for most benchmarks. The only exceptions are EP and FT where the multi-variable model is slightly better in terms of mean error and mean power estimation, though the absolute differences between them are very small in terms of Watts. The slightly larger mean errors of W-Classifier for benchmarks EP and FT, compared with the multi-variable model, can mainly be attributed to the errors of base power estimation in W-Classifier, which affects the power estimation of W-Classifier for all benchmarks to a different degree. One of the possible factors contributing to this is the varied impact of temperature on the micro-benchmark and the NPB benchmarks, which will be discussed further in Section IV-C.

From Figure 4, we find W-Classifier has an average MAE of 6.95% for all benchmarks, while the multi-variable model has an average MAE of 40.74%. However, the average MAE for the multi-variable model is negatively impacted by the significant errors of 55.6% and 188.6% for the DC and SP benchmarks respectively. When these two benchmarks are excluded from the results, W-Classifier has an average MAE of 6.69%, while the multi-variable model has a significantly improved average MAE of 8.18%. W-Classifier has a smaller MAE than the multi-variable model for all benchmarks except EP. This demonstrates that W-Classifier has a better run-time tracking ability of power changes during the execution of each benchmark.

From the figure, it can also be seen that there is a large difference between mean error and MAE in W-Classifier for benchmarks like DC and FT. This difference is caused by large variations in real power use during the execution of DC and FT, with standard deviations of 41.1W and 32.2W respectively. All other benchmarks exhibit more constant power use during execution, with standard deviations ranging from 2.3-8.5W for real power use.

The overall trend between the estimated and measured power values can be more easily understood by visualizing the raw data points on a scatter plot. For example, Figure 5 plots the power values for the DC benchmark, which had a high standard deviation caused by large variations in power during execution. The x-axis is the execution time in seconds, while the y-axis is the corresponding power in Watts. Now it is possible to easily recognize that the trend of power estimates

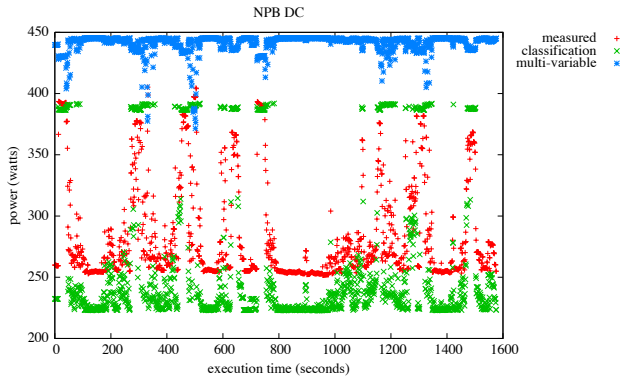

Fig. 5. Measured and estimated power for NPB DC

made by W-Classifier closely follows that of measured real power use. However, the estimates are consistently about 30W below measured values, indicating the model is currently not capturing the right value for the base power, as mentioned previously. The overall strength of the trend supports the current selection of PMCs for the execution workloads.

It can clearly be seen in Figure 5 that the multi-variable model significantly over-estimates power consumption. This is because the DC benchmark largely consists of a memory dominant workload, with significantly lower processor utilization levels than most of the other NPB benchmarks. For instance, the median retired-uops for FT is 22 times higher than the DC median value, indicating the scale of this difference. Since retired-uops is the dominant PMC used in the resulting multi-variable model, it contains a bias towards high utilization workloads and is therefore not well suited to such memory dominant workloads. In contrast, W-Classifier detects two separate workloads due to workload classification and uses a different linear function for each type of workload. For some periods of time when the utilization level is high (with power around 400W), a FPU/cache workload classification is used for power estimation. For other time periods when the utilization level is much lower, the workload is classified as being memory dominant, allowing a different power estimation function to be used.

In summary, W-Classifier can better track run-time power changes than the multi-variable model. This is reflected in the smaller average MAE across all benchmarks, which is 6.95%. While the multi-variable model is able to estimate power with an average MAE of only 8.18% for the more compute intensive benchmarks, the error significantly increases to 40.74% after the inclusion of DC and SP. This shows W-Classifier has the distinct advantage of being better able to estimate power for a broader range of application workloads. This will prove to be an important feature when estimating power for a wider range of more general applications, as they are more likely to have varying execution phases and workloads than the NPBs.

B. Importance of sampling rate

In this section we will discuss the impact of sampling rate on power estimation. The rate at which PMCs are sampled during real-time power estimation is not often considered when using a multi-variable model, as it will not have an impact on the mean power estimation. This is because the model uses the intensity values for a constant set of PMCs, where the sum of each PMC's intensity values will remain the same for a given time, regardless of the number of samples collected. However, this is not the case for W-Classifier due to the selective use of PMC-based linear functions for different workload types.

For instance, the DC benchmark shown in Figure 5 consists of two workload types, modeled by different PMCs and linear functions, as discussed in the previous section. It is the relatively high frequency, periodic sampling in power estimation that enables W-Classifier to detect all of these workload changes at run-time, resulting in the estimation trend matching the measured real power use. A decrease in the sampling rate can result in modest increases in the estimation error. The sampling rate does not have a very significant impact on the results for NPB benchmarks, because the inherent nature of these benchmarks exhibit few detected workload changes if any. However, we would anticipate that sampling rate would have a significant impact when operating over more general applications that exhibit a wider range of workload variations. In these cases, W-Classifier will likely work better for them than the multi-variable model, based on the results that we have presented, e.g., the DC and SP benchmarks.

C. Temperature effects

While it is well known that temperature has an effect on CPU power usage, it remains less well known what steps should be taken to mitigate some of the impact. Therefore, in this work we have adopted a range of techniques and good practices, some of which have been discussed in our previous work [17].

The most notable technique is the use of long execution times to ensure the processor reaches a stable operating temperature. For instance, the micro-benchmarks were configured to run for about 25 minutes. While this may seem excessive, we had previously found that the processor may take up to 6 minutes before reaching a stable temperature during the execution of our micro-benchmark. The longer the execution continues after reaching a stable temperature, the less weight the warm-up period will have on power modeling. This technique was additionally applied to the NPB benchmarks shown in Table III, where each benchmark was configured to run the smallest class size providing an execution time as close to 10 minutes as possible. While this was not achievable for all benchmarks due to system resource limitations, it was achieved on most.

An alternative to mitigating warm-up effects through increasing the execution time, is to merely trim all data points from the beginning of a data set that are affected by the warm-up period. A comparison of these two approaches is made in Figure 6, which plots the MAE for W-Classifier run on the two alternative configurations of NPBs. The results for long executions of the benchmarks are given by 'Including thermal effect', while the results for 'Excluding thermal effect' have had the first half of the respective sample data set removed. The most noticeable variation in MAE is for the DC benchmark, where the error for warm-up removal is much higher than for long execution times. However, this variation in estimation error can easily be attributed to the characteristics of the benchmark. Looking at the scatter plot for DC in Figure 5, it can be seen that towards the end of each execution, the power

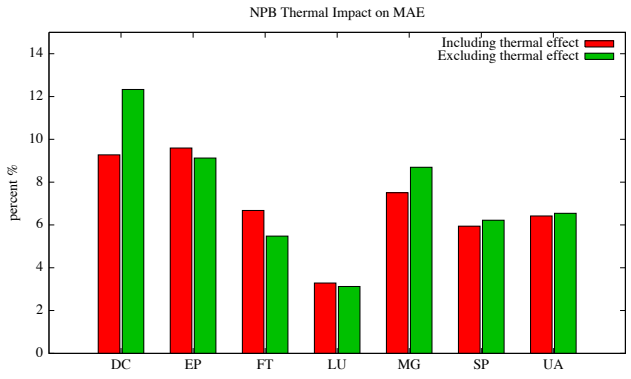

Fig. 6. Impact of excluding thermal effect from NPB

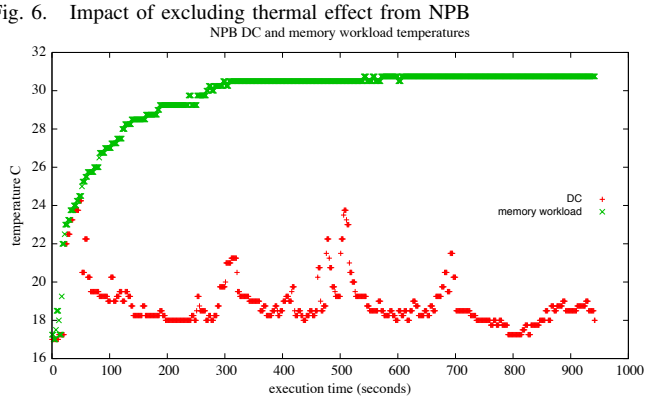

Fig. 7. DC temperature comparison with a memory workload

values begin to fluctuate frequently, resulting in a significantly noisier data set once the warm-up is removed. Therefore, neither technique provides a significant advantage over the other in helping to improve power estimation.

The techniques discussed so far are only capable of affecting the dynamic temperature changes from processor warmup. They do not address the potentially more significant impact of differences between steady-state operating temperatures. Such a difference can be seen in Figure 7, which plots the temperatures (in degrees Celsius) measured while executing the DC benchmark and our micro-benchmark with a memory dominant workload. The observed difference in the measured temperatures (about 12 degrees) is quite significant, which is over a third of the measured maximum for our micro-benchmark.

Such a significant difference in temperature between two similar workloads in DC and our benchmarks has unquestionably impacted on the accuracy of our derived power model in W-Classifier. Because the sampled power values when the micro-benchmark is run is lower than the real power use of DC for a similar workload due to the temperature difference, the estimated power of DC by W-Classifier is no doubt lower than the measured real power, as shown in Figure 5. This highlights the necessity of incorporating temperature values into the power model, which can help correct the variation between the measured and estimated power by W-Classifier. Any difference in operating temperature between the training micro-benchmark and evaluated benchmarks will contribute some error to the power estimation. This temperature difference partly contributed to the errors of the base power estimation in W-Classifier, as mentioned previously.

V. Related Work

Much of the prior work on PMC-based power estimation has taken the approach of using PMCs to model the underlying architectural components. For this, a set of architecture specific micro-benchmarks try and affect the power use of targeted processor units. Singh et al. [11] proposed such a model, which used micro-architectural knowledge to decompose the processor into its four main functional units: Floating Point Units, Memory, Stalls, and Instructions Retired. PMC selection is made from initial data collected from the execution of the SPEC benchmark suite. A separate micro-benchmark is designed for each of the four PMCs most strongly correlated with power for each functional unit. The micro-benchmark data is used to form a piece-wise linear function.

Bertran et al. [13] take an even finer-grained approach by starting with a set of 97 micro-benchmarks designed to individually highlight all possible processor power components. This results in multiple linear equations with an input for each of the seven derived power components. During the runtime period, PMC multiplexing is required, as the hardware micro-architecture does not allow that many PMC values to be collected simultaneously.

Such models are derived from a very selective set of micro-benchmark workloads, designed with the sole intention of affecting the power use of target hardware components. This results in a model strongly correlated to a specific micro-architecture, which may not generalize well across a large variety of workloads. In our work we have taken the opposite approach, where we attempt to generalize the power model by starting with a set of varied workload types. During the execution of each workload, the PMC values are collected for a range of hardware components, allowing the most applicable components to be used to model each specific workload type.

Da Costa et al. [12] present a methodology intended to broaden the range of modeled workloads by supplementing PMC values with process and system level statistics. This means the resulting model, derived through multivariate regression, is not limited to estimating the power of CPU and memory workloads, also allowing accurate power estimation of the network and disk synthetic benchmarks.

A limitation for this existing work is the requirement for a number of benchmarks and micro-benchmarks to be used while deriving the model. The use of specific benchmark suites can restrict portability, while the micro-benchmarks can be time consuming, both for execution and their architecture-specific development. In contrast, our approach uses a single parameterized micro-benchmark, which is general enough to be cross-platform, with little requirement for further development time. A further limitation of the related research work is the development of single, global power estimation functions. While developing such functions is a desirable goal, it will inevitably lead to some form of over-generalization, where a given application is outside the range of the behavior that is able to be reliably estimated with the current set of PMCs. This limitation is the key principle behind our use of a workload classification system and the corresponding workload specific power models.

Workload classification is adopted in [20], where the PMC details and static power for each workload is stored in a

single lookup table. The power for a given workload is obtained by locating the corresponding table entry. However, this technique relies on the assumption that a corresponding workload has previously been stored, creating the requirement of an exhaustive training process to be performed. Dhiman et al. [21] propose an alternative classification method using a gaussian mixture model for power estimation in virtualized environments. The resulting model provided for more accurate power estimation than the linear and multi-variate regression models. In comparison, our proposed modeling methodology uses simple linear regression to model power, while overcoming many of the previous limitations. W-Classifier is able to achieve general workload classification across the entire NPB suite, without imposing any additional requirements on model training data collection.

VI. Conclusion and future work

In this paper we have presented W-Classifier: a PMC-based power model for estimating the power consumption caused by software tasks on modern multicore CPUs. We have evaluated the model on a variety of applications, such as the benchmarks within the NPB suite. For those benchmarks, we have shown that the MAE (Mean Absolute Error) is 6.95%. This is significantly lower than the 40.74% MAE incurred by a multi-variable estimation model that is run on the same set of benchmarks—such multi-variable estimation models are the dominant technique currently used in software power estimation systems. The error for W-Classifier is sufficiently low to allow fine-grained decisions to be made within power-aware task scheduling in an operating system. Our future work will include further efforts to determine the cause of the remaining error, and hopefully provide bounds on it.

A key concept that we have emphasized is the importance of including workload classification into W-Classifier. This enables accurate power estimates for a wide range of applications, through the use of workload specific power models. This has been shown to increase the precision of power estimation compared to typical multi-variable approaches that do not perform workload classification. We are able to derive W-Classifier from the training dataset gathered from a single parameterized micro-benchmark.

Our future work aims to include additional system metrics into our energy model, such as determining how temperature affects base power draw, and how the base power draw operates across multicore configurations that are operating with different per-core utilization levels. Finally, we would like to incorporate CPU temperature metrics themselves into the estimation model, given the significant effect to which temperature has been found to affect power consumption. These factors are not currently included within the existing software power consumption estimation models that we are aware of.

References

[1] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," in *Proceedings of the 2005 ACM/IEEE conference on Supercomputing*. IEEE Computer Society, 2005, p. 1.

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

[3] J. Park, D. Shin, N. Chang, and M. Pedram, "Accurate modeling and calculation of delay and energy overheads of dynamic voltage scaling in modern high-performance microprocessors," in *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, 2010, pp. 419–424.

[4] W. Bircher and L. John, "Complete system power estimation: A trickle-down approach based on performance events," in *Performance Analysis of Systems and Software, IEEE International Symmposium on*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2007, pp. 158–168.

[5] Y. Zhang, Y. Wang, and X. Wang, "Electricity bill capping for cloud-scale data centers that impact the power markets," in *Proc of International Conference on Parallel Processing (ICPP)*, 2012.

[6] Z. Zong, X. Qin, X. Ruan, K. Bellam, M. Nijim, and M. Alghamdi, "Energy-efficient scheduling for parallel applications running on heterogeneous clusters," in *Proceedings of the International Conference on Parallel Processing*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2007, p. 19.

[7] M. A. Suleman, M. K. Qureshi, and Y. N. Patt, "Feedback-driven threading: Power-efficient and high-performance execution of multi-threaded workloads on CMPs," in *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, 2008, pp. 277–286.

[8] X. Chen, C. Xu, R. Dick, and Z. Mao, "Performance and power modeling in a multi-programmed multi-core environment," in *Proceedings of the 47th Design Automation Conference*. ACM, 2010, pp. 813–818.

[9] A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, "Power management architecture of the 2nd generation Intel Core microarchitecture, formerly codenamed Sandy Bridge," in *Hot Chips: A Symposium on High Performance Chips*, 2011.

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

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

[12] G. Da Costa and H. Hlavacs, "Methodology of measurement for energy consumption of applications," in *2010 11th IEEE/ACM International Conference on Grid Computing (GRID)*, 2010, pp. 290–297.

[13] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in *ICS '10: Proceedings of the 24th ACM International Conference on Supercomputing*. New York, NY, USA: ACM, 2010, pp. 147–158.

[14] P. Alonso, R. M. Badia, J. Labarta, M. Barreda, M. F. Dolz, R. Mayo, E. S. Quintana-Ortí, and R. Reyes, "Tools for power and energy analysis of parallel scientific applications," in *Proc of International Conference on Parallel Processing (ICPP)*, 2012.

[15] S. Wang, H. Chen, and W. Shi, "SPAN: A software power analyzer for multicore computer systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 23–34, 2011.

[16] H. Jin, M. Frumkin, and J. Yan, "The OpenMP implementation of NAS parallel benchmarks and its performance," NASA, Tech. Rep. NAS-99-011, 1999.

[17] J. Mair, Z. Huang, D. Eyers, and H. Zhang, "Myths in PMC-based power estimation," in *Energy Efficiency in Large Scale Distributed Systems*, 2013, pp. 35–50.

[18] AMD, "BIOS and kernel developer's guid (BKDG) for AMD family 10h processors," http://support.amd.com/us/Processor_TechDocs/31116.pdf, 2013.

[19] O. A. R. Board, "OpenMP application program interface version 3.0," http://www.openmp.org/mp-documents/spec30.pdf, May 2008.

[20] G. L. T. Chetsa, L. Lefevre, J.-M. Pierson, P. Stolf, and G. Da Costa, "DNA-inspired scheme for building the energy profile of HPC systems," in *Energy Efficient Data Centers*, 2012, pp. 141–152.

[21] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using Gaussian mixture models," in *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, 2010, pp. 807–812.