A Statistical Approach to Power Estimation for x86 Processors

Mohak Chadha*, Thomas Ilsche†, Mario Bielert†, Wolfgang E. Nagel†
†Technische Universität Dresden, Germany
Center for Information Services and High Performance Computing (ZIH)
Email: {firstname.lastname}@tu-dresden.de
*BITS Pilani K.K. Birla Goa Campus
Department of Computer Science and Information Systems
Email: mohak.chadha08@gmail.com

***Abstract*—With the growing significance of green computing and difficulty in obtaining accurate real time power measurements, there is an increasing need for accurate and reliable power estimation techniques for energy-aware performance optimization. In this paper we present a statistical approach for building accurate power models using Performance Monitoring Counters (PMC) as effective proxies for x86 systems. The selection of PMC events is based on statistical methods described in literature for ARM systems. The models are trained and validated through a synthetic workload generator as well as standardized benchmarks, using k-fold cross validation technique. We demonstrate the accuracy of the resultant models across different voltage and frequency states using sophisticated reference measurements. Furthermore, we analyze the significance of the chosen PMC events, which form the framework for the regression based power models.**

I. Introduction

Modern HPC systems, ranging from the current petascale to future exascale supercomputers, are constrained by power and energy consumption. As such, to balance performance and power consumption, there is a growing need for accurate real-time power information for efficient power management. Power usage information can be obtained either by using power meters or through model-based power estimation techniques implemented in hardware or software. While physical power measurements are becoming more widely available, they always present a trade-off between resolution, accuracy, scalability and cost [1]. In particular, measurement sensors need to be placed somewhere in the power distribution chain, effectively limiting the possibility to observe components with a common voltage source (e.g. multiple cores). Power estimation models can complement measurements in terms of general availability, component resolution and temporal granularity.

The most common approach to estimating the power of processors is using Performance Monitoring Counters (PMC) and statistical regression methods. Walker et al. [2] have presented an approach for statistical power modeling that particularly focuses on the stability of the generated model. While Walker et al. focus on embedded CPUs with the ARM architecture, the key contribution of this paper is to adapt the proposed modeling approach to high performance x86 CPUs. In doing so, we highlight the practical differences to power modeling arising from the different processor architectures. To build power models using PMCs, we implemented a workflow following the proposed steps data acquisition, post-processing, PMC selection, and model formulation. We practically demonstrate the approach and analyze the selected performance counters, which form the framework for our power models to quantify their significance. Furthermore, we validate our power models for both, standardized benchmarks and synthetic workloads, and emphasize additional limitations.

The rest of the paper is organized as follows. Section II describes the existing techniques and methodologies for power estimation. In Section III the implemented workflow and the proposed statistical approach are outlined. The implementation and experimental verification of our approach is described in Section IV. In Section V we discuss the significance of the selected performance counters. Section VI concludes the paper and presents an outlook.

II. Related Work

Power models for processors can be broadly classified into two categories, namely bottom-up and top-down. While bottom-up models use theoretical knowledge of each component, top-down models characterize specific devices experimentally. Top-down models commonly utilize performance counters as a basis for statistical regression modeling. This approach does not require any knowledge regarding the power consumption of individual components. However, the estimation accuracy and stability of the resultant regression model highly depends upon the chosen hardware event counters and the selected benchmarks used for the formulation of the model.

These techniques propose ways to formulate power models to estimate power at various denominations ranging from the entire CPUs [3], [4], [5] to individual micro architectural components [6], [7]. Some of the initial contributions in top-down CPU power estimation utilizing performance counters is from Bellosa [8] and Bircher et. John [3]. Bellosa [8] has proposed a Joule Watcher Energy Accounting framework which utilizes hardware counters to register events which indicate the consumption of a certain amount of energy. On the other hand, Bircher et. John [3] employ a trickle down approach for modeling complete system power as it allows the creation of accurate, performance counter based models using events native to a processor.

An important aspect of regression based modeling using performance counters is the separation of dynamic and static power components and the incorporation of Dynamic Voltage and Frequency Scaling (DVFS) into the power model. According to Mair et al. [9], dynamic and static elements of a processor-power model should be separated due to the fundamental differences in parameters and their behavior across different DVFS states. While voltage and frequency influence the dynamic power, static power is influenced by voltage and leakage current. Towards this, Su et al. [10] distinguish static and dynamic power and propose PPEP:Online Performance, Power, and Energy Prediction Framework which can predict performance, power, and energy across different DVFS states. The PPEP framework comprises of a per core power model and a cycles-per-instruction model based on hardware events. Goel & Mckee [11] build on this and propose a functional bottom-up power model for multicore processors. The power model is divided into four components: uncore dynamic power, core dynamic power, uncore static power and core static power respectively. The static power models are developed using both voltage and temperature as parameters.

Some interesting research has also been done on the selection of performance counters, which form the framework for accurate regression based power models. As such, Rodrigues et al. [12] propose a universal subset of performance counters which can be used for formulating power models for different underlying architectures within an average error of 5%. The proposed subset of performance counters namely fetched instructions, L1 hit and dispatch stalls were obtained by analyzing the different regression models developed by using different combinations of performance counters from a larger set. However, the proposed approach does not account for multicollinearity between the selected performance counters.

A more recent research on power modeling using performance counters is done by Walker et al. [2]. They propose a modeling methodology for developing run-time power models using performance counters and demonstrate it for ARM processors. Their approach involves statistical analysis for selection of performance counters and accounts for multicollinearity between them, addresses heteroscedasticity in power modeling in order to ensure the stability of formulated models. In this paper, we adapt this approach for the Intel x86 architecture.

III. Modeling Workflow

The methodology described in [2] suggests a four step workflow comprising data acquisition, PMC event selection, model formulation and validation, and CPU voltage model. We use a slightly modified workflow with an additional post-processing step after the actual data acquisition. Further there is no need for a CPU voltage model, given that it is possible to read actual core voltages during runtime on contemporary Intel processors. The following section describes the adapted workflow in detail, an overview is shown in Figure 1.

A. Data Acquistion & Post-Processing

The first step is concerned with the collection of data required to build and validate the models. On x86 systems, performance counter values can be obtained using several tools, one of the most prominent being Performance Application Programming Interface (PAPI) [13]. We use the available PAPI counters on the experimental platform as PMCs for developing the regression power models.

For collecting PAPI counter data along with CPU power and voltage information for different workloads, we generate the application trace by utilizing the tracing infrastructure provided by Score-P [14]. Score-P is a tool suite for tracing, profiling and online analysis of HPC applications. The relevant information such as power values, PAPI data and voltage values are added to the application trace by using metric plugins [15]. A metric plugin is an external dynamic linked library, which implements the Score-P metric plugin interface. The metric plugins used in this work include scorep_x86_adapt[1], scorep_plugin_apapi[2] and scorep_ni. The plugin scorep_x86_adapt supports per core metrics and is used for obtaining core voltage values. scorep_plugin_apapi is an asynchronous plugin used to sample and add PAPI performance counter data to the application trace. The plugin scorep_ni is used for obtaining power information on our measurement platform.

The format of the application trace generated by Score-P is Open Trace Format 2 (OTF2) [16]. It consists of a stream of events chronologically ordered by the time of their occurrence, and information about the state and configuration of the target system. The obtained application trace is used for generating the subsequent phase profile. The phase profiles for the traces from roco2 workload kernels (see Section IV) are generated using a HAEC-SIM module introduced in [17]. On the other hand, phase profiles for any standardized benchmarks are generated by using a custom python OTF2 post-processing tool. The resulting phase profile contains the start and end time, the average over time for each async metric, the average value of the recorded PMC values, the number of active threads, and the identification of the application. Overall the data acquisition with intermediate trace files is complex, but it allows to reuse existing tools instead of building prototypes only for the singular research effort. Further most of the used tools are publicly available.

Multiple runs of the same application are required due to the hardware limitation on simultaneous recording of multiple PAPI counters. The operating frequency f_{clk} is always fixed to one particular value during one particular execution of a workload. Following data acquisition, the data from multiple runs is processed to calculate average power and voltage across all runs. Furthermore, the phase profiles from multiple runs are combined together for selecting optimal performance counters.

B. Performance Counter Selection

The method to select the PMC events is shown in Algorithm 1. It is based on the proposed algorithm by Walker et

[1]https://github.com/score-p/scorep_plugin_x86_adapt
[2]https://github.com/score-p/scorep_plugin_apapi

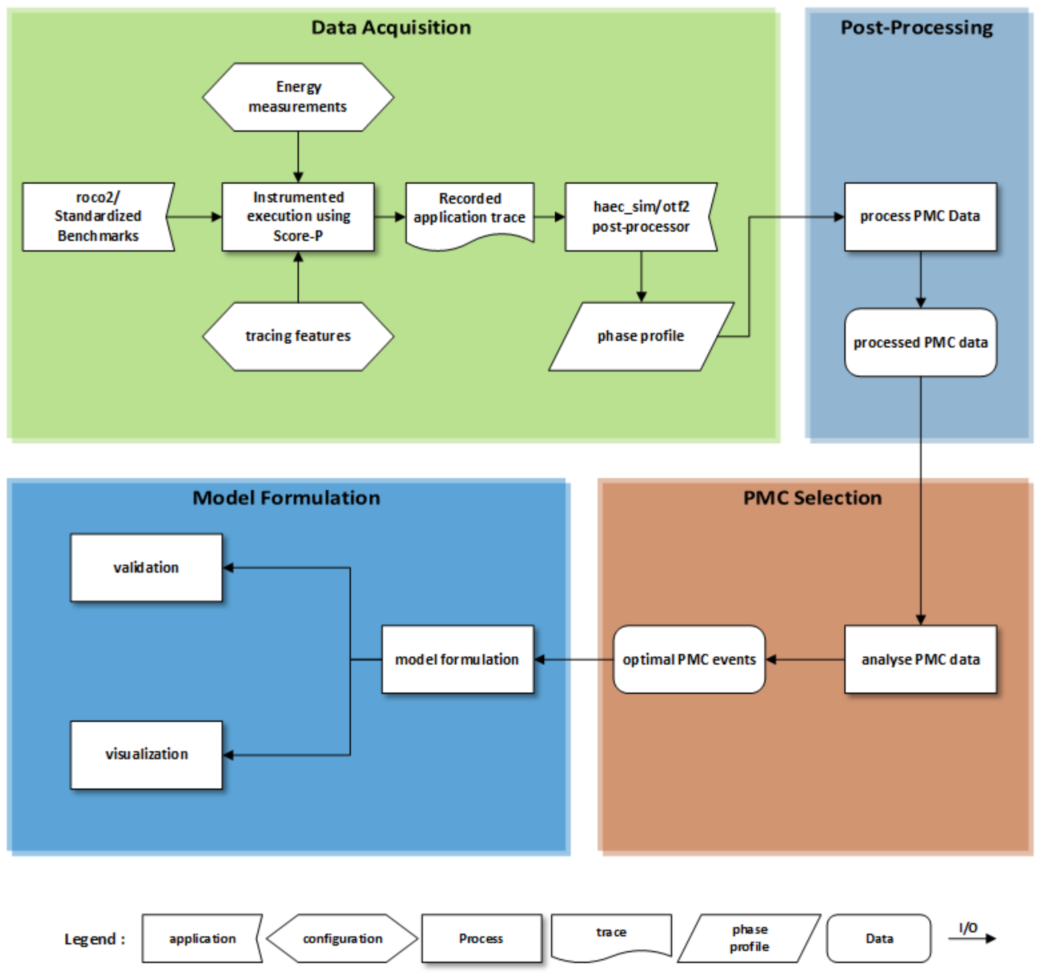


Fig. 1: Overview of the Workflow for Power Estimation

Algorithm 1 Algorithm for selecting PMC events adapted from [2]

```
1:  function SELECT_EVENTS($allEvents$,$\#Events$)
2:      $selectedEvents \leftarrow \emptyset$
3:      while $|selectedEvents| < \#Events$ do
4:          $\widetilde{R^2} \leftarrow 0$
5:          for $event \in allEvents$ do
6:              $R^2 \leftarrow \text{model}(selectedEvents \cup \{event\})$
7:              if $R^2 > \widetilde{R^2}$ then
8:                  $\widetilde{event} \leftarrow event$
9:                  $\widetilde{R^2} \leftarrow R^2$
10:             end if
11:         end for
12:         $selectedEvents \leftarrow selectedEvents \cup \{\widetilde{event}\}$
13:     end while
14:     return $selectedEvents$
15: end function
```

al. [2], with the notable difference that we do not initialize the $selectedEvents$ with a cycle counter. Preliminary tests have shown, that initializing the events with the processor cycle counter neither improves nor worsen the accuracy of the resulting model significantly [18]. The set $allEvents$ is initialized with the standardized PAPI counters available on the experimental platform as discussed in Section IV. The first stage iteratively selects events that give the most information regarding the power consumption. This is done by building a regression model with power as the dependent variable for each additional PMC event with the existing selected PMC events as additional independent variables. The PMC event that results in the highest R^2 value is then added to the $selectedEvents$.

To quantify multicollinearity between the chosen PMC events and to ensure stability of the model, the Variance Inflation Factor (VIF) is calculated. The VIF for a particular PMC event, is calculated using an ordinary least squares (OLS) based linear regression model, which predicts this variable using the other variables. A lower mean VIF for a chosen set of PMC events, ensures the stability of the coefficients of a regression based model, when different sets of workloads are considered. A VIF of 1 indicates no correlation between the independent and the dependent variables, while a VIF value greater than 10 generally indicates multicollinearity problems [19], [20].

The second stage of the PMC event selection proposed by Walker et al. involves identifying events which are correlated with each other, resulting in a high mean VIF. To accord for such cases, the authors suggest a mathematical transformation for the selected event with respect to the event with which it is highly correlated so as to reduce the mean VIF and improve the stability of the model. In our case, such a transformation was not practically applicable as discussed in Section IV.

C. Model Formulation

$$P_{Total} = \underbrace{(\sum_{n=0}^{N-1} \alpha_n E_n V_{DD}^2 f_{clk}) + \beta V_{DD}^2 f_{clk}}_{\text{dynamic power}} + \underbrace{\gamma V_{DD} + \delta Z}_{\text{static power}} \quad (1)$$

We adapt the power model proposed by Walker et al. [2] for x86 Processors. The proposed model (see Equation 1) includes static and dynamic power. Total power P_{total} is chosen as the dependent variable and is modeled with respect to rate of chosen PMC events (E_n), core operating voltage (V_{DD}) and operating frequency (f_{clk}). Whereas Walker et al. suggest a general function $f(V_{DD}, f_{clk})$ for static and background dynamic power, we add three specific terms: Dynamic power that is not represented by captured events ($\beta V_{DD}^2 f_{clk}$), static processor power (γV_{DD}), and system power that is independent of the processor core voltage (δZ). All of those are assumed to be constant for a given voltage and frequency. The coefficients of the model are calculated using Ordinary Least Squares (OLS) regression using python3 *statsmodel* [21].

Walker et al. also suggest the use of heteroscedasticity consistent standard error (HCSE) estimator as a parameter for regression using OLS to overcome the problem of heteroscedasticity, which is common in regression models. Heteroscedasticity refers to a situation in which size of the error term differs across values of independent variables. It leads to reduction in accuracy of the coefficients (α_n), the standard error of the coefficients and confidence intervals [22], [23]. We use a HC3 estimator provided by the python3 *statsmodel* as HCSE.

In order to reduce the multicollinearity of the model, which in turn reduces VIF, making the model more stable, we are not using the obtained PMC event value, i.e., the number of events per second. Instead, since the value of the PMC events are related to the operating frequency f_{clk}, the PMC event rate E_n, i.e., the number of events per cpu cycle, is used.

IV. Experimental Results

To validate the models experimentally, we use a set of small synthetic workload kernels (roco2 [17]) as well as the SPEC OMP2012 benchmark suite [24]. We excluded the benchmarks `kdtree`, `imagick`, `smithwa`, and `botsspar` from SPEC OMP2012 because they failed to build or crashed on our test system.

We run the test on a dual socket Intel Xeon E5-2690v3 (Haswell-EP) system with a total of 24 cores. As possible input to the power model, we use 54 PAPI counters that are available on the system. The selection of the used counters is done by the proposed algorithm (Algorithm 1). Note that there are even more native counters (162), each of with many possible configuration. We focus on the standardized PAPI conters to keep the amount of measurements needed feasible. Also the standardized PAPI counters represent a more generic view of the processor architecture. Hyper-Threading and Turbo Boost is disabled on the system.

TABLE I: Selected performance counters based on all workloads.

Counter	R^2	**Adj.**R^2	**VIF**
PRF_DM	0.735	0.730	n/a
TOT_CYC	0.897	0.893	1.062
TLB_IM	0.933	0.930	1.405
FUL_CCY	0.962	0.959	1.472
STL_ICY	0.979	0.976	1.573
BR_MSP	0.984	0.982	1.787

A custom-built energy measurement instrumentation is used for selection of PMC events, model parameter training, and verification. The system under test is instrumented with calibrated high resolution power sensors at the 12 V inputs to each socket. During the experimentation, the power measurements are collected on a separate system, avoiding perturbation on the measurement itself. The power measurement system is described in detail in [1].

The SPEC OMP2012 benchmarks are built using a custom configuration file with Score-P compiler instrumentation and use the Intel compiler suite[3]. Power, voltage, and performance trace are added to the benchmark trace by using the appropriate Score-P metric plugins.

A. PMC Event Selection

For selecting the PMC events, we run all roco2 and SPEC benchmarks at a fixed operating frequency f_{clk} of 2400 MHz with all available counters. Table I show the performance counter events in the order they were selected by Algorithm 1, along with the corresponding mean VIF values. It is notable that the VIF for six counters is consistently low, and no additional transformation is required. However, if we let the algorithm select one more performance counter, it chooses the event `CA_SNP` which denotes the number of snoop requests. Although, the R^2 value rises to a value of 0.989 the mean VIF increases to 26.42, signifying high collinearity between the selected events. Since `CA_SNP` is a non-derived event and there is no direct mathematical relationship between `CA_SNP` and previously selected events in Table I, such a transformation to reduce the VIF is not applicable. This scenario acts as a limitation to the modeling approach, since selecting the event `CA_SNP` will make the model less stable and not selecting the event will prevent the model from utilizing all the available information for estimating power. For the following, we utilize the 6 events in Table I.

Figure 2 portrays the changing R^2 and Adj.R^2 when the proposed algorithm is used. Adj.R^2 is a modified version of R^2 that has been adjusted for the number of predictors in the model. The Adj.R^2 value increases only if the new predictor variable improves the model more than that would be expected by chance.

[3]Compiler flags: `-openmp -O3 -ipo1 -xAVX`

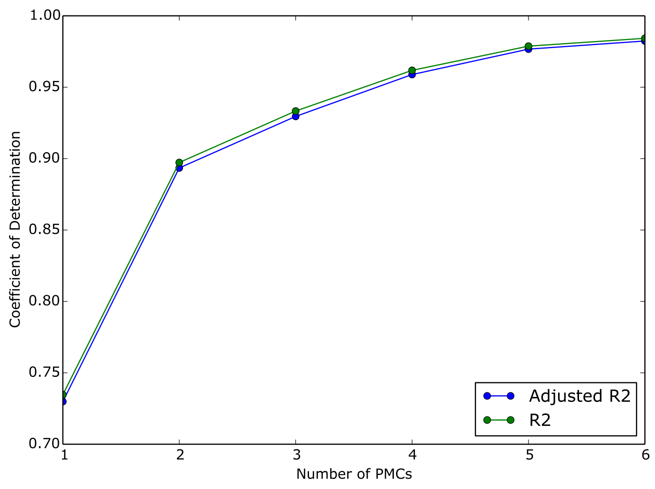


Fig. 2: Changes in R^2 and Adj.R^2 values with selection of performance counters.

TABLE II: Summary of results for 10-fold cross validation

Metric	Min	Max	Mean
R^2	0.9904	0.9913	0.9910
Adj.R^2	0.9900	0.9910	0.9906
MAPE	6.6114	8.3198	7.5452

In Table I `PRF_DM` describes the number of data prefetch misses, `TOT_CYC` describes the total number of cycles, `TLB_IM` describes the instruction translation look aside buffer misses, `FUL_CCY` describes the cycles with maximum instructions completed, `STL_ICY` describes the cycle with no instruction completed and `BR_MSP` describes the total number of mispredicted branch instructions. The selected performance counters are analyzed in detail in Section V.

B. Model Parameter Training

To verify that our model estimates power accurately across different voltage-frequency steps and, for the short-running roco2 kernels, different thread counts. The values of the selected performance counters along with power and voltage data are obtained at 5 distinct operating frequencies between 1200 and 2600 MHz. Following this, the model (Equation 1) is trained and validated using 10-fold cross validation with random indexing. Figure 3 shows the Mean Absolute Percentage Error (MAPE) of all workloads across the five DVFS states. The maximum error is observed for the SPEC benchmark `ilbdc`, while the minimum is observed for the roco2 square root workload `sqrt`. The formulated model achieves an R^2 value greater than 0.99 with a MAPE of 7.54 across all DVFS states for multi-threaded benchmarks (see Table II). The mean Adj.R^2 for the formulated model is only 0.0004 lower than the respective R^2 value, indicating that the present predictor variables (Equation 1) in the model add relevant information to it and don't inflate the value of R^2.

To analyze the effect of unseen workloads on the power model and assess it's stability we consider four scenarios in which we choose different training and validation sets for formulating our model. The scenarios considered are:

1) The model is trained using four random workloads from roco2 and SPEC OMP2012 and validated using the rest.
2) The model is trained using all workloads from roco2 and validated using all workloads from SPEC OMP2012.
3) The model is trained and validated using 10 fold cross validation for roco2 and SPEC OMP2012 workloads (Table II). Due to separate frequencies, the training set includes most workloads.
4) The model is trained and validated using 10 fold cross validation for roco2 workloads. Due to separate frequencies, the training set includes most workloads.

Note that the selected performance counters are fixed for those scenarios due to practical considerations on the total amount of measurements. We discuss the impact of selected training workloads on counter selection later in this section.

Figure 4 compares the mean absolute percentage error for the different scenarios. Although the MAPE obtained across all DVFS states in the considered scenarios remain relatively close to the value obtained from 10 fold cross validation (see Table II), a considerable increase is seen in scenario 2 (see Figure 4). The highest error of 15.10 % occurs in scenario 2 where only synthetic workloads are used for training the model parameters. The synthetic workloads are not diverse enough to create a stable model that can be applied to more realistic benchmarks. Using random workloads, both synthetic and closer to applications (scenario 1), results in a better MAPE. Scenario 4, where training and validation is done on only synthetic benchmarks, shows the best accuracy. However that is the least realistic use case. This shows the need to use diverse and large number of workloads with varying characteristics such as remote NUMA memory accesses, different memory accesses patterns, but also instruction cache usage for training in power modeling. Compared to the implementation on ARM, which has a MAPE of 2.8 % and 3.8 % our results on Intel with the comparable scenario 3 turn out to be less accurate (7.54 %).

The mean absolute percentage error is commonly used in literature to describe the accuracy of power and energy models. We also use it for a single metric comparison of the different scenarios. However, the MAPE can hide information about the particular accuracy of a model across a variety workloads. To reveal more detail about the accuracy of the model prediction across different workloads and value ranges, Figure 5a and 5b compare the actual and estimated average power for the scenarios 2 and 3 respectively. Each data point in the charts show the average power for one specific experiment as a combination of workload, core frequency, and for the synthetic workload kernels, thread count. Figure 5a reveals that the estimated power has a systematic bias for certain workloads, that is often independent from frequency or used thread count. For example, the average power for benchmarks `md` and `nab` are consistently overestimated, when trained only with synthetic workloads. Figure 5b shows similar errors for both the synthetic kernels that the model was trained with, and the more realistic SPEC OMP2012 workloads. On the

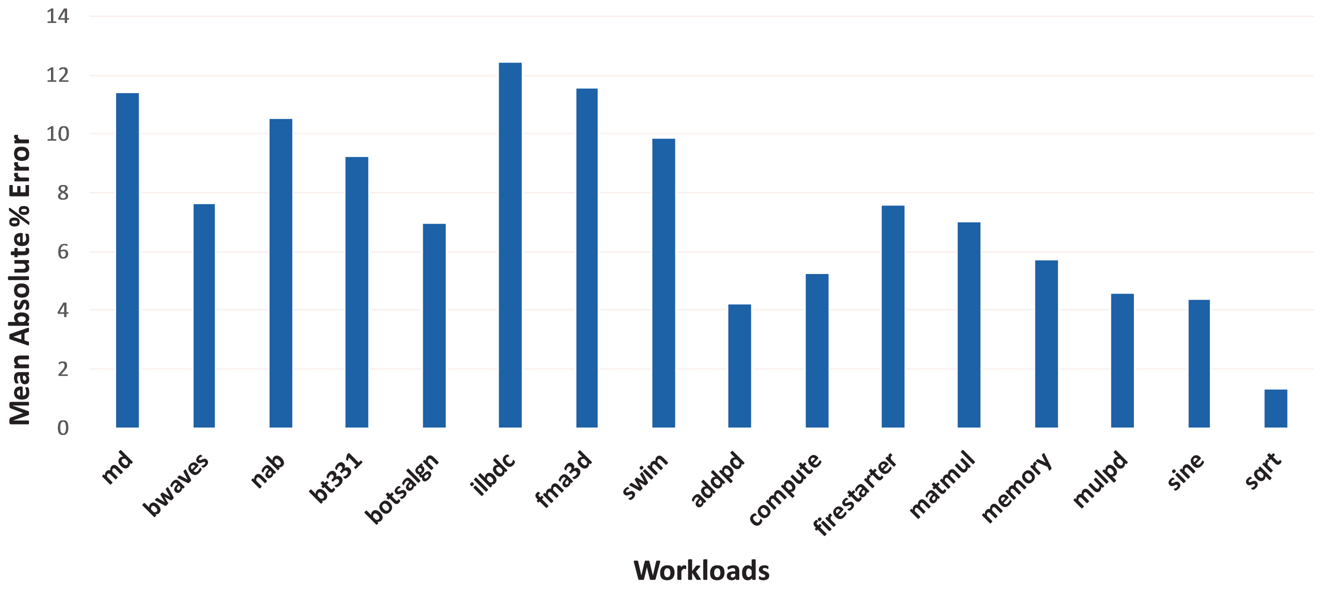


Fig. 3: Mean absolute percentage error for 16 workloads across all DVFS states.

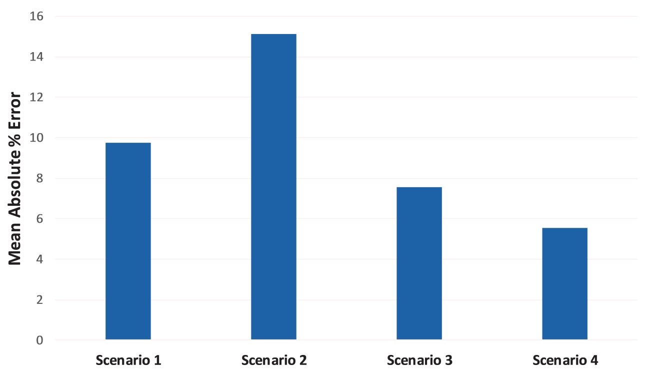


Fig. 4: Mean absolute percentage error for scenarios: 1) training on four random workloads, validation on rest, 2) training on synthetic workloads, validation on SPEC OMP2012, 3) 10-fold CV on all experiments, 4) 10-fold CV on all synthetic workload experiments.

one hand, the synthetic kernels can be estimated more easily considering that they were part of the training set. On the other hand, the SPEC workloads have more internal variability that can even out the error on overall average power estimation. In general the model exhibits no strong tendency towards over- or underestimation and it's residuals show heteroscedasticity, i.e. the absolute error grows with increasing power values.

V. Analysis of the Selected Performance Counters

To quantify the significance of the selected performance counters shown in Table I, we calculate the Pearson Correlation Coefficient (PCC) between the supported PAPI counters and power. The PCC represents the degree of linear correlation between two variables and can have values in the range between $+1$ and -1. $+1$ denotes total positive linear correlation and -1 signifies total negative linear correlation. A value of 0 indicates no correlation. To calculate the PCC, we use the python3 module scipy [25].

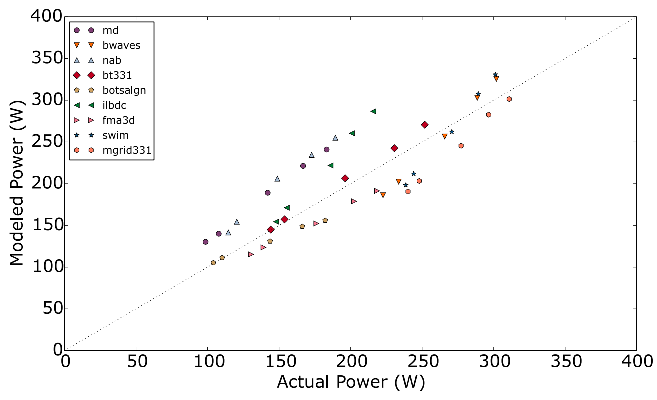
TABLE III: Pearson correlation coefficient of selected performance counters with power

Counter	PCC
PRF_DM	0.85
TOT_CYC	0.59
TLB_IM	0.33
FUL_CCY	0.57
STL_ICY	0.38
BR_MSP	-0.01

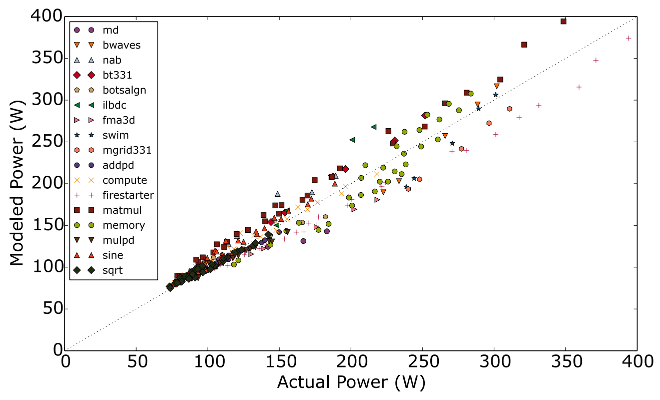
$$P(x,y) = \frac{\sum_i (x_i - \bar{x}) * (y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{2}$$

Figure 6 displays the PCC of the supported PAPI counters for our experimental platform and Table III lists the ones selected by Algorithm 1. Interestingly, statistically chosen counters do not show a particularly strong correlation with power, except for the very first one. This signifies that the model will be provided with unique information from the selected performance counters, since those with similar correlation with power will tend to be correlated with each other, which would lead to increased multicollinearity and a less stable model.

Even though `BR_MSP` has negligible negative correlation with power, it is selected as performance counter. It appears to offer relevant information for the workload kernels `compute` and `md`, since it has relatively high values. Therefore, this performance counter does improve the regression model the most, given the already selected PMC events.



(a) Scenario 2, training with synthetic workloads, verification with SPEC.



(b) Scenario 3, training on random samples of all workload/frequency combination.

Fig. 5: Comparison of actual and modeled power values for different workloads and frequencies. The diagonal dotted line represents a perfect agreement of model and measurement.

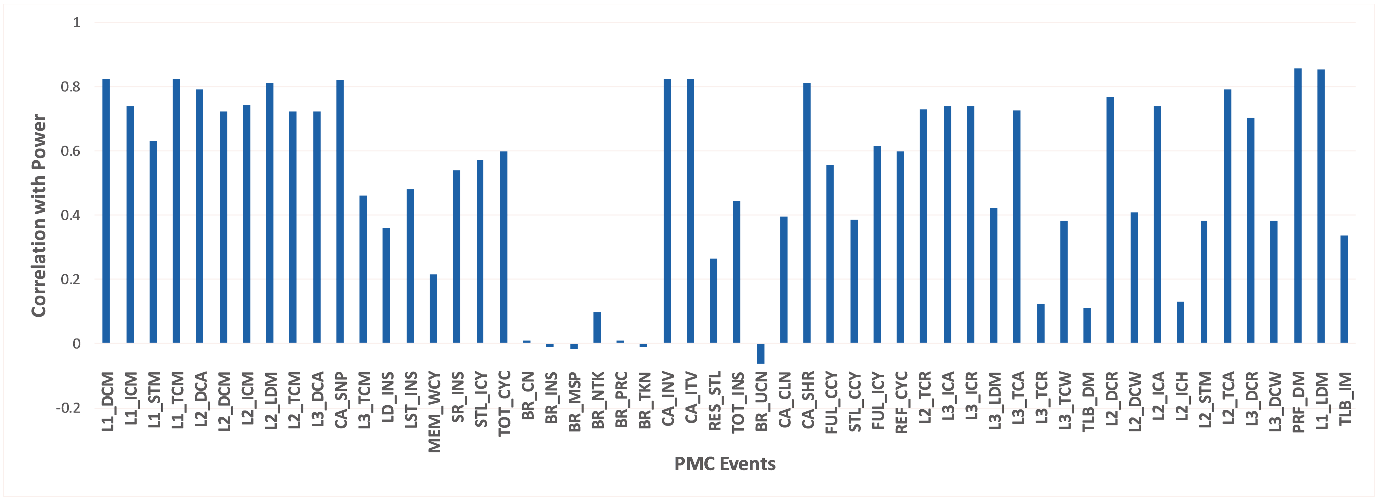


Fig. 6: PCC Values of PAPI Counters for our experimental platform

TABLE IV: Selected performance counters based on small synthetic workloads

Counter	R^2	Adj.R^2	mean VIF
L1_LDM	0.839	0.836	n/a
REF_CYC	0.941	0.938	1.084
BR_PRC	0.973	0.971	1.340
L3_LDM	0.990	0.989	1.341
FUL_CCY	0.993	0.993	8.982
STL_ICY	0.995	0.994	13.617

If the counter selection of Algorithm 1 is performed on a subset of the training data that includes only the synthetic workloads, the resulting counters are different as shown in Table IV. In our case, it also leads to a significantly higher VIF from the fifth added counter. Again while there is a strong correlation, there is no clear transformation for the added counters `FUL_CCY` (cycles with maximum instructions completed) and `STL_ICY` (cycles with no instruction issue).

In [18], we have shown when counter selection and training is done on a limited set of benchmark kernels, the stability (as in accuracy on unseen workloads) can suffer. A low VIF does not prevent overfitting in cases where the counter values during the training are very low compared to counter values of unseen workloads. Once the model gets input of counter values that are order of magnitude higher than the ones during training, the result becomes unstable and inaccurate.

VI. Summary

In this paper, we adapted a statistically rigorous CPU power modeling approach by Walker et al. [2]. While originally described on an embedded ARM system, we demonstrate how it can be used on a high performance Intel systems. There are several practical differences in power modeling in our implementation. First, extracting performance counter data and real-time voltage measurements for x86 CPUs is well-supported by existing software libraries and tools such as PAPI [13] and Score-P [26]. Hence it was not necessary to develop a separate voltage model. In general it was possible to apply the suggested counter selection algorithm, However, it was not possible to transform the selected performance counters in order to reduce the VIF, because there is no clear relationship between the correlating selected counters. Together with the high intricacy of the x86 CISC architecture and PMCs, this likely contributes to a reduced accuracy on our x86 system compared with the original implementation on ARM. The formulated models are able to estimate power with a mean absolute percentage error of 7.54 across all DVFS states for multi-threaded synthetic kernels and standardized benchmarks from SPEC OMP2012.

Vastly different accuracies for different training scenarios show, that the selection of model training workloads has considerable impact on the accuracy and stability of the model. In particular, only using a limited set of micro workloads is not sufficient for either selecting good performance counters or calibrating the model parameters. Such limited workloads do not cover the vast range of states a complex modern architecture comprises. In our experiments a low VIF was no guarantee for a stable model.

Building reliable power models for complex architectures remains an ongoing research challenge. Statistically sound modeling approaches are a key contribution towards this goal. Nevertheless our work shows that even statistical black box techniques, which use little architecture specific knowledge, require an extensive evaluation when applied to different architectures or workloads.

The future work of this project will focus on analyzing different statistical algorithms and heuristic criterion's for selecting PMC events as variables for the regression based power models. To strengthen the general validity of the approach, more experiments should be performed on different generations of x86 processors. Further investigation also includes the adaptation of the model to a larger scale such that it can be applied to peta- or exa-scale systems instead of individual nodes.

Acknowledgment

This work has been funded in part by the German Research Foundation (DFG) in the Collaborative Research Center "Highly Adaptive Energy-Efficient Computing" (HAEC, SFB 912). We thank the reviewers for their exceptionally constructive reviews to improve this work and inspire future work.

References

[1] T. Ilsche, D. Hackenberg, S. Graul, J. Schuchart, and R. Schöne, "Power measurements for compute nodes: Improving sampling rates, granularity and accuracy," ser. THE Sixth INTERNATIONAL GREEN and SUSTAINABLE COMPUTING CONFERENCE, Dec. 2015.

[2] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modeling for mobile and embedded cpus," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. PP, no. 99, pp. 1–1, 2016.

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

[4] G. Contreras and M. Martonosi, "Power prediction for intel xscale®processors using performance monitoring unit events," in *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, ser. ISLPED '05. New York, NY, USA: ACM, 2005, pp. 221–226.

[5] D. Economou, S. Rivoire, and C. Kozyrakis, "Full-system power analysis and modeling for server environments," in *In Workshop on Modeling Benchmarking and Simulation (MOBS*, 2006.

[6] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *Proceedings of the 2001 International Symposium on Low Power Electronics and Design*, ser. ISLPED '01. New York, NY, USA: ACM, 2001, pp. 135–140.

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

[8] F. Bellosa, "The benefits of event-driven energy accounting in power-sensitive systems," in *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System*, ser. EW 9. New York, NY, USA: ACM, 2000, pp. 37–42.

[9] J. Mair, Z. Huang, D. Eyers, and H. Zhang, "Myths in pmc-based power estimation," in *Revised Selected Papers of the COST IC0804 European Conference on Energy Efficiency in Large Scale Distributed Systems - Volume 8046*, ser. EE-LSDS 2013. New York, NY, USA: Springer-Verlag New York, Inc., 2013, pp. 35–50.

[10] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "Ppep: Online performance, power, and energy prediction framework and dvfs space exploration," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-47. Washington, DC, USA: IEEE Computer Society, 2014, pp. 445–457.

[11] B. Goel and S. A. McKee, "A methodology for modeling dynamic and static power consumption for multicore processors," in *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, May 2016, pp. 273–282.

[12] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, "A study on the use of performance counters to estimate power in microprocessors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 882–886, Dec 2013.

[13] P. J. Mucci, S. Browne, C. Deane, and G. Ho, "Papi: A portable interface to hardware performance counters," in *In Proceedings of the Department of Defense HPCMP Users Group Conference*, 1999, pp. 7–10.

[14] A. Knüpfer, C. Rössel, D. an Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony *et al.*, "Score-p: A joint performance measurement run-time infrastructure for periscope, scalasca, tau, and vampir," in *Tools for High Performance Computing 2011*. Springer, 2012, pp. 79–91.

[15] R. Schöne, R. Tschüter, T. Ilsche, J. Schuchart, D. Hackenberg, and W. E. Nagel, "Extending the functionality of score-p through plugins: Interfaces and use cases," in *Tools for High Performance Computing 2016 (accepted for publication)*. Springer, 2017.

[16] M. Wagner, A. Knüpfer, and W. E. Nagel, "Enhanced encoding techniques for the open trace format 2," *Procedia Computer Science*, vol. 9, no. Complete, pp. 1979–1987, 2012.

[17] M. Bielert, "Evaluating power estimation techniques: A methodological approach," Master's thesis, Technische Universitat Dresden, 2015.

[18] M. Chadha, "A statistical approach to power estimation for x86 processors," Bachelor's Thesis, Technische Universitat Dresden, 2016.

[19] M. Kutner, *Applied linear regression models*. Boston New York: McGraw-Hill/Irwin, 2004.

[20] J. Hair, *Multivariate data analysis*. Upper Saddle River, NJ: Prentice Hall, 2010.

[21] S. Seabold and J. Perktold, "Statsmodels: Econometric and statistical modeling with python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 57 – 61.

[22] J. S. Long and L. H. Ervin, "Using Heteroscedasticity Consistent Standard Errors in the Linear Regression Model," *The American Statistician*, vol. 54, no. 3, pp. 217–224, 2000.

[23] R. C. Dorf, *The Technology Management Handbook*. CRC Press, 1998.

[24] M. S. Müller, J. Baron, W. C. Brantley, H. Feng, D. Hackenberg, R. Henschel, G. Jost, D. Molka, C. Parrott, J. Robichaux, P. Shelepugin, M. van Waveren, B. Whitney, and K. Kumaran, "Spec omp2012 – an application benchmark suite for parallel systems using openmp," in *Proceedings of the 8th International Conference on OpenMP in a Heterogeneous World*, ser. IWOMP'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 223–236.

[25] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–. [Online]. Available: http://www.scipy.org/

[26] A. Knüpfer, C. Rössel, D. a. Mey, S. Biersdorff, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W. E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, and F. Wolf, *Score-P: A Joint Performance Measurement Run-Time Infrastructure for Periscope,Scalasca, TAU, and Vampir*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 79–91.