Power Modelling for Heterogeneous Cloud-Edge Data Centers

Published in:
 International Conference on Parallel Computing: Proceedings

Document Version:
Peer reviewed version

Queen's University Belfast - Research Portal:
[Link to publication record in Queen's University Belfast Research Portal](#)

Power Modelling for Heterogeneous Cloud-Edge Data Centers

Kai CHEN [1], Blesson VARGHESE, Peter KILPATRICK, and
Dimitrios S. NIKOLOPOULOS

School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast, UK

Abstract. Existing power modelling research focuses not on the method used for developing models but rather on the model itself. This paper aims to develop a method for deploying power models on emerging processors that will be used, for example, in cloud-edge data centers. Our research first develops a hardware counter selection method that appropriately selects counters most correlated to power on ARM and Intel processors. Then, we propose a two stage power model that works across multiple architectures. The key results are: (i) the automated hardware performance counter selection method achieves comparable selection to the manual selection methods reported in literature, and (ii) the two stage power model can predict dynamic power more accurately on both ARM and Intel processors when compared to classic power models.

Keywords. power modelling, cloud-edge computing, heterogeneous data centers

Introduction

Power monitoring has become a significant task for data-center management [1]. Direct power measurement obtained by physical meters or model-based interfaces has been widely supported in platforms [2]. However, fine-grained power measurement of individual hardware/software components, which plays a significant role in runtime energy/performance management/optimisation, is not easy [3]. For instance, both the model-based energy interface of the Intel Sandy Bridge server and the physical power meter of ARM Odroid-Xu3 board can measure the power of the entire processor rather than of individual computing cores in the processor or of the executing programs.

Developing accurate power models of computing cores or more fine-grained execution units, therefore, is an important avenue of research. A large proportion of existing models rely on multiple hardware activities of the processor represented by hardware performance counters, referred to as hardware counters, for estimating power [2].

However, the hardware counters necessary to build an accurate power model may substantially differ across processors due to the differences in the instruction set, pipeline, cache architecture and on-chip interconnect. The hardware counters are usually selected on the basis of experimental knowledge of the processor [3]. Such an approach

---

[1] Corresponding Author. E-mail: kai.chen@qub.ac.uk

used in traditional data centers cannot easily scale for heterogeneous processors (hosting multiple generations of server processors), or in emerging distributed computing environments like fog/edge computing and mobile cloud computing (in these environments, an application is distributed across data center processors, for example Intel Xeon processors [4], and low power processors, such as ARM [5] popularly employed in embedded systems for edge computing [6]). In this paper, we make novel contributions by developing a hardware counter selection method that is employed across multiple processors without compromising the accuracy of the power model.

With hardware counters selected by our proposed method we evaluate three classic power models based on Linear Regression (LR), Support Vector Machines (SVM) and Neural Networks (NN) on Intel and ARM processors. This evaluation motivates the design of a two stage power model that takes advantage of LR to estimate a basic power value and then uses SVM to optimise this value for improved accuracy. It is observed that our model predicts dynamic power more accurately on both ARM and Intel processors when compared to classic power models.

The research contributions of this paper are: (i) the design and implementation of an automated hardware counter selection method to simplify the hardware counter selection process without sacrificing the accuracy of the power model, and (ii) the proposal of a Two Stage Power Model which takes advantage of both LR and SVM algorithms.

The remainder of this paper is organised as follows. Section 1 presents the mathematical notation and the hardware platform employed. Section 2 proposes and evaluates a method for selecting hardware counters. Section 3 presents a two stage power model. Section 4 presents related work and concludes this paper.

1. Definitions

This section presents mathematical notation employed and the hardware platform used.

Notations: Classic power models that are used for estimating dynamic power of processors and the concept of vectors are defined as used in this paper.

Classic Power Models: Consider a processor power model in which the estimated power, P, is the sum of the idle power consumed by the processor (static), and the power required for various activities of the processor (dynamic). Thus, $P = P_{static} + P_{dynamic}$. This paper explores dynamic power which is a function of the volume of hardware activities on the processor, obtained from a set of hardware counters.

Consider n hardware counters whose values obtained from a processor during time interval t_i are denoted as $e_{i_1}, e_{i_2}, \cdots, e_{i_n}$ and let the measured dynamic power during interval t_i be denoted by $P_{i_{dynamic}}$. We consider the following three classic power models.

a. Linear Regression Power Model (LRPM): In this model, dynamic power is defined as $P_{i_{dynamic}} = \sum_{j=1}^{n} c_j e_{i_j}$, where c_j is the coefficient of the jth hardware counter.

b. Neural Network Power Model (NNPM): Compared to LRPM which captures linear relationships, NNPM can also model non-linear relationships. Hardware counter values provided as input pass through layers of NNPM where a linear (i.e. the weighted sum) and non-linear function (i.e. activation function) are applied to map the input to the output.

c. Support Vector Machine Power Model (SVMPM): This model captures both linear and non-linear relationships between dynamic power and the hardware counters. A set of hyperplanes are fitted using the training data to estimate the dynamic power.

The ideal configuration of input parameters for both NNPM and SVMPM was manually chosen by extensively exploring the space. In this paper parameters that provide the most accurate estimation of dynamic power are chosen.

Vectors: We define vector $V_i = \{P_{i_{dynamic}}, e_{i_1}, e_{i_2}, \cdots, e_{i_n}\}$ where $P_{i_{dynamic}}$ and the e_{i_j} are defined as above for the time interval t_i.

Each vector is normalised to bring values of all variables into the same range between 0 and 1. The normalised vector of V_i is represented as $\hat{V}_i = \{\hat{P}_{i_{dynamic}}, \hat{e}_{i_1}, \hat{e}_{i_2}, \cdots, \hat{e}_{i_n}\}$, where $\hat{P}_{i_{dynamic}} = P_{i_{dynamic}}$, $\hat{e}_{i_1} = \frac{e_{i_1} - min(e_1)}{max(e_1) - min(e_1)}$, $\cdots$, $\hat{e}_{i_n} = \frac{e_{i_n} - min(e_n)}{max(e_n) - min(e_n)}$.

Platform: Distributed computing environments such as those employed in fog/edge computing make use of both cloud data center and edge nodes. Typically, data center servers, for example Amazon cloud servers, make use of Intel Xeon processors[2], which are designed for high-performance computing. On the other hand edge nodes do not make use of large processors, instead employing low power processors, such as ARM[3]. Next generation power models will need to work for emerging distributed computing environments and therefore both an Intel Xeon processor representing servers used in data centers and an ARM processor representing edge nodes are used in our investigation.

The first processor is the Intel Xeon Sandy Bridge server comprising two Intel Xeon E5-2650 processors with 8 cores on each processor.The processor runs CentOS 6.5. We measure power consumption using the Running Average Power Limit (RAPL) interface [7]. The second processor is the ODROID-XU+E[4] board which has one ARM Big.LITTLE architecture Exynos 5 Octa processor. There are four Cortex-A15 cores and four Cortex-A7 cores and 2 GBytes of LPDDR3 DRAM. The system runs Ubuntu 14.04 LTS. We use the on-chip power meter to measure the power of the Cortex-A15 cores.

The hardware counters are obtained by real-time profiling using Performance API (PAPI) [8]. Power is obtained from the on-chip power sensor on ARM and from the RAPL interface on Intel. This research employs 16 scientific benchmarks using MPI and OpenMP (such as the Buffon Laplace, Monte-Carlo and molecular dynamics simulations, solvers for Poisson and wave equations, and fast fourier transform) which captures a wide range of workloads[5]. Vectors with hardware counters and measured power are continuously sampled during the execution of benchmarks approximately every 1 second. On ARM, we used the Cortex-A15 cores at their maximum frequency of 2.0GHz to execute the benchmarks. The Cortex-A7 cores at their maximum frequency of 1.4 GHz are used to obtain vectors. Similarly, on Intel, we used one processor at its maximum frequency to execute the benchmarks and the second processor at its maximum frequency to obtain vectors (both 2.0GHz).

2. Hardware Counter Selection (HCS) Method

Design of the HCS method: To design a hardware counter based power model, the *selection of hardware counters* must first be addressed. This requires addressing 'how many' and 'which' hardware counters should be selected.

---

[2] https://aws.amazon.com/ec2/instance-types/

[3] http://www.arm.com/products/iot-solutions/mbed-iot-device-platform

[4] http://www.hardkernel.com

[5] http://people.sc.fsu.edu/~jburkardt/c_src/c_src.html

If more hardware counters are employed, then a more accurate power model is built. Up to a maximum of 16 and 50 hardware counters are supported on ARM and Intel processors, respectively. However, using PAPI a maximum of only six hardware counters can be obtained simultaneously on both the ARM and Intel processors employed in this work. It should be noted that the number of hardware counters that are available and can be profiled simultaneously may vary between different processors even if they are produced by the same vendor. Multiplexing (profiling a different set of hardware counters sequentially to obtain a large number of hardware counters) techniques can surmount the limitation of the number of hardware counters that can be profiled, but introduces overheads that cannot be ignored. In this paper, we attempt to eliminate extra overheads by using only six hardware counters to build power models on both platforms, while retaining accuracy. Additional multiplexing techniques can easily be integrated into the modelling process proposed in this work.

In existing research, hardware counters that contribute to the power function are usually selected on the basis of experimental knowledge of the processor. Typically, all possible hardware counters that can be obtained are extensively explored using a cumbersome trial and error approach [5]. Then a combination of counters is chosen to develop a power model. However, this approach will not be practical for data centers in distributed cloud environments, such as in fog computing. It would be impossible to manually determine the suitable hardware counters of each processor for developing a power model. This motivates the need for the automated hardware selection method we propose.

We develop a generic Hardware Counter Selection (HCS) method that can be employed on multiple processors. The method selects a set of six hardware counters from all the available hardware counters that best correlates to power for a given processor. The method is based on a Random Forest (RF) algorithm that maps the hardware counters to power. We choose RF due to its accuracy in regression [9]. We note that an RF based power model will not be feasible for on-line power monitoring due to its high computing complexity. However, the RF algorithm can determine and quantify the relative importance of each hardware counter to power during the model fitting process. Therefore, we leverage this characteristic of RF algorithms to build the HCS method that works off-line.

The HCS method is designed to generate a list of hardware counters that are most relevant to power estimation. Algorithm 1 shows the proposed method. The key design principle is that the HCS method should be suitable for all applications and the hardware counters selected by the approach should not be dependent on a particular application. To obtain a general HCS method, we partition the dataset and obtain hardware counters for each subset to break any dependence on the dataset. The inputs to the algorithm are:

1) *all_vectors*, which is the set of all vectors (including all hardware counters from a processor) obtained from executing benchmarks using the multiplexing function of PAPI.

2) *n*, which is the number of hardware counters to be selected. In our case, we use six, which is the maximum number of counters obtained simultaneously using PAPI.

3) *ntree*, which is the number of trees that are used to build the random forest model for selection. This parameter is determined through an experimental exploration which estimates the effect of different values of *ntree* on the hardware counter selection result; the experimental results are not within the scope of this paper. As a conclusion, the HCS method is not sensitive to *ntree*. In detail, employing values of *ntree* which are not less than 2 on ARM and 16 on Intel, the HCS method selects the same hardware counters.

Algorithm 1 Hardware Counter Selection (HCS) Method

1: **procedure** SELECT_COUNTERS(*all_vectors*, *n*, *ntree*)
2:     *counters_selected* ← *list* ()
3:     **for** $i = 1$ to *M* **do**   ▷ The entire dataset is partitioned into *M* subsets
4:         *part_vectors* ← *extract* (*all_vectors*, *i*, *M*)   ▷ Extract the ith subsets from overall M subsets
5:         *rfes* ← randomForest(*part_vectors*, *ntree*)
6:         *counters_importance*[*i*] ← *rfes.importance*
7:     Find n hardware counters with largest average value of importance.
8:     **Return** n events with largest average value of importance

---

The algorithm first partitions *all_vectors* into a set of subsets (i.e. *M* subsets) (lines 3-4). During each run of the *for* loop (line 3) for each subset *i* which is extracted from the overall *M* subsets (line 4), a Random Forest model is used to map hardware counters to power (line 5). The importance of each hardware counter for a given partition is obtained and stored in the *counters_importance* array (line 6). Finally, *n* hardware counters with largest average importance values which are calculated by averaging the importance values of all subsets are found (line 7) and returned (line 8).

Evaluation of the HCS method: To evaluate the Hardware Counter Selection (HCS) method, hardware counters selected by the HCS method and by the manual expert based method reported in the literature are compared. Then we compare the accuracy of classic power models presented in Section 1, when using hardware counters obtained from our selection method against a baseline using hardware counters reported in the literature.

We use a rigorous training and testing strategy. All vectors obtained from profiling the execution of the benchmarks are partitioned equally into four parts. Then we use a combination of three parts to train the LRPM. The trained model is used to test: (i) vectors from the three parts used to train the model (75% of the vectors), referred to as *'Known'* vectors since they are known to the model through the training process; and (ii) vectors from the fourth part which were not used for training the model (25% of the vectors), referred to as *'Unknown'* vectors since they are not known to the model and were not used for training.

Table 1 and Table 2 show the hardware counters reported in the literature (which we use as a baseline) and selected by the the HCS method for ARM and Intel, respectively. The baseline is determined by reviewing existing research [5,10,11] and by considering the characteristics of our experimental platform and the profiling tool PAPI.

On the ARM and Intel processors we note that the hardware counters obtained from the HCS method are quite similar to those from the baseline (on ARM only one hardware counter is different and on Intel only two hardware counters differ). We infer from this that given different hardware processors our selection method can obtain appropriate hardware counters that capture dynamic power. It is also observed that the hardware counters for the ARM and Intel processors are different (4 out of the 6 hardware counters differ). The HCS method we propose can select processor dependent hardware counters that are suitable for developing power models.

We evaluated the accuracy in predicting power based on estimation error *Error*, which is defined as $Error = \frac{|P_{estimated} - P_{dynamic}|}{P_{dynamic}}$. Figure 1 shows the percentage of Known and Unknown vectors that can be accurately predicted with different error percentages

Table 1. Hardware counters from the baseline and selected by the HCS method on ARM

Hardware Counters		Description
Baseline	HCS Method	
PAPI_TOT_CYC	PAPI_TOT_CYC	Total cycles
PAPI_TOT_INS	PAPI_TOT_INS	Instructions completed
PAPI_TLB_IM	PAPI_TLB_IM	Instruction TLB misses
PAPI_L1_DCA	PAPI_L1_DCA	L1 data cache accesses
PAPI_L1_ICA	PAPI_L1_ICA	L1 instruction cache accesses
PAPI_L2_DCA	-	Level 2 data cache accesses
-	PAPI_L2_TCM	Level 2 cache misses

Table 2. Hardware counters from the baseline and selected by the HCS method on Intel

Hardware Counters		Description
Baseline	HCS Method	
PAPI_TOT_CYC	PAPI_TOT_CYC	Total cycles
PAPI_TOT_INS	PAPI_TOT_INS	Instructions completed
PAPI_LD_INS	PAPI_LD_INS	Load instructions
PAPI_SR_INS	PAPI_SR_INS	Store instructions
PAPI_FP_OPS	-	Floating point operations
PAPI_L3_TCA	-	L3 total cache accesses
-	PAPI_REF_CYC	Reference clock cycles
-	PAPI_L3_TCM	L3 cache misses

when using hardware counters of the baseline and the hardware counters from the HCS method on the ARM processor. Figure 2 corresponds to the Intel processor. In the best case, the HCS method on both processors performs better for Known and Unknown vectors than the baseline. The HCS method is automated in contrast to the baseline, but even in the worst case it provides near similar accuracy to the baseline. We evaluated the accuracy of power models based on Linear Regression (LRPM), Support Vector Machine (SVMPM) and Neural Network (NNPM). However, we present results based on LRPM, since the results and conclusions on all power models are similar.

(a) Known vectors

(b) Unknown vectors
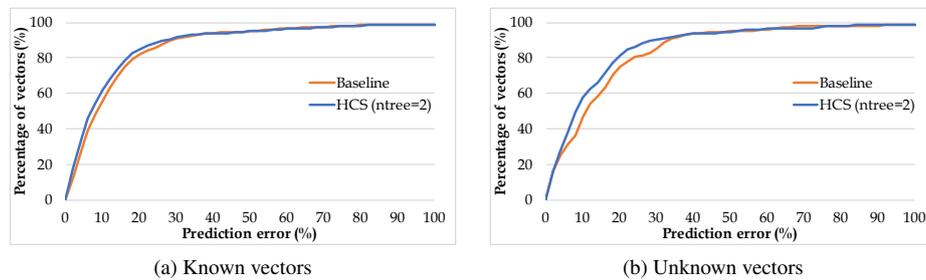
Figure 1. Accuracy of the LRPM employing hardware counters reported in the literature referred to as baseline and selected by the HCS method (*ntree* = 2) on the ARM processor
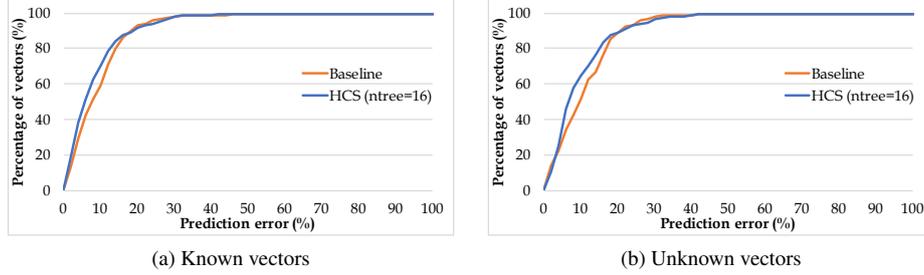
(a) Known vectors

(b) Unknown vectors

Figure 2. Accuracy of the LRPM employing hardware counters reported in the literature referred to as baseline and selected by the HCS method (*ntree* = 16) on the Intel processor
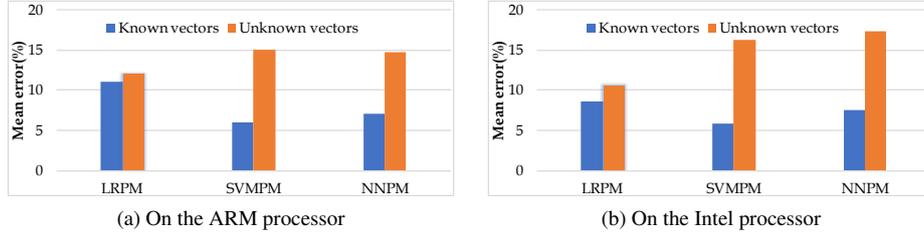

(a) On the ARM processor

(b) On the Intel processor

Figure 3. The accuracy of classic power models for Known and Unknown vectors

3. Design of a Two Stage Power Model

In this section, we explore three classic power models to understand their accuracy. This exploration motivates the need for the Two Stage Power Model (TSPM).

Motivation: We evaluate the three classic models by measuring the estimation error *Error* in predicting power. The HCS method selects the hardware counters as shown in Table 1 and Table 2. The training/testing strategy presented in Section 2 was used.

Figure 3 shows the mean error of the classic power models for predicting dynamic power when testing Known and Unknown vectors on the ARM and Intel processors. On both processors, it is evident that the SVMPM is more accurate for predicting dynamic power of Known vectors. This indicates that SVMPM fits the training data well. Compared to SVMPM, LRPM relatively under fits the data resulting in lower accuracy. However, for Unknown vectors LRPM is most accurate. This is surprising, but is because more sophisticated models, such as SVMPM and NNPM over fit data [12] and lead to lower accuracy for Unknown vectors than a simpler model, such as LRPM. There is no off-the-shelf power model that achieves accuracy of the best performing power model for both Known and Unknown vectors. This motivates the need for a new power model that reduces the effect of over-fitted models in predicting Unknown vectors than classic power models, but at the same time achieves low error rates for known vectors.

Design: In this section, we propose a Two Stage Power Model (TSPM) which takes advantage of the low variance of simple models, such as Linear Regression (LR) and of the low bias of sophisticated models, such as Support Vector Machine (SVM). We empirically identified that the combination of LR and SVM provided better accuracy than alternative classic models and was therefore chosen.

The TSPM operates in two stages. In the first stage, a LR based model is used to estimate a basic power value of an incoming vector. In the second stage, a SVM based model is employed to refine the basic power value to improve estimation accuracy.

Algorithm 2 Training process of TSPM

1: **procedure** TRAIN_MODEL(*training_vectors*)
2:    $LRPM \leftarrow build\_model(LR, training\_vectors)$ ▷ LR is the abbreviation of Linear Regression
3:    $difference\_vectors \leftarrow training\_vectors$ ▷ Initialize the training set for the difference model (DM)
4:    $n \leftarrow sizeof(training\_vectors)$ ▷ n is the number of vectors in *training_vectors*
5:    **for** $i = 0$ to $n-1$ **do**   ▷ Construct the training set for the difference model
6:        $basic\_value \leftarrow predict(LRPM, training\_vectors[i])$
7:        $difference \leftarrow training\_vectors[i, 1] - basic\_value$
8:        $difference\_vectors[i, 1] \leftarrow difference$
9:    $SVMDM \leftarrow build\_model(SVM, difference\_vectors)$
10:   **Return** $LRPM, SVMDM$

Training of TSPM: Algorithm 2 describes the training process of TSPM. First, an LRPM is developed using a training dataset consisting of profiled vectors as shown in Line 2. Then a difference based training dataset is constructed (Lines 3-8) by replacing the measured power of each vector in the original training set with the difference between the measured power and the value predicted by LRPM (Line 8). Finally, using the difference training set, a SVM based difference model is built (Line 9).

Prediction of TSPM: Algorithm 3 describes prediction of TSPM. For an incoming vector, both LRPM and SVMDM obtained from Algorithm 2 are used. The LRPM is used to predict the basic power value (Line 2) and the SVMDM is used to estimate the difference between the measured power and the estimated power of LRPM (Line 3). We offset the basic power value with the difference, such that the final predicted power is obtained by summing the basic power and the difference (Line 4).

Algorithm 3 Prediction process of TSPM

1: **procedure** PREDICT_POWER(*test_vector*, *LRPM*, *SVMDM*)
2:    $basic\_value \leftarrow predict(LRPM, test\_vector)$
3:    $difference \leftarrow predict(SVMPM, test\_vector)$
4:    $power \leftarrow basic\_value + difference$
5:    **Return** $power$

Comparing TSPM and Classic Power Models: In this section, the accuracy of the proposed TSPM against classic power models is evaluated. Figure 4 shows the prediction accuracy of TSPM in comparison to classic power models for both Known and Unknown vectors on ARM. For Known vectors, TSPM can achieve near similar accuracy to SVMPM (which has lowest prediction error for Known vectors). For Unknown vectors, TSPM obtains accuracy similar to the best classic model, which is LRPM when compared to SVMPM and NNPM. For example, nearly 60% of Unknown vectors can be predicted with error less than 10% using TSPM and LRPM.
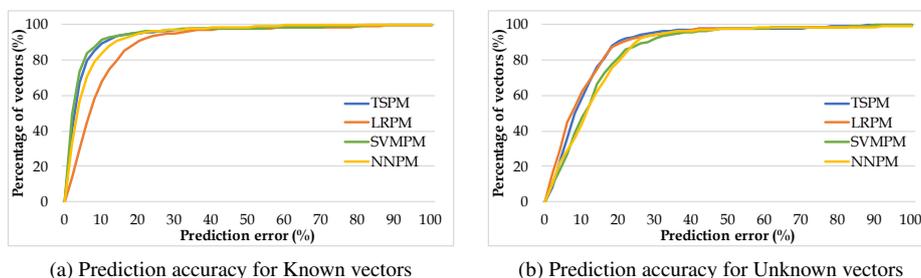
(a) Prediction accuracy for Known vectors

(b) Prediction accuracy for Unknown vectors

Figure 4. Prediction accuracy of different models on the ARM processor

(a) Prediction accuracy for Known vectors
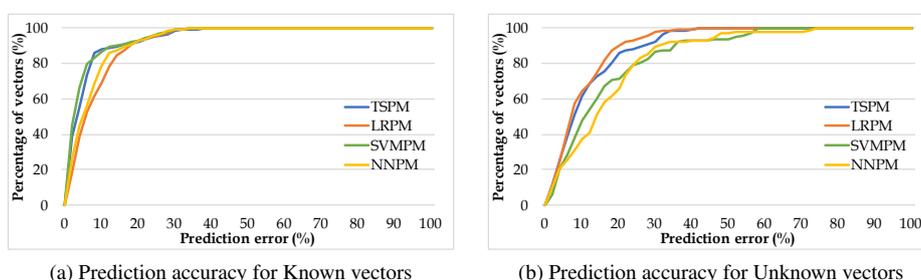
(b) Prediction accuracy for Unknown vectors

Figure 5. Prediction accuracy of different models on the Intel processor

Figure 5 shows prediction accuracy of TSPM when compared to classic power models for Known and Unknown vectors on Intel. On Intel for Known and Unknown vectors, TSPM performs similarly to the best classic power model.

4. Discussion and Conclusions

Research in power modelling has led to (i) instruction-level [13], (ii) coarse-grain utilisation [14], and (iii) hardware counter-based [2] models. Instruction-level models require extensive knowledge of the entire instruction set. It is cumbersome to obtain the power of each instruction and the overhead of all instruction pairs, thereby rendering these models impractical for real use. Although coarse-grained utilisation-based power models are easy to implement, they are not accurate since power depends not only on utilisation, but also on the type of operation. For example, floating point operations require more power than integer operations. Hardware counter-based power models are fine-grained utilisation models. These models are relatively simpler than instruction-level models, but at the same time are more accurate when compared to coarse-grained utilisation models.

However, distributed computing models, such as fog/edge computing make use of processors in data centers and at the edge of the network. There are two challenges that will limit the use of existing power models in these settings. Firstly, hardware counter selection is usually dependent on human expertise. This becomes challenging when heterogeneous processors are used. Secondly, existing models focus on either large processors or edge-like processors, but do not work across both architectures. This limits the use of existing models for end-to-end power modelling since platform independent cross

architectural models are required. Our research tackles both these challenges on multiple architectures by developing (i) an automated method for selecting hardware counters, and (ii) a two stage power model that performs better than existing models.

The research in this paper firstly developed a hardware counter selection method that appropriately selects hardware counters that capture power for both ARM and Intel processors. This selection method simplifies the hardware counter selection process without compromising accuracy. Secondly, we developed a two stage power model that surmounts the challenges in using existing power models across multiple architectures. We demonstrated that our model predicts dynamic power more accurately on both ARM and Intel processors when compared to classic power models.

References

[1] Y. Li, D. Wang, S. Ghose, J. Liu, S. Govindan, S. James, E. Peterson, J. Siegler, R. Ausavarungnirun, and O. Mutlu, "SizeCap: Coordinating Energy Storage Sizing and Power Capping for Fuel Cell Powered Data Centers," in *Proceedings of the IEEE Symposium on High-Performance Computer Architecture*, 2016, pp. 444–456.

[2] W. Huang, C. Lefurgy, W. Kuk, A. Buyuktosunoglu, M. Floyd, K. Rajamani, M. Allen-Ware, and B. Brock, "Accurate Fine-grained Processor Power Proxies," in *Proceedings of the Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 224–234.

[3] R. Bertran, M. Gonzelez, X. Martorell, N. Navarro, and E. Ayguade, "A Systematic Methodology to Generate Decomposable and Responsive Power Models for CMPs," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1289–1302, 2013.

[4] Y. S. Shao and D. Brooks, "Energy Characterisation and Instruction-level Energy Model of Intel's Xeon Phi Processor," in *Proceedings of the International Symposium on Low Power Electronics and Design*, 2013, pp. 389–394.

[5] 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 Modelling for Mobile and Embedded CPUs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 1, pp. 106–119, 2015.

[6] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A Framework for Edge Node Resource Management," *IEEE Transactions on Services Computing*, vol. PP, no. 99, pp. 1–14, 2017.

[7] Intel Corporation, "Intel 64 and IA-32 Architectures Software Developer's Manual," December 2015. [Online]. Available: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

[8] S. Browne, J. Dongarra, N. Garner, K. London, and P. Mucci, "A Scalable Cross-platform Infrastructure for Application Performance Tuning Using Hardware Counters," in *Proceedings of the ACM/IEEE Conference on Supercomputing*, 2000, pp. 42–54.

[9] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[10] S. K. Rethinagiri, O. Palomar, R. Ben Atitallah, S. Niar, O. Unsal, and A. C. Kestelman, "System-level Power Estimation Tool for Embedded Processor based Platforms," in *Proceedings of the 6th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, 2014, pp. 5–12.

[11] 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, 2013.

[12] "Model Fit: Underfitting vs. Overfitting," in *Amazon Machine Learning*. [Online]. Available: https://goo.gl/zcUzh6

[13] V. Tiwari, S. Malik, A. Wolfe, and M. T. C. Lee, "Instruction Level Power Analysis and Optimisation of Software," in *Proceedings of the International Conference on VLSI Design*, 1996, pp. 326–328.

[14] W. Dargie, "A Stochastic Model for Estimating the Power Consumption of a Processor," *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, 2015.