WoMA: An Input-Based Learning Model to Predict Dynamic Workload of Embedded Applications

Dongning Ma, Xun Jiao
Villanova University, PA, USA
{dma2, xjiao}@villanova.edu

***Abstract*—Embedded applications often have real-time requirements and thus meeting the user-specified deadline are essential for good user experience. Frequency (down)scaling is a widely-used technique to improve energy efficiency but may lead to violations of the user-specified deadline. Thus, a workload prediction model is of critical importance to guide the frequency scaling. In this letter, we propose `WoMA`, a supervised learning model to predict dynamic workload (execution cycles) of embedded applications based on their input data. `WoMA` is built on our key observation that the workload of some embedded applications is primarily determined by their input size. Using the "input size" as features, we apply a logistic regression method to construct `WoMA` trained and tested using five popular embedded applications: adpcm, aes, dfsin, blowfish, and gsm. Since frequency scaling is typically performed at discrete levels, for each application, we classify its workload into five distinct classes. For a given test input, `WoMA` will predict the class of its corresponding workload. Cycle-accurate simulation results show that 100% of `WoMA` predictions are accurate. Thus, we use `WoMA` to guide frequency scaling, resulting in power saving by 9.9%–61.8% across these benchmark applications.**

***Index terms*—** Embedded Systems, Frequency Scaling, Machine Learning

I. Introduction

Embedded systems have been increasingly used in our society due to a wide adoption of emerging Internet of Things (IoT) including smart health [6] and autonomous cars [2]. Regardless of the application scenario, power consumption is a primary concern in the design of embedded systems [10]. Dynamic voltage and frequency scaling (DVFS) is a typical method to reduce system power dissipation [1], [9], [11]. In particular, DFS is used to dynamically scale the frequency level of the system so as to provide "just-enough" circuit speed to process the system workload.

Many embedded applications are real-time applications that require a certain amount of response time. In this letter, we focus on applications with soft real-time requirements. That means, while meeting the timing deadline specified by users are essential for good user experience, finishing applications earlier than the timing deadline does not help improve the user experience. This creates an available slack. A DFS technique typically exploits this slack to improve power efficiency. Essentially, this exploitation requires a knowledge of application workload in advance and dynamically adjust the DFS effort to provide "just-enough" circuit speed. The prediction of dynamic workload, however, is challenging because the workload is determined by the input data.

Existing DFS research can be roughly classified into three categories based on their assumption of workload models. The first category assumes that the application workload is known in advance [11]. The second category utilizes static compilation analysis to determine worst-case workload [9]. The last category uses runtime statistics of applications to determine the workload [1]. Unfortunately, none of these works can actually predict the dynamic workload of embedded applications for a specific input stimuli before its execution. This prediction, however, is critical to design a smart DFS strategy before the actual execution of embedded applications.

To tackle this challenge, we develop **`WoMA`**, a machine learning-based model to predict the workload of an embedded application for a given input stimuli. This prediction is possible through our key observation between input stimuli and dynamic workload. The advantages of **`WoMA`** are threefold: (1) it does not require the knowledge of the source code of the programs; (2) it is an input-aware model and can predict the workload for different input; (3) it is a light-weight model using the logistic regression method. Specifically, our contributions are as following:

- We present our key observation of the relationship between application workload and input stimuli in five popular embedded applications, based on which we define such embedded applications as *size-oriented* applications.
- We develop **`WoMA`**, a machine learning model that can predict the application workload based on the input size. Across various *size-oriented* applications, **`WoMA`** exhibits 100% accuracy in classifying the workload.
- We demonstrate the application of **`WoMA`** by using it to guide dynamic frequency scaling. Results show that we can save 9.9% – 61.8% power consumption across five *size-oriented* applications.

II. Motivation Study

Fig. 1 shows an example program control flow graph with four nodes, where each node represents a basic block of which the workload is fixed, e.g., the workload of the first node is 1. The overall workload of this program is determined by the execution path. For example, if the program takes the left path and has 2 iterations on the last node, then its workload is $1 + 10 + 5 * 2 = 21$. If the program takes the right path and has one iteration on the last node, then the workload would be $1 + 1 + 5 = 7$. Hence, the workload is purely dependent on the program execution path. For programs without random

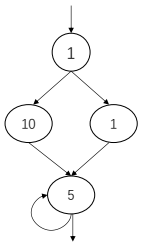


Fig. 1. Example control flow graph. Each node is annotated with its number of cycles (workload).

functions internally, (which is our assumption in this work), the program execution path is purely dependent on the input stimuli. That is, for a fixed program and fixed input stimuli, the workload is also fixed. Thus, the key to predict the workload is to establish the mapping relationship between the input stimuli and the program execution path.

III. Workload Analysis

In this section, we present our key observation of workload of several widely-used embedded applications in Mibench [4]: adpcm, dfsin, aes, blowfish, and gsm, which are defined as *size-oriented* applications. We characterize their workload as follows: First, we provide different input datasets as stimuli to the applications. For each input stimuli, we use the cycle-accurate Gem5 simulator under ARM architecture model to simulate the corresponding cycles, i.e., workload. Second, we use a Monte-Carlo method by repeating such simulation process for $1K$ times where we randomize the input stimuli each time. By this, we refer to randomizing both the input value and the input size. Note that for some applications, the workload is monotonically increasing with the increase of input data size, so we limit our input data size within a certain range to accommodate simulation time. For example, we constrain our input data size in adpcm less than $1K$.

Therefore, we design controlled experiments as follows:

- Case 1: We fix the input data size while randomly changing the input data value, and then check the workload accordingly.
- Case 2: We fix the input data value while randomly changing the input data size, and then check the workload accordingly. (We start with a large input data size and reduce the data size while keeping the input value the same.)

We illustrate some example results in Fig. 2. Each graph presents the workload under 9 different input stimuli, where each stimuli has different input value but some of them share the same input size. For example, in *adpcm*, the first three bars represent three different input stimuli with the same input size (195) but different input values. Specifically, for all the applications illustrated in Fig. 2, we can see that under a fixed input size, the workload has no noticeable change even if the input value are different. This indicates that the input value has no significant impact on the workload for these applications. On the other hand, once the input size changes, the workload also changes significantly. We define such applications of which the workload are strongly determined by input size as *size-oriented* applications. Moreover, we observe that in *size-oriented* applications, the workload increases monotonically with the increase of the input size. This is reasonable as we look into the source code of the applications. For example, adpcm needs to do encoding and decoding for every two consecutive data, where each encoding and decoding operation will consume the same amount of cycles no matter what the input values are. The same principle applies to the other applications where the input size is the primary factor that drives the workload. Thus, a larger input data size implies a larger workload.

IV. **WoMA** Construction

The strong correlation between input size and application workload inspires us to develop a supervised learning model to predict the application workload.

To get the training data, we measure $1K$ data points where for each point, we generate a random input dataset with a random size I and then measure its corresponding workload W through the Gem5 simulation. For each workload, we classify it into one of the five workload classes: high, mild-high, middle, mild-low, low. These five classes are divided evenly of the whole workload range observed in the Monte-Carlo experiments under $1K$ stimuli. We use five classes because the workload is used to guide dynamic frequency scaling, which, empirically, is performed at discrete levels such as five levels [7].

Thus, we set the input size, I as our input feature and $C(W)$ (workload class) as our output labels. We apply logistic regression methods on these training data to train **WoMA** based on the analysis of the near-linear relationship between input size and application workload. We use logistic regression for two reasons: (1) we only have one dimension of the feature (size) and based on the preliminary analysis, the workload monotonically increases with the input size; (2) logistic regression is light-weight and hence suitable for embedded system.

V. Experimental Results

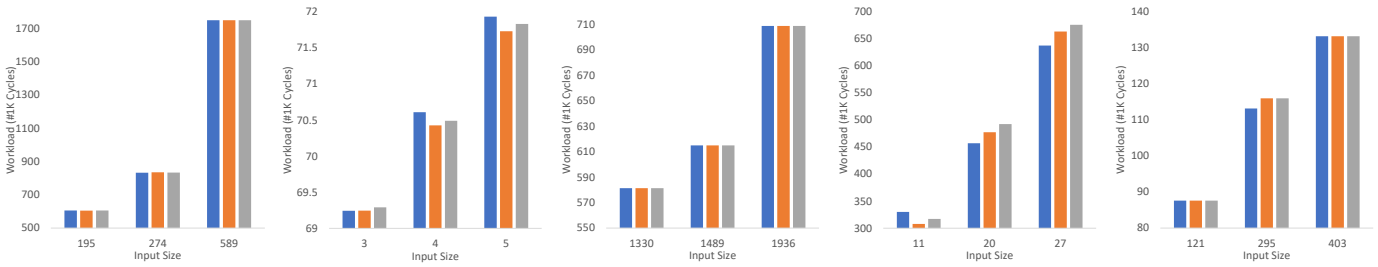
A. Experimental Setup

We use 500 random data points as our testing input stimuli. For each input, we extract the output workload through Gem5 simulation and map it into one of the five workload classes. We use five *sized-oriented* applications: adpcm, aes, dfsin, blowfish, and gsm.

B. Model Accuracy

Since there are no previous works on predicting dynamic workload of applications based on a given input stimuli, we compare **WoMA** against following baseline methods that can help us evaluate the true performance **WoMA**:

- **rand:** predict the dynamic workload class among the non-empty classes which contain at least one instance randomly. Some classes might have no instance , thus we ignore those empty classes.



(a) adpcm (b) dfsin (c) aes (d) blowfish (e) gsm

Fig. 2. Example workload snapshot of *size-oriented* applications

- **naive:** use a naive classifier to always predict the class which contains most instances. If the dataset is heavily biased, e.g., 99% of the data belongs to one class, then even a trivial class can achieve 99% prediction accuracy by always predicting that class.

Table. I presents the prediction accuracy of workload of five applications under 500 different input stimuli using three models: **WoMA**, **rand** and **naive**. **WoMA** exhibits the prediction accuracy always at 100% across all applications. The **rand** model achieves average prediction accuracy at 20.0% while **naive** can achieve 30.2% on average. Thus, compared to these baseline models, **WoMA** exhibits 5.0X and 3.3X higher prediction accuracy.

TABLE I
PREDICTION ACCURACY OF THREE DIFFERENT MODELS.

Applications	**WoMA**	**rand**	**naive**
adpcm	100%	20.0%	25.2%
aes	100%	20.1%	22.3%
gsm	100%	20.0%	47.1%
dfsin	100%	20.0%	24.7%
blowfish	100%	20.0%	31.9%

C. *WoMA-based Dynamic Frequency Scaling*

One application of **WoMA** is to guide DFS. Fig. 3 presents the workflow of **WoMA**-guided DFS with two phases: *workload prediction* and *frequency selection*. In *workload prediction* phase, the logistic regression method can be easily implemented with a MAC unit. Specifically, a MAC unit with pre-stored coefficients and bias will compute the prediction result based on the input size. Then in the *frequency selection* phase, the predicted workload w will be compared among several different workload-frequency (w_i, f_i) pairs and select the proper frequency. Note that the frequency is selected to provide "just enough" speed.

$$P = CfV^2 \tag{1}$$

Let us take Blowfish application as an example and assume it has a deadline of 0.007s. There are five frequency settings, corresponding to the five workload classes as shown in Table II. We calculate the frequency which can satisfy the deadline for a given specific workload range. For example, if a workload falls into a "low" class (360K-430K), it would need a frequency at $61.4MHz$ ($430K/0.007$) to meet the deadline. Actually, any workload that falls into the "low" class can meet the deadline with $61.4MHz$. In contrast, without the **WoMA**-based DFS, the processor would always run at worst case $101MHz$ ($710K/0.007$) frequency based on the "high" class. Then, according to Eq. 1, dynamic power consumption is proportional to the frequency. Thus, if we run the processor at $60MHz$ compared to $100MHz$, we can save dynamic power consumption by 40% = ($\frac{100-60}{100}$). Generally, for an application with deadline T, the power saving at frequency f_i is $\frac{f_5-f_i}{f_5}$ = $\frac{w_5/T-w_i/T}{w_5/T}$ = $\frac{w_5-w_i}{w_5}$. For example, running processor at $60MHz$ is the best case scenario because the workload falls into "low" class. If the workload falls into the "middle" class at which the frequency is $80MHz$, the power consumption would be 20% according to Eq. 1. Thus, to calculate the average power saving across multiple input stimuli, we use a weighted sum over all workload classes. That is, we compute the probability of each workload class p_i and multiply their corresponding power saving $\frac{w_5-w_i}{w_5}$ as shown in Eq. 2.

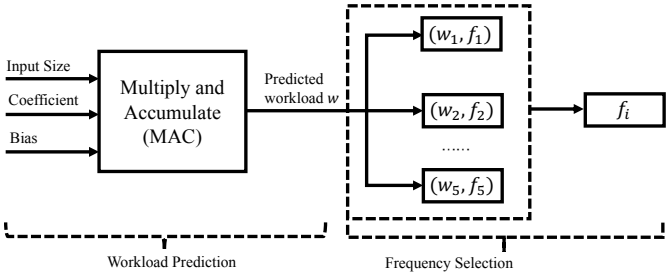


Fig. 3. **WoMA** guided dynamic frequency scaling

TABLE II
AN EXAMPLE OF (w_i, f_i) PAIRS OF BLOWFISH APPLICATION FOR A DEADLINE AT 0.007S.

class	workload	frequency	power saving
low	360k-430K	$\approx 60MHz$	40%
mild-low	430K-500K	$\approx 70MHz$	30%
middle	500K-570K	$\approx 80MHz$	20%
mild-high	570K-640K	$\approx 90MHz$	10%
high	640K-710K	$\approx 100MHz$	0

$$pwr\_saving = \sum_{i=1}^{4} \frac{w_5 - w_i}{w_5} * p_i \quad (2)$$

where w_i is the max workload of class i (e.g., the 500K in "mild-low" class for blowfish), p_i is the probability of a specific class. Note that the "high" class would not contribute to the power reduction as it requires the worst case frequency.

Using Eq. 2, we can compute the average power saving of the five applications as illustrated in Fig. 4. We can observe several facts. First, aes achieves lowest power savings (9.9%). This is because the workload distribution of aes is very narrow, limiting its extent of frequency scaling. Second, both adpcm and dfsin can achieve more than 60% power savings because their workload distribution is widely spread across five classes. Third, the gsm also achieves relatively low power saving because its "high" class accounts for a large portion of the workload distribution and the "high" class would not contribute to the power reduction because they belong to the worst-case scenario.

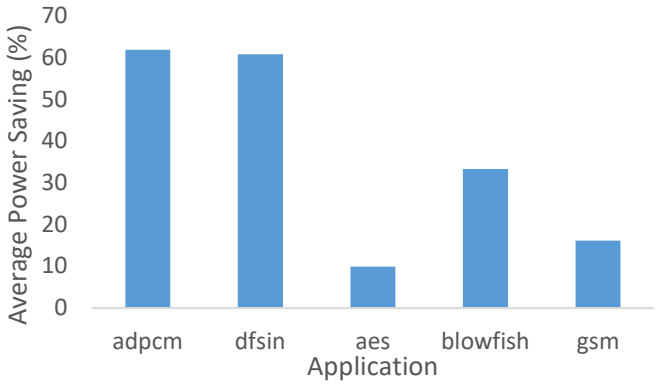


Fig. 4. Average power savings using **WoMA**-based DFS

.

VI. Related Work

DVFS have been studied intensively over past decades. One category of DVFS assumes that the application workload is fixed and known in advance [11]. Thus, designers can tune DVFS level at the granularity of application-level to meet the timing constraints of applications. This assumption is not practical as application workload is determined by inputs. The second category of DVFS-related work utilizes static analysis at the code level to characterize the workload for performing DVFS. Typically, the compiler will analyze the application and generate useful information about application workload. For example, in [9], the compiler divides the application code into a set of blocks and for each block, the compiler can generate the worst-case execution time (WCET). Thus, the total workload of the application is the sum of WCET of each block. This category assumes that the knowledge of the source code is available for analysis. The third category of DVFS-related work make no assumptions about application workload and does not rely on compiler support. Typically, these approaches will gather the runtime statistics information of the applications through the use of simulation or a hardware-based unit. For example, in [8], the simulator generates the information of cache miss to drive the voltage scaling setting. In [3], the simulator generates the information of IPC (instruction per cycle) rate to direct the voltage scaling. These works relied on either simulations or extra hardware monitor modules to obtain the runtime information.

Our work is different from all of these works because our work does not require compiler nor hardware support to model the application workload and hence guiding DFS. In addition, our work is complementary to recent power saving methods such as approximate computing [5] because our work does not degrade application quality.

VII. Conclusion

In this letter, we present our analysis on the workload of embedded applications, based on which we define *size-oriented* applications. We then construct a machine learning model **WoMA** that can predict the workload of *size-oriented* applications given a specific input. We train the model with data generated from cycle-accurate simulations and use logistic regression as training method to build **WoMA**. We evaluate the performance of **WoMA** across several benchmarks and demonstrate its high prediction accuracy. By using it to guide dynamic frequency scaling, we can save 9.9%-61.8% power across five *size-oriented* applications.

References

[1] Kihwan Choi, Ramakrishna Soma, and Massoud Pedram. Dynamic voltage and frequency scaling based on workload decomposition. In *Proceedings of the 2004 international symposium on Low power electronics and design*, pages 174–179. ACM, 2004.

[2] Mario Gerla, Eun-Kyu Lee, Giovanni Pau, and Uichin Lee. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In *IEEE world forum on internet of things*, pages 241–246. IEEE, 2014.

[3] Soraya Ghiasi, Jason Casmira, and Dirk Grunwald. Using ipc variation in workloads with externally specified rates to reduce power consumption. In *Workshop on Complexity Effective Design*, 2000.

[4] Matthew R Guthaus, Jeffrey S Ringenberg, Dan Ernst, Todd M Austin, Trevor Mudge, and Richard B Brown. Mibench: A free, commercially representative embedded benchmark suite. In *IEEE International Workshop on Workload Characterization*, pages 3–14. IEEE, 2001.

[5] Jie Han and Michael Orshansky. Approximate computing: An emerging paradigm for energy-efficient design. In *2013 18th IEEE European Test Symposium (ETS)*, pages 1–6. IEEE, 2013.

[6] Yu Jiang, Houbing Song, Rui Wang, Ming Gu, Jiaguang Sun, and Lui Sha. Data-centered runtime verification of wireless medical cyber-physical system. *IEEE transactions on industrial informatics*, 13(4):1900–1909, 2016.

[7] Xun Jiao, Yu Jiang, Abbas Rahimi, and Rajesh K Gupta. Wild: A workload-based learning model to predict dynamic delay of functional units. In *IEEE 34th International Conference on Computer Design (ICCD)*, pages 185–192. IEEE, 2016.

[8] Diana Marculescu. On the use of microarchitecture-driven dynamic voltage scaling. In *Workshop on Complexity-Effective Design*, 2000.

[9] Dongkun Shin, Jihong Kim, and Seongsoo Lee. Low-energy intra-task voltage scheduling using static timing analysis. In *Proceedings of the 38th annual Design Automation Conference*. ACM, 2001.

[10] Tajana Simunic, Luca Benini, and Giovanni De Micheli. Cycle-accurate simulation of energy consumption in embedded systems. In *Proceedings of Design Automation Conference*. IEEE, 1999.

[11] Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In *Annual Symposium on Foundations of Computer Science*. IEEE, 1995.