GPGPU Power Estimation with Core and Memory Frequency Scaling

Qiang Wang
Department of Computer Science, Hong Kong Baptist University
qiangwang@comp.hkbu.edu.hk

Xiaowen Chu
Department of Computer Science, Hong Kong Baptist University
chxw@comp.hkbu.edu.hk

ABSTRACT

With the increasing installation of Graphics Processing Units (GPUs) in supercomputers and data centers, their huge electricity cost brings new environmental and economic concerns. Although Dynamic Voltage and Frequency Scaling (DVFS) techniques have been successfully applied on traditional CPUs to reserve energy, the impact of GPU DVFS on application performance and power consumption is not yet fully understood, mainly due to the complicated GPU memory system. This paper proposes a fast prediction model based on Support Vector Regression (SVR), which can estimate the average runtime power of a given GPU kernel using a set of profiling parameters under different GPU core and memory frequencies. Our experimental data set includes 931 samples obtained from 19 GPU kernels running on a real GPU platform with the core and memory frequencies ranging between 400MHz and 1000MHz. We evaluate the accuracy of the SVR-based prediction model by ten-fold cross validation. We achieve an average Mean Square Error (MSE) of 0.797 Watt and Mean Absolute Percentage Error (MAPE) of 3.08%, which are much better than existing results. Combined with an existing performance prediction model, we can find the optimal GPU frequency settings that can save an average of 17% energy across 12 GPU kernels.

1 INTRODUCTION

Over the past few decades the Graphics Processing Units (GPUs) have been increasingly adopted to massive parallel computing across a wide range of industrial and academic areas, including big data analysis [7], bio-informatics [26], image recognition based on deep neural network [10], etc. Apart from 4.36 TFLOPS of the peak single-precision performance, modern GPUs also bring brilliant advantages in performance-per-watt. In the GREEN500 supercomputer list [8] of November 2016, 6 of the Top 10 are equipped with GPUs. Especially the top 1 machine can provide up to 9 GFLOPS per watt with NVIDIA Tesla P100. However, to deal with huge amout of data generated every day, to manufacture such systems could suffer from high electricity cost. One example is the Google's DeepMind infrastructure [21], which is built with about 600 million dollars but consumes 150 million dollars of electricity annually. These facts reveal great emergency and potential of energy conservation of GPUs with hunger of efficient power management techniques.

Dynamic Voltage and Frequency Scaling (DVFS) [12] is a traditional and useful technique to save energy of modern computers. It allows the processors to achieve better energy efficiency with proper voltage and frequency settings. Compared with relatively mature CPU DVFS technology, GPU DVFS is still at an early stage. Unfortunately, according to existing studies [1, 16, 24], CPU DVFS technology can not be directly adopted to GPUs. That might be caused by the fact that modern GPUs have two main frequency domains (core and memory frequencies), which makes it more complicated. Besides, scaling up the frequency is proved be energy efficient for CPUs but not always for GPUs. To achieve the maximum energy conservation of one given GPU application with DVFS, it is necessary to predict its energy consumption, which requires precise power estimation under different frequency settings. Besides, such power estimation could also help solve energy efficient tasking scheduling problem on CPU-GPU grid which has been discussed by [4, 15].

Several previous literatures [1, 16] have explored much of GPU power characteristics with DVFS and reveal the fact that to achieve the best energy efficiency needs careful frequency determinations. Usually neither the lowest nor the highest is the optimal solution. To model those complex correlations, recent research papers [1, 6, 25] attempt to apply machine learning methods. From linear regression to K-means clustering, they reveal the considerable potential to precisely catch the statistical correlations between performance metrics of the GPU applications and the final power consumptions. However, there might exist one or more drawbacks of each work. For example, some of them applied too few performance features of the GPU kernels which may result in large bias from the ground truth. Furthermore, some works adopted too simple pre-processing, especially like normalization, to the performance features before feeding them to the machine learning algorithm. Even few of them considered the effects of core and memory frequency scaling on the final power consumption.

To address those problems, this paper proposes a power estimation model based on support vector regression machine (SVR) which is applicable to real GPU hardware. With special treatments to the features extracted by the profiling tool during kernel runtime, we can utilize them as input of the SVR model to let it master the effects on the power consumption caused by both core and memory frequency scaling. Our experimental results show that our power prediction model achieves 3.08% Mean Average Percentage Error (MAPE) among 19 tested kernels when estimating the average power consumptions of new frequency settings, which includes totally 49 different frequency settings with up to 2.5x scaling range.

The rest of this paper is scheduled as follows. Section 2 summarizes some related work about power characterization and modeling of GPUs. Section 3 gives the basic idea of DVFS as well as an overview of power characteristics of GPU with frequency scaling. Section 4 proposes our GPGPU power estimation model with SVR. Model evaluation and experimental results are revealed in Section 5. Finally we conclude our work in Section 6.

2 RELATED WORK

In this section we review some previous work about GPU DVFS characteristics and power modeling. As mentioned before, GPU DVFS can be very different from traditional CPU DVFS. Mei et al. [16] conducted real hardware experiments on Fermi GTX560Ti and Maxwell GTX980 and observed that the best energy efficient setting of core voltage and two frequency domains varies from kernel to kernel. Even the default setting is usually not the best choice, which may bring up to 34% potential energy wastes. Jared et al. [9] explored energy, power and performance characterizations of 34 selected GPGPU benchmark programs by varying the frequency, input data size and ECC switch and revealed some relationships between average runtime power and those three factors, which suggests the potential of energy conservation with proper settings.

As for power modeling work, Hong and Kim [11] estimated the access rates of different components on the GPU based on the dynamic number of instructions and the cycles of each instruction. They then designed a suite of micro-benchmarks to search for separated power of those components. After that they could estimate the runtime GPU power consumption of a new kernel. However, today's GPU architecture can be more complicated with new hardware features and instructions, which gives much more challenges of power modeling. Leng et al. packed Hong and Kim's power modelling to GPGPUSim, to form GPUWattch, which could estimate the runtime GPU power with different voltage/frequency settings at cycle-level [13]. The authors refined Hong and Kim's model with supplemental micro-benchmarks, to overcome the power uncertainties brought by the new Fermi hardware. The prediction error was 15% for the micro-benchmarks, and 9.9% for the general GPU kernels, on the Fermi GTX480 GPU. Although those papers achieve good accuracy, they can not avoid trivial micro-benchmark procedures for new hardware and long time simulations.

Recent years witness the popularity of statistical methods applying to GPU power modeling, which treats the GPU hardware as a black box and predicts the power with some monitored runtime signals. Abe et al. built regression models to estimate the average power consumption of the NVIDIA Tesla, Fermi and Kepler GPUs under different levels of core and memory frequency [1]. Particularly, they defined three core/memory frequency settings as part of the model inputs. They also chose 10 most relative performance counters who provided the best-fitting results. The average prediction errors range from 15% to 23.5% depending on the kernel characteristics and the generations of GPU architecture. Newer hardware had larger prediction error, which indicates that simple linear regression is not sufficient to catch those correlations between power and performance counters.

Song et. al [22] attempted to use artificial neural network (ANN) to model the non-linear relationship between runtime events and runtime average power of a given GPU kernel. They use two different independent threads to collect performance events with CUPTI API and power metrics with NVML API respectively. Then they select a collection of most relevant events as input of ANN. Their model achieved average prediction error of 2.1%. Wu et al. extensively studied the GPU power and performance, with different settings of GPU frequency, memory bandwidth and core number [25]. They adopted K-means clustering and ANN simultaneously. In the ANN modeling process, they first used K-means to cluster the kernels according to the similarity of scaling behaviors. Then for each cluster, they trained an ANN with two hidden layers. The reported average power prediction error over all frequency/unit configurations was 10%.

3 BACKGROUND AND MOTIVATION

3.1 Dynamic Voltage and Frequency Scaling

As mentioned before, DVFS is one of the most important techniques for energy conservations of not only traditional CPUs but also GPUs. The dynamic power is usually modeled by Equation (1), where a denotes a utilization ratio, C denotes the capacity, V denotes the chip supply voltage and f denotes the frequency [12]. Since the total energy consumption of one application is the product of average runtime power and total execution time, power modeling plays an important role in energy conservation with different DVFS settings.

$$P_{dynamic} = aCV^2f \tag{1}$$

Notice that the power has a linear correlation with the frequency. Generally, it is true for traditional CPUs. However, some previous GPU DVFS work indicates that GPUs have more complex power scaling behaviors when adopting different frequencies [16]. One reason is that modern GPUs have two main frequency domains. One is core frequency, which mainly controls the speed of stream multiprocessors (SMs), while the other is memory frequency, which affects the bandwidth of DRAM. Table 1 summarizes the dominating frequency for different types of memory. Note that only DRAM works under memory frequency and L2 cache works under core frequency though they both serve the global memory requests.

Table 1: Dominating Frequency for different components.

Components	Dominating Frequency
DRAM	memory frequency
L2 Cache	core frequency
Shared Memory	core frequency
Texture Cache	core frequency
Register	core frequency
CUDA cores	core frequency
Special Function Units	core frequency

3.2 Power Characteristics of Frequency Scaling

Because different applications can have diverse workloads on different units on the GPU, scaling both core and memory frequency may result in completely disparate power scaling behaviors on them. In this section, we demonstrate an example of Nvidia GTX 980 [17] to illustrate the effects of different frequency settings on its power consumption.

We first fix the core frequency to 400 MHz and 1000 MHz respectively and scale the memory frequency from 400 MHz to 1000 MHz with a step size of 100 MHz. Figure 1(a) and 1(b) show the results of six GPU kernels with those settings. The power scaling behaviors could be either simple or complicated among different GPU kernels as well as different frequency settings. For example, the

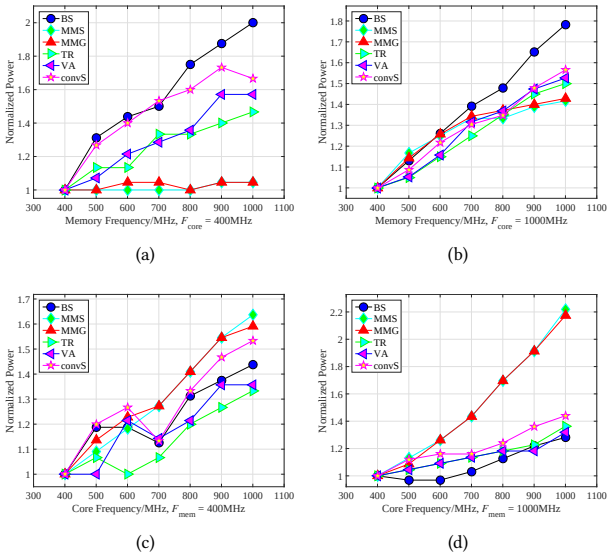

Figure 1: Power scaling behavior under different frequency settings. The upper two figures show the normalized power consumption of different GPU kernels when increasing memory frequency with fixed core frequency. The below two figures show the normalized power consumption of different GPU kernels when increasing core frequency with fixed memory frequency.

approximately linear correlations between power and frequency can be observed from Figure 1(b) while there exist more complex relationships than linearity in Figure 1(a). Furthermore, some kernels have different sensitivity to two frequency domains. These phenomena happen to Matrix Multiplication with global memory (*MMG*) and with shared memory (*MMS*). Their power consumptions have very few changes when scaling up core frequency with fixed 400 MHz memory frequency, which differs from the linear increments in 1000 MHz memory frequency setting. Then we fix the memory frequency to 400 MHz and 1000 MHz respectively and scale the core frequency from 400 MHz to 1000 MHz. Figure 1(c) and 1(d) show that *MMG* and *MMS* have linearly increasing power consumptions with scaling up either core or memory frequency this time. Other kernels also display different scaling behaviors from the core-frequency-fixed cases.

Notice that the power scaling behaviors are rather complex and somehow non-linear with respect to not only frequency settings but also GPU kernel characteristics. We would like to use statistical methods to help estimate the power consumption under different frequency settings of the target GPU kernel.

4 POWER ESTIMATION WITH CORE AND MEMORY FREQUENCY SCALING

4.1 Feature Selection

Thanks to the programmability provided by Compute Unified Device Architecture (CUDA), not only the software developments become simpler on modern NVIDIA GPUs, but also the performance profiling becomes available in a convenient way. We adopt CUDA version 8.0 in our work since it has good compatibility among a wide range of GPU generations and also provides sufficient supports for new features of recent GPU architectures. CUDA 8.0 provides *nvprof* [19] which can capture more than 50 performance counters for analyzing kernel performance. We choose 13 major counters listed in Table 2 which significantly affect the average power. *achieved_occupancy* and *eligible_warps_per_cycle* indicate the utilization of stream-multiprocessors on GPUs. *dram_read_transactions* and *dram_write_transactions* mean the number of DRAM transactions happened during kernel execution while *l2_read_transactions* and *l2_write_transactions* mean the number of L2 cache transactions. Since shared memory is also widely used in GPU kernels for performance optimization, we also include *shared_load_transactions* and *shared_store_transactions*. *branch_efficiency* and *cf_executed* reflect the divergence level of control flow while the rest three *float_count_** evaluate the workload of single/double precision floating-point operations.

However, it might cause large errors if such metrics data are directly adopted to statistical methods. First, since most of the profiling metrics represent the total number of instructions or transactions of the corresponding events, they could have extreme different magnitudes if the original GPU kernels have different workloads. Second, even for the same kernel, the distribution among different types of instructions can be considerably uneven. To better take advantage of a data-driven model, we should conduct careful feature pre-processing according to not only the variable value itself but also the characteristic of the feature.

4.2 Power measurement

To measure the power consumption of the tested GPU kernels, we use the tool *nvidia-smi* [20], which is a command line utility based on top of NVIDIA GPU driver API designed for the management and monitoring of NVIDIA GPU devices. We choose the sampling frequency at 1 read per second so that the impact of *nvidia-smi* on the application performance is negligible. We also add ten seconds of sampling before and after the kernel execution as safeguards. We revise each tested application to let them run sufficient iterations so that the GPUs are running for at least 10 minutes and generate more than 600 power samples. It is also well-known that the GPU temperature can also affect the runtime power consumption. Our current work focuses on the effect of frequency scaling, hence we control the GPU temperatures between 45 °C to 55 °C through fan speed adjustment. To verify that our temperature range does not bring obvious variance to the sampling results, we conduct significance test with t-distribution on the power samples of each kernel and achieve 95% confidence interval. The result also suggests the repeatability of our experiments. We will leave the investigation of the impact of GPU temperature on power consumption as our future work.

4.3 Power modeling with SVR

Some researchers established statistical models to estimate the GPU runtime power consumption. Those models include square linear regression (SLR) [1], support vector regression with linear kernel (SVR) [14], etc. Most of them attempt to fit a linear relationship

Table 2: Adopted CUDA performance counters

Metrics	Description
achieved_occupancy	Ratio of the average active warps per active cycle to the maximum number of warps supported on a multiprocessor
eligible_warps_per_cycle	Average number of warps that are eligible to issue per active cycle
dram_read_transactions	Device memory read transactions
dram_write_transactions	Device memory write transactions
l2_read_transactions	Memory read transactions seen at L2 cache for all read requests
l2_write_transactions	Memory write transactions seen at L2 cache for all write requests
shared_load_transactions	Number of shared memory load transactions
shared_store_transactions	Number of shared memory store transactions
branch_efficiency	Ratio of non-divergent branches to total branches expressed as percentage
cf_executed	Number of executed control-flow instructions
flop_count_dp	Number of double-precision floating-point operations executed by non-predicated threads
flop_count_sp	Number of single-precision floating-point operations executed by non-predicated threads
flop_count_sp_special	Number of single-precision floating-point special operations executed by non-predicated threads

tightly so that the model can explain the contribution of each input variable. However, since the GPU architectures are becoming more and more sophisticated, simple linear interpreters become insufficient to model the relationship between the extracted kernel features and the power consumptions, which is also validated by our motivation examples in Section 3.2.

Support Vector Regression (SVR) is a classical supervised learning algorithm for data regression. Mathematically, the basic idea is to predict y given the input x with the function f in Eq. (2) by solving the optimization problem in Eq. (3). $< \omega, x >$ represents the kernel function used in the model, which helps map the original input data to a higher dimension. This is useful when exploiting the non-linear correlation between the input variables and the output. Thus, we apply SVR in modeling the average GPU runtime power consumptions. The concrete algorithms can be found in [2].

$$f(x) =< \omega, x > +b, \omega \in \mathbb{R}^d, b \in \mathbb{R} \tag{2}$$

$$\begin{aligned} & min \frac{1}{2} \parallel \omega \parallel \\ & s.t. \parallel y - (< \omega, x > -b) \parallel < \epsilon \end{aligned} \tag{3}$$

- **Pre-processing of the input features:** Instead of directly using the profiling metrics as input data of the SVR model, we conduct two special pre-processing on them. First, since some performance counters represent the number of different types of operations and are typically very large integers, we normalize this type of metrics, including (a)*dram_read_transactions*, (b)*l2_read_transactions*, (c)*shared_load_transactions*, (d)*dram_write_transactions*, (e)*l2_write_transactions*, (f)*shared_store_transactions*, (g)*flop_count_dp*, (h)*flop_count_sp* and (i)*flop_count_sp_special*, by dividing each dimension by the summation of them. Since our objective is to predict the average runtime power consumption, what we should concern is the relative workload information of different units on the GPU, which can be retained with that normalization. Second, to introduce the effects of frequency scaling, we multiply all the transaction and operation variables by the ratio of the relative frequency domain to the baseline frequency setting. For example, global memory speed is influenced by the memory frequency while l2 cache and shared memory speed is influenced by the core frequency. The baseline frequency setting is 400MHz core frequency and 400MHz memory frequency.
- **Model Selection:** We have tried different kernel functions including linear kernel, gaussian kernel and polynomial kernel. We also use grid search to find the optimal hyper-parameters of each kernel, which is not listed here due to space limitation. Finally we choose the polynomial kernel with degree three. We use libSVM [3] as the implementation and set *Epsilon*, which is half the width of epsilon insensitive bend, to be 0.1 and $OutlierFraction$, which is the expected fraction of outliers, to be 0.2. To evaluate the generalization capability of the model, we adopt ten-fold cross validation.

5 PERFORMANCE EVALUATION

We apply the proposed power prediction model to 15 GPU kernels from CUDA SDK 8.0 [18] and 4 from Rodinia [5], which are listed in Table 3, to evaluate its accuracy. In the ten-fold cross validation, the data set is separated evenly to ten subsets. For each subset, we use mean squared error (MSE) to assess the model accuracy trained by the other nine subsets. The details of the hardware platform are listed in Table 4.

5.1 Training and Evaluation Samples

We first run **nvprof** on all the tested GPU kernels under the configuration of 400MHz core frequency and 400MHz memory frequency and collect all the required performance counters data. Then we measure the power consumption of each kernel under both core and memory frequency scaling from 400MHz to 1000MHz with a step size of 100MHz so that all the power data of totally 49 frequency settings are collected. Combined with the profiling data, we finally obtain 931 samples for SVR modeling.

We randomly select 40 frequency pairs and pick out the corresponding samples to train the SVR model and record the model as well as the *MSE*s of the cross validation and the mean absolute percentage error (MAPE) of the final model on the remaining testing samples. We call it one *training group* and repeat it one hundred times so that a total of one hundred training groups are obtained.

5.2 Experimental Results

We first analyze the distributions of profiling metrics among all the tested kernels. As Fig. 2 demonstrates, our tested kernels have various partitions of different types of profiling metrics, which somehow makes it more convincing of the generalization of our SVR model if the prediction accuracy is high. Unlike the performance modeling, it seems that the relationship between these profiling

Table 3: Tested Applications

abbr.	Application Name	Benchmark Suite
BP	backprop	Rodinia
BFS	breath First Search	Rodinia
BS	BlackScholes	CUDA SDK
CG	conjugateGradient	CUDA SDK
convSp	convolutionSeparable	CUDA SDK
convTx	convolutionTexture	CUDA SDK
FWT	fastWalshTransform	CUDA SDK
Hist	histogram	CUDA SDK
MMG	matrixMul(Global)	CUDA SDK
MMS	matrixMul(Shared)	CUDA SDK
MS	merge sort	CUDA SDK
NN	k-Nearest Neighbors	Rodinia
quasG	Quasi random Generator	CUDA SDK
SP	scalarProd	CUDA SDK
SC	scan	CUDA SDK
SQ	Sobol QRNG	CUDA SDK
SN	sortingNetworks	CUDA SDK
TR	transpose	CUDA SDK
VA	vector addition	CUDA SDK

Table 4: Target GPU frequency configurations

Device	GTX 980
Compute apability	5.2
SMs * cores per SM	16 * 128
Global Memory bus width	256-bit
Global Memory size	4GB
Core frequency scaling	[400MHz, 1000MHz]
Memory scaling	[400MHz, 1000MHz]
Scaling stride	100MHz

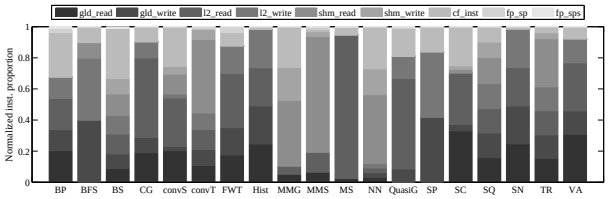

Figure 2: Breakdown of profiling metrics

metrics and power consumption scaling is more complicated. For example, **MMS** has larger partition of shared memory transactions while the global memory transactions take majority of **MMG**. However, their power scaling behaviors seem to be close according to Figure 1. This phenomenon indicates the necessity of using machine learning methods.

We present the results of all the training groups in Figure 3. The mean square error (MSE) of the leave-out validation subset of each training group is used to evaluate the model accuracy and stability. The average MSE is 0.797 W with 5.5×10^{-3} variance. These results indicate the high accuracy and decent stability of our SVR model. Based on them we can conduct further validation on the testing samples.

We aggregate the absolute precision errors of the testing samples of all the training groups in terms of frequency settings and use a heatmap shown in Figure 4 to illustrate the results. Each entry in Figure 4 is the average error of the certain frequency domain. Notice that the upper left triangle area has shallower gray scale indicating higher error values show up in lower frequency domains while the model performs better on the higher frequency domains with less than 3% error on some particular frequency settings.

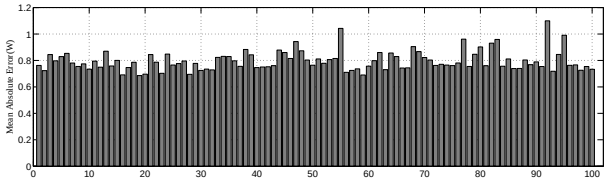

Figure 3: Mean square errors across all training groups

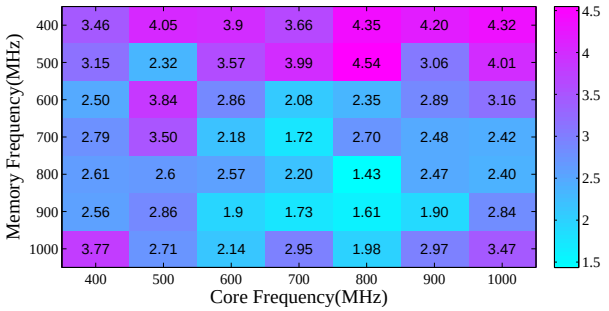

Figure 4: Mean absolute precision Error heat map among all tested frequency pairs. Each point represents the average error of all validation samples of the certain frequency setting.

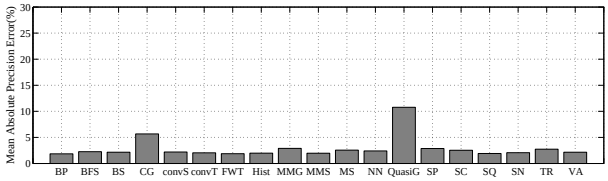

Figure 5: Mean absolute percentage error average across all available frequency pairs

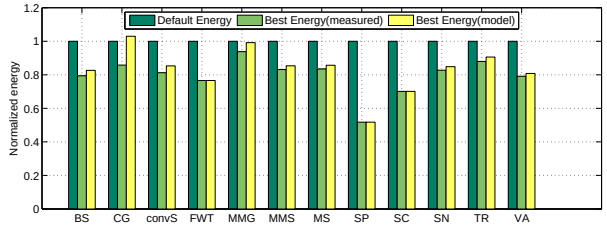

Figure 6: Energy Conservation achieved most by ground-truth and modeling

We demonstrate the average MAPE of each GPU kernel in Figure 5. The MAPE ranges from 1.5% to 5.7% across all the testing frequency pairs of each kernel except **QuasiG**. We finally achieve 3.08% MAPE across all the testing samples, which indicates remarkably high accuracy of our SVR model.

5.3 Energy Conservation

Combined with the performance model proposed by [23], we can use our SVR-based power prediction model to find the optimal frequency setting for a given kernel, which can achieve the best energy efficiency. We take the energy consumption under 1000 MHz for both core and memory frequency as the baseline. Then we compare the best energy efficiency achieved in measurements with that predicted by our model. Figure 6 illustrates the results. First, to conserve most energy, the highest frequency, which drives the program run fastest in general, is rarely the best choice. Second, except *CG* and *MMG*, the modeling results are basically similar to the ground truth. With the prediction from our model, we finally achieve up to 45% energy saving and an average of 17% among 12 tested GPU kernels. Both the performance and power models only need one-run profiling to extract the features of the GPU kernel and an off-line training, which makes it possible to determine the best frequency settings for energy conservation in runtime.

6 CONCLUSION

In this paper, we derive an SVR-based prediction model to estimate the average power consumptions of different core and memory frequency settings. Our approach provides not only good stability but also decent accuracy. The model takes the profiling data of a given kernel as input to estimate the average power consumptions under new frequency combinations which do not appear in the training data. We show that our model can achieve 3.08% MAPE across up to 2.5x both core and memory frequency scaling. Combined with a proper performance model, we can easily determine the most energy-efficient frequency configurations for a give GPU kernel. Our experimental results show that using GPU frequency scaling alone can save an average of 17% energy consumption across 12 GPU kernels.

There exist several directions to further improve our prediction model. First, we could further analyze the final SVR model to see how the model performs on different input performance features, which can help us understand the contributions of each individual feature. Second, GPU temperature is also an important factor affecting the runtime power. Third, our current work focuses on frequency scaling only. However to incorporate voltage scaling into our model is also an important issue.

ACKNOWLEDGMENTS

The authors would like to thank all the reviewers for their insightful comments and valuable suggestions. This work is partially supported by research grant HKBU FRG2/14-15/059 and Shenzhen Basic Research Grant SCI-2015-SZTIC-002.

REFERENCES

[1] Y. Abe, H. Sasaki, S. Kato, K. Inoue, M. Edahiro, and M. Peres. 2014. Power and Performance Characterization and Modeling of GPU-Accelerated Systems. In *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. 113–122. DOI: http://dx.doi.org/10.1109/IPDPS.2014.23

[2] Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. 2007. Support vector regression. *Neural Information Processing-Letters and Reviews* 11, 10 (2007), 203–224.

[3] Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 2, 3, Article 27 (May 2011), 27 pages. DOI: http://dx.doi.org/10.1145/1961189.1961199

[4] Vincent Chau, Xiaowen Chu, Hai Liu, and Yiu-Wing Leung. 2017. Energy Efficient Job Scheduling with DVFS for CPU-GPU Heterogeneous Systems. In *Proceedings of e-Energy '17*. Shatin, Hong Kong, 11 pages. DOI: http://dx.doi.org/10.1145/3077839.3077855

[5] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. H. Lee, and K. Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *2009 IEEE International Symposium on Workload Characterization (IISWC)*. 44–54. DOI: http://dx.doi.org/10.1109/IISWC.2009.5306797

[6] X. Chen, Y. Wang, Y. Liang, Y. Xie, and H. Yang. 2014. Run-time technique for simultaneous aging and power optimization in GPGPUs. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. DOI: http://dx.doi.org/10.1145/2593069.2593208

[7] X. Chu, C. Liu, K. Ouyang, L. S. Yung, H. Liu, and Y. W. Leung. 2015. PErasure: A parallel Cauchy Reed-Solomon coding library for GPUs. In *2015 IEEE International Conference on Communications (ICC)*. 436–441. DOI: http://dx.doi.org/10.1109/ICC.2015.7248360

[8] Wu chun Feng and Tom Scoglands. 2016. GREEN500. [Online] https://www.top500.org/green500/lists/2016/11/. (2016).

[9] J. Coplin and M. Burtscher. 2016. Energy, Power, and Performance Characterization of GPGPU Benchmark Programs. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 1190–1199. DOI: http://dx.doi.org/10.1109/IPDPSW.2016.164

[10] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*. 1223–1231.

[11] Sunpyo Hong and Hyesoon Kim. 2010. An Integrated GPU Power and Performance Model. In *Proceedings of the 37th Annual International Symposium on Computer Architecture (ISCA '10)*. ACM, New York, NY, USA, 280–289. DOI: http://dx.doi.org/10.1145/1815961.1815998

[12] V. Kursun and E. G. Friedman. 2006. Supply and Threshold Voltage Scaling Techniques. *Multi-Voltage CMOS Circuit Design* (2006), 45–84. DOI: http://dx.doi.org/10.1002/0470033371.ch3

[13] Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M. Aamodt, and Vijay Janapa Reddi. 2013. GPUWattch: Enabling Energy Optimizations in GPGPUs. In *Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA '13)*. ACM, New York, NY, USA, 487–498. DOI: http://dx.doi.org/10.1145/2485922.2485964

[14] Xiaohan Ma, Mian Dong, Lin Zhong, and Zhigang Deng. 2009. Statistical power consumption analysis and modeling for GPU-based computing. In *Proceeding of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower)*.

[15] Xinxin Mei, Xiaowen Chu, Yiu-Wing Leung, Hai Liu, and Zongpeng Li. 2017. Energy Efficient Real-time Task Scheduling on CPU-GPU Hybrid Clusters. In *Proceedings of IEEE infocom 2017*. Atlanta, GA, USA.

[16] Xinxin Mei, Qiang Wang, and Xiaowen Chu. 2017. A Survey and Measurement Study of GPU DVFS on Energy Conservation. *Accepted by Digital Communication and Network (DCN)* (2017). https://arxiv.org/abs/1610.01784

[17] NVIDIA. 2014. GeForce GTX 980 Whitepaper. [Online] http://www.geforce.com/hardware/notebook-gpus/geforce-gtx-980/specifications. (2014).

[18] NVIDIA. 2016. GPU Computing SDK. [Online] https://developer.nvidia.com/gpu-computing-sdk. (2016).

[19] NVIDIA. 2016. NVIDIA Profiler. [Online] http://docs.nvidia.com/cuda/profiler-users-guide. (2016).

[20] NVIDIA. 2016. NVIDIA System Management Interface (nvidia-smi). [Online] https://developer.nvidia.com/nvidia-system-management-interface. (2016).

[21] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[22] S. Song, C. Su, B. Rountree, and K. W. Cameron. 2013. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*. 673–686. DOI: http://dx.doi.org/10.1109/IPDPS.2013.73

[23] Qiang Wang and Xiaowen Chu. 2017. GPGPU Performance Estimation with Core and Memory Frequency Scaling. *arXiv preprint arXiv:1701.05308* (2017).

[24] Qiang Wang, Pengfei Xu, Yatao Zhang, and Xiaoowen Chu. 2017. EPPMiner: An Extended Benchmark Suite for Energy, Power and Performance Characterization of Heterogeneous Architecture. In *Proceedings of e-Energy '17*. Shatin, Hong Kong, 11 pages. DOI: http://dx.doi.org/10.1145/3077839.3077858

[25] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou. 2015. GPGPU performance and power estimation using machine learning. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*. 564–576. DOI: http://dx.doi.org/10.1109/HPCA.2015.7056063

[26] Kaiyong Zhao and Xiaowen Chu. 2014. G-BLASTN: accelerating nucleotide alignment by graphics processors. *Bioinformatics* 30, 10 (2014), 1384. DOI: http://dx.doi.org/10.1093/bioinformatics/btu047