Improving Energy Efficiency of GPU based General-Purpose Scientific Computing through Automated Selection of Near Optimal Configurations

Xiaohan Ma, Marion Rincon, and Zhigang Deng

Computer Science Department
University of Houston
Houston, TX, 77204, USA
http://www.cs.uh.edu

UH-CS-11-08
October 20th, 2011

Keywords: List of at most five keywords and phrases goes here

Abstract

Modern GPUs have been rapidly and increasingly used as a powerful engine for a variety of general-purpose computing applications due to their enormous parallelism and throughput capabilities. However, GPU power consumption still remains high since more and more transistors are integrated into its chip. Until now, how to increase and optimize energy efficiency (e.g., performance-per-Watt ratio) of GPU-based computing applications is still a largely unsolved challenge. In this paper, we propose a novel framework to improve the energy efficiency of GPU-based General-Purpose Computing (GPGPU) applications. Based on a statistical regression model capable of dynamically estimating the runtime GPU power consumption, our framework can infer and select near-optimal GPGPU programming configurations to improve the energy efficiency of any given GPGPU program. Through preliminary empirical validation of a number of GPGPU benchmarks, we demonstrated that our framework can be robustly used to measurably improve the energy efficiency of various GPGPU programs.

Improving Energy Efficiency of GPU based General-Purpose Scientific Computing through Automated Selection of Near Optimal Configurations

Xiaohan Ma, Mario Rincon, and Zhigang Deng
Department of Computer Science, University of Houston
Emails: xiaohan@cs.uh.edu, mrinconnigro@uh.edu, zdeng@cs.uh.edu
Published in October 20th, 2011

ABSTRACT

Modern GPUs have been rapidly and increasingly used as a powerful engine for a variety of general-purpose computing applications due to their enormous parallelism and throughput capabilities. However, GPU power consumption still remains high since more and more transistors are integrated into its chip. Until now, how to increase and optimize energy efficiency (e.g., performance-per-Watt ratio) of GPU-based computing applications is still a largely unsolved challenge. In this paper, we propose a novel framework to improve the energy efficiency of GPU-based General-Purpose Computing (GPGPU) applications. Based on a statistical regression model capable of dynamically estimating the runtime GPU power consumption, our framework can infer and select near-optimal GPGPU programming configurations to improve the energy efficiency of any given GPGPU program. Through preliminary empirical validation of a number of GPGPU benchmarks, we demonstrated that our framework can be robustly used to measurably improve the energy efficiency of various GPGPU programs.

Categories and Subject Descriptors

I.3.1 [**Computer Graphics**]: Hardware Architectures—*Graphics Processors*

General Terms

Algorithm

1. INTRODUCTION

In recent years, the ever-increasing computing power of modern GPUs comes with the cost of an increasingly high power consumption. As such, the performance-per-Watt ratio (e.g., GFLOPS per Watt) has caught growingly more attentions in the community. Currently, researchers have proposed various idle energy-aware schemes where processor clock frequency and voltage can be dynamically modulated based on predicted workloads [9, 5, 13, 10]. Some of these methods are even used in industry practice (e.g., ATI Powerplay and NVidia PowerMizer). However, even with the above energy-aware schemes, GPU power consumption still remains high. Therefore, under this context, increasing and optimizing the energy efficiency of GPU-based computing applications has become a highly demanded, yet largely unsolved problem.

In this paper we propose a novel framework to improve the energy efficiency of GPU-based General-Purpose Computing (GPGPU) applications. Based on the recorded GPU power consumption, and runtime workload signals of benchmark programs, we first perform in-depth analysis to identify the subset of the most statistically significant GPU workload signals with regard to power consumption, and further build a statistical regression model capable of dynamically and accurately estimating the power consumption of GPGPU applications. Based on the GPU power estimation model, we introduce an automated selection approach to infer and choose near-optimal energy efficient GPGPU programming configurations from a set of exhaustive combinations of program features (called *candidates* in this work). It is noteworthy that in this preliminary work, we picked and studied a representative off-the-shelf GPU, namely, the NVidia GeForce 8800 GT graphics card. However, we believe the methodology and model described in this work can be straightforwardly applied or generalized to other types of commercial GPUs.

The remainder of this paper is organized as follows. Section 2 briefly reviews the recent work that is most related to this work. Section 3 describes how we recorded and processed the power and workload datasets of the target GPU. Section 4 describes how we build a statistical model for power consumption estimation. Section 5 describe how to improve the energy efficiency of various GPGPU computing applications based on the constructed GPU power consumption prediction model. Lastly, discussion and conclusion remarks are presented in Section 6.

2. RELATED WORK

To date, limited work has focused on simulating and modeling GPU power consumption. Sheaffer et al. [13, 14] proposed the first GPU architectural simulation framework (called Qsilver) by extending well-studied CPU power consumption models. However, its primary use is limited to GPU ar-

chitectural design, particularly at ISA and microarchitecture definition design stages. Following the same trend, Ramani et al. [10] proposed a modular power estimation framework "PowerRed" at the architectural level that is primarily for GPU designers. The PowerRed framework is similar to the Qsilver, except the following difference: the Qsilver does not model power consumption of interconnects (e.g., global and local buses on chip) on GPUs, while the PowerRed framework takes it into consideration. As such, the PowerRed framework is supposed to provide a more accurate framework for GPU designers than Qsilver. However, these power models use graphics API interceptors (e.g., Chromium, http://chromium.sourceforge.net/) to trace OpenGL streams and produce GPU workload annotations. Hence, they cannot model power consumptions of most GPGPU applications where no graphics APIs is called.

The relation between workload and power consumption at different stages of the graphics pipeline was previously studied by Mochocki et al. [5]. In their approach, depending on the workload configuration of specific applications, energy bottlenecks may be located at different stages of the graphics pipeline and they can even shift dynamically. Also, Mochocki et al. proposed the use of Dynamic Voltage and Frequency Scaling (DVFS) schemes to decrease energy usage under certain conditions. Recently, Ma et al. [4] proposed a scheme to estimate the power consumption of a runtime commodity GPU based on an empirically-selected subset of GPU workload signals. However, certain critical limitations exist in their approach. First, their approach is originally designed for GPU-based graphics applications. Therefore, its performance on GPGPU computing is less accurate and robust. Second, their used workload signals were empirically chosen and not soundly justified. By contrast, this work employs step-wise statistical analysis to accurately identify the optimal set of used workload signals. Third, their approach does not provide any practical guidelines or procedures to improve or optimize the energy efficiency of various GPU-based computing applications, while this work aims to demonstrate how to exploit an accurate GPU power consumption model to improve the energy efficiency of various GPGPU applications.

Takizawa et al. [15] presented the SPRAT programming framework that dynamically selects an appropriate processor (CPU or GPU) in the runtime so as to improve the overall energy efficiency. Their approach showed that the runtime processor selection for the execution of each kernel with a given data stream is promising for energy-efficient computing on a hybrid (CPU+GPU) computing system. However, in their approach, the power consumption of the selected GPU is assumed to be constant regardless of its runtime workload, which is typically invalid in many real-world GPU computing [4]. Later, Rofouei et al. [12] conducted a broad range of experiments to study the energy efficiency of different computing platform selections (e.g., CPU-only or GPU-only). Their study demonstrated that the GPU-only solution led to a higher energy efficiency if the performance gain is above a certain bound. However, it did not enclose any quantitative method to determine this critical bound parameter (i.e., using GPU or not). Similarly, Collange et al. [1] studied how different GPU computing factors such as memory access operations will impact the power consumption of GPUs and further analyzed the energy cost of various GPGPU operations. Their findings provide valuable implications for energy-efficient GPU computing. However, their approach does not provide any generalized guidelines or procedures to optimize the energy efficiency of various GPU computing applications.

3. DATA ACQUISITION AND PROCESSING

3.1 GPU Power Data Acquisition

We recorded power consumption data of graphics cards (GPUs) using a custom data acquisition setup (Figure 1). As shown in this setup, the target computer runs GPU programs (e.g., benchmarks), the power recording device is a FLUKE Hydra logger 2625A power acquisition system, and the host computer runs its specialized data recording software. Due to the variety of modern GPU architectures, in this work we are not able to comprehensively cover, measure and model the energy aspect of all types of GPUs. Without loss of generality, we pick and study a representative mainstream GPU, namely, the NVidia GeForce 8800 GT graphics card. The methodology and model described in this paper can be easily applied or generalized to other types of GPUs and GPGPU applications.

In our experiment, besides the test graphics card, the target computer's configuration includes AMD Athlon 64x2 3.0GHz Dual-Core Processor, 2GB memory, and Corsair TX750W power supply. The used FLUKE Hydra logger 2625A power acquisition device with one fast-analog-input module is able to perform 10 readings per second, which is competent for this work. We ran benchmark programs that stress test all the stages of modern GPGPU pipeline on the target computer. 20 GPGPU benchmarks enclosed in NVidia GPU Computing SDK were used in this work.

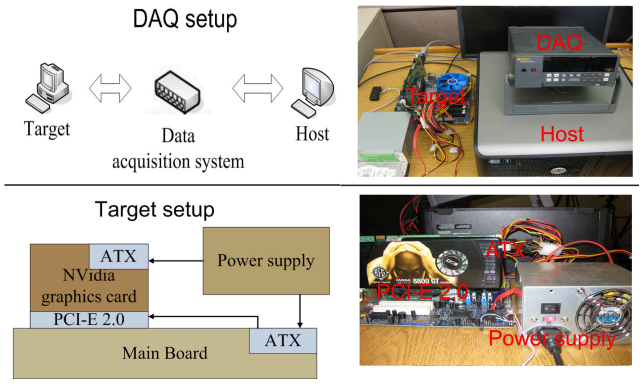

Figure 1: Data acquisition setup used in this work. Note that the graphics card had two power supplies (ATX and PCI-E 2.0). DAQ is a FLUKE Hydra logger 2625A power acquisition system, and the target computer is equipped with the chosen NVidia GeForce 8800 GT graphics card (200 Watt power specification), AMD Athlon 64x2 3.0GHz Dual-Core Processor, 2GB memory, and Corsair TX750W power supply.

3.2 GPU Workload Signal Recording

In the above data acquisition step (Figure 1), we also simultaneously recorded GPU workload signals of the running benchmarks using NVidia CUDA Visual Profiler [6]. The CUDA Profiler that employs an abstracted CUDA programming model is capable of dynamically extracting 21 major GPGPU workload variables such as *gld_incoherent* (the number of non-coalesced global memory loads), *gld_coherent* (the number of coalesced global memory loads), and *branch* (the number of branches taken by threads executing a kernel), *etc.* For the complete description of the 21 extracted GPGPU workload variables, please refer to [6].

The CUDA Profiler sampled the normalized GPGPU workload signals whenever a CUDA kernel function is finished. The number of kernel functions called per second was also dynamically acquired through the CUDA Profiler. Hence, we were able to align the recorded GPGPU workload data with the acquired GPU power consumption data using a linear interpolation based resampling scheme. In this way, the GPU workload signals were resampled (supersampled or downsampled depending on the varying number of kernel functions called per second) to 10 measurements per second that is the same as the acquired power consumption data. Figure 2 shows an example of the recorded and resampled GPU workload signals on NVidia GeForce 8800 GT. We use $\mathbf{X}_t$ to denote the extracted GPGPU workload set at time t.

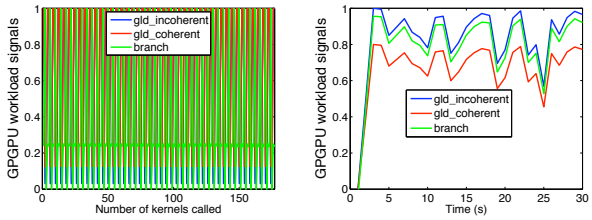

Figure 2: The original recorded GPGPU workload signals (only showing a 2 seconds portion due to the signal density) and the resampled GPGPU workload signals (30 seconds) when one NVidia CUDA SDK sample (N Body Simulation) ran on the GPUs. For the sake of clear illustration, we only plot three selected GPGPU workload variables: *gld_incoherent*, *gld_coherent*, **and** *branch*.

4. GPGPU POWER CONSUMPTION ANALYSIS AND MODELING

Given the recorded GPU power consumption dataset $\mathbf{P} = \{P_{t_1}, P_{t_2}, ..., P_{t_n}\}$ (where t_i denotes a time index), and the time-aligned GPGPU workload dataset $\mathbf{X}^j = \{X^j_{t_1}, X^j_{t_2}, ..., X^j_{t_n}\}(1 < j < N)$, where $X^j_{t_i}$ denotes the j^{th} extracted GPGPU workload signal value at time t_i, our goal is to construct a statistical multivariable function (model) $\hat{P} = F(X^1_t, X^2_t, ..., X^N_t)$ that can accurately predict the GPU power consumption $\hat{P}$, given any GPGPU workload signal variables $(X^1_t, X^2_t, ..., X^N_t)$.

We use a two-step process to construct such a statistical multivariable function (model). First, we perform a statistical analysis to find out which GPGPU workload variables (out of the total 21 variables) - called *workload signature* - are statistically significant to GPU power consumption. Then, we train a Gaussian process regression function to model the inherent association between the GPGPU workload signature and its GPU power consumption. To the end, the trained model is capable of predicting the GPU power consumption based on any inputted GPGPU workload signature.

4.1 Energy-Significance Statistical Analysis

A total of 21 major GPGPU workload variables are acquired (Section 3). However, some workload variables may be dependent to other ones or do not have a strong correlation with the GPU power consumption. In addition, due to the curse of dimensionality, the amount of training data needed to learn a robust statistical model is exponential to the number of input variables. Therefore, we first use a stepwise multiple regression analysis procedure [16] to compute the statistical significance of workload variables in terms of power consumption prediction. In each step, we either add the most statistically significant workload variable or remove the least significant variable based on the computed P-values. In this work, if the P-value of a workload variable is less than 0.05, it will enter into the stepwise selection; otherwise, it will be removed from the selection. As the result of this analysis process, there are 9 (out of 21) GPGPU workload variables used for the following power modeling.

Table 1 shows the nine most statistically significant GPGPU workload variables and their corresponding statistics information. In the remaining writing, we use $(X^1, X^2, ..., X^9)$ to represent the retained, statistically significant nine GPGPU workload variables (called the *GPGPU workload signature*). As shown in Table 1, the GPGPU workload instructions signal has the most significant effect on GPU power consumption estimation. This finding is easy to interpret since computation (executing instructions) is the first thrust of GPUs. The subsequent three GPGPU workload signals, *warp_serial*, *gld_incoherent*, and *gld_coherent*, that are related to GPU memory access operations also have significant impact on GPU power consumption estimation. This finding is, to some extent, consistent with previous GPU energy analysis results [2], that is, moving data across the memory subsystem of processor is one of the most energy-expensive bottlenecks. Grid size and thread block size are also significantly related to the resultant GPU power consumption since these configurations determine the computational burden on GPUs at runtime.

From our analysis we observed that memory store related workload signals (i.e., *gst_incoherent* and *gst_coherent*) do not have significant effects on GPU power consumption, while memory load related workload signals (e.g., *gld_incoherent* and *gld_coherent*) have the significant effects, as discussed in the above paragraph. One possible explanation is that memory store operations benefit from GPU cache more than memory load operations. Store operations can always write data to cache first to avoid frequent and heavy data-moving across the GPU memory subsystem. Since on-chip data movements do not go across the entire GPU memory hierarchy, GPGPU workload signals related to local memory and Translation Lookaside Buffer (TLB) operations do not have significant effects on GPU power consumption as well. We also notice that branch divergence does not have a significant effect on GPU power consumption, since the workload

	GeForce 8800 GT	
Variable	**P-value**	**Std. Error**
instructions	0	1.6463e-007
warp_serial	6.5995e-018	1.5475e-005
gld_incoherent	2.5360e-017	0.4305
gld_coherent	1.1265e-017	9.7682e-006
gridSizeX	9.9186e-016	0.0117
gridSizeY	5.6322e-016	1.4475e-006
blockSizeX	2.2204e-016	0.1273
blockSizeY	2.4748e-004	1.4840e-007
blockSizeZ	0.0077	1.1951e-005

Table 1: Stepwise multiple regression results in terms of GPU power consumption estimation for the NVidia GeForce 8800 GT. It only shows the 9 retained (most significant) GPU workload variables. Refer to [6] for the details of the extracted GPU workload signals.

of the instructions executed on cores per time step remains the same regardless branches [7].

4.2 GPU Power Consumption Modeling

Inspired by the previous GPU statistical power modeling scheme proposed by Ma et al. [4] that uses a support vector regression model to learn the association between GPU workload signals and power consumption successfully, in this work, we employ the Gaussian process regression (GPR) model for such statistical power modeling task. As described above, a total of 9 major GPGPU workload variables are selected for our GPGPU power estimation model training. We first split the processed dataset $< \mathbf{X}_{i=1}^{9}, \mathbf{P} >$ into a training subset (80% of the data) and a cross-validation subset (the remaining 20%). Then, we use the training subset to learn the GPR model, $P_t = F(X_t^1, X_t^2, ..., X_t^9)$, that is capable of predicting the GPU power consumption, P_t, given any given GPGPU workload signature, $(X_t^1, X_t^2, ..., X_t^9)$.

we choose to rely on the Gaussian Process Regression (GPR) model because it offers compelling advantages over other regression models [18]. First, the GPR model is non-parametric, so it does not require a considerable effort for tuning parameters to achieve good training results. Second, the GPR model is context-dependent, and hence it is able to automatically handle GPGPU workload signals with different properties (e.g., computation-intensive or memory-intensive workloads) in different ways.

Mathematically, a GPR model is characterized by its hyper-parameter vector θ that includes a characteristic length-scale parameter θ_1 and a signal magnitude parameter θ_2. Hence, the crucial step of training a GPR model is to learn the hyper-parameter vector θ properly. In this work, we learn θ by optimizing the following marginal log-likelihood function (Equation 1).

$$\begin{aligned} L_{GP} &= -\log\ P(P|X,\theta) \\ &= \frac{1}{2}\log|K+\sigma^2\ I| + \frac{1}{2}P^T(K+\sigma^2\ I)^{-1}P \\ &\quad + \frac{N}{2}\log\ 2\pi \end{aligned} \tag{1}$$

Here L_{GP} is the negative log-posterior of the model, σ^2 is the variance of noise (0.012 in this work), and K is a used kernel function (we use the ARD covariance function [17] as described in Equation 2).

$$K_{i,j} = k(X_i, X_j) = \theta_2 \exp(-\frac{1}{2\theta_1^2}||X_i - X_j||^2) \tag{2}$$

Then, Rasmussen's minimization algorithm [11] is chosen to optimize L_{GP} due to its efficiency. The maximum number of iterations is experimentally set to 1024. After θ is optimally solved, the trained GPR model yields a likelihood function for any predicted output. Concretely, for any new GPGPU workload signals, x, we can obtain a distribution of its predicted GPU power consumption, p. In addition, we can evaluate the negative log probability of the predicted output. This log-likelihood function is shown in Equation 3.

$$\begin{aligned} L_S &= -\log\ P(p|x,\theta) \\ &= \frac{1}{2}\log(2\pi(V(x)+\sigma^2)) + \frac{||y-U(x)||^2}{2(V(x)+\sigma^2)} \end{aligned} \tag{3}$$

$$\begin{aligned} U(x) &= \kappa(x)^P(K+\sigma^2 I)^{-1}P \\ V(x) &= k(x,x) - \kappa(x)^P(K+\sigma^2 I)^{-1}\kappa(x)^T \end{aligned}$$

Here $\kappa(x)$ is a vector in which the i^{th} entry is $k(x, X_i)$, function U returns the mean of the posterior distribution of the learned model given new input x, and function V returns the variance of the learned posterior distribution. In all, we just need to minimize L_S to obtain the predicted output p.

Cross-Validation: We also compared the cross-validation performance of the chosen GPR model with other two widely-used regression models: simple stepwise least square based linear regression (SLR) and Support Vector Machine regression (SVR) [4]. Comparison results are shown in Figure 3. In this work, the Percentage of Mean Prediction Error (PMPE) is used as a quantitative measure for the cross-validation comparisons. We computed the PMPE values of the entire cross-validation subset (the retained 20% dataset, 1428 seconds). The resultant PPE values of GPR, SLR, and SVR are 9.2%, 26.1%, and 12.5% on NVidia GeForce 8800 GT, respectively. As such, the GPR model chosen in this work measurably outperformed the other two models (SLR and SVR) on the retained cross-validation dataset.

5. IMPROVING ENERGY EFFICIENCY FOR GPGPU COMPUTING

In this section, we describe how to improve the energy efficiency of various GPGPU computing applications based on the above GPGPU power consumption prediction model. The pipeline of our proposed GPGPU energy efficiency optimization approach can be briefly summarized as follows: (1) First, we select certain GPGPU programming configurations (e.g., thread number per block, and loop unrolling level) of the original GPGPU program to generate a number

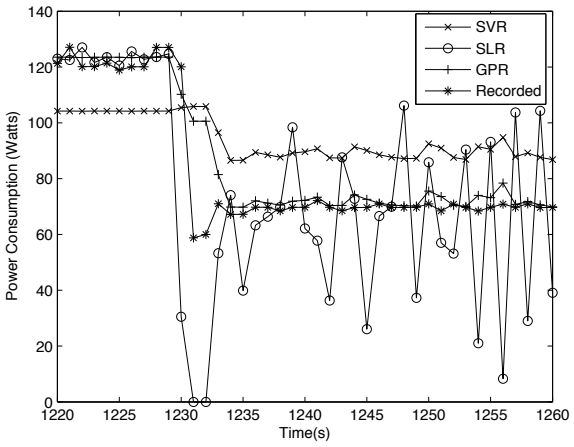

Figure 3: A part (80 seconds) of cross-validation comparison results when a NVidia CUDA SDK sample "OceanFFT" ran on NVidia GeForce 8800 GT.

of different combinations by varying their values. (2) Then, we estimate the GPU power consumption of each combination using the above constructed GPGPU power estimation model. (3) Lastly, based on the estimated GPGPU power consumptions, we can calculate the energy-efficiencies of all the combinations and choose the most energy-efficient one to improve and optimize the energy efficiency of the original GPGPU program.

5.1 GPGPU Energy-Conscious Programming Configurations

Before we select certain GPGPU programming configurations to generate a number of different candidates, we need to study the energy-consciousness of available GPGPU programming factors/configurations [8]. We use a logistic regression analysis technique [3] to compute the statistical significances of energy efficiency of these GPGPU programming factors. Concretely, by using the same benchmark set as used in Section 3, we perform the analysis and record the change of the deviances of Chi-Square test statistics if one programming factor is not considered into logistic regression. As shown in Table 2, based on the logistic regression results, we are able to rank the significance of energy efficiency of the top-six GPGPU programming factors as follows: "input data size of kernel function" > "number of threads per tile" > "unroll level" > "24-bit ALU instruction" > "shared memory access" > "number of tiles per grid". As such, we select these programming factors (except the user-specified "input data size") as the energy-conscious programming configurations of interest to generate candidates in this work.

5.2 Selected Benchmarks and Results

To validate our GPGPU energy efficiency optimization approach, we selected five GPGPU benchmarks from the NVidia GPU Computing SDK (listed in Table 3) as the test programs. The five benchmarks were not used in the previous power consumption model training (Section 4) and their graphical outputs were disabled in order to isolate the GPGPU effect. Then, (1) we selected the energy-conscious

Variable	Deviance χ^2	d.f.	5% χ^2
input data size	13.1	6	12.59
num of threads per tile	9.0	3	7.82
unroll level	8.6	3	7.82
24-bit ALU	4.5	1	3.84
shared memory access	3.8	1	3.84
num of tiles per grid	2.2	1	3.84

Table 2: Logistic regression results of different CUDA-based GPGPU programming factors in terms of their significances of energy efficiency (GFLOPS per Watt)

GPGPU programming configurations (Section 5.1) for energy efficiency optimization. We varied the above programming configurations to generate a number of GPGPU programming configuration combinations/candidates for each benchmark. After that, (2) we estimated the energy-efficiencies of all the combinations (shown in Fig. 4). We used the above constructed GPGPU power estimation model, as well as the recorded program performance data, to predict the energy efficiency of each of the generated GPGPU configuration candidates. Finally, (3) we determined the optimal GPGPU configuration combination in terms of energy efficiency. For example, for the "fastWalshTransform" benchmark, as depicted in Figure 4, the configuration combination that is predicted to achieve the maximal energy efficiency (0.1531 GFLOPS/Watt) is candidate #40 (i.e., thread per tile = 512, unroll level = 8, ALU instruction = 24-bit, memory access = S (shared)). Analogously, Figure 4 shows the predicted information of the other four benchmarks. The optimal GPGPU configuration combinations that are predicted to achieve the maximal GPGPU energy efficiency are: "particles" candidate #24 (0.0266 GFLOPS/Watt, thread per tile = 512, unroll level = 4, memory access = S (shared)), "matrixMul" candidate #22 (0.1677 GFLOPS/Watt, thread per tile = 512, unroll level = 1, memory access = S (shared)), "scalarProd" candidate #9 (0.0610 GFLOPS/Watt, thread per tile = 128, unroll level = 16), and "MersenneTwister" candidate #14 (0.1705 GFLOPS/Watt, thread per tile = 128, unroll level = 2, memory access = non-shared memory).

5.3 Validation

In order to quantify the accuracy and effectiveness of our GPGPU energy efficiency optimization approach, we also recorded the power consumption data (as the ground truth) for all the GPGPU configuration combinations considered in this study. The ground truth GPGPU energy-efficiencies (GFLOPS/Watt) of all the candidates are also depicted in Fig. 4. Table 3 shows the energy efficiencies for the selected (by our approach), average, and best (i.e. ground-truth) candidates running on the chosen NVidia 8800 GT graphics card.

From Fig. 4, we can draw the following two conclusions. First, *the optimized GPGPU configuration combination, chosen by our approach, is able to substantially improve the energy-efficiencies of the chosen GPGPU applications*. For example, for the "fastWalshTransform" benchmark, as shown in Figure 4, the optimized configuration combination (candidate #40) chosen by our approach, generates a measur-

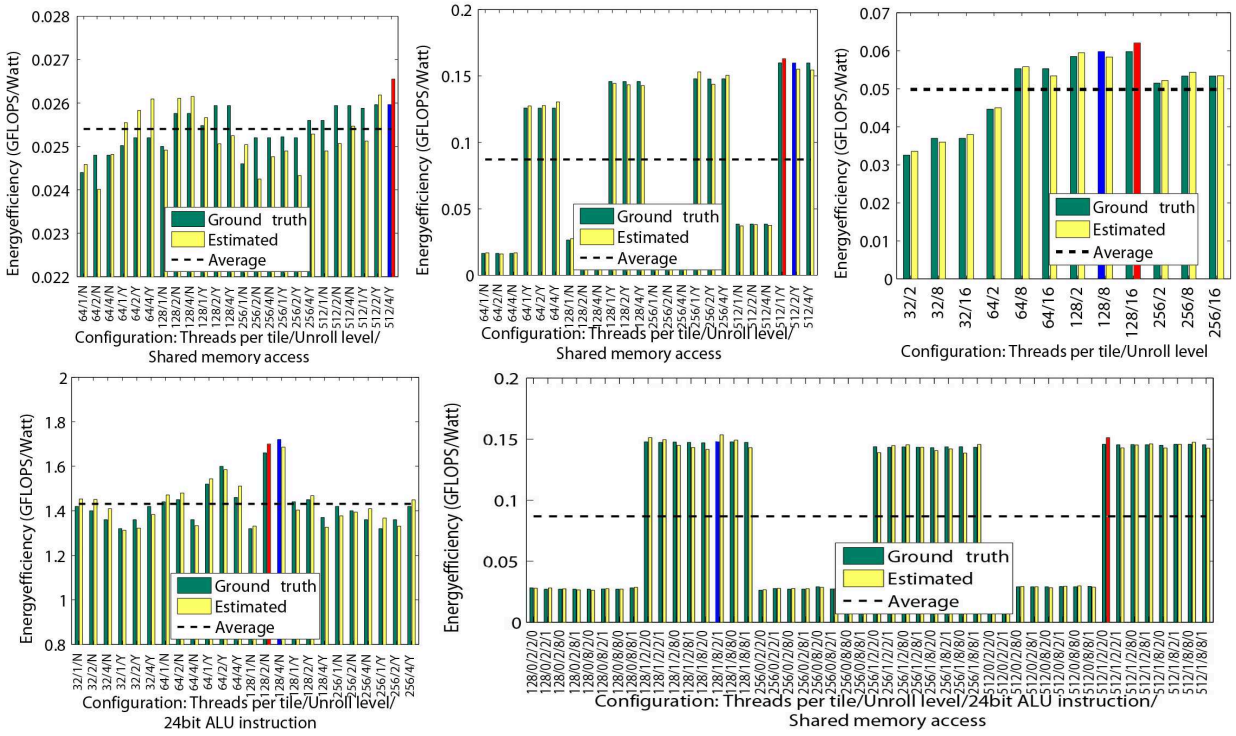

Figure 4: The estimated and ground-truth energy-efficiencies for all the generated candidates of the selected benchmarks (from top-left to bottom-right): "particles", "matrixMul", "scalarProd", "MersenneTwister", and "fastWalshTransform". The blue bars indicate the candidates of maximal ground truth energy-efficiencies while the red bars indicate the candidates of maximal estimated energy-efficiencies.

Benchmark	Selected	Average	Best
particles	0.0260	0.02547	0.0260
matrixMul	0.151	0.0909	0.159
scalarProd	0.0551	0.0501	0.0554
MersenneTwister	0.1615	0.1420	0.1729
fastWalshTransform	0.1525	0.0919	0.1578

Table 3: Ground truth energy-efficiencies (GFLOPS/Watt) for the selected, average, best candidates of the benchmarks

ably higher energy efficiency (0.1525 GFLOPS/Watt) than the average case (0.0919 GFLOPS/Watt). The percentage of GPGPU energy efficiency improvement with respect to the average efficiency is 65.607%. The energy efficiency of the selected candidate by our approach is considerably close to the best possible energy efficiency among all candidates - 0.1578 GFLOPS/Watt for candidate #14. Take the "MersenneTwister" benchmark for another example (the bottom-left panel of Fig. 4), the selected candidate #14 by our approach produces a higher energy efficiency (0.1615 GFLOPS/Watt) than the average case (0.1420 GFLOPS/Watt). The percentage of GPGPU energy efficiency improvement is 13.735% with respect to the average energy efficiency. The energy efficiency of the selected candidate is a 93.40% of the best possible case among all candidates. The predicted energy efficiency for the selected candidate was 0.1705 GFLOPS/Watt, which is higher than the ground-truth energy efficiency. The deviation is due to the difficulty of our GPU power estimation model since it does not take GPU factors such as varying I/O activities and GPU fan speeds into explicit consideration. Nonetheless, our model is capable of consistently predicting the overall energy usage tendency and thus selecting the near-optimal candidate (i.e. significantly above the average) in most cases.

Second, *our approach can robustly predict the energy efficiencies of different GPGPU configuration combinations, in particular, their general trend; as such, it is effective to improve and optimize the energy efficiency of various GPGPU computing applications*. Fig. 4 show the comparisons between all the predicted and recorded (ground-truth) energy-efficiencies for the test benchmarks. We further computed the average percentage of prediction errors: the average error (percentage) of "particles" is 8.19%, it is 2.52% for "matrixMul", 1.43% for "scalarProd", 4.78% for "MersenneTwister", and 4.60% for "fastWalshTransform". As such, we can observe that though our GPGPU power estimation model is not able to perfectly predict the energy-efficiencies, the errors are nonetheless within a manageable range. In particular, the predicted GPGPU energy-efficiencies are approximately consistent with the ground-truth in terms of their general trend, e.g., the energy efficiency variations. Therefore, our approach is measurably effective to optimize the energy efficiency of a variety of GPGPU computing applications.

6. DISCUSSION AND CONCLUSIONS

In this paper, we propose a novel framework to improve the energy efficiency of GPGPU computing applications. Its first element is to statistically analyze and model the GPGPU power consumption by exploiting the inherent association among GPU power consumption, runtime performance, and dynamic GPGPU workload. Our trained statistical model is capable of robustly and accurately predicting power consumption for the chosen two commodity NVidia GPUs. Based on this GPU power model, we present an automated selection approach of choosing the near-optimal energy efficient GPGPU programming configuration from a set of exhaustive combinations of program features (which we call *candidates* in this work) on the power signals predicted by our power model. Furthermore, through the energy efficiency improvement of a number of selected GPGPU benchmarks, we demonstrated that our proposed framework can be used to measurably improve the energy efficiency of various GPGPU applications.

Certain limitations still remain in the current work. (1) Our current framework cannot be directly applied to non-NVidia GPUs, e.g., ATI Radeon series GPUs. The main reason is that most of ATI GPU performance counters or workload recorders do not expose sufficient workload signal information to external developers. Hence, it is difficult to use their limited performance counters to train a well-behaviored statistical model for GPU power estimation. (2) Our current framework employs the CUDA profiler to extract GPGPU workloads, which means all our training and test GPGPU benchmarks need to be implemented with CUDA. In the future, we will conduct more studies on other GPGPU programming models such as ATI Stream and OpenCL and generalize our framework to other GPGPU platforms.

7. REFERENCES

[1] S. Collange, D. Defour, and A. Tisserand. Power consumption of gpus from a software perspective. In *ICCS '09: Proc. of the 9th International Conference on Computational Science*, pages 914–923, 2009.

[2] B. Dally. Low-power architecture. http://cva.stanford.edu/people/dally/ISSCC2005.pdf, 2005.

[3] J. M. Hilbe. *Logistic Regression Models*. Chapman & Hall/CRC Press, 2009.

[4] X. Ma, M. Dong, L. Zhong, and Z. Deng. Statistical power consumption analysis and modeling for gpu-based computing. In *Proc. of SOSP Workshop on Power-Aware Computing and Systems (HotPower)'09*, 2009.

[5] B. Mochocki, K. Lahiri, and S. Cadambi. Power analysis of mobile 3D graphics. In *DATE'06: IEEE Conf. on Design, Automation, and Test in Europe (DATE)*, pages 502–507, 2006.

[6] NVIDIA . NVidia Visual CUDA Profiler. http://developer.nvidia.com/object/cuda.html, 2010.

[7] NVIDIA. GeForce 8800 GPU architecture overview. Technical report, NVIDIA, Nov 2006.

[8] NVIDIA. *NVidia GPU Programming Guide*, 2008.

[9] T. Pering, T. Burd, and R. Brodersen. The simulation and evaluation of dynamic voltage scaling algorithms. In *ISLPED'98: Proc. of international symposium on Low power electronics and design*, pages 76–81, 1998.

[10] K. Ramani, A. Ibrahim, and D. Shimizu. PowerRed: A flexible power modeling framework for power efficiency exploration in GPUs. In *Worskshop on General Purpose Processing on Graphics Processing Units (GPGPU)*, 2007.

[11] C. E. Rasmussen. Minimize function, http://www.kyb.tuebingen.mpg.de/bs/people/carl/, 2006.

[12] M. Rofouei, T. Stathopoulos, S. Ryffel, W. Kaiser, and M. Sarrafzadeh. Energy-aware high performance computing with graphic processing units. In *HotPower'08: Proc. of ACM SOSP Workshop on Power Aware Computing and Systems (HotPower) 2008*, Dec 2008.

[13] J. W. Sheaffer, D. Luebke, and K. Skadron. A flexible simulation framework for graphics architectures. In *HWWS '04: Proc. of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 85–94, 2004.

[14] J. W. Sheaffer, K. Skadron, and D. Luebke. Studying thermal management for graphics-processor architectures. In *Proc. of IEEE International Symposium on Performance Analysis of Systems and Software*, pages 54–65, 2005.

[15] H. Takizawa, K. Sato, and H. Kobayashi. SPRAT: Runtime processor selection for energy-aware computing. In *2008 IEEE International Conference on Cluster Computing*, pages 386–393, Cct 2008.

[16] H. Valiaho and T. Pekkonen. *A procedure for stepwise regression analysis*. Akademie-Verlag, 1976.

[17] C. K. I. Williams and C. E. Rasmussen. Gaussian processes for regression. In *Advances in Neural Information Processing Systems 8*, pages 514–520. MIT Press, 1996.

[18] T. S. P. Withers and C. Bailer-Jones. Accelerated learning using gaussian process models to predict static recrystallization in an al-mg alloy. *Modelling and Simulation in Materials Science and Engineering*, 8(5):1–14, 2000.