Statistical Power Consumption Analysis and Modeling for GPU-based Computing

Xiaohan Ma
University of Houston

Mian Dong
Rice University

Lin Zhong
Rice University

Zhigang Deng
University of Houston

ABSTRACT

In recent years, more and more transistors have been integrated within the GPU, which has resulted in steadily rising power consumption requirements. In this paper we present a preliminary scheme to statistically analyze and model the power consumption of a mainstream GPU (NVidia GeForce 8800gt) by exploiting the innate coupling among power consumption characteristics, runtime performance, and dynamic workloads. Based on the recorded runtime GPU workload signals, our trained statistical model is capable of robustly and accurately predicting power consumption of the target GPU. To the best of our knowledge, this study is the first work that applies statistical analysis to model the power consumption of a mainstream GPU, and its results provide useful insights for future endeavors of building energy-efficient GPU computing paradigms.

1. INTRODUCTION

Modern GPUs are integrating more and more transistors on their chips (*e.g.*, NVidia GeForce GTX 280 contains 1.4 billion transistors), and thus suffer from increasingly higher power consumption requirements. The direct consequences of its higher power consumptions are growing dissipation of heat, more complex cooling solutions, and noisier fans. From the perspective of GPU programmers, how to develop energy-efficient GPU codes (i.e., performance per watt ratio) becomes a challenging and open ended issue. Before this research issue can be fully resolved, analyzing and modeling the power consumption of runtime GPUs is clearly a priority.

Real-time power modeling on CPU typically uses detailed analytical power models [3, 2] or high-level, black-box models [1, 5] based on CPU performance counters. Our work employs a similar high-level methodology to model GPU power consumption. Earlier studies related to GPU power modeling have different focuses. Sheaffer *et al.* [7, 8] proposed a power consumption model based on a hypothetical GPU architectural simulation framework by extending existing well-studied CPU power models. Ramani *et al.* [4] proposed a modular power estimation framework at an architectural level primarily for GPU designers. Takizawa *et al.* [9] proposed the SPRAT programming framework that dynamically selects an appropriate processor (CPU or GPUs) so as to improve the overall energy efficiency; however, it ignores GPU runtime workloads.

In this work, we present a novel scheme for analyzing and modeling the power consumption of GPU. Based on the recorded power consumption, runtime workload signals, and performance data, we build a statistical regression model capable of dynamically estimating the power consumption of a runtime GPU based on a selected subset of GPU workload signals. Our statistical model bridges the dynamic workloads of runtime GPUs and their estimated power consumptions. To the best of our knowledge, our work is the first-of-a-kind effort that applies statistical analysis and modeling to the power consumption of a commercial mainstream GPU (our work uses the NVidia GeForce 8800gt graphics card), and its results provide useful insights for future endeavors to construct energy-aware GPU computing paradigms.

2. DATA ACQUISITION

Power Consumption Data Acquisition: We recorded power consumption data for the chosen graphics card (we define its power as GPU power consumption in this paper) using a customized data acquisition setup. The test computer ran programs designed to test the GPU (*e.g.*, benchmarks), the host computer ran specialized data recording software, and a FLUKE 2680A power acquisition system was used to record power readings. In our experiment, the test computer was equipped with a NVidia GeForce 8800gt graphics card with a 200 Watt power specification, AMD Athlon 64x2 3.0GHz Dual-Core Processor, 2GB memory, and a Corsair TX 750W power supply. The FLUKE 2680A power acquisition device with two fast-analog-input modules is able to perform 1000 readings per second.

The chosen graphics card has two separate power planes. One is supplied by the 12V power plane of the PCI-E bus and the other is directly connected to a 12V output of the ATX power supply. Therefore, we sought to measure the power supplied by each power plane separately and use the sum of the two as the total power consumption of the graphics card. To obtain the power supplied by the PCI-E bus, we choose to use a riser card in-between the graphics card and the PCI-E slot on the motherboard. Then, we placed a 0.1 ohm resistor at the power pin of the riser card to enable current sensing. We calculated the power supplied by the PCI-E bus by multiplying the current and voltage at the power pin of the riser card. To get the power supplied by the ATX power supply, similarly, we placed a 0.1 ohm resistor in the power cable that connects the graphics card and the ATX power supply. Again, we calculated the power supplied by the ATX power supply by multiplying the current and voltage of the power cable.

We ran several benchmark programs to stress test different stages of the GPU pipeline on the test computer and also measured GPU power consumption during runtime. We chose to use four different GPU benchmarks: *OpenGL Geometry Benchmark 1.0*, *Furmark*, *Jorik*, and *Parboil*. The OpenGL Geometry Benchmark 1.0 fully exploits the traditional graphics pipeline, especially its geometry transform & lighting units. The Furmark benchmark mainly focuses on the GPU shader units. The Jorik and the Parboil benchmarks were selected to exhaustively test the general-purpose computation capabilities of GPUs including parallel sorting and dynamic PDE solvers. In our initial data acquisition stage, we ran the OpenGL Geometry Benchmark 1.0 for 6730 seconds, the Furmark for 240 seconds, the Jorik for 185 seconds, and the Parboil for 384 seconds.

To increase the efficiency and stability of follow-up analysis and modeling algorithms, we processed the original power consumption data (1000 frames per second) through an averaging and downsampling operation. Essentially, the operation generates a new frame in the processed data by averaging corresponding M (a preset averaging window size) frames in the original data. M=25 was experimentally determined in this work.

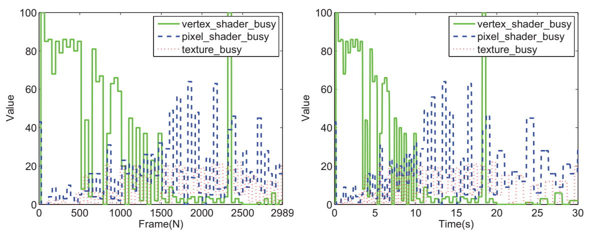

Figure 1: The original recorded GPU workload signals while the GPU ran the OpenGL Geometry Benchmark 1.0 (left), and the corresponding aligned/resampled GPU workload signals (right). For the sake of clear illustration, here we only plot three GPU workload variables: vertex_shader_busy, pixel_shader_busy, and texture_busy.

GPU Workload Signal Recording: In the above data acquisition step, we also recorded the workload signals of the runtime GPU using the *NVidia PerfKit performance analysis tool* simultaneously. The NVidia PerfKit tool which employs an abstract GPU programming model is capable of dynamically extracting 39 GPU workload variables. Inspired by the previous study on GPU workload characterization [6], in our work we choose to use 5 major variables from the recorded 39 GPU workload signals: *vertex_shader_busy* (the percentage of time when the vertex shader is busy), *pixel_shader_busy* (the percentage of time when the pixel shader is busy), *texture_busy* (the percentage of time when the texture unit is busy), *goem_busy* (the percentage of time when the geometry shader is busy), and *rop_busy* (the percentage of time when the ROP unit is active). These 5 signals (variables) represent the runtime utilizations of major pipeline stages on the GPU, which together profile GPU workloads in a compact and robust manner. For a complete description of the above 5 chosen GPU workload signals, refer to the NVidia PerfKit user guide.

The NVidia PerfKit tool records the GPU workload signals whenever a front frame is generated. The number of generated frames per second is dynamically acquired through the FPS counter. Hence, we aligned the recorded GPU workload data with the above processed 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 FPS) to 40 frames/second, making their

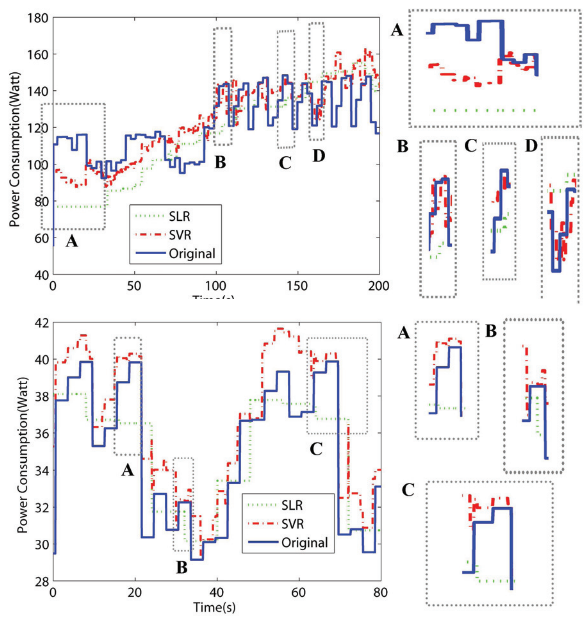

Figure 2: (Top panel) A part (200 seconds) of the cross-validation comparison results when the GPU ran graphics programs (OpenGL Geometry Benchmark 1.0). (Bottom panel) A part (80 seconds) of the cross-validation comparison results when the GPU ran the GPGPU Jorik benchmark.

sampling rate the same as the processed power consumption data. Figure 1 shows an example of the recorded and resampled GPU workload signals.

3. STATISTICAL GPU POWER MODEL

Now we describe how to construct a statistical model for GPU power consumption estimation, based on the above power consumption and GPU workload dataset. Assuming the processed power consumption data is $Y = \{Y_{t_1}, Y_{t_2}, ..., Y_{t_n}\}$ (t_i denotes a time index), and the aligned GPU workload data is $X^j = \{X^j_{t_1}, X^j_{t_2}, ..., X^j_{t_n}\}$ ($1 \leq j \leq N$, X^j represents the j^{th} GPU workload variable), we want to construct a statistical multivariable function (model) $Y_t = F(X^1_t, X^2_t, ..., X^N_t)$ that can robustly and accurately predict the GPU power consumption, Y_t, given any GPU workload variables $(X^1_t, X^2_t, ..., X^N_t)$.

As mentioned in Section 2, a total of 5 major GPU workload variables were selected in this study. Therefore, in the remaining sections, we use (X^1, X^2, ... , X^5) to represent the five chosen GPU workload variables. We first split the above dataset $< \{X^5_{i=1}\}, Y >$ into a training subset (80%, 6031 seconds) and a test or cross-validation subset (the remaining 20%, 1508 seconds). Then, we use the training subset to learn a Support Vector Regression (SVR) model. Mathematically, the basic idea of SVR is to predict result y from input x by optimizing the Eq. 1 where w and b are used to describe the SVR regression model. The LIBSVM is used for SVR implementation. We compared the cross-validation results of the chosen SVR model with a simple least square based linear regression (SLR) model. Figure 2 shows the cross-validation comparison results between SLR and SVR. Here the sum of square errors is used as a metric to measure the prediction quality (*i.e.*, the discrepancy between the predicted and the original power consumption data). The top panel of Figure 2 shows a part (200 seconds) of the cross validation comparison results when the GPU ran graphics programs (OpenGL Geometry Benchmark 1.0). In this case, the sum-square-error metrics of SLR and SVR are 656.83 and 589.78, respectively. Its bottom panel shows a part (80 seconds) of the cross-validation comparison results when the GPU ran the GPGPU Jorik benchmark, and the sum-square-error metrics of SLR and SVR are 44.523 and 39.427, respectively. As clearly shown in this figure, regardless of whether we used graphics computing or GPGPU applications, the chosen Support Vector Regression (SVR) model measurably outperformed the traditional SLR on the retained cross-validation dataset.

$$\begin{aligned} &minimize\ \frac{1}{2}\ ||w||^2 \\ &subject\ to\ ||y - (w \cdot x - b)|| < \epsilon \end{aligned} \qquad (1)$$

4. EVALUATION AND VALIDATION

In this section, we further look into the generality and robustness of our statistical power consumption model, such as, *what are the accuracy and robustness of our statistical model if the GPU runs non-benchmark programs?* The chosen eight test programs can be divided into two categories: graphics programs and GPGPU computing applications. An open-source, first-person shooter game - *Nexuiz*, and three widely-used program samples enclosed in the NVidia OpenGL SDK ("XMas Tree", "HDR", and "Dual Depth Peeling") are used to evaluate the accuracy of our model for graphics programs. Partial graphics settings of the above three SDK program samples are: 1046x768 as the resolution of rendering scene, and 2x as Multisample anti-aliasing (MSAA) level. The Nexuiz game ran under a full-screen resolution with disabled MSAA. In our experiment, each of the above four programs ran for 100 seconds.

The selected GPGPU programs include a GPGPU-

	OpenGL Geom. Benchmark	Jorik	X Mas Tree	HDR	Dual Depth Peeling	Nexuiz	N-body Simulation	Option Pricing	Fast Walsh Transform	"GNN"
SVR	3.3%	13.0%	9.0%	29.4%	6.5%	27.7%	3.5%	1.7%	12.0%	20.8%
SLR	5.1%	11.1%	9.8%	39.0%	6.4%	29.1%	8.9%	2.0%	18.9%	33.7%

Table 1: Summary of power prediction errors as a percentage of the mean GPU power consumption in our evaluation experiment.

based neural network application "GNN" and three program samples enclosed in the NVidia CUDA SDK ("N-body simulation", "Option Pricing", and "Fast Walsh Transform"). The program settings for the CUDA SDK programs are detailed as follows. The body number of "N-body simulation" is 16K. The configurations of the "Option Pricing" include: the maximum price option number is $4,000$K, the risk-free rate of return is 0.02, the volatility of the underlying stock is 0.30, and the iteration number is 512. The configurations of the "Fast Walsh Transform" sample include: 128 as kernel size, and $8,388,608$ as data size. Finally, for the "GNN" program we tested the recognition of 28 handwritten digits. In our experiment, the "N-body simulation" sample ran for 20 seconds and other three programs stopped automatically once they generated outputs.

4.1 Results and Discussion

We recorded the energy consumption (Watt) and workload signals of the chosen GPU, when each of the above eight programs was running on the test computer. The recorded data was used as the ground truth in our evaluation. We used our GPU power statistical model (described in Section 3) to predict the power consumption of all the chosen eight programs based on the recorded GPU workload signals. Figures 3 and 4 show comparisons between the ground truth and the predicted power consumption data by our model. We also computed the mean square errors as an objective metric to measure prediction accuracy and obtained the following results: "XMas Tree" (2.7), "HDR" (10.3), "Dual Depth Peeling" (2.5), Nexuiz (13.4), "N-body simulation" (8.4), "Option Pricing" (4.4), "Fast Walsh Transform" (15.0) and "GNN" (13.9).

As shown in Figures 3 and 4, for most of the time intervals the power consumption prediction errors by our model are small and consistent across frames. However, for certain time intervals the predicted power consumption errors are measurably larger. For example, the mean square error of the "XMas Tree" sample (2.7) is significantly smaller than that of the "HDR" sample (10.3). Also, the power consumption curve of the "XMas Tree" sample has more regular shapes than the "HDR" sample in which peaks frequently occurred during a short time period. It is difficult to model these kinds of variations using our statistical model. One plausible explanation is that our current statistical power consumption estimation model completely depends on the recorded workload signals of the runtime GPU (as input), but in certain cases, these GPU workload signals may fail to indicate the power consumption of the underlying GPU. For example, at the 5^{th} second of the "HDR" sample, the GPU workload keeps a stable level while the ground-truth GPU power consumption drops rapidly.

In addition, our approach cannot accurately model power consumption peaks, *e.g.*, some parts of "HDR", "Nexuiz" and "Option Pricing" in Figures 3 and 4. In these cases, the GPU power consumption rapidly reached a high level, while our model failed to keep pace. The resulting prediction errors are non-trivial (*e.g.*, >20 Watt). One possible explanation is that certain parts of those GPU power peaks might not be completely due to the execution of the GPU graphics pipeline and that the power contribution of some other factors such as bus communication or memory access varied dramatically. Thus, our current model can not sufficiently model these power consumption peaks. In the future, workloads on other components such as memory and I/O activities should be recorded and incorporated to enhance the coverage of the utilization units on GPU. Table 1 summarizes the power prediction errors as a percentage of the mean GPU power consumption in the evaluation experiment.

Another limitation of our model (actually a limitation of statistical models in general) is that it is hard to predict how much training data is sufficient and will be needed in advance. Here is a concrete example: as shown in the Nexuiz sample in Figure 3, our model has difficulty in predicting a case that involves extremely low power consumptions (*e.g.*, <20 Watt in the ground-truth) due to the lack of similar extreme cases in our training dataset.

5. CONCLUSIONS

We present a novel statistical power analysis and modeling scheme for GPU-based computing, by exploiting the intrinsic coupling among power consumption, runtime performance, and dynamic work-

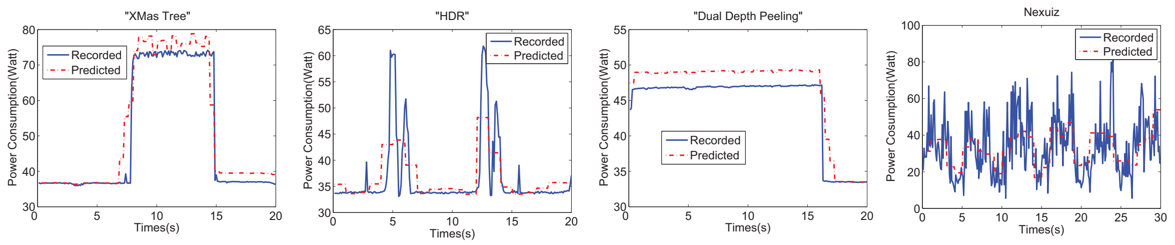

Figure 3: Comparisons between the ground truth (blue) and the predicted GPU power consumption data (red) for the chosen four graphics programs. The four panels (from left to right) show "XMas Tree", "HDR", "Dual Depth Peeling", and the Nexuiz game, respectively.

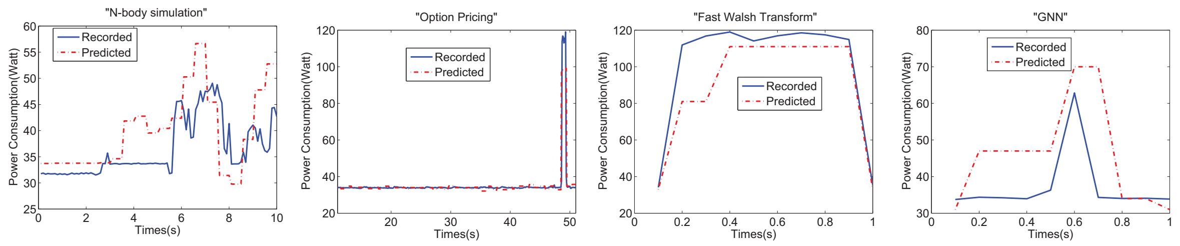

Figure 4: Comparisons between the ground truth (blue) and the predicted GPU power consumption data (red) for the four chosen GPGPU programs. The four panels (from left to right) show "N-body simulation", "Option Pricing", "Fast Walsh Transform", and "GNN", respectively.

loads of GPUs. The core piece of our scheme is a statistical model for dynamic power consumption estimation during the GPU runtime. We showed that our statistical model is able to accurately and robustly estimate the power consumption during the GPU runtime, especially for graphics applications. Compared with CPU, GPU has a relatively simpler cache hierarchy, more parallelism, less complex control requirements, and more computation units, which makes GPU power modeling vary from general-purpose processing units. In the future, detailed micro architectural knowledge of GPU needs to be relied on for providing more complex and accurate modeling approaches. Also, quantitative analysis of GPU workloads and statistical selection of the power consumption correlated workloads are necessary in the data preprocessing step. Despite these current limitations, we believe this work can still serve as an intriguing starting point for future endeavors to develop energy-efficient tools for GPU-based computing applications.

Acknowledgments

Xiaohan Ma and Zhigang Deng were supported in part by the Texas Norman Hackerman Advanced Research Program (NHARP 003652-0058-2007). Lin Zhong and Mian Dong were supported in part by NSF CNS-0720825 and IIS-0713249 and by the Texas Instruments Leadership University program. The authors would like to thank Michael Jahn for his proofreading and the anonymous reviewers for their constructive comments.

6. REFERENCES

[1] F. Bellosa. The benefits of event: driven energy accounting in power-sensitive systems. In *Proc. of 9th workshop on ACM SIGOPS European workshop*, pages 37–42, 2000.

[2] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *ISLPED '01*, pages 135–140, 2001.

[3] T. Li and L. K. John. Run-time modeling and estimation of operating system power consumption. In *SIGMETRICS '03*, pages 160–171, 2003.

[4] K. Ramani, A. Ibrahim, and D. Shimizu. PowerRed: A flexible power modeling framework for power efficiency exploration in GPUs. In *Worskshop on GPGPU*, 2007.

[5] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *HotPower'08*, 2008.

[6] J. Roca, V. M. D. Barrio, C. González, C. Solis, A. Fernández, and R. Espasa. Workload characterization of 3d games. In *IISWC*, pages 17–26, 2006.

[7] J. W. Sheaffer, D. Luebke, and K. Skadron. A flexible simulation framework for graphics architectures. In *HWWS '04*, pages 85–94, 2004.

[8] 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.

[9] 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.