A Measurement Study of GPU DVFS on Energy Conservation

Xinxin Mei, Ling Sing Yung, Kaiyong Zhao, Xiaowen Chu
Department of Computer Science, Hong Kong Baptist University
{*xxmei, lsyung, kyzhao, chxw*}*@comp.hkbu.edu.hk*

Abstract

Energy conservation on computing systems has become an important research topic in recent years. Dynamic voltage/frequency scaling (DVFS) has been shown as an appealing method for saving energy. Nowadays, GPUs have been widely used to accelerate many high performance computing applications. However, there is a lack of study on the effectiveness of GPU DVFS on energy conservation. This paper presents a thorough measurement study that aims to explore how GPU DVFS affects the system energy consumption. We have conducted experiments on a real GPU platform with 37 benchmark applications. Our results show that GPU DVFS is an effective approach to conserving energy. For example, by scaling down the GPU core voltage and frequency, we have achieved an average of 19.28% energy reduction compared with the default setting, while giving up no more than 4% of performance. For all tested GPU applications, core voltage scaling is significantly effective to reduce system energy consumption. On the other hand, the effect of scaling core frequency and memory frequency depends on the characteristic of GPU applications.

1 Introduction

In the past few years, general purpose GPUs have become increasingly popular among high performance computing area. With a large number of cores, GPUs can be much faster than traditional CPUs in a variety of scientific and commercial applications [4, 5, 16]. Although GPUs have higher power efficiency than most CPUs, they still consume a lot of power. For example, the Titan supercomputer at Oak Ridge National Laboratory is equipped with 18,688 NVIDIA Tesla K20X GPU cards and consumes more than 8 million watts at full load. How to conserver energy on such GPU platforms becomes an important problem.

Dynamic voltage/frequency scaling (DVFS) has been shown as an effective approach to saving energy for CPUs [6, 7, 17]. However, the study of GPU DVFS on energy conservation is still at an early stage. Existing research work on GPU energy consumption mostly focused on evaluating the GPU power efficiency, and the modeling of GPU power consumption [1, 2, 8–13, 15, 19]. In this paper, we aim to answer one question: can GPU DVFS save energy in practice? We conduct extensive experiments on a set of 37 GPU applications to investigate the impact of GPU DVFS on the whole system energy consumption.

Our major findings are summarized as follows. First of all, our experimental results show that GPU DVFS is an effective approach to saving system energy for a broad range of applications. Secondly, we find that scaling down the GPU core voltage is effective in saving the system idle power. Thirdly, scaling down the GPU core voltage is very effective in saving system runtime energy when working at an appropriate core frequency. Lastly, we show that GPU memory frequency scaling can also save system runtime energy for some applications, but the optimal setting of memory frequency depends on the application characteristics.

The rest of this paper is organized as follows. Section 2 summarizes related work. Section 3 states our measurement platform and methodology. Section 4 presents our experimental results. The last section concludes our findings and proposes some possible future work.

2 Related Work

The pioneer work on GPU energy conservation was done by Hong and Kim [9–11]. They used PTX code analysis to predict GPU kernel execution cycles. With similar approach, they could also estimate kernel power consumption. They came up with an integrated GPU performance and power prediction model eventually. Then they proposed to use fewer GPU SMs when possible to reduce GPU energy consumption. Nagasaka *et al.* offered an alternate to estimate GPU power consumption using performance counters [19].

Jiao *et al.* studied GPU core and memory frequency scaling on NVIDIA GeForce GTX 280 [12]. They disclosed that power efficiency is largely determined by the ratio of memory transactions to computation instructions. Scaling down GPU core frequency could save energy for memory intense kernels. Abe *et al.* reported a 28% of system energy reduction on 64×64 matrix multiplication by scaling down memory frequency with NVIDIA GeForce GTX 480 [1]. Ge *et al.* applied frequency scaling to both CPU and GPU with three typical parallel applications. They found that scaling GPU frequency higher would not consume more energy [8]. Distinct from these related work, we investigate the impact of

Table 1: Platform configuration

CPU	Intel Core^{TM} i5-750 (4 core)
Clock rate	2.67 GHz
RAM	Kingston DDR3 1333MHz 2GB
MainBoard	ASUS P7P55D PRO
Harddisk	Seagate ST31000528AS 1TB
Power Supply	MaxPower GPX850
GPU	NVIDIA GeForce GTX 560 Ti
Shading clock rate	1900 MHz
Memory interface	1 G GDDR5
Memory clock rate	2100 MHz
GPU driver	306.97
CUDA runtime version	4.1

GPU voltage scaling as well as frequency scaling on a much broader range of applications.

Lee and Kim analyzed the optimal core number and voltage/frequency setting of many-core processors using predictive technology models [13, 14]. They found that doubling core number with lower supply voltage could reduce up to 65% power. Very recently, Leng *et al.* simulated GPU core DVFS and showed 14.4% energy reduction through GPGPU-Sim [2, 15]. Our measurement study is complementary to these theoretical analysis.

3 Experimental Methodology

A GPU board has four scalable variables: core frequency, core voltage, DRAM I/O frequency, and DRAM voltage, denoted as f_{core}, V_{core}, f_{mem} and V_{mem} respectively. We use E to denote the entire system energy consumption during program execution period. Our target is to observe how E varies to f_{core}, V_{core}, f_{mem} and V_{mem}.

Some software are available to adjust GPU voltages and frequencies [18, 21]. In order to achieve the widest range of GPU core voltage, we first use *NVIDIA Inspector* 1.9.7.1 to do coarse adjustment; and then we use *Afterburner* 2.3.0 to do fine adjustment. We also trace GPU temperature and some other runtime information via TechPowerUp's *GPU-Z* [22]. In this paper, we only focus on GPU DVFS and do not apply CPU DVFS.

We perform our experiments on a personal computer equipped with a graphic card MSI N560GTX-Ti Hawk. We choose this card because our scaling tool *Afterburner* supports up to Geforce GTX 500 family GPUs at the time we started this research work. Our system specification is given in Table 1. The measured idle system power is about 85 W in which 29 W is contributed by the graphic card. Table 2 lists the obtained scaling interval of our graphic card.

We measure the whole system power consumption by a commercial power meter, *Watts Up? Pro*, which takes a power sample every second, denoted as P_i for the ith sample. This meter has independent power supply, so it nearly has no influence on our system power consumption. We estimate the average power consumption of a program, $\bar{P}$, by $\bar{P} = \sum_i P_i/N$, in which N is the number of samples obtained during the execution of the application. We obtain the application execution time, t, by function *gettimeofday()*. Then we estimate the whole system power consumption E by $E = \bar{P} \times t$.

Table 2: Geforce GTX 560 Ti allowable scaling interval

Category	Default	Adjustable range
V_{core} (V)	1.049	[0.849, 1.149]
f_{core} (MHz)	950	[480, 1000]
V_{mem} (V)	1.50	[1.40, 1.58]
f_{mem} (MHz)	2100	[1050, 2300]

The system energy consumption at default GPU configuration is represented as $\hat{E}$. For the same application, the minimum and maximum energy consumption under different voltage/ frequency settings are denoted as E_{min} and E_{max}. We use two metrics $\hat{R}$ and R_{max} to evaluate energy conservation:

$$\hat{R} = 1 - E_{min}/\hat{E} \quad (1)$$

which quantizes how much energy could be saved compared to default configuration;

$$R_{max} = 1 - E_{min}/E_{max} \quad (2)$$

which indicates the maximum energy saving capability. Due to limited space, we don't present the ratio of reduced energy over GPU energy, which is much larger than $\hat{R}$ and R_{max}.

Our benchmark suit consists of 37 GPU applications taken from CUDA SDK 4.1 [20] and Rodinia [3]. Due to the RLC effect [15], programs should execute for a relatively long time to get accurate measurements. We revise the source codes of these GPU applications so that each program would last for more than half a minute. We use control variate method to explore the impact of GPU DVFS on system energy consumption. Namely for each group of experiments, we fix part of the four variables, and observe system energy's response to remaining variables. Detailed configurations are given in Section 4.

4 Experimental Results

4.1 System Idle Power

GPU idle power is a non-negligible component of the whole system power consumption. It is well known that the GPU idle power is irrelevant to f_{core} or f_{mem}, which is also confirmed by our experiments. So we measure the system idle power at different combinations of $\{V_{mem}, V_{core}\}$. Our measurement results are shown in Figure 1. The system idle power varies from 78 W

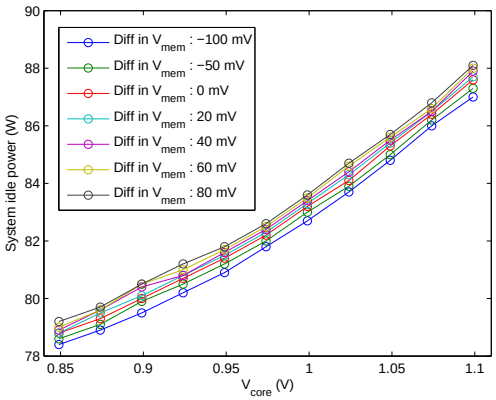

Figure 1: Idle power consumption of different voltages

to 89 W, where the variation is mainly caused by scaling V_{core}. The impact of V_{mem} on system idle power is very marginal, mostly within 1 W. On our platform, the idle GPU power drops from 29 W to 22 W when scaling down V_{core} from 1.049 V to 0.849 V, which corresponds to about 8% of system power reduction.

4.2 Core Scaling

Core scaling refers to the adjustment of V_{core} and/or f_{core}. We first fix V_{core} and explore the impact of f_{core} on E. We do experiments under two different core voltages: 1.049 V and 0.849 V. For each core voltage, we change f_{core} from 480 MHz to 880 MHz incrementally.

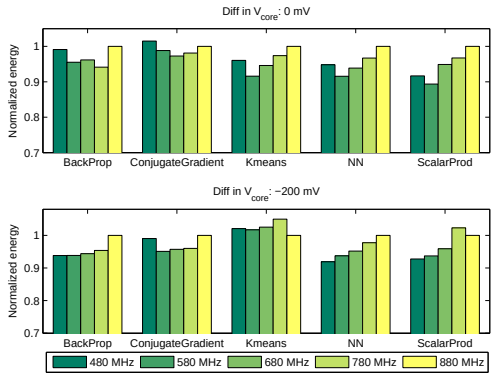

Figure 2: Benchmarks benefit from low core frequencies

For CPUs, it is generally believed that with a fixed voltage, using high processor frequencies would reduce total energy because of shorter execution time. However, we find that it is not always true for GPUs. Among our 37 testing benchmarks, 5 actually benefit from lower core frequencies. They are: *BackProp*, *NN*(nearest neighbourhood), *ConjugateGradient*, *Kmeans* and *ScalarProd*. These applications have frequent CPU-GPU data transactions, and core frequency scaling almost has no impact on their execution time. Figure 2

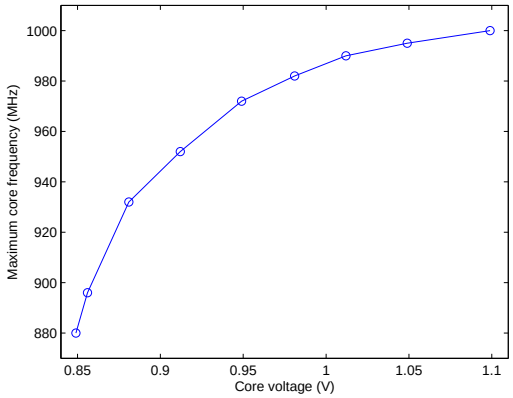

Figure 3: Core voltage and maximum stable frequency

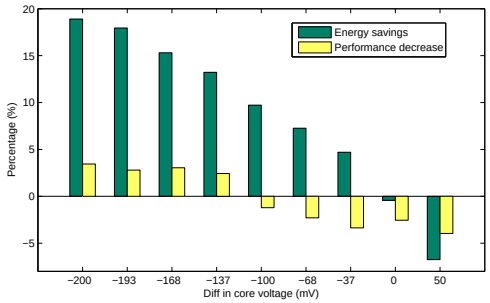

Figure 4: Energy savings of separate core voltages

gives their normalized energy (by normalizing to the energy consumption at $f_{core} = 880$ MHz). Up to 10% of energy can be saved by scaling down f_{core}. It is interesting to see that for *Kmeans*, scaling down f_{core} can save energy when $V_{core} = 1.049$ V, but this situation does not hold anymore when $V_{core} = 0.849$ V. It is an interesting research problem to find the optimal f_{core} that can save the most energy for such special applications.

But for the other 32 benchmark applications, energy consumption suffers a lot under low f_{core}. A majority of applications become much slower when running at low core voltages, and hence they consume much more energy. Since for most of applications, scaling down f_{core} will increase energy consumption, we use the maximum stable core frequency, f^*_{core}, of each V_{core} to do the subsequent voltage scaling experiments.

We obtain f^*_{core} by fixing V_{core} and running pressure tests by increasing f_{core} until GPU turns to be unstable. Figure 3 illustrates the relationship between f^*_{core} and V_{core}. When core voltage changes from 0.849 V to 1.099 V, f^*_{core} increases accordingly. The (f^*_{core}, V_{core}) pairs construct a space in which GPU hardware has the best compute capacity. Notice that the default GPU core frequency setting (i.e., 950MHz) is relatively conservative.

We do core voltage scaling experiments within above space. The mean energy savings ($\hat{R}$) and performance de-

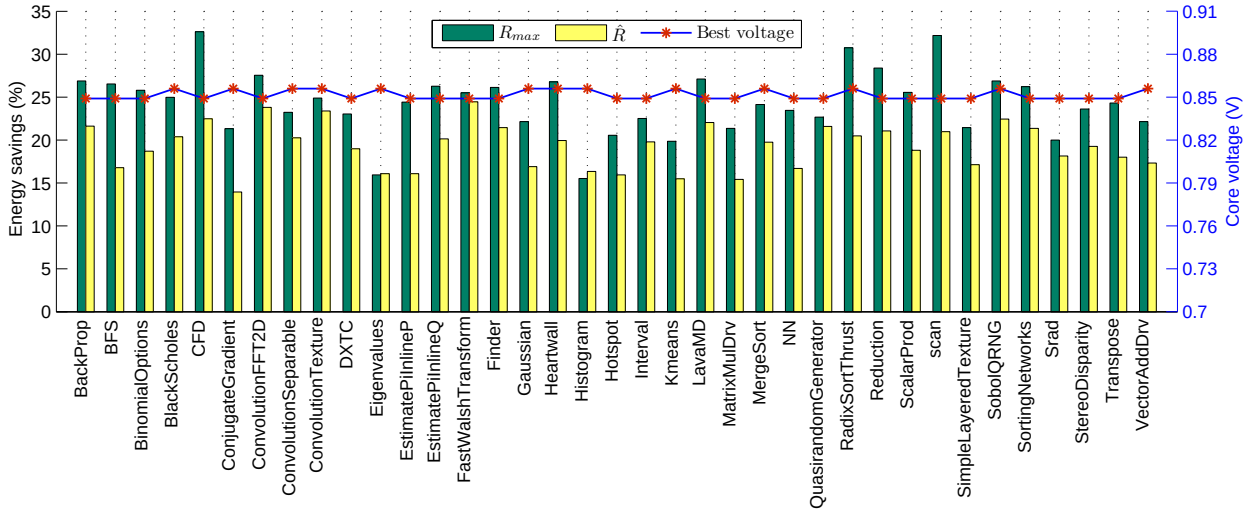

Figure 5: Core voltage scaling efficiency of all programs

Table 3: Runtime temperature of *ScalarProd*

V_{core} (V)	0.849	0.949	1.012	1.049	1.099
Temp inc (°C)	18	24	27	30	32

crease (as compared to the default setting) of 37 benchmarks is shown in Figure 4. Notice that negative performance decrease means better performance due to higher core frequencies. We also gain an average of 2% of speedup at default V_{core} since we have scaled up default f_{core} (950 MHz) to f^*_{core} (995 MHz).

For all benchmarks, low V_{core} is significantly effective to conserve energy. In particular, an average of 18.91% energy is reduced with 3.45% performance decrease at $V_{core} = 0.849$ V, $f_{core} = 880$ MHz. $\hat{R}$ decreases almost linearly as V_{core} increases.

Figure 5 summarizes $\hat{R}$ and R_{max} of all benchmarks. Applications that can save the most energy include *CFD*, *FastWalshTransform*, *convolutionFFT2D etc*. These applications are both memory intensive and computation intensive. The average $\hat{R}$ is 19.28% while average R_{max} is 24.40%. In the best case (*CFD*), up to 32.62% of energy can be reduced.

We also find that the GPU runtime temperature is closely related to energy conservation. Low temperature means less energy is converted into heat, so that electricity can be used more efficiently. We record the difference between GPU chip peak temperature during program execution and temperature in idle state. We find that for the same application, the increase of temperature at high voltage is much bigger than that of low voltage. Table 3 is a runtime chip temperature example of *ScalarProd*. The execution time of *ScalarProd* is insensitive to f_{core} or V_{core} scaling. The large difference between the increase of chip temperature at different voltages is mainly caused by V_{core} variation.

4.3 Memory Scaling

We continue our experiments by adjusting V_{mem} and f_{mem}. We find that scaling V_{mem} does not have obvious influence on whole system energy. This can be explained by two reasons. First, V_{mem} only offers a narrow adjusting range, and increasing V_{mem} doesn't lead to higher stable f_{mem}. Second, GPU DRAM just accounts for a small part of the whole GPU board power consumption [9]. As a result, we focus on the experiments of memory frequency scaling.

We test all our benchmarks with f_{mem} varying from 1500 MHz to 2300 MHz where $V_{mem} = 1.50$ V, $V_{core} = 1.049$ V, $f_{core} = 990$ MHz. The memory scaling results show strong individual characteristic. Figure 6 plots the optimal f_{mem} that can achieve minimum energy consumption for 37 benchmarks. The computation intensive kernels, like *MatrixMulDrv etc*, benefit from low f_{mem}, while memory intensive kernels, like *ConvolutionFFT2D etc*, can save energy with high f_{mem}. Default f_{mem} is the optimal setting for eight kernels.

Figure 6 also shows R_{max} and $\hat{R}$ of all benchmarks. f_{mem} affects energy mainly by changing the execution time. Applications with large R_{max}, like *SobolQRNG*, *EstimatePiInlineQ*, *Reduction etc.* last much longer with low f_{mem}. Such programs have large thread divergence. Both memory parallelism and computation parallelism are low, so that memory access latency cannot be hidden effectively.

In summary, the average R_{max} and average $\hat{R}$ are 10.20% and 3.52% respectively. The average energy saving is lower than that of core scaling because many kernels work quite well under the default memory frequency. In fact, memory frequency scaling can save energy significantly for some applications. In the best case (*SobolQRNG*), up to 28.65% energy can be reduced by

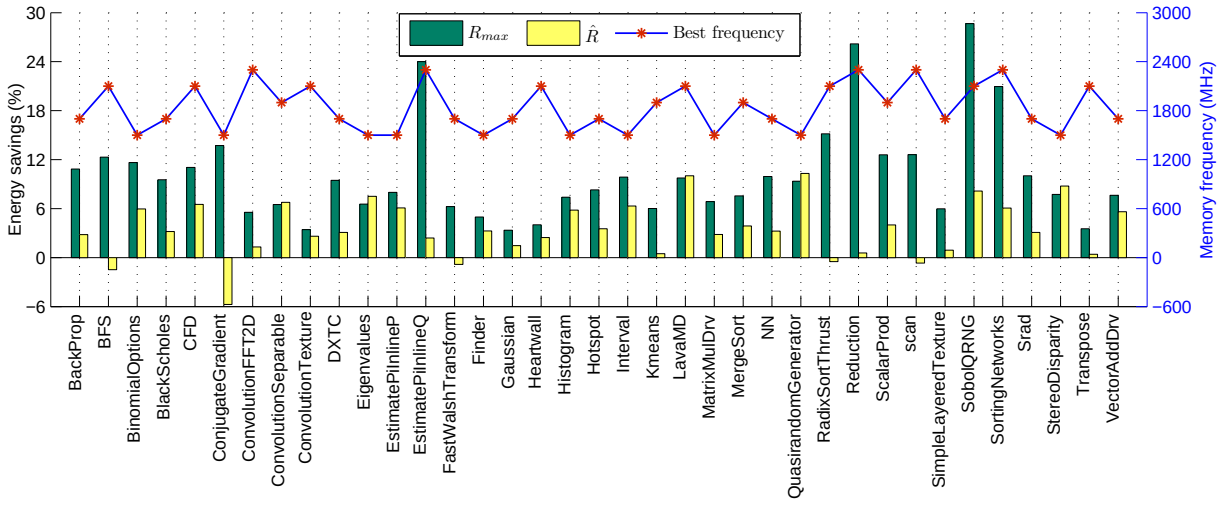

Figure 6: Memory frequency scaling efficiency of all programs

memory frequency scaling.

5 Conclusions and Future Work

In this paper, we conducted a comprehensive measurement study of GPU DVFS on energy conservation. We studied 37 benchmark applications and illustrated that GPU DVFS is an effective approach in saving system energy. In general, scaling down the GPU core voltage can significantly save the energy when working at appropriate core frequency. Memory frequency scaling can also save energy for some applications. However, it is not a trivial task to find the optimal setting of GPU DVFS. In the future, we plan to develop a statistical power model that incorporates GPU DVFS. When combined with an execution time prediction model, it becomes feasible to analytically find out the optimal setting of GPU DVFS that can minimize the system energy consumption.

Acknowledgement

This work is supported by Hong Kong General Research Fund HKBU 210412.

References

[1] Abe, Y., Sasaki, H., Peres, M., Inoue, K., Murakami, K., and Kato, S. Power and performance analysis of gpu-accelerated systems. In *HotPower12* (2012), ACM.

[2] Bakhoda, A., Yuan, G. L., Fung, W. W., Wong, H., and Aamodt, T. M. Analyzing cuda workloads using a detailed gpu simulator. In *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on* (2009), IEEE, pp. 163–174.

[3] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S.-H., and Skadron, K. Rodinia: A benchmark suite for heterogeneous computing. In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on* (2009), IEEE, pp. 44–54.

[4] Chu, X., and Zhao, K. Practical random linear network coding on gpus. In *GPU Solutions to Multi-scale Problems in Science and Engineering*. Springer, 2013, pp. 115–130.

[5] Chu, X., Zhao, K., and Wang, M. Massively parallel network coding on gpus. In *Performance, Computing and Communications Conference, 2008. IPCCC 2008. IEEE International* (2008), IEEE, pp. 144–151.

[6] David, H., Fallin, C., Gorbatov, E., Hanebutte, U. R., and Mutlu, O. Memory power management via dynamic voltage/frequency scaling. In *8th ACM international conference on Autonomic computing* (2011), ACM, pp. 31–40.

[7] Etinski, M., Corbalan, J., Labarta, J., and Valero, M. Understanding the future of energy-performance trade-off via dvfs in hpc environments. *Journal of Parallel and Distributed Computing 72*, 4 (2012), 579–590.

[8] Ge, R., Vogt, R., Majumder, J., Alam, A., Burtscher, M., and Zong, Z. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *2nd International Workshop on Power-aware Algorithms, Systems, and Architectures* (2013), IEEE.

[9] Hong, S. *Modeling performance and power for energy-efficient gpgpu computing*. PhD thesis, Georgia Institute of Technology, 2012.

[10] Hong, S., and Kim, H. An analytical model for a gpu architecture with memory-level and thread-level parallelism awareness. In *ACM SIGARCH Computer Architecture News* (2009), vol. 37, ACM, pp. 152–163.

[11] Hong, S., and Kim, H. An integrated gpu power and performance model. In *ACM SIGARCH Computer Architecture News* (2010), vol. 38, ACM, pp. 280–289.

[12] Jiao, Y., Lin, H., Balaji, P., and Feng, W. Power and performance characterization of computational kernels on the gpu. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)* (2010), IEEE, pp. 221–228.

[13] Lee, J., and Kim, N. S. Optimizing total power of many-core processors considering voltage scaling limit and process variations. In *14th ACM/IEEE international symposium on Low power electronics and design* (2009), ACM, pp. 201–206.

[14] Lee, J., Sathisha, V., Schulte, M., Compton, K., and Kim, N. S. Improving throughput of power-constrained gpus using dynamic voltage/frequency and core scaling. In *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on* (2011), IEEE, pp. 111–120.

[15] Leng, J., Hetherington, T., ElTantawy, A., Gilani, S., Kim, N. S., Aamodt, T. M., and Reddi, V. J. Gpuwattch: Enabling energy optimizations in gpgpus. In *ISCA* (2013), vol. 40.

[16] Li, Y., Zhao, K., Chu, X., and Liu, J. Speeding up k-means algorithm by gpus. *Journal of Computer and System Sciences 79*, 2 (2013), 216–229.

[17] Lim, M. Y., and Freeh, V. W. Determining the minimum energy consumption using dynamic voltage and frequency scaling. In *Parallel and Distributed Processing Symposium* (2007), IEEE, pp. 1–8.

[18] MSI. Afterburner, graphics card performance booster. `http://event.msi.com/vga/afterburner/download.htm`.

[19] Nagasaka, H., Maruyama, N., Nukada, A., Endo, T., and Matsuoka, S. Statistical power modeling of gpu kernels using performance counters. In *Green Computing Conference, 2010 International* (2010), IEEE, pp. 115–122.

[20] NVIDIA. Gpu computing sdk. `https://developer.nvidia.com/gpu-computing-sdk`.

[21] Orbmu2k. Nvidia inspector. `http://blog.orbmu2k.de/tools/nvidia-inspector-tool`.

[22] TechPowerUp. Gpu-z. `http://www.techpowerup.com/gpuz/`.