Power and Performance Analysis of GPU-Accelerated Systems

Yuki Abe
Kyushu University

Hiroshi Sasaki
Kyushu University

Martin Peres
Laboratoire Bordelais de Recherche en Informatique

Koji Inoue
Kyushu University

Kazuaki Murakami
Kyushu University

Shinpei Kato
Nagoya University

Abstract

Graphics processing units (GPUs) provide significant improvements in performance and performance-per-watt as compared to traditional multicore CPUs. This energy-efficiency of GPUs has facilitated use of GPUs in many application domains. Albeit energy efficient, GPUs still consume non-trivial power independently of CPUs. It is desired to analyze the power and performance charateristic of GPUs and their causal relation with CPUs. In this paper, we provide a power and performance analysis of GPU-accelerated systems for better understandings of these implications. Our analysis discloses that system energy could be reduced by about 28% retaining a decrease in performance within 1%. Specifically, we identify that energy saving is particularly significant when (i) reducing the GPU memory clock for compute-intensive workload and (ii) reducing the GPU core clock for memory-intensive workload. We also demonstrate that voltage and frequency scaling of CPUs is trivial and even should not be applied in GPU-accelerated systems. We believe that these findings are useful to develop dynamic voltage and frequency scaling (DVFS) algorithms for GPU-accelerated systems.

1 Introduction

Graphics processing units (GPUs) have been increasingly deployed in general-purpose application domains due to their significant improvements in performance and performance-per-watt. As depicted in Figure 1, the performance-per-watt of GPUs outperforms highly that of traditional multicore CPUs. Albeit energy efficient, GPUs still consume non-trivial power during operation. Commodity system software for GPUs is unfortunately not well designed to control their power consumption while primarily tailored to accelerate computations. To the best of our knowledge, commodity system software does not employ even a basic scheme of voltage and frequency scaling for GPUs, though most computational pieces of GPU-accelerated systems are offloaded on to
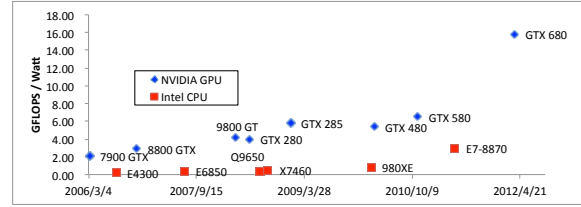

Figure 1: Performance-per-watt trends on representative NVIDIA GPUs and Intel CPUs.

GPUs. In order to develop truly energy-efficient GPU-accelerated systems, it is essential to identify the trade-off in power and performance of GPUs and its causal relation with CPUs.

Despite a rapid growth of GPU technology, there has been not much understanding of power and performance implications of GPU-accelerated systems. According to vendor's specifications, the thermal design power (TDP) of state-of-the-art GPUs is around 200W, while that of today's multicore CPUs is typically below 100W. Such a difference in the scale of TDP between CPUs and GPUs prevents system designers from predicting the power and performance of GPU-accelerated systems, which makes it difficult, if not impossible, to optimize their energy savings. Previous work [2, 3, 5, 6, 7] on the power and performance analysis of GPU-accelerated systems are based on either simulation studies or limited hardware functionality. None of previous work has ever disclosed a fundamental approach to the power and performance analysis of GPU-accelerated systems.

The contribution of this paper is to provide a power and performance analysis of GPU-accelerated systems using NVIDIA's *Fermi* architectures. Specifically, we identify when to scale the frequency and voltage of GPUs and CPUs in order to minimize overall system energy. Our analysis opens up important problems of dynamic voltage and frequency scaling (DVFS) algorithms for growing GPU-accelerated systems. We also provide an open method and tool to scale voltage and frequency of GPUs. The black box feature of current GPU drivers and

Table 1: Performance levels of GTX 480 (GPU).

Clock Domains	Min [MHz]	Low [MHz]	High [MHz]
Core	50	405	700
Memory	135	324	1848

Table 2: Performance levels of Core i5 2400 (CPU).

Platforms	Min [MHz]	Low [MHz]	High [MHz]
Core i5-2400	1600	2700	3300.1

Figure 2: Power consumption and execution time of the 512×512 matrix addition program.

runtimes prevents researchers from tackling correlative power and performance optimization problems. Sharing such a common method and tool with researchers would further facilitate use of GPUs.

The rest of this paper is organized as follows. Section 2 presents our system platform. Section 3 provides our evaluation. Section 4 discusses related work, and this paper concludes in Section 5.

2 System Platform

We use an NVIDIA GeForce GTX 480 graphics card and Intel Core i5 2400 processor with the Linux kernel 3.3.0. Table 1 and 2 present their available performance levels respectively. To perform the experiment, we use NVIDIA's proprietary software [9] and Gdev [10] case by case. NVIDIA's proprietary software does not provide a system interface to scale the performance level of the GPU. We hence provide the modified BIOS files for the GPU that force the binary driver to configure the GPU with the specified performance level when loaded. This method enables us to choose any set of the GPU core and memory clocks, but requires the driver to reload, and the configuration is static while the driver is running. On the other hand, Gdev allows the system to change the performance level of the GPU dynamically at runtime through the Linux "/proc" file system interface. However, it is available only for the GPU core clock at the moment, and the GPU memory clock is fixed at 135MHz. This is limited due to an open-source implementation of Linux, but will be removed in the future release.

The experimental workload executes the Rodinia benchmark suite 2.0.1 [1] and our original microbenchmark programs of matrix computation. All input data for the Rodinia programs use the maximum feasible size, while the microbenchmark programs vary data size to conduct fine-grained measurements, all of which are written in CUDA. We use the NVIDIA CUDA Compiler (NVCC) 4.0 [8] to compile the programs. Both NVIDIA's proprietary software and Gdev receive the same program binary.
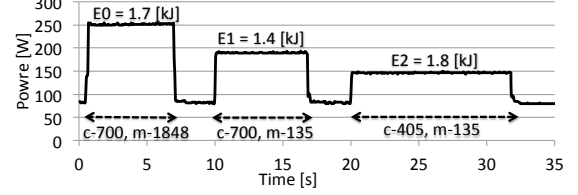
The power and energy consumption of the system is measured by the YOKOGAWA WT1600 digital power meter. This instrument obtains the voltage and electric current every $50ms$ from the power plug of the machine. The power consumption is calculated by multiplying the voltage and current, while the energy consumption is derived by accumulation of power consumption.

3 Evaluation

We first investigate the impact of GPU core and memory clocks on GPU-intensive workload executing twenty thousands loops of 512×512 matrix addition. The voltage and frequency of the GPU is changed three times during the operation, while the CPU is fixed at the minimum level to focus on the behavior of the GPU. Figure 2 shows the power consumption of the system in this setup, where "c-*" and "m-*" stand for the GPU core and memory clocks respectively, while "E*" represents the cumulative energy consumption of the corresponding duration. What is learned from this experiment is that energy consumption is sensitive to the GPU core and memory clocks. Lowering the memory clock to 135MHz successfully reduces energy consumption, but the further downscaling of the core clock to 405MHz counter-increases energy consumption. This indeed implies DVFS algorithms dominate the power and performance of GPU-accelerated systems.

We next coordinate the GPU and the CPU using more realistic workload from the Rodinia benchmark suite. To simplify the setup, we consider only high (maximum) and low (minimum) core clocks, meaning that we evaluate four configurations of (GPU-L, CPU-L), (GPU-H, CPU-L), (GPU-L, CPU-H), and (GPU-H, CPU-H), where "*-L" and "*-H" represent the low and high core clocks. In an idle state, however, the clocks are always down-scaled to the minimum level. We also add another configuration that keeps at the maximum clocks even though the GPU is idle, in order to see the impact of elementary coordinated DVFS on GPU-accelerated systems. Figure 3 and 4 respectively show the execution time and energy consumption of four representa-
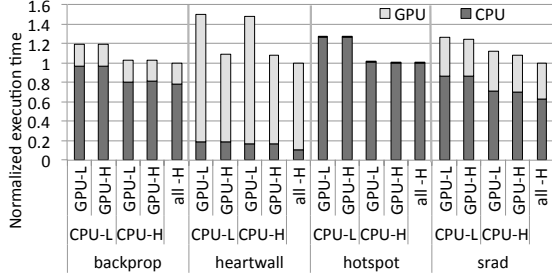
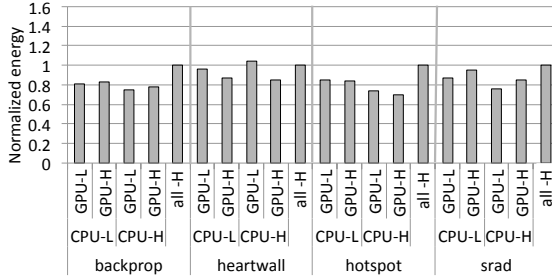Figure 3: Execution time of the Rodinia programs.

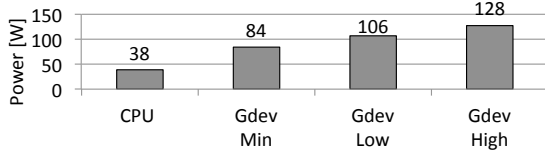

Figure 4: Energy consumption of the Rodinia programs.

Figure 5: Power consumption in an idle state.

(a) Execution Time

(b) Energy Consumption
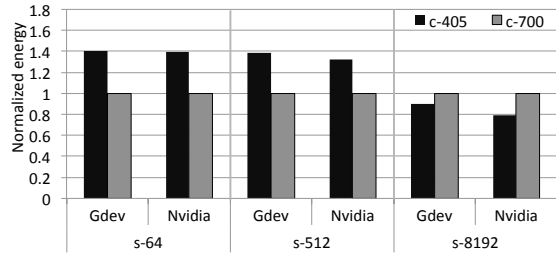
Figure 8: Comparison of the NVIDIA proprietary and the Gdev open-source runtimes and drivers.

tive programs of the Rodinia benchmark suite. Regarding the execution time, "all-H" always takes the shortest execution time, as it consistently keeps at the maximum performance level. Other configurations however depend on workload. For example, the execution time of heartwall – GPU-intensive workload – can be decreased by setting the high GPU clock, whereas that of hotspot is rather affected by the CPU clock. The characteristic of energy consumption is more complicated. For some workload, lowering the clock causes an increase in energy consumption, as the duration of execution is increased, consuming more cumulative power consumption. In other words, GPU-intensive workload should generally use the high GPU clock so that it completes operation as soon as possible to minimize energy. Apparently, "all-H" is not a good idea in terms of energy; the clock should be minimized when the device is not used. Hence, DVFS is certainly desired but the design of its algorithms is left an open issue.

In the above experiments, we have never observed that energy consumption is reduced by lowering the CPU clock. This is because lowering the CPU clock causes the GPU to increase the duration of an idle state, and there is no power gating support for the GPU at the moment. Hence, energy is always wasted when the GPU is idle. We demonstrate how energy is wasted in an idle state, when (i) the GPU is not present and (ii) is present with three levels of a set of voltage and frequency. Figure 5 shows the average power consumption of those four cases obtained by running the system for 60 seconds. The CPU consumes no more than 38W on average, whereas the GPU-installed systems consume a different scale of power depending on the configured set of voltage and frequency. This observation encourages the system not to downscale the voltage and frequency of the CPU, unless the GPU supports power gating to totally cut off its consuming power. Another lesson learned from this experiment is that the power consumption of the GPU is significant even in an idle state, meaning that DVFS is strongly desired for the GPU with whatever overhead it has to pay for changing the performance level.

The preceding evaluation indicates that the CPU is a weak factor for energy savings of GPU-accelerated systems. Henceforth, we restrict our attention to the GPU. According to the traditional power modeling [4], lowering the core clock is often effective for memory-intensive workload. Our next evaluation verifies if the same is true for the GPU. We use matrix addition and multiplication programs with varied sizes of data. A small size of
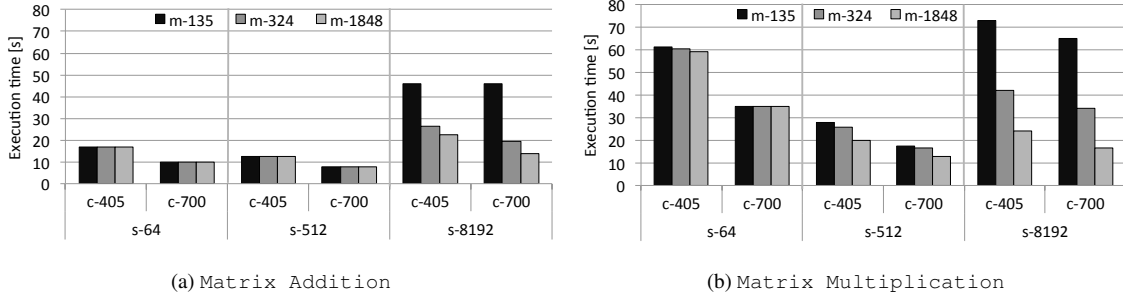
| (a) Matrix Addition | (b) Matrix Multiplication |

Figure 6: Execution time of the matrix addition and multiplication programs.

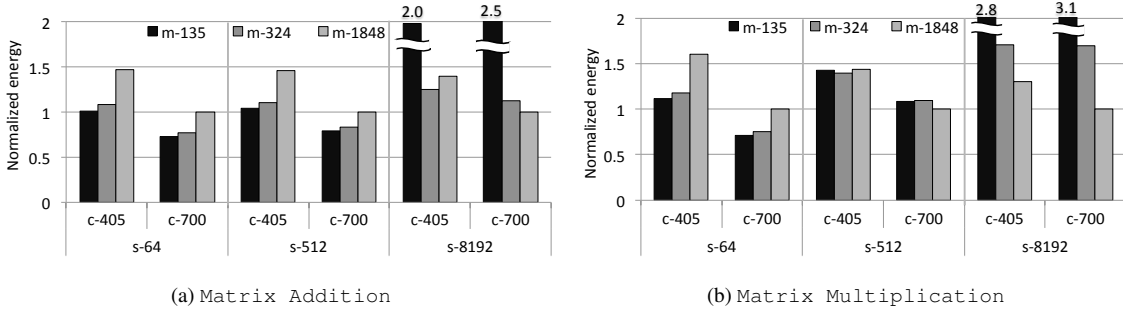| (a) Matrix Addition | (b) Matrix Multiplication |

Figure 7: Energy consumption of the matrix addition and multiplication programs.

data reduces memory accesses, while a large size of data makes the workload memory-intensive. Figure 6 and 7 show the execution time and energy consumption of those matrix computations, where "s-*" represents the number of matrix row/column. A difference between "s-64" and "s-8192" explains that memory-clock scaling is more effective for such computations that use a smaller size of data. This is because the execution time of such computations is not affected by lowering the memory clock. Another observation is that energy cannot be saved by lowering the core clock, because these matrix computations are consistently compute-intensive. If the core clock is downscaled, their execution time is highly increased, which results in an increase in cumulative power consumption.

Seen from the above experiments, system energy could be reduced by about 28% retaining a decrease in performance within 1%. These experimental results encourage that DVFS algorithms for GPU-accelerated systems should be weighted on the GPU rather the CPU, though their energy optimization is very challenging, given many factors of design knobs including CPU/GPU, core/memory, and workload characteristics.

Finally, we compare NVIDIA's proprietary software and Gdev. This is an important and practical investigation in that NVIDIA's proprietary software does not expose a system interface to change the voltage and frequency of the GPU dynamically at runtime, and hence

the development of DVFS algorithms in future work will inevitably depend on Gdev. The basic performance of Gdev is competitive to NVIDIA's proprietary software [10], but we have to evince that Gdev is also reliable for power management. The test program exploits matrix addition with varied sizes of data. Figure 8 shows the execution time and energy consumption of the matrix addition programs using different scales of GPU core clocks, where the GPU memory clock is fixed at 135MHz. In this experiment, "s-8192" benefits from lowering the core clock, because the workload is memory-bound due to a large size of matrix, and the execution time is not much affected by the core clock, while every is effectively saved. The most remarkable observation is that NVIDIA's proprietary software and Gdev exhibit almost the same results on the execution time and energy consumption. This implies that the result of our on-going research using Gdev could be easily propagated to the real product, once energy management interfaces are employed by vendor's software.

4 Related Work

Nagasaka *et al.* conjectured energy consumption of GPUs based on the hardware performance counter [7]. This performance counter, however, is not adequate in that power consumption rises even in an idle state when

voltage and frequency are scaled, though the performance counter does not change in an idle state. Hence, this approach would require an additional model to precisely analyze the power consumption of the GPU.

Hong *et al.* studied energy savings of GPUs, assuming power gating available to limit the number of active cores [2, 3]. In particular, they analyze PTX code to model the power and performance of GPUs based on the number of instructions and memory accesses. Unfortunately, none of the current GPU architectures yet supports power gating, which limited their contribution to simulation studies. Therefore, it is questionable if the presented power and performance model is applicable to the real-world, and GPU power gating is also not a realistic assumption at the moment. Nonetheless, we consider that an offline PTX analysis for power and performance prediction is a useful approach to the design of DVFS algorithms. What lacks in this approach, however, is a runtime analysis for input data. In this paper, we have analyzed the power and performance characteristics depending on the size of input data.

Lee *et al.* presented a method to apply DVFS algorithms to the GPU. They particularly aimed at maximizing performance under the given power constraint [6]. A strong limitation of their work, however, is that the evaluation of power consumption is based on a conceptual model but not on real-world hardware. They also failed to discuss how to determine the voltage and frequency. In this paper, we have rather explored how to minimize the energy consumption of GPU-accelerated systems using the cutting-edge real-world hardware.

Jiao *et al.* evaluated the power and performance of an old NVIDIA GTX 280 GPU [5]. They examined compute-intensive and memory-intensive programs. According to their analysis, energy consumption could often be reduced by lowering the core clock when workload is memory-intensive. This is exactly the same as what we have identified for an NVIDIA's GTX 480 GPU. Therefore, we conjecture that this observation and knowledge could be applied to future GPU architectures as well. In addition, we have disclosed that energy consumption could also be reduced by scaling the memory clock. This opens up a new insight into DVFS algorithms for GPU-accelerated systems.

Ying *et al.* analyzed the power and performance of an AMD HD 5870 GPU using a random forest method with the profile counter [11]. They revealed that activating a fewer number of ALUs reduces power consumption. However, this approach incurs an increase in execution time, and does not successfully reduce energy consumption. This is attributed to the fact that they use only software management. Meanwhile, we have demonstrated that energy can be reduced by scaling the voltate and frequency of the GPU.

5   Conclusion and Future Work

We have presented a power and performance analysis of GPU-accelerated systems based on the NVIDIA Fermi architecture. Our findings include that the CPU is a weak factor for energy savings of GPU-accelerated systems unless power gating is suppoted by the GPU. In contrast, voltage and frequency scaling of the GPU is significant to reduce energy consumption. Our experimental results demonstrated that system energy could be reduced by about 28% retaining a decrease in performance within 1%, if the performance level of the GPU can be scaled effectively.

In future work, we plan to develop DVFS algorithms for GPU-accelerated systems, using the characteristic identified in this paper. We basically consider such an approach that controls the GPU core clock for memory-intensive workload while controls the GPU memory clock for compute-intensive workload. To this end, we integrate PTX code analysis [2, 3] into DVFS algorithms so that energy optimization can be provided at runtime. We also consider a further dynamic scheme that scales the performance level of the GPU during the execution of GPU code, whereas we restricted a scaling point at the boundary of GPU code in this paper.

References

[1] CHE, S., SHEAFFER, J. W., BOYER, M., SZAFARYN, L. G., WANG, L., AND SKADRON, K. A characterization of the rodinia benchmark suite with comparison to contemporary cmp workloads. In *Proc. of the IEEE international symposium on Workload characterization* (2010), pp. 1–11.

[2] HONG, S., AND KIM, H. An analytical model for a gpu architecture with memory-level and thread-level parallelism awareness. In *Proc. of the annual international symposium on Computer architecture* (2009), pp. 152–163.

[3] HONG, S., AND KIM, H. An integrated gpu power and performance model. In *Proc. of the annual international symposium on Computer architecture* (2010), pp. 280–289.

[4] HSU, C.-H., KREMER, U., AND HSIAO, M. Compiler-directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. In *Proc. of the international symposium on Low power electronics and design* (2001), pp. 275–278.

[5] JIAO, Y., LIN, H., BALAJI, P., AND FENG, W. Power and performance characterization of computational kernels on the gpu. In *Proc. of the IEEE/ACM international conference on Green computing and communications & international conference on Cyber, physical and social computing* (2010), pp. 221 –228.

[6] 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 *Proc. of the international conference on Parallel architectures and compilation techniques* (2011), pp. 111–120.

[7] NAGASAKA, H., MARUYAMA, N., NUKADA, A., ENDO, T., AND MATSUOKA, S. Statistical power modeling of gpu kernels using performance counters. In *Proc. of the international conference on Green computing* (2010), pp. 115–122.

[8] NVIDIA. CUDA 4.0. http://developer.nvidia.com/cuda-toolkit-40, 2011.

[9] NVIDIA. Linux X64 Display Driver. http://www.nvidia.com/object/linux-display-amd64-295.59-driver.html, 2012.

[10] SHINPEI, K., MICHAEL, M., CARLOS, M., AND SCOTT, B. Gdev: First-class gpu resource management in the operating system. In *Proc. of the USENIX conference on USENIX annual technical conference* (2012).

[11] ZHANG, Y., HU, Y., LI, B., AND PENG, L. Performance and power analysis of ati gpu: A statistical approach. In *Proc. of the IEEE international conference on Networking, architecture, and storage* (2011), pp. 149 –158.