Open Journal of Energy Efficiency, 2015, 4, 77-86
Published Online December 2015 in SciRes. http://www.scirp.org/journal/ojee
http://dx.doi.org/10.4236/ojee.2015.44009

Analysis of DVFS Techniques for Improving the GPU Energy Efficiency

Ashish Mishra, Nilay Khare

Department of Computer Science and Engineering, Maulana Azad National Institute of Technology, Bhopal, India
Email: ashishmishra81@gmail.com

Received 2 October 2015; accepted 11 December 2015; published 14 December 2015

Abstract

Dynamic Voltage Frequency Scaling (DVFS) techniques are used to improve energy efficiency of GPUs. Literature survey and thorough analysis of various schemes on DVFS techniques during the last decade are presented in this paper. Detailed analysis of the schemes is included with respect to comparison of various DVFS techniques over the years. To endow with knowledge of various power management techniques that utilize DVFS during the last decade is the main objective of this paper. During the study, we find that DVFS not only work solely but also in coordination with other power optimization techniques like load balancing and task mapping where performance and energy efficiency are affected by varying the platform and benchmark. Thorough analysis of various schemes on DVFS techniques is presented in this paper such that further research in the field of DVFS can be enhanced.

Keywords

GPGPUs, DVFS, Task Mapping, Energy Efficiency

1. Introduction

As we move from mega scale to petascale era, the requirements of data processing and computation are growing exponentially. In order to accomplish this high computation demand, researchers have moved from serial computation platforms to high performance computation (HPC) platforms such as multicore processor, FPGAs and heterogeneous system (GPU supported systems) etc. GPUs, in particular, have been widely used for HPC applications due to their extremely high computational powers. A large number of supercomputer found in TOP500 list use GPU to achieve unprecedented computational power [1].

Today, GPU has become the core part of high performance system having hundreds to thousands of processor cores and much higher peak performance than CPUs. Hence, many HPC applications utilize the power of GPUs. For example, the recently built supercomputer Tianhe-1A has won the second spot on the TOP 500 list [1] and is equipped with Intel Xeon 5670 processors and NVidia's CUDA-enabled Tesla M2050 general purpose GPUs. Having GPU in Tianhe-1A makes supercomputers able to achieve more than two fold energy efficiency than the third place CPU-based Jaguar TOP 500 list. However, the electricity bill of Tianhe-1A is estimated around annual electricity bill of $2.7 million [2]. This high power consumption is another reason that forces researchers to work in a direction to reduce the power consumption of GPUs.

On the other hand, manufacturers increase the number of processing core to gain the high performance which has resulted in raising the power consumption of GPUs. They consume much high power as compared to CPUs and the raised levels of power consumption of GPUs have significant impact on reliability, architecture design, economic feasibility and deployment into widespread range of application domains. In recent years, several research has been accomplished for the reduction of power consumption of homogenous as well as heterogeneous systems and various techniques to reduce the power consumption of both the systems have been proposed. In [3], Sparsh and Jeffery present superior categorization of various power reduction techniques which are categorized as follows:

1. DVFS based techniques;
2. CPU-GPU workload division techniques;
3. Saving energy in GPU components;
4. Dynamic resource allocation techniques;
5. Application specific and programming level techniques.

In this paper, literature survey and thorough analysis of various schemes on DVFS techniques to reduce the energy efficiency of GPUs only are presented. The rest of the paper is organized as follows.

2. Background

2.1. GPU

As shown in **Figure 1**, all GPUs have two important component one is number of parallel streaming processor and second one is memory used by GPUs core. Each streaming processor again has a number of processing elements. Performance of kernel application is vastly depending on the frequency on which these two components operated.

2.2. Need to Improving Energy Efficiency of GPU

Due to application limitation, it is not always possible for an application to map all the available cores. In several applications, memory bandwidth [4] [5] of GPU act as bottleneck to affect the performance of GPU. Due to this bottleneck, the core of GPU remains unutilized. Therefore, a good power management technique is required

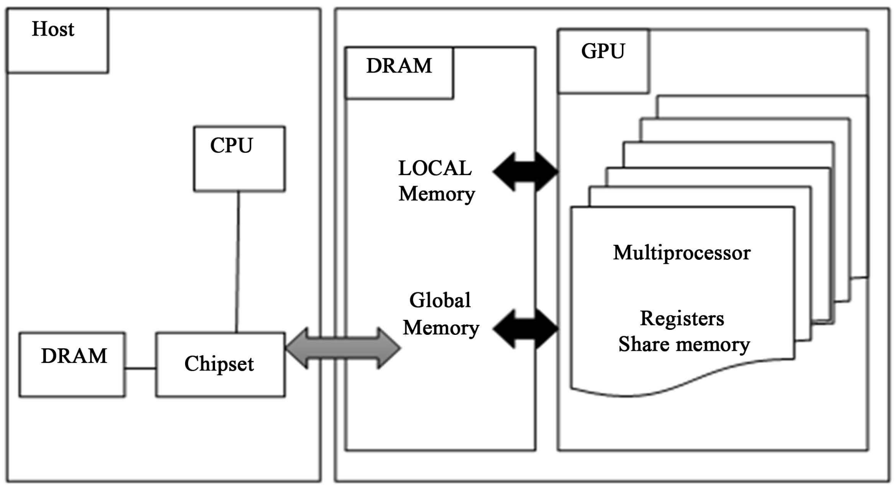

Figure 1. GPU architecture.

to save the power consumed by these unutilized cores. As said by Anderson in [6], a 15 degree increase in temperature is responsible to increase the failure rates of component by a factor of two. This component failure may lead to system malfunction that in turn affect economic of the system as GPU becomes popular accelerator among the super computer and business services. Thus, an efficient energy model is needed to ensure the reliability. Although, GPU's get admired for the performance improvement and have to be energy efficient to ensure the reliably and improve business gain.

3. Dynamic Voltage Scaling Techniques

Dynamic voltage and frequency scaling (DVFS) is a technique widely used for reducing energy consumption of processors by varying the voltage and frequency at run time [7]. The main idea is to reduce frequency or voltage during periods when the processor has a reduced workload. If DVFS is done wisely, energy can be saved without any noticeable effects on the speed at which the processor performs its tasks. Most systems are designed with fixed voltage and frequency settings in order to make the system stable. However, the activity levels of applications are variable, and in many cases, applications have idle periods when no useful task is performed. By reducing the processor voltage and frequency levels at run-time when the application has low-activity or idle periods, energy can be saved.

DVFS can be used to eliminate power-wasting idle times by lowering the processor's voltage and frequency during low workload periods so that the processor will have meaningful work at all times leading reduction in the overall power consumption. The energy consumed by GPU is given by the following equation [3] [8]:

$$E = CV^2 * f \tag{1}$$

where,

E = Energy consumed by GPU Measured in joules (J);

C = Capacitance;

V = Voltage supply to GPU;

f = Clock frequency of GPU.

Thus, the power consumed by a task may be decreased by reducing V or F, or both. However, for tasks that require a fixed 5 amount of work, reducing the frequency may simply take more time to complete the work. As a result, little or no energy will be saved. Therefore, intelligent DFVS techniques are required to improve the energy efficiency of GPUs. Many techniques are used to control power consumption by controlling the frequency, since processor frequency has a strong effect on power consumption and temperature. Dynamic voltage and frequency scaling (DVFS) are the most commonly used techniques in modern processors [9].

This section describes various DVFS techniques explored by the researchers exclusively for GPUs. We found that DVFS not only work solely but also work in coordination with other techniques like workload divisions/task division techniques to give the best result. This section categorized the DVFS techniques in the following heads:

A. Schemes using core DVFS technique

B. Schemes using DVFS with other GPU optimization (Hybrid DVFS)

In Section 3.1, detailed description of energy saving methods with only DVFS technique is presented. In Section 3.2, those methods which not only used DVFS but also used some other optimization such as task mapping, workload division, load balancing etc. in coordinated manner are discussed.

3.1. Schemes Using Core DVFS Technique

Intelligent use of DVFS technique may reduce energy consumption of GPU's energy demand. It is a challenging task for DVFS to save energy while preserving the performance [8]. The schemes presented in [10]-[14] apply only DVFS technique for conserving energy on various benchmarks. All of them managed to save energy while maintaining the performance aspects. A triple domain DVFS scheme is proposed in [11] for graphical processor where the frequency and voltage of RISC, geometry processor (GP) and rendering engine (RE) are independently managed. This multi-domain power management scheme used three power management unit that is fully integrated on chip to apply DVFS on graphical processing unit (GPU).With this triple domain power scheme in [11], authors managed to save power up to 65%. GPU only consume 52.4 mW at runtime benchmarks test in contrast to 154 mW without power management techniques. In [8], performance and power of various application kernels under varying frequency settings are characterized. In order to conduct research, the authors choose three computationally diverse applications namely:

1. Compute Intensive Application
2. Memory Intensive Application
3. Hybrid Application

The authors identified these three classes of kernel on the basis of two metrics proposed in [15]:

1. Rate of instruction issues
2. Ratio of Global memory transaction to Computation Instruction

On the basis of above ration, following application kernel belonging to above three categories are identified. The kernel categories and application kernels are shown in **Table 1**.

In [8], the number of utilized GPU cores is not varied because of the restrictions of GPUs in "suspending" unused GPU cores so as to utilize lesser power causing inconsistent results. At the end of study [8], the authors concluded that performance and power consumption of GPU are largely determined by two characteristics: the rate of issuing instructions and the ration of global memory transactions to computation instructions. The vital goal of research work in [8] is to investigate the power and performance, and power consumption of typical GPU application kernel under different memory and core frequency.

In [12], the authors improve throughput of GPUs by adjusting the number of operating cores and voltage/ frequency of cores and/or on chip interconnects/cached for different application under the power constraint environment. Even they further improve throughput by dynamically scaling the number of core and voltage/frequency of both core and on-chip caches at runtime. The objective of [12] is to improve the throughput only by keeping power consumption constant and the results are shown in **Table 2** on the basis of experiments conducted on GPGPU-SIM [16] simulator.

On average, [12] achieved 20% improvement in power constraint environment. In addition to core/memory frequency, the method proposed in [12] also varies the number of active core. Instead of adding one more parameter, the proposed method does not have robust runtime mechanism to deal with all scenarios. However, the benefits of implementing DVFS on GPU without describing the detailed process of the runtime system [17] are shown. The scheme do not focus on memory side DVFS at all as DVFS can be useful for embedded GPUs.

A power management approach is presented in [10] that takes a unified view of the CPU-GPU DVFS to reduce the power consumption of latest 3D mobile games on android platform as compared to independent CPU-GPU DVFS based power management approach. The main objective of scheme is to provide expected frame per second for games while reducing the power consumption. Besides Asphalt 7, high end android games like Anomaly 2, Call of Duty, Need for Speed Most Wanted, Final Strike, Real Football 2013 and AVP are used. In [10], the authors examined that increasing the GPU frequency has no impact on the frame per second. Instead, they employ the concept of CPU COST and GPU COST [18] which is given by:

Table 1. Kernel categories and application kernels.

S. No	Kernel Category	Kernel
1	Compute intensive	Dense matrix multiplication
2	Memory intensive	Dense matrix transpose
3	Hybrid	Fast Fourier transform

Table 2. Throughput improvement under various uses case and power constraints.

Method Adopted	Throughput Improvement	Power Constraint
Appropriately choosing the number of operating cores and their voltage/frequency for a given application.	29%	No Power Constraint
Changing the number of operating cores and the voltage/frequency of on-chip interconnects/caches for a given application	13%	No Power Constraint
Vary the number of operating cores and the voltages/frequencies of both cores and on-chip interconnects/caches.	10%	Power Constraint
Vary the number of operating cores and the voltages/frequencies of both cores and on-chip interconnects/caches every 20 μs within the power constraint	38%	Power Constraint

$$\text{Cpu cost} = \text{Cpu utilization} * \text{frequency} \tag{2}$$

$$\text{Gpu cost} = \text{Gpu utilization} * \text{frequency} \tag{3}$$

Proposed integrated approach is able to reduce power consumption of 3-D games by up to 26% for comparable frame per second range.

A broad study of GPU DVFS conducted on 37 benchmark kernel is presented in [13]. The scheme not only increase performance by 4% but also conserves energy by 19.28% and it is shown that frequency scaling is effective approach to save the energy. By scaling down the core frequency, run time energy can also be saved. The scheme is compared with performance matrices obtained from default setting.

Matrices $\hat{R}$ and $R_{\max}$ are used to evaluate energy conservation in [13].

$$\hat{R} = 1 - E_{\min} \big/ \hat{E} \tag{4}$$

$\hat{E}$ = Energy consumption at default GPU Configuration.

$E_{\min}$ and $E_{\max}$ are the minimum and maximum energy consumption under different voltage/frequency setting for a given application.

$$R_{\max} = 1 - E_{\min} \big/ E_{\max} \tag{5}$$

Core scaling and memory scaling does not work well for every application kernel and some application kernel gives best result at default frequency settings.

Ge *et al.* [14] applied frequency scaling on both CPU and GPU with three typical parallel applications. They found that scaling GPU frequency higher would not consume more energy [14]. To investigate the impact of DVFS, they use following four classes of the performance metrics:

1. Performance
2. Power
3. Energy
4. Energy Efficiency

Experiments are performed on Tesla K20 series GPU form the family of Keplers architecture that support power management and power accounting features. The scheme presented in [14] not only concentrate on GPU energy but also focus on system lever energy and concluded that GPU DVFS affect system energy less as compared to CPU DVFS.

3.2. Schemes Using DVFS with Other GPU Optimization (Hybrid DVFS)

DVFS shown in the previous section suffers from major energy/performance trade-off issues. If not intelligently selected, it may affect performance/energy or both. Therefore, researchers combine DVFS with some other optimization techniques to further improve the performance as well as energy efficiency of running kernel. Over the years, considerable work has been proposed in [9] [17] [19]-[22] to improve the energy efficiency by adding some more optimization approaches to DVFS.

Liu *et al.* [19] proposed power aware time sensitive mapping technique for heterogeneous system that are able to meet application timing requirement while reducing power consumption of applying DVFS on both CPU and GPU. The scheme is executed in three phases as shown in **Figure 2**.

Assignment phase is responsible to assign application to processor like GPU/CPU. Thereafter, Load Balancing phase will manage the Load among CPU and GPU. Finally, DVFS phase scale the frequency as per requirement while meeting all the deadlines. Assignment phase calculate the heterogeneous ratio for each of application to take the assignment decision. Heterogeneous ratio is given by H_i

$$H_i = \max\left\{\frac{e_i^c}{e_i^g}, \frac{e_i^g}{e_i^c}\right\} \tag{6}$$

where e_i^c is the worst case execution time of i^{th} workload on CPU under maximum voltage and e_i^g is worst case execution time of i^{th} workload on GPU under maximum voltage.

If $\frac{e_i^c}{e_i^g} > 1$, applications are more suitable to run on GPU than on CPU otherwise CPU will be more suitable.

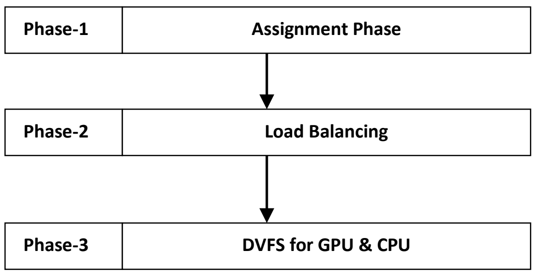

Figure 2. Different phases of equalizer.

With the proposed method in [19], Liu *et al.* managed to save energy more than 20%. Although they develop algorithm for time sensitive applications like data analysis, stock trading, real time scoring of bank transaction, live video processing etc., but does not experiment with them. Behavior of proposed method should be investigate with memory/core bounded applications.

There has been lot of work done on saving energy consumption of either CPU or GPU but, the work in isolated manner cannot achieve maximum performance. Ma *et al.* [20] proposed an energy-management framework known as GreenGPU for GPU-CPU heterogeneous architecture. The framework presented 2-Tier design framework for saving energy as shown in **Figure 3**. As an example, the workload share of CPU and GPU may be 15% and 85% respectively. Tier-1 ensure load balancing which avoid the energy-waste due to idling.

Tier-2 adjusted the frequency of GPU cores and memory is adjusted along with the frequency and voltage of the CPU to achieve largest possible energy savings with marginal performance degradation. However, [20] use DVFS and workload division individually, and so their method cannot set optimal parameters of DVFS and task mapping [22], otherwise existing marginal performance degradation can be improved. An efficient power capping technique through coordinating task mapping and DVFS in a CPU-GPU heterogeneous system is proposed in [22]. The proposed empirical model predict execution time and power consumption of heterogeneous system in order to avoid power violation and load imbalance between the CPU and GPU. The scheme was using benchmark application form Rodinia and form BLAS library under the power constrained of 200 w, 220 w, 240 w, 260 w, 280 w. Their proposed Power model is represented by

$$p_{node} = p_{idle}\left(f_{cpu}, f_{gpu}\right) + p_{cpu}\left(f_{cpu}, f_{gpu}, r_{cpu}\right) + p_{gpu}\left(f_{cpu}, f_{gpu}, r_{gpu}\right) \tag{7}$$

where p_{idle} is represent idle power consumption which is dependent on frequency of CPU (f_{cpu}) and GPU (f_{gpu}) but independent of task mapping function. On the other hand, power consumption of CPU (p_{cpu}) and GPU (p_{gpu}) is dependent on the percentage of task mapping along with the frequencies. The proposed power capping techniques can achieve more than 93% of performance as compared to the ideal one. In [17], Sethia *et al.* proposed a runtime system known as Equalizer that provides adaptive approach so that the hardware will best match the needs of running kernel. Equalizer was designed to work on two modes:

1. Energy efficiency mode
2. High performance mode

Working of Equalizer can easily understand with the help of **Figure 4**. In energy efficient mode, equalizer throttled the frequency of underutilized resources. No performance degradation reported, in fact it saves 15% energy by improving the performance by 5%. On the other hand, in high performance mode only bottleneck resource is boosted by scaling up the frequency to provide higher performance.

On the cost of 6% extra energy consumption, this mode achieves 22% performance improvement. To achieve this improvement in both the modes, equalizer tunes three major architectural parameters: No of Concurrent Thread, Core Frequency, and Memory Frequency according to the mode selected. As per the requirement, Equalizer tunes these three parameters. Wang and Nagarajan [9] proposed a feedback controlling algorithm, known as Proportional integral derivative dynamic frequency scaling (PIDDFS) to scale the core and memory frequencies for GPU architecture. PIDDFS minimized the energy consumption for the memory intensive applications. This technique basically targeted to memory bound application and can be able to save more power if application is memory bounded. The reason is: when memory access intensity is higher, low frequency period maintained by PIDDFS will be more and power saving during that time will be more. The technique was simulated

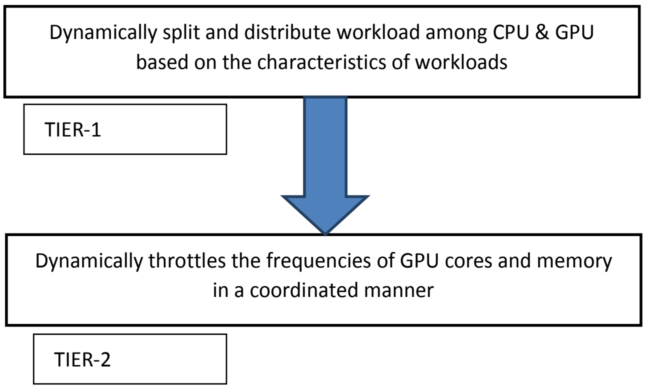
Dynamically split and distribute workload among CPU & GPU based on the characteristics of workloads
TIER-1
Dynamically throttles the frequencies of GPU cores and memory in a coordinated manner
TIER-2

Figure 3. Green GPU two tier design architecture [20].

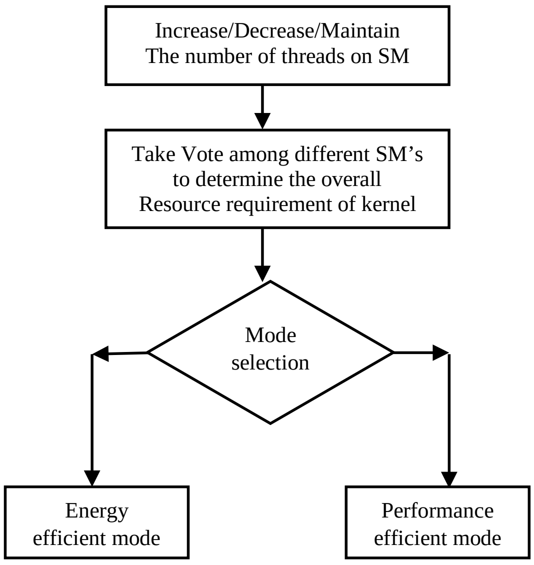
Increase/Decrease/Maintain The number of threads on SM
Take Vote among different SM's to determine the overall Resource requirement of kernel
Mode selection
Energy efficient mode
Performance efficient mode

Figure 4. Flow chart of equalizer [17].

with GPGPU-SIM [16] in coordination with power model GPUWattch [23]. Simulation results show that application gain 23% on average power saving with performance improvement of average 4% for all benchmarks. Although proposed method is less complicated and has cross platform adaptability but give best result only for the memory bound applications. The authors compared PID based approach with only CPU DVFS rather than GPU DVFS.

Since the authors in this field usually applied DVFS on single GPU, Ren *et al.* [21] proposed a method that apply DVFS on CPU and load balancing on multiple GPUs. In this scheme, they identify that the power consumption behavior of the application is highly dependent on the underlying design of the algorithm. The scheme used the algorithmic level power model to predict the execution time and power related parameters. The method converts instruction mixture information, pipelining structure and out of order processing in SIMD flow, so that it can be measured in optimum accuracy. This is the only document that used the CPU DVFS technique with load balancing in multiple GPU and successfully saved the 4.4% energy. The scheme used CPU DVFS only for improving the executing time and load balancing technique to improve the energy consumption of applications. Sufficient overhead was saved by avoiding the GPU DVFS. The authors created three scenarios depicted in **Table 3** and test their proposed algorithm model.

The scheme demonstrated that intelligent use of GPU parallelization, CPU frequency scaling and power load scheduling methods will improve the performance of application while reducing the energy consumption of processing elements in multiple GPU platforms. Wu *et al.* [24] proposed a machine learning based power estimation model that learn itself to scale application according to different hardware configuration. The ultimate objective of their research is to predict the power and performance of GPUs across a range of hardware configuration. The machine learning algorithm requires training data set which is formed by varying the number of computing core, frequency of core and memory. Although DVFS not directly used but varying the frequency of core and memory are used to get the performance metrics. Total 448 training sets are acquired by varying the range of eight computing unit (4, 8, …, 32) eight core frequencies (300, 400, …, 100 MHz) and eight memory frequencies(475, 625, …, 1375 MHz).

4. Comparative Study

DVFS can be used either in isolated manner or in coordination with some other techniques. As shown in **Table 4**, energy/performance improvement depends on type of application kernel and platform chosen. DVFS can be

Table 3. Measurement result under three power aware CPU-GPU configuration [21].

	Load Balancing	DVFS (CPU ONLY)	Execution Time Improvement	Energy Improvement
CPU + GPU	NO	YES	YES	NO
CPU + MULTIPLE GPU	NO	YES	YES	NO
CPU + MULTIPLE GPU	YES	YES	YES	YES

Table 4. Comparative analysis.

Author	Technique Used	No of Benchmark Used	Benchmark Or Application Kernel	Energy Improvement	Performance Improvement	Platform for Parallel Implementation
			Core DVFS			
Lee *et al.* [11]	DVFS	Not specified	Not specified	65%	Not specified	Not specified
Jiao *et al.* [8]	DVFS	3	• Dense matrix multiplication • Dense matrix transpose • Fast Fourier transform	4%	Not specified	NVidiaGTX-280
Lee *et al.* [12]	DVFS	39	• GPGPU-Sim • Rodinia • ERCBench	Power constraint	20%	GPGPU-Sim (Simulate Quadro FX 5800)
Mei *et al.* [13]	DVFS	37	• CUDA SDK 4.1 • Rodinia	19.28%	4	NVIDIA GeForce GTX 560 Ti
Ge *et al.* [14] K20c	DVFS	1	• Matrix multiplication • Traveling salesman problem • Finite state machine	Not specified	Not specified	NVIDIA Tesla K20c
			Hybrid DVFS			
Liu *et al.* [19]	DVFS with Load Balancing	4	• AMD OPENCL Sdk • IBM	20%	Performance constraint	AMD Radeon HD 5770
Ma *et al.* [20]	DVFS with Task Mapping	9	• Rodinia	21.04%	Marginal performance degradation	NVIDIA GeForce 8800 GTX GPU
Komoda *et al.* [22]	DVFS with Task Mapping	25	• Rodinia • BLAS Library	Power constraint	93%	NVIDIA Tesla K20c
Sethia and Mahlke [17]	DVFS with Vary No of Thread	27	• Rodinia • Parboil	15% (Energy efficiency mode)	20% (Performance mode)	GPGPU-Sim (Simulate GTX480)
Wang & Nagarajan [9]	DVFS with PID	12	• CUDA Sdk	23%	4%	GPGPU-Sim (Simulate GTX480)

designed to improve either performance [19] or energy [12] or both [9] [12] [13] [17] [19]. A variation in implementation platform is observed although NVidia is the favorite for researchers [14]. Presented a deep study on advance GPU K20c by varying the frequency of CPU and GPU and observe the performance/energy efficiency improvement. It is observed that applying CPU DVFS and Load balancing in multiple GPU can also improve the energy and performance efficiency [21].

5. Conclusion

In this paper, survey and analysis of several DVFS techniques aimed at analyzing and improving the energy efficiency of GPUs are presented. The key emphasis is on the need of power management in GPUs and identification of important trends in DVFS which are admirable for future study. In our study, we classify the research on DVFS into schemes using core DVFS technique and schemes using DVFS with other GPU optimization (Hybrid DVFS) and highlight the underlying similarities and differences between them. Energy efficiency and performance variation of applications running on GPU are presented in this paper such that breakthrough invention of designing Green GPUs for further research can be accomplished. In future, DVFS can be pooled with other techniques such that energy saving in an optimized way can be attained and electric bill as well as carbon footprint of IT infrastructure can be reduced.

References

[1] TOP500 Supercomputing Sites. http://www.top500.org/

[2] The Green500 List. http://www.green500.org/lists/2010/11/top/list.php

[3] Mittal, S. and Vetter, J.S. (2014) A Survey of Methods for Analyzing and Improving GPU Energy Efficiency. *ACM Computing Surveys*, **47**, 1-23. http://dx.doi.org/10.1145/2636342

[4] Hong, S. and Kim, H. (2010) An Integrated GPU Power and Performance Model. *ACM SIGARCH Computer Architecture News*, **38**, 280. http://dx.doi.org/10.1145/1816038.1815998

[5] Cebri'n, J.M., Guerrero, G.D. and Garcia, J.M. (2012) Energy Efficiency Analysis of GPUs. 2012 *IEEE* 26*th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, Shanghai, 21-25 May 2012, 1014-1022. http://dx.doi.org/10.1109/ipdpsw.2012.124

[6] Dave, A., Dykes, J. and Riedle, E. (2003) More than an Interface-SCSI vs. ATA. *FAST*'03 *Proceedings of the* 2*nd USENIX Conference on File and Storage Technologies*, 2003, 245-257.

[7] Hsu, C.-H. and Kremer, U. (2002) Compiler-Directed Dynamic Voltage Scaling for Memory-Bound Applications. In: Hsu, C.-H. and Kremer, U., *Compiler-Directed Dynamic Voltage Scaling for Memory-Bound Applications*, Technical Report DCS-TR-498, Department of Computer Science, Rutgers University, New Brunswick/Piscataway, Camden and Newark.

[8] Jiao, Y., Lin, H., Balaji, P. and Feng, W. (2010) Power and Performance Characterization of Computational Kernels on the GPU. *2010 IEEE/ACM International Conference on & In Conference on Cyber, Physical and Social Computing* (*CPSCom*) *Green Computing and Communications* (*GreenCom*), Hangzhou, 18-20 December 2010, 221-228. http://dx.doi.org/10.1109/greencom-cpscom.2010.143

[9] Wang, Y. and Ranganathan, N. (2014) A Feedback, Runtime Technique for Scaling the Frequency in GPU Architectures. 2014 *IEEE Computer Society Annual Symposium on VLSI*, Tampapp, 9-11 July 2014, 430-435. http://dx.doi.org/10.1109/isvlsi.2014.34

[10] Pathania, A., Jiao, Q., Prakash, A. and Mitra, T. (2014) Integrated CPU-GPU Power Management for 3D Mobile Games. *Proceedings of the the* 51*st Annual Design Automation Conference on Design Automation Conference*, 2014, 1-6. http://dx.doi.org/10.1145/2593069.2593151

[11] Lee, J., Nam, B.-G. and Yoo, H.-J. (2007) Dynamic Voltage and Frequency Scaling (DVFS) Scheme for Multi-Domains Power Management. 2007 *IEEE Asian Solid-State Circuits Conference*, 12-14 November 2007, Jeju, 360-363.

[12] Lee, J., Sathisha, V., Schulte, M., Compton, K. and Kim, N.S. (2011) Improving Throughput of Power-Constrained GPUs Using Dynamic Voltage/Frequency and Core Scaling. 2011 *International Conference on Parallel Architectures and Compilation Techniques*, *PACT*, Galveston, 10-14 October 2011, 111-120. http://dx.doi.org/10.1109/pact.2011.17

[13] Mei, X., Yung, L.S., Zhao, K. and Chu, X. (2013) A Measurement Study of GPU DVFS on Energy Conservation. *Proceedings of the Workshop on Power-Aware Computing and Systems*, *HotPower* '13, Farmington, 3-6 November 2013, Article No. 10. http://dx.doi.org/10.1145/2525526.2525852

[14] Ge, R., Vogt, R., Majumder, J., Alam, A., Burtscher, M. and Zong, Z. (2013) Effects of Dynamic Voltage and Frequency Scaling on a K20 GPU. 2013 42*nd International Conference on Parallel Processing*, Lyon, 1-4 October 2013,

826-833. http://dx.doi.org/10.1109/ICPP.2013.98

[15] Ryoo, S., Rodrigues, C.I., Baghsorkhi, S.S., Stone, S.S., Kirk, D.B. and Hwu, W.W. (2008) Optimization Principles and Application Performance Evaluation of a Multithreaded GPU Using CUDA. *Proceedings of the* 13*th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, *PPoPP* '08, Salt Lake City, 20-23 February 2008, 73-82. http://dx.doi.org/10.1145/1345206.1345220

[16] Bakhoda, A., Yuan, G.L., Fung, W.W.L., Wong, H. and Aamodt, T.M. (2009) Analyzing CUDA Workloads Using a Detailed GPU Simulator. 2009 *IEEE International Symposium on Performance Analysis of Systems and Software*, Boston, 26-28 April 2009, 163-174. http://dx.doi.org/10.1109/ISPASS.2009.4919648

[17] Sethia, A. and Mahlke, S. (2014) Equalizer: Dynamic Tuning of GPU Resources for Efficient Execution. 2014 47*th Annual IEEE/ACM International Symposium on Microarchitecture*, Cambridge, 13-17 December 2014, 647-658. http://dx.doi.org/10.1109/MICRO.2014.16

[18] Bai, Y. and Vaidya, P. (2009) Memory Characterization to Analyze and Predict Multimedia Performance and Power in Embedded Systems. 2009 *IEEE International Conference on Acoustics*, *Speech and Signal Processing*, Taipei, 19-24 April 2009, 1321-1324. http://dx.doi.org/10.1109/ICASSP.2009.4959835

[19] Liu, C., Li, J., Huang, W., Rubio, J., Speight, E. and Lin, X. (2012) Power-Efficient Time-Sensitive Mapping in Heterogeneous Systems. *Proceedings of the* 21*st International Conference on Parallel Architectures and Compilation Techniques*, *PACT* '12, Minneapolis, 19-23 September 2012, 23-32.

[20] Ma, K., Li, X., Chen, W., Zhang, C. and Wang, X. (2012) GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU Heterogeneous Architectures. 2012 41*st International Conference on Parallel Processing*, Pittsburgh, 10-13 September 2012, 48-57. http://dx.doi.org/10.1109/icpp.2012.31

[21] Ren, D.Q., Bracken, E., Polstyanko, S., Lambert, N., Suda, R. and Giannacopulos, D.D. (2012) Power Aware Parallel 3-D Finite Element Mesh Refinement Performance Modeling and Analysis with CUDA/MPI on GPU and Multi-Core Architecture. *IEEE Transactions on Magnetics*, 48, 335-338. http://dx.doi.org/10.1109/TMAG.2011.2177814

[22] Komoda, T., Hayashi, S., Nakada, T., Miwa, S. and Nakamura, H. (2013) Power Capping of CPU-GPU Heterogeneous Systems through Coordinating DVFS and Task Mapping. 2013 *IEEE* 31*st International Conference on Computer Design* (*ICCD*), Asheville, 6-9 October 2013, 349-356. http://dx.doi.org/10.1109/ICCD.2013.6657064

[23] Leng, J., Hetherington, T., Tantawy, A.E., Gilani, S., Kim, N.S., Aamodt, T.M. and Reddi, V.J. (2013) GPUWattch: Enabling Energy Optimizations in GPGPUs. *Proceedings of the* 40*th Annual International Symposium on Computer Architecture—ISCA*'13, New York, 2013, 487.

[24] Wu, G., Greathouse, J.L., Lyashevsky, A., Jayasena, N. and Chiou, D. (2015) GPGPU Performance and Power Estimation Using Machine Learning. 2015 *IEEE* 21*st International Symposium on High Performance Computer Architecture* (*HPCA*), Burlingame, 2015, 564-576.