

A mechanism to reduce energy waste in the post-execution of GPU applications

Emmanuel D. Carreño, Adiel S. Sarates Jr. and Philippe O. A. Navaux

Informatics Institute – Federal University of Rio Grande do Sul (UFRGS)
P.O Box 15.064 – 91501-970 – Porto Alegre – RS – Brazil

E-mail: {edcarreno, assaratesj, navaux}@inf.ufrgs.br

Abstract. With the increasing demand of GPU accelerators for general purpose in HPC, the impact of energy consumption of these resources cannot be overlooked. To reduce the power consumption some strategies have been applied, but their approaches have been mostly focused on power savings during the application execution. This work focuses on post-execution energy savings. When the post-execution behavior is analyzed in newer GPU cards, it is observed that the power draw does not return to the idle state in an efficient way, creating an unexpected power waste. To overcome this inefficient return to idle and power draw waste in the post-execution, we developed a strategy to reduce the energy consumption considering a minimal impact on global performance. Using this strategy, we achieved energy savings up to 73 percent in the post-execution phase of a single run of a GPU application. In the case of sequential runs, the energy saving percentage depends of the waiting time gap between executions.

1. Introduction

In the last few years, the usage of Graphics Processing Units (GPU) as accelerators for computing have become an important technology for High-Performance Computing (HPC).

A recent survey [1] of the top 500 supercomputers shows that eight percent of them are heterogeneous systems based on GPU cards used for General Purpose (GPGPUs). Each GPU consists of thousands of highly specialized processing cores, during computation, the CPUs offloads some compute-intensive, highly parallelizable code segments to these GPUs for execution acceleration.

The pioneer GPUs were developed for computer graphics that require high computational throughput capability, but actually they also support the execution of general purpose applications. The Compute Unified Device Architecture (CUDA) from Nvidia have been developed to support such applications on GPUs [2]. With the CUDA development, programming GPUs becomes simpler, being no longer necessary to use libraries like OpenGL.

However, the electricity costs of operating HPC systems and data centers have increased by 36 percent in the period of 2005 to 2010 (from seven gigawatts to almost ten gigawatts total in the United States), according to a 2011 report on recent electricity growth within data centers [3]. This growth was actually dampened by the global recession, it had been forecasted by the Environmental Protection Agency of the United States to double in 2016 as its Department of Energy enters the *exascale* era of supercomputing [4].



The emerging green-thinking that claims for HPC systems and programs more energetic efficiently is nowadays more present in HPC laboratories. Unfortunately, even with those efforts aiming green computing, a large part of research about the applications' energy consumption is focused mostly on application performance. To achieve the exascale era, reducing the power consumption, saving more energy and, as a consequence saving money, we need to look at the application *post-execution* behavior too. When an application finishes, what happens after that point is also important, as it can be wasting valuable joules.

There is a lot of effort to save energy during the application execution, some techniques are very helpful and can be adapted to reduce the consumption where is it possible yet. Improving energy efficiency is a continuous challenge in HPC because of the constant need for performance despite the constraints in the power budget and economic cost. As a result, it is necessary to invest in power-aware technologies on GPU-based heterogeneous systems.

The main motivation of this work is the fact that when a GPU application ends, its power consumption does not return as fast as possible to its lowest power draw. Our contribution is a tested mechanism to reduce the time that a GPU takes to return to its idle state, saving energy and money in the process without impacting application performance.

2. Related Work

The power consumption for GPUs have been studied using experimental methods where power is measured directly on the whole system, also using energy models or energy optimization techniques [5, 6, 7, 8].

The Dynamic Voltage Frequency Scale (DVFS) is a technique used to decrease the frequency-voltage dynamically, allowing a lower power draw. Ge et al. [9] developed a mechanism using DVFS for GPUs. However, they found an undesirable side effect of this technique, it causes a potential loss in performance when they lower the clock frequency by creating delays in the application execution. They conclude that there is a careful balance that must be struck between lowered power draw and loss of performance in order to achieve a reduced energy consumption.

The work of Lee et al. [10], although includes CPU usage, shows how to improve the throughput of power-constrained GPUs. They optimized the number of operating cores, the voltages and frequencies of caches, cores and on-chip interconnects depending on applications' characteristics. Using their method they analyzed the impact of the number of operating cores and their voltage and frequency on GPU improving the throughput nearly 20% on average.

The green-thinking brings the economic aspect related to the energy costs. To help reducing these costs, projects like Green Queue were developed. Green Queue is based on a series of application-aware Dynamic Voltage-Frequency Scaling (DVFS) techniques for use within MPI applications [11]. This scalable scheduler was deployed at the San Diego Supercomputer Center. Results of using this technique on their machines allow them to save as much as 31.7% of the operational energy bill.

In order to reduce energy waste, either for economic reasons or to reach the exascale era, techniques like frequency scaling on GPU cores and memory and workload division between CPU and GPU are used. Frameworks to control this scaling have been developed and GPGPU's are analyzed for HPC purposes. Benchmarks, algebra libraries or actual applications have been used to validate the new ideas and concepts of Green Computing [12, 13, 14]. Recently, high-end GPUs that support frequency scaling and power monitoring, such as Nvidia's K20, entered the market. The benefits of GPU frequency scaling appear like an effective approach to save system power. In general, scaling down the GPU core voltage can significantly save energy when working at appropriate core frequencies. Scaling memory frequencies can also save energy for some applications. However, it is not easy to find optimal setting for GPU frequency scaling [15].

3. Resources

The newer GPUs are not concerned only with performance, they also have several hardware counters and information about the power draw and usage. Those metrics and data can be retrieved using appropriate libraries and methods without generating a significant energy waste.

In this section, we discuss how these metrics were obtained by a developed application. Besides, we present some technical background on the Nvidia Tesla K20 and the benchmarks and applications used to evaluate our methodology.

3.1. Nvidia Tesla K20

The Nvidia Tesla K20 is a GPU card based on the Kepler architecture that comprises more than seven billion transistors adding new features focused on computing performance. The Kepler family was designed especially to attract the HPC market. This GPU card has 2496 processor cores working at 706MHz and 5GB of memory operating at 2.6GHz.

This architecture was also designed to improve power efficiency, achieving a better performance per watt versus the previous Nvidia GPU architecture. The execution units run at a high clock rate, allowing the chip to achieve higher throughput with few copies of the execution units [16]. A trade-off of GPU cards is that their power consumption is usually higher than a CPU. The thermal design power for the K20 is 225W, while its idle power is 14W.

3.2. Nvidia NVML

Nvidia offers the Nvidia Management Library (NVML), a C-based API that allows the monitoring and managing of states in a Nvidia GPU device [17]. This library allows low-level access to the features on the GPU card, sensors data and accounting information. Also, NVML allows administrators to query the GPU device state and, with the appropriate privileges, it is possible to modify some of these GPU device states. NVML is compatible with some Tesla and Fermi devices and it has limited support on older Nvidia GPUs.

3.3. Benchmarks

To obtain the post-execution power draw behavior of applications on the K20 GPU, we used a set of applications from two benchmarks commonly used in GPU, MAGMA and Rodinia.

3.3.1. MAGMA

MAGMA is a framework to develop linear algebra applications for hybrid and heterogeneous architectures [18]. Hybrid many core and GPU systems can enable applications to fully exploit the power that each of the hybrid components offers.

MAGMA has a testing suite which comprises single and double precision test for real and complex matrix operations. It is possible to set different input sizes for each instance, that allows a wide range of processing loads and consequently a different power consumption behavior for each test [19, 20].

We used a subset of the available benchmarks of MAGMA. The tests executed with MAGMA used single and double precision, in real and complex domains, performing QR, LQ and triangular factorization operations. The matrix sizes used were 512, 1024, 2048, 4096.

3.3.2. Rodinia

Rodinia is a set of free and open benchmarks. Its applications are designed for heterogeneous computing infrastructures targeting both GPUs and multicore CPUs [21].

The Rodinia benchmarks span a range of parallelism and compute patterns, it provides applications from Linear Algebra problems to Medical Images processing. To perform the tests we used all Rodinia tests that do not require the GPU Texture Area. Each application or kernel represents a different type of behavior called dwarf. A Dwarf is a terminology proposed in Berkeley as the Dwarf taxonomy and was used to classify parallel patterns [22].

4. GPUtool

In order to gather GPU metrics, it is possible to use profilers like the Nvidia Visual Profiler [23]. However, this profiler is unable to obtain data during the post-execution interval.

To overcome this problem, we developed an application (GPUtool) to retrieve and analyze this data even after the end of the execution. GPUtool is a C-based profiling-tool running in the background using the NVML libraries.

This application retrieves the necessary metrics to analyze the energy consumption in *post-execution* from the Tesla K20 counters, without increasing significantly the power draw while running in parallel with the analyzed application.

This tool is executed in CPU collecting relevant values stored in the GPU counters. The collected metrics are, among others, execution time, power draw, GPU memory and SMX-core frequency, fan-speed and temperature. To avoid getting erroneous data about the analyzed application, it was checked that only one application was running on the GPU.

To minimize computing overhead, which could cause some power draw in the GPU, the access time to retrieve the metrics was defined to be slower than 15ms. It was confirmed by us that a faster measuring frequency increased the power draw in the card.

When the GPUtool is used only to collect data, the frequencies are controlled by the GPU included mechanisms at all time. On the other hand, when the tool is responsible for controlling the frequency at post-execution, it sets both frequencies to the lowest possible values right after the GPU application ends.

The main reason to develop a profiling application versus using one already available, was based on the lack of a tool that permits collecting data after the application finished. Available profiling tools did not work without an application context, that means that while there is no application running on the GPU it is not possible to profile and obtain some valuable metrics from the card.

5. Mechanism to reduce energy waste in Post-Execution

When a GPU application finishes, it is expected that the power draw in the GPU card return to its idle state if no other application starts. The idle state is characterized by a low power draw and both memory and core frequencies standing at their lowest values. However in some GPU devices this decrease in the power draw is not instantaneous, taking some seconds to start and some other time before reaching that state to avoid unnecessary energy consumption.

The mechanism to reduce this return to idle time and its subsequently energy waste is to force a frequency scaling when the post-execution phase starts. In this way, both memory and core frequencies are set to their lowest values, resulting in a fast reduction in energy consumption. By applying the frequency scaling at the start of the post-execution phase, both core and memory frequencies are set to the lower states, resulting in a quick decrease of power draw. This process reduces the energy wasting in this non-computing state.

The applications we used to validate our mechanism were executed on the GPU without frequency limitations, allowing the GPU energy management to set the adequate frequency by default. The highest values of the frequencies are 2600MHz for the memory and 706MHz for the GPU cores. The lowest values they both could be set are 324MHz.

This mechanism can be analyzed in two scenarios with a difference at what happens after the post-execution phase starts. The first scenario is when there are no more applications waiting to be executed and the second one when there are consecutive executions of GPU applications. In both scenarios, it was analyzed the post-execution behavior while implementing the proposed mechanism. In the next subsections both scenarios will be detailed.

5.1. Single run scenario

This scenario consists of the execution of a GPU application and its posterior post-execution phase until it reaches the idle state. In this scenario no other application starts during the post-execution phase. This setting captures the behavior of the post-execution energy consumption without the effects of more executions of GPU applications.

Figure 1 shows a timeline with each one of the phases of a GPU application run. In this scenario GPUtool activates the energy saving mechanism at the only post-execution phase.

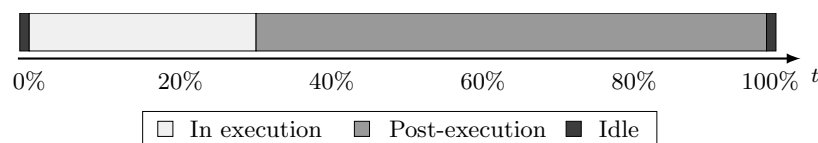


Figure 1. Phases in a single run scenario.

5.2. Multiple run scenario

This scenario consists in the consecutive execution of one GPU application, starting each execution during the post-execution phase of the previous one. This creates a behavior similar to an scheduler, with several applications running in the GPU.

Figure 2 depicts a detailed timeline for each of the phases taken into account in the multiple run scenario. The methodology to save energy is applied only at the post-execution intervals, i.e., between two consecutive executions and at the last post-execution interval until reaching the idle state.

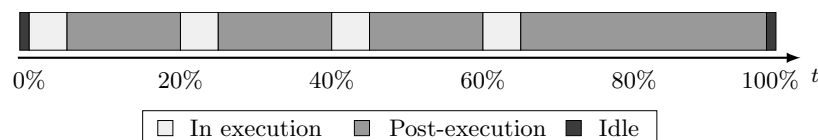


Figure 2. Phases in a multiple run scenario.

The multiple run scenario was considered to illustrate that the proposed mechanism does not generate a significant performance loss in posterior executions. Because GPUTool does not know beforehand how many applications will be executed or the duration of them, the tool needs to be launched before the application execution starts to act when it is necessary.

Since this tool does not produce a significant impact on application performance along the application execution, it is possible to save energy at every moment that there are no applications running on the GPU without compromising performance.

6. Results

All tests were executed on two different machines using four different devices, two Nvidia Tesla K20m and two Nvidia Tesla K20c. In order to discard manufacturing problems with the cards and to compare the results between them. All the results presented here are the arithmetic average of multiple executions.

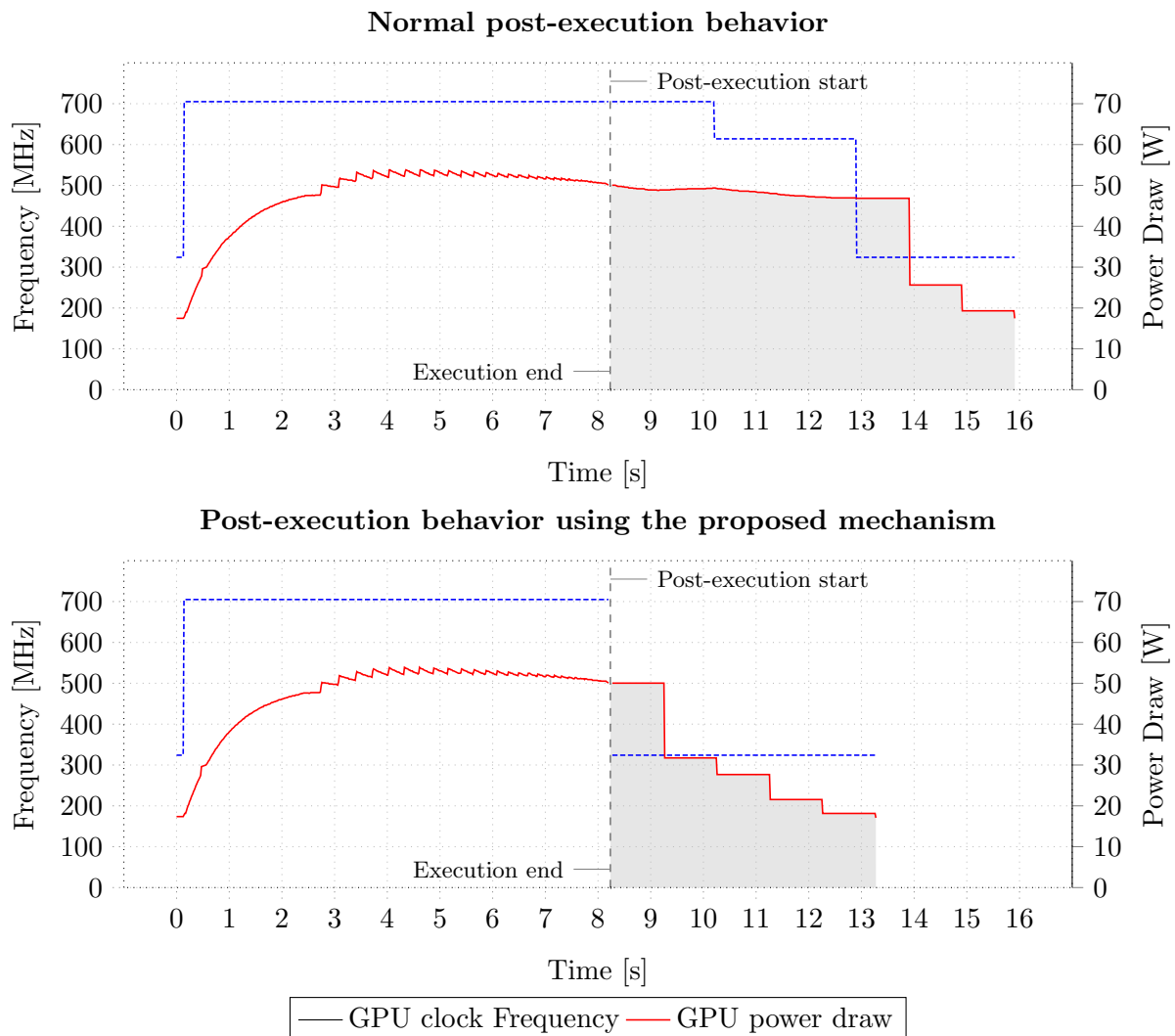


Figure 3. Energy consumption of CGEQRF with input size 4096 in a single run scenario. Note that in the normal post-execution behavior the idle state is reached near 16 seconds, on the other hand, using the proposed mechanism this state is achieved in a shorter time with a lower power draw, minimizing the energy waste. The dark area represents the energy consumption.

A comparison of an execution of the benchmark CGQRF and its power draw with and without our mechanism is depicted in Figure 3. In the Figure, the power draw is the same in both in-execution phases because the mechanism is activated only after the start of the post-execution phase. The vertical dashed line indicates where this phase starts. The area under the GPU power draw curve represents the energy consumption. The dark area allows a visual comparison of the post-execution savings. The power draw stays near its last value before the application ended, returning to idle after some seconds.

Without the proposed mechanism, the GPU card persists in a high frequency during non-computing periods and this high frequency consumes unnecessary power for GPU card. With the mechanism active, both frequencies are returned to their lowest values and the power draw is reduced quickly. Savings in the post-execution of this benchmark were 52.96 percent. As shown in the Figure, the execution time is not altered by the mechanism, allowing energy savings in the post-execution phase of a single run without affecting the application in-execution behavior.

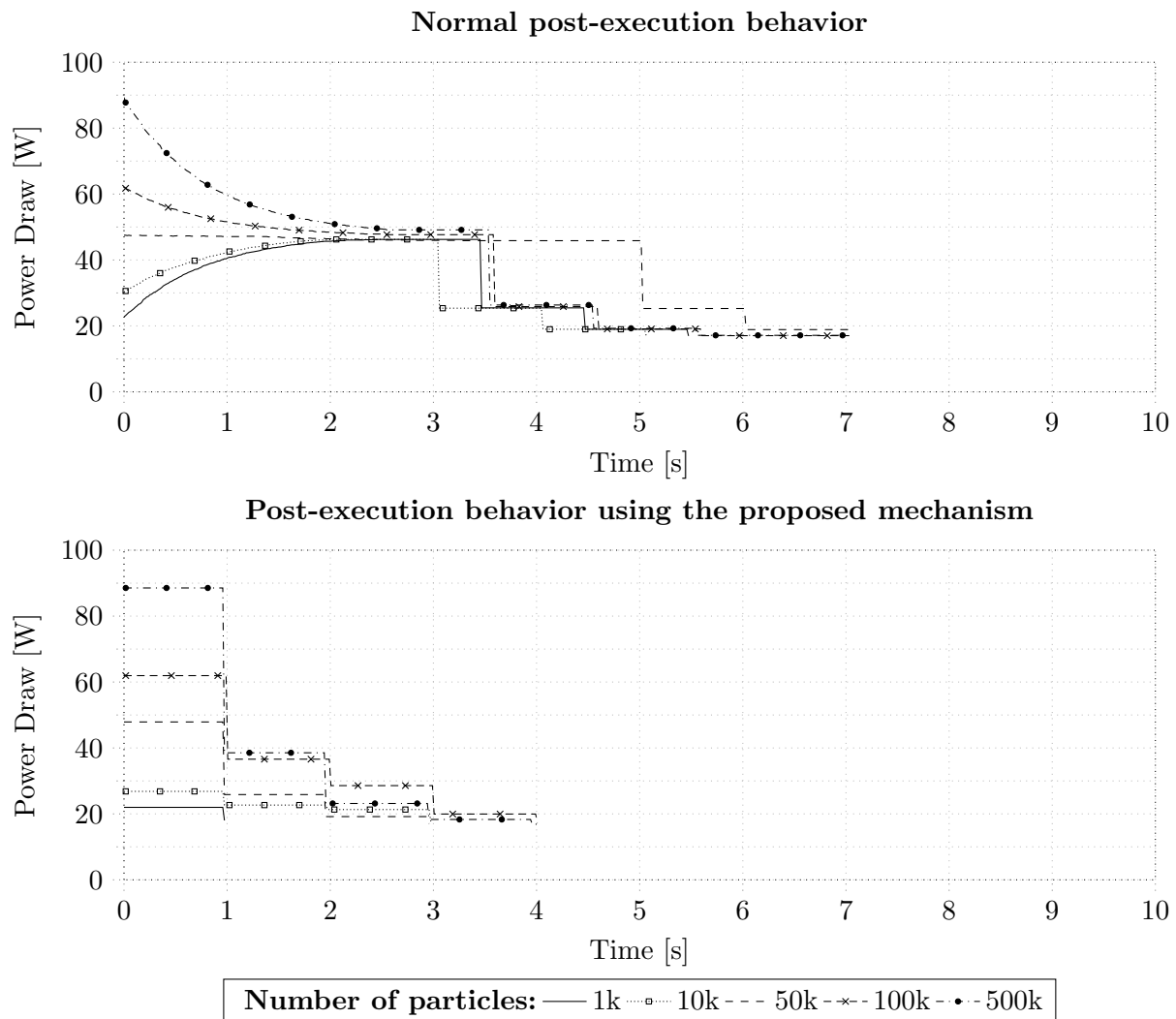


Figure 4. Post-execution energy consumption using different input data size to achieve different power draw behaviors for Particle Filter benchmark. Time zero indicates the start of the post-execution phase.

A more detailed view of the post-execution behavior is showed in Figure 4. The Figure depicts only the post-execution phase of the Particle Filter benchmark using different number of particles as input size. As the figure shows, the time-to-idle period is different based on the input size, it seems to be a high correlation between the power draw at the end of an application execution and the post-execution phase length.

The post execution savings of Rodinia's Particle Filter benchmark, with and without the usage of the mechanism in the same order as the input sizes of Figure 4 are 88.55%, 59.91%, 67.03%, 35.12%, 33.33%. As illustrated in Figure the clock frequency behavior right after the end of the execution impacts on power draw. That means that the time that power draw takes to return to idle and the post-execution savings depends on the power draw at the very end of the execution.

Using the MAGMA benchmarks in all of them we obtained savings during its post-execution phase. Figure 5 shows the normalized power savings of the MAGMA tests. As depicted in the Figure, in some benchmarks our mechanism is able to achieve energy savings up to 73 percent.

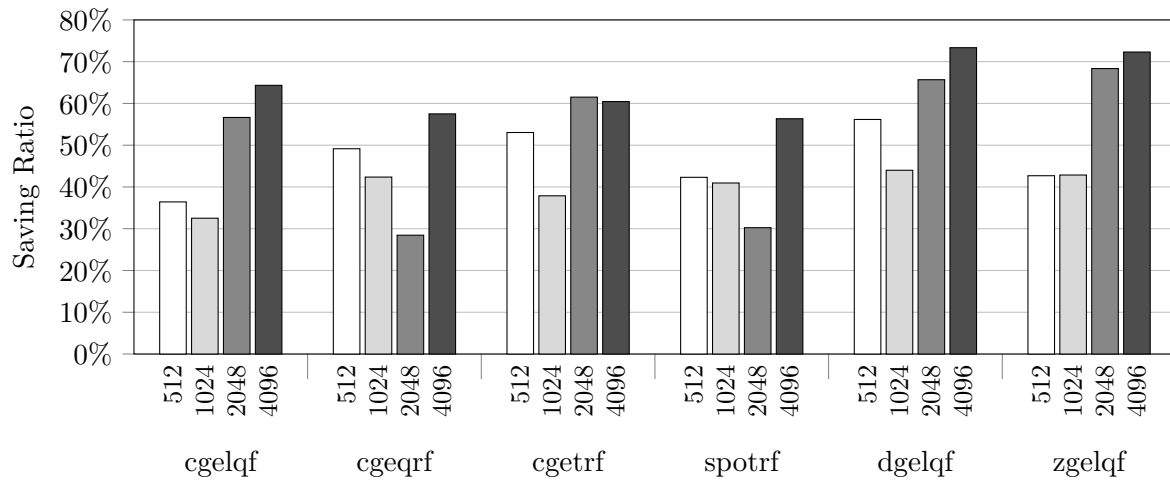


Figure 5. MAGMA benchmarks. Average saving ratio of each benchmark using different input data sizes.

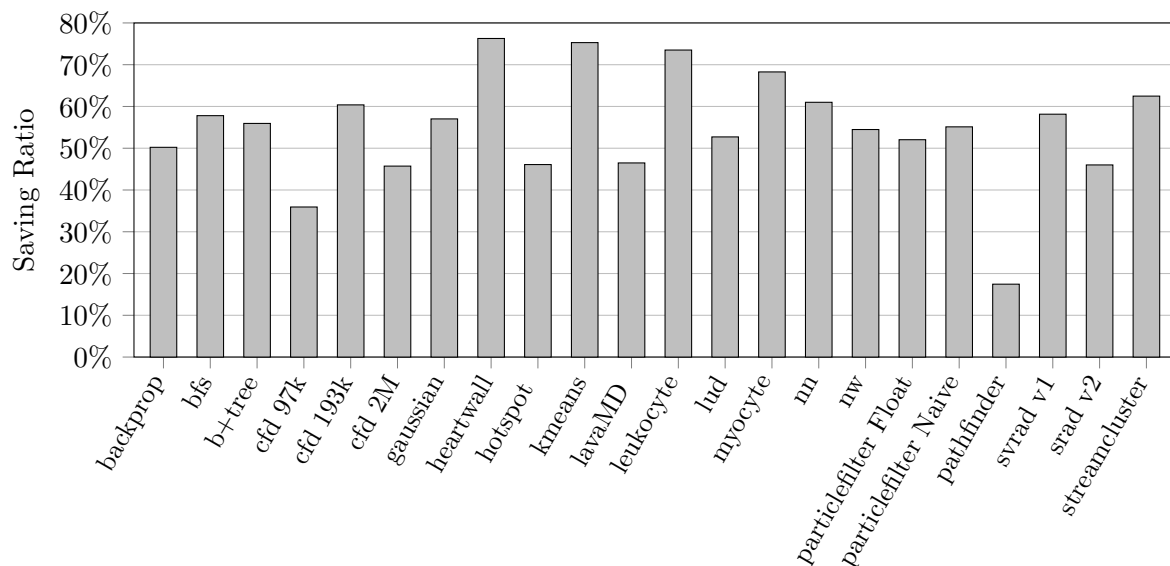


Figure 6. Rodinia benchmarks. Average saving ratio of each benchmark using the default input size.

In relation to the minimum savings, the MAGMA benchmark with the lowest value reached 28.45 percent of energy savings, even this value could be considered as a good result.

The post-execution behavior of this benchmark also allowed us to confirm that applications with higher power draw at the end of execution achieved bigger savings. In all of the experiments using MAGMA benchmarks our mechanism reduced the power consumption during the post-execution phase.

Regarding the Rodinia benchmarks we obtained the following results. Using the Heart Wall Tracking benchmark with its default parameters, our mechanism reduced 65.59 percent the time to idle, saving more than 245J. In the Back Propagation benchmark were achieved energy savings of 50.22 percent representing 94.29J. Saving ratios of the Rodinia benchmarks are depicted in Figure 6. As in the previous set of benchmarks, in all of the cases we were able to reduce power consumption during the post-execution phase.

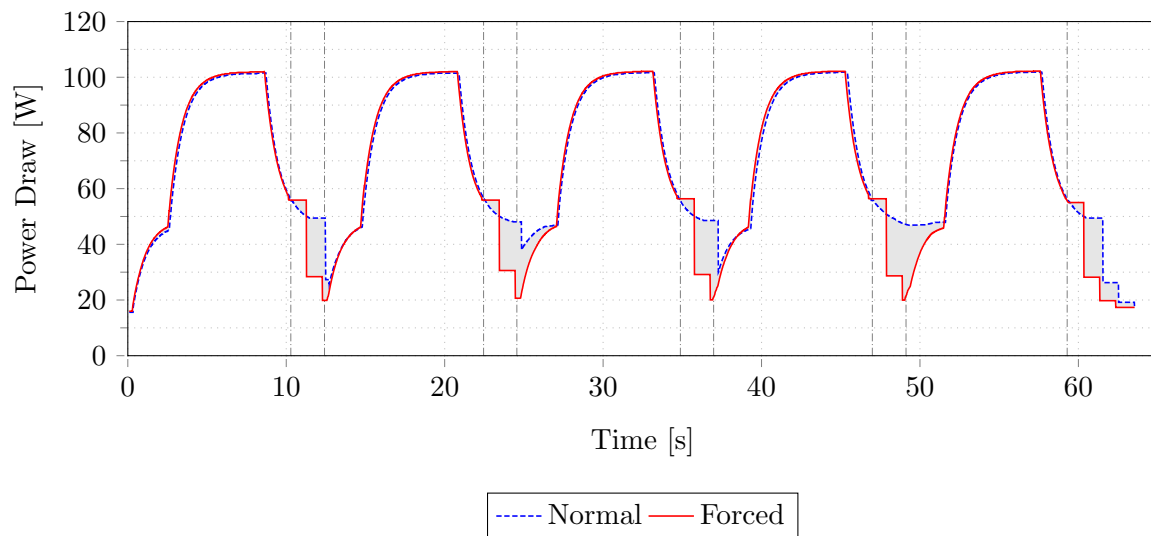


Figure 7. Energy consumption on five executions of the CFD benchmark with 2 million elements. Two seconds of post-executions phase between executions. The final post-execution phase goes until idle state. The dark area between plot lines represents the total energy savings.

As proposed for the second scenario, in order to reduce the power consumption between two runs we limited the waiting time between the consecutive executions of the same GPU application. This scenario is shown in Figure 7 using CFD, one of the Rodinia benchmarks, with an interval of two seconds between five executions. In this benchmark we achieved savings of 118J in the post-execution phase, almost 21 percent. As depicted in the Figure, application performance was not impacted by using our mechanism between the executions. Because the mechanism is activated as soon as the GPU application finishes, if the post-execution runs for a significant amount of time it will allow greater energy savings. However, it is not realistic to wait a lot of time between executions when running short-time GPU applications sequentially.

The results obtained shows that is possible to obtain savings in the two scenarios proposed in our methodology, without impacting performance.

7. Conclusions

In this work we conducted a measurement study of the power consumption savings by analyzing the behavior of the GPU card after a running application finished. We used some benchmarks and variations in their input sizes to generate different post-execution behavior. We demonstrated that power waste exists in GPU cards and that this could be mitigated using frequency scaling. While frequency scaling already is a common strategy to save energy during the application's execution, there are no previous works performed in post-execution energy savings using this approach.

For short running applications, scaling down the GPU and Memory frequency can significantly save energy during the post-execution phase. Using this approach, savings up to 73 percent could be achieved. This mechanism could be implemented directly on the GPU power manager, optimizing the resources usage for every GPU application.

By aiming to reduce the energy consumption of configurable GPUs, our approach can be a starting point to claim the attention for an overlooked area of energy waste, the post-execution consumption. These results could be useful in the road to the exascale era where there are efforts to save every watt as possible for computation.

8. Future Work

In the future, an analysis of the consumption behavior when several applications are executed in parallel could be performed. With this approach it could be possible to reduce power consumption in the execution gaps, that is when there are no kernels running in the device, extending our mechanism to in-execution savings. Also, it may be analyzed the behavior of the applications that require more than one GPU device.

Acknowledgments

The authors would like to thank all the members of the Parallel and Distributed Processing Group (GPPD) at Federal University of Rio Grande do Sul (UFRGS), their help, expertise and advice were of great value.

This research have been partially supported by CNPq under Grant 132123/2013-4, by FAPERGS under grant ME 14/2012 and by the UFRGS Informatics Institute.

References

- [1] Top500.org, "China's Tianhe-2 Supercomputer Maintains Top Spot on 42nd TOP500 List," <http://www.top500.org/blog/lists/2013/11/press-release/>, 2013.
- [2] Nickolls, John and Buck, Ian and Garland, Michael and Skadron, Kevin, "Scalable Parallel Programming with CUDA," *Queue*, vol. 6, no. 2, pp. 40–53, Mar. 2008. [Online]. Available: <http://doi.acm.org/10.1145/1365490.1365500>
- [3] Koomey, Jonathan, "Growth in data center electricity use 2005 to 2010," *Oakland, CA: Analytics Press. August*, vol. 1, p. 2010, 2011.
- [4] Brown, Richard and others, "Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431," *Lawrence Berkeley National Laboratory*, vol. 109, p. 431, 2007.
- [5] Chen, Jianmin and Li, Bin and Zhang, Ying and Peng, Lu and Peir, Jih-kwon, "Statistical GPU power analysis using tree-based methods," in *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 2011, pp. 1–6.
- [6] Kasichayanula, Kiran and Terpstra, Dan and Luszczek, Piotr and Tomov, Stan and Moore, Shirley and Peterson, Gregory D, "Power aware computing on GPUs," in *Application Accelerators in High Performance Computing (SAAHPC), 2012 Symposium on*. IEEE, 2012, pp. 64–73.
- [7] Ukidave, Yash and Kaeli, David R, "Analyzing Optimization Techniques for Power Efficiency on Heterogeneous Platforms," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 1040–1049.
- [8] Ukidave, Y. and Ziabari, AK. and Mistry, P. and Schirner, G. and Kaeli, D., "Quantifying the energy efficiency of FFT on heterogeneous platforms," in *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, April 2013, pp. 235–244.
- [9] Ge, Rong and Vogt, Ryan and Majumder, Jahangir and Alam, Arif and Burtscher, Martin and Zong, Ziliang, "Effects of dynamic voltage and frequency scaling on a k20 gpu," in *2nd International Workshop on Power-aware Algorithms, Systems, and Architectures*, 2013.
- [10] Lee, Jungseob and Sathisha, Vijay and Schulte, Michael and Compton, Katherine and Kim, Nam Sung, "Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling," in *Parallel Architectures and Compilation Techniques (PACT), 2011 International Conference on*. IEEE, 2011, pp. 111–120.
- [11] Ananta Tiwari and Michael Laurenzano and Joshua Peraza and Laura Carrington and Allan Snaveley, "Green Queue: Customized Large-Scale Clock Frequency Scaling," in *Cloud and Green Computing (CGC) , 1-3 November , 2012*, IEEE. Xiangtan, Hunan, China: IEEE, 2012. [Online]. Available: <http://users.sdsc.edu/~lcarring/Papers/2012-CGC.pdf>
- [12] Padoin, Edson Luiz and Pilla, Laércio Lima and Boito, Francieli Zanon and Kassick, Rodrigo Virote and Velho, Pedro and Navaux, Philippe OA, "Evaluating application performance and energy consumption on hybrid CPU+ GPU architecture," *Cluster Computing*, pp. 1–15, 2012.
- [13] Ma, Kai and Li, Xue and Chen, Wei and Zhang, Chi and Wang, Xiaorui, "Greengpu: A holistic approach to energy efficiency in gpu-cpu heterogeneous architectures," in *Parallel Processing (ICPP), 2012 41st International Conference on*. IEEE, 2012, pp. 48–57.

- [14] Huang, Song and Xiao, Shucai and Feng, Wu-chun, "On the energy efficiency of graphics processing units for scientific computing," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 1–8.
- [15] Mei, Xinxin and Yung, Ling Sing and Zhao, Kaiyong and Chu, Xiaowen, "A measurement study of GPU DVFS on energy conservation," in *Proceedings of the Workshop on Power-Aware Computing and Systems*. ACM, 2013, p. 10.
- [16] Nvidia, "Whitepaper Nvidia's Next Generation CUDA Compute Architecture: Kepler GK110," <http://www.nvidia.com/content/PDF/kepler/Nvidia-Kepler-GK110-Architecture-Whitepaper.pdf>, 2012.
- [17] Nvidia, "Nvidia Management Library (NVML)," <https://developer.nvidia.com/nvidia-management-library-nvml>, 2013.
- [18] Haidar, A. and Tomov, S. and Yamazaki, I. and Solca, R. and Schulthess, T. and Dong, T. and Dongarra, J., "MAGMA: A Breakthrough in Solvers for Eigenvalue Problems," <http://icl.utk.edu/magma/>, 2008.
- [19] Agullo, Emmanuel and Demmel, Jim and Dongarra, Jack and Hadri, Bilel and Kurzak, Jakub and Langou, Julien and Ltaief, Hatem and Luszczek, Piotr and Tomov, Stanimire, "Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects," in *Journal of Physics: Conference Series*, vol. 180. IOP Publishing, 2009, p. 012037.
- [20] Innovative Computing Laboratory - UTK, "Matrix Algebra on GPU and Multicore Architectures," <http://icl.utk.edu/magma/>, 2008.
- [21] Che, Shuai and Sheaffer, Jeremy W and Boyer, Michael and Szafaryn, Lukasz G and Wang, Liang and Skadron, Kevin, "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads," in *Workload Characterization (IISWC), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1–11.
- [22] Che, Shuai and Boyer, Michael and Meng, Jiayuan and Tarjan, David and Sheaffer, Jeremy W and Lee, Sang-Ha and Skadron, Kevin, "Rodinia: A benchmark suite for heterogeneous computing," in *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*. IEEE, 2009, pp. 44–54.
- [23] Nvidia, "Nvidia Visual Profiler," <http://developer.nvidia.com/nvidia-visual-profiler>, 2013.