# Performance and Power-Aware Classification for Frequency Scaling of GPGPU Applications

João Guerreiro, Aleksandar Ilic, Nuno Roma and Pedro Tomás

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

**Abstract.** The increased adoption of Graphics Processing Units (GPUs) to accelerate modern computational intensive applications, together with the strict power and energy constraints of many computing systems, has pushed for the development of efficient procedures to exploit dynamic voltage and frequency scaling (DVFS) techniques in GPUs. Although previous works have applied several pattern recognition techniques for GPGPU application classification, these approaches often result in many misclassifications when trying to identify which applications can benefit from DVFS. To circumvent this limitation, a new lightweight methodology for classifying GPU applications based on their performance and power consumption in the presence of GPU core frequency scaling is presented. The proposed methodology is based on a set of performance counters, such as memory bandwidth utilization and memory-related stalls, which are extracted during the application execution. Experimental results for a set of 20 applications from the Parboil, Rodinia and Polybench benchmark suites show that the proposed classification approach is able to correctly identify applications that can benefit from frequency scaling.

## 1 Introduction

Modern high performance computing (HPC) systems are increasingly making use of general purpose accelerators, such as graphics processing units (GPUs), in order to increase the resulting system performance. This is confirmed by an analysis of the most recent version of the TOP500 list (November 2015), where 103 of these systems are equipped with accelerators (75 and 90 in the two previous editions of this list). However, with the established adoption of GPUs, it is gradually important to find mechanisms that ensure the maximum efficiency of the computing system, both in terms of performance and (most importantly) energy. Accordingly, significant research efforts are being put forth in the investigation of dynamic voltage and frequency scaling (DVFS) techniques, due to the inherent potential for significant power and energy savings in many of the computer system components, including the processor cores [1].

General-purpose applications can largely vary in the way they use the computational and memory resources of the devices where they are executing [2]. While some applications perform a large number of computational operations for each loaded data (more *compute-bound*), other applications may perform very few operations for each portion of fetched data (more *memory-bound*). Although in

the former type of applications the resulting performance is more likely to scale proportionally with the frequency of the cores (highest frequency $\equiv$ best performance), this behaviour is not guaranteed for the latter set of applications. This opens an interesting window of opportunity, since some of these applications can be executed at lower frequency levels with negligible performance drop-off. Consequentially, and considering that (under DVFS) power consumption increases with the operating frequency of the device, the identification of these classes of applications can potentially be considered as an interesting opportunity for energy savings. However, just as power scales up with the operating frequency, the execution time scales down, making the definition of the optimal operating frequency a non-trivial choice. Hence, to perform this type of analysis, it is therefore fundamental the adoption of appropriate methodologies that allow the proper classification of the applications workloads, in order to identify which cases could potentially result in power- or energy-savings.

Although some previous works have already addressed the topic of workload classification, they have mostly focused on CPUs [2], even though addressing many different goals, e.g. *characterization*, *diversity analysis*, *subsetting*, *etc.*. In particular, the majority of the previous works on workload characterization involves the combination of principal component analysis (PCA) and hierarchical clustering [3]. As a consequence, some previous studies on workload characterization in the GPU-domain have tried to use similar approaches to the ones that were applied to the CPUs. In particular, Kerr et al. [4] characterized PTX workloads using a GPU simulator with the purpose of optimizing these applications. Che et al. [5] also performed a diversity analysis on the Rodinia benchmark suite, by using a real GPU (NVIDIA GTX 480). However, when looking at the majority of the state-of-the-art works in the area of GPU workload classification, it can be seen that most of the research relies on the usage of GPU simulators instead of real hardware. Although this allows for a detailed profiling of the workloads, usually by considering performance counters that are non-existent in real hardware, this renders these approaches impossible to replicate in real systems. Additionally, the existing GPU simulators are based on the NVIDIA's Fermi microarchitecture, which has already been followed by Kepler (2013), Maxwell (late 2014) and more recently Pascal (2016).

In the same trend, Adhinarayanan et al. [6] also provided an automated framework for characterizing and subsetting GPU workloads, by also relying on PCA and hierarchical clustering. While this approach has the advantage of reducing the dimensionality of the problem, usually by transforming a large number of metrics into a smaller number of principal components, it makes the understanding of each resulting class harder (from the computing architecture perspective) and does not necessarily result in a energy-aware classification.

In contrast with the described approaches, this work specifically addresses the definition of alternate classification methodologies in order to unveil which workloads will benefit from the application of DVFS techniques to provide energy savings. In fact, while it makes sense to classify applications and workloads as

compute-bound or memory-bound when analysing their performance on a given GPU, it is observed that these notions cannot be applied in the same way when the power consumption is considered. In particular, we show that classifying for performance and for power consumption may result in different application classifications, confirming the need for separate classification techniques that depend on the considered goal. Additionally, unlike the previous proposals, this work is performed by using real and modern hardware systems. Accordingly, the major contributions of this paper are the following:

- Analysis of different types of performance and power consumption metrics using several relevant GPU benchmarks on real hardware;
- novel application classification algorithms based on GPU performance and power metrics, able to characterize the execution of each application on a range of GPU frequencies based on its execution on a single core frequency;
- comparison with other state-of-the-art classification techniques for GPU applications.

To conduct this work, we study 20 applications from different relevant benchmark suites (Parboil [7], Rodinia [8] and Polybench [9]), and analyse how the core frequency scaling affects their performance and their power consumption. The obtained experimental results show that the proposed algorithms are able to accurately and consistently classify the considered GPU applications in terms of their behaviour in performance, power and energy consumption. Finally, the proposed approach is compared with other state-of-the-art classification methodologies, which result in classes of applications that do not exhibit similar performance/power behaviour in the presence of frequency scaling.

## 2 Application Classification for GPGPU DVFS

This work focuses on the classification of GPGPU applications, with the objective of identifying which applications can benefit from DVFS in order to provide energy-savings. The goal is to be able to properly classify one given application for all operating frequencies of a given GPU device, after the execution of that application on a single operating frequency.

However, changing the frequency of the GPU cores may affect an application execution in very different ways, depending on each application's characteristics. While it can be expected that a decrease in the core frequency ($f$) will cause the kernel execution time to increase ($T \propto \frac{1}{f}$) and power consumption to decrease ($P_{dynamic} \propto V^2 f$ and $P_{static} \propto V e^{\gamma V}$), the actual values for the application's performance and power consumption over different frequencies are highly dependent on the application. In fact, it has been shown that accurately predicting the impact of DVFS in the execution time or power consumption requires complex predictive models [10]. Hence, given that the energy ($E$) consumed by the GPU is computed as the product between power ($P$) and execution time ($t$), $\mathrm{E} = \mathrm{P} \times t$, it is important to understand the effects of DVFS on both the execution time and power consumption of applications.

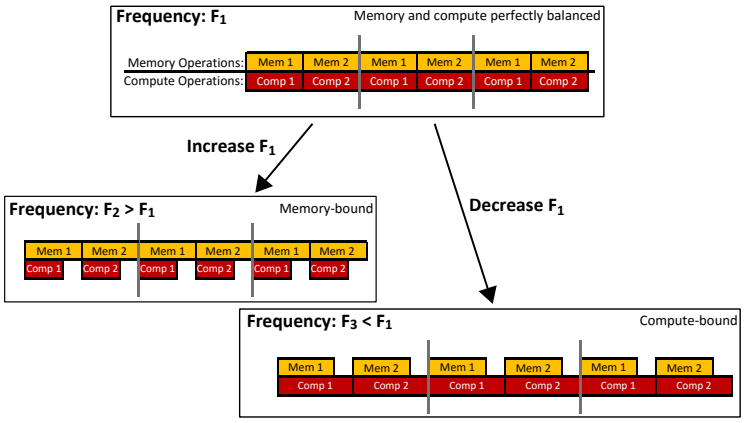



**Fig. 1:** Example of memory and compute operations overlap on three different core frequencies, with $F_2 > F_1 > F_3$. The instructions pairs (**Mem1**, **Comp1**) and (**Mem2**, **Comp2**) require full synchronization.

The DVFS impact on execution time is a complex problem that require a better understanding of the GPU architecture. In particular, one of the GPU main design goals concerns the use of multiple groups of parallel threads (*warps* in NVIDIA nomenclature) to hide instruction latency. However, in many general purpose applications, it is not always possible to hide the instruction latency with other warps. Therefore, when analysing the performance of applications, from the perspective of their bottleneck, most works tend to consider two main types of applications: *compute-bound* and *memory-bound*. Compute-bounded applications are defined as those where the execution time is mainly determined by the performance of the processing components, and is a direct consequence of an intensive utilization of the processing pipeline and functional units. On the other hand, memory-bound applications have their execution time mainly dependent on the bandwidth and latency of the memory hierarchy when satisfying the memory access requests.

Accordingly, when applying core level DVFS, the performance of a memory-bound kernel is limited by the communication with the GPU global memory, since the operating frequency of such component does not scale with the core frequency. However, such limitation is a consequence of the applied setup in terms of the core and memory operating frequencies. Hence, while one kernel may be compute-bound at one core operating frequency, it may become memory-bound if the operating frequency increases. To illustrate such condition, Figure 1 presents a simplification of relative weight represented by the memory and compute operations of one given kernel at three different core frequencies. At frequency $F_1$, the threads start executing both the **Mem1** and **Comp1** instructions at the same time and both finish their execution at the same time. In this example, instructions **Mem2** and **Comp2** require full synchronization but since both instructions finish at the same time, the latency of the threads waiting to be issued is fully hidden by the threads currently executing.

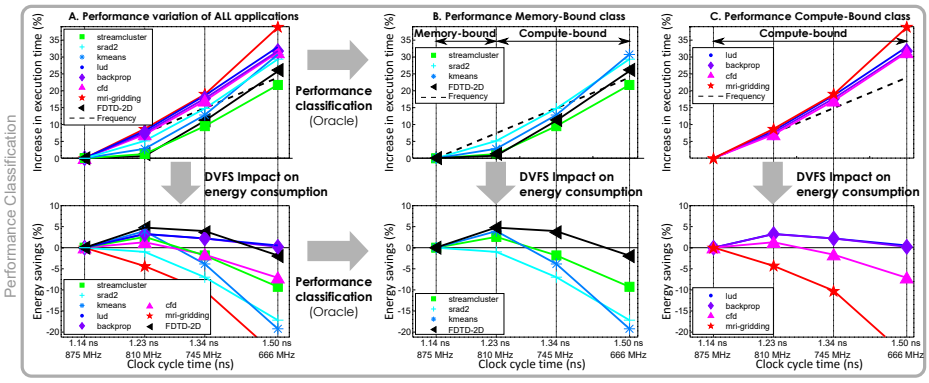



**Fig. 2:** Execution of the training set applications on NVIDIA Tesla K40c.

However, if the core frequency is changed to a higher value ($F_2$), there will be a time interval where only the **Mem** instructions are executing on the GPU, meaning that there are not enough compute threads to hide the latency of the threads waiting to be issued. This will cause an increase of the number of stalls caused by memory dependencies, therefore making the corresponding performance counter a good indicator of the bottleneck of one application. Hence, at core frequency $F_2$, the application is memory-bounded, since its performance bottleneck depends on the latency of the memory operations. If, on the contrary, the core frequency is set to a value $F_3$ that is lower than $F_1$, the duration of the **Comp** instructions will be longer than that of the **Mem** instructions. In this case, the performance bottleneck will be determined by the critical path of the compute instructions, thus resulting in a compute-bounded classification.

However, although such classification strategy is valid for the application execution time, it is not entirely accurate for power classification. To demonstrate such conclusion, a set of benchmark applications from the previously referred Parboil, Rodinia and Polybench benchmark suites were executed on an NVIDIA Tesla K40c GPU at different core frequency levels, namely 875 MHz (default level), 810 MHz, 745 MHz and 666 MHz. Additionally, the DVFS impact on the execution time and energy consumption was measured (see Figure 2). Also, the applications were hand-classified as *memory-bound* and *compute-bound* at the default operating frequency (Figure 2, top-right), by considering the execution time increase when operating at the different frequencies, and each group's energy variation was analysed (Figure 2, bottom-right). As can be observed, the presented compute/memory bound classification presents uninteresting results when energy consumption is considered. Hence, different methodologies must be employed. As a consequence, and keeping in mind that $E = P \times t$, a separate methodology to characterize effects of DVFS on the power consumption of applications is required.

In most GPUs (and in particular in modern NVIDIA GPUs) there are two independent frequency domains: 1) *Core domain*, which includes the streaming multiprocessors (SMs); and 2) *Off-chip domain*, which includes the off-chip DRAM. Accordingly, the power consumption can be expressed as

$$P(f_{\text{CORE}}, f_{\text{MEM}}) = P_{\text{CORE}}(f_{\text{CORE}}) + P_{\text{MEM}}(f_{\text{MEM}}) \quad (1)$$

where $P_{CORE}$ is the power consumed by the components of the core domain, and $P_{MEM}$ is the power consumed by the memory components, i.e. by the DRAM, which is herein assumed to remain constant. Based on Equation 1, it is possible to characterize the applications depending on their usage of the core and memory components. Therefore, a given application shall be considered to have a high memory utilization (i.e. with high activity of the memory resources) if the power consumed by the GPU is dominated by the $P_{MEM}$ parcel, i.e. if $P_{MEM} >> P_{CORE}$. As a consequence, when the frequency of the core is changed, since the power consumed by the memory is independent of the frequency of the core domain, the total power consumed will remain almost constant. On the other hand, if $P_{CORE} >> P_{MEM}$, the GPU power consumption will scale linearly with the core frequency and the applications are considered to have a high core utilization. In order to perform the power-aware classification of GPU applications, a third scenario is also herein considered where both the memory and core components present a low utilization. Hence, since both components present a low activation of their resources, the average power consumption variation will be reduced when the core frequency is scaled.

Accordingly, when looking at the behaviour of the GPU power consumption over different operating frequencies, three different classes will be considered: *High Core Utilization*, *High Memory Utilization* and *Low Device Utilization*.

However, in order to perform such classification without having to execute the kernel at a different operating frequency, it is necessary to retrieve some profiling information regarding each application, which in this case must characterize its usage of the memory components (specifically, of the off-chip DRAM). One metric that gives a good indicator of the level of utilization of the memory resources is the percentage of stalls caused by memory dependencies (Eq. 2). However, it is still possible for one application to have other dominant causes for stalls and still a have high utilization of the memory components. In accordance, a complementary metric that quantifies the ratio of the achieved memory bandwidth over the device's peak (Eq. 3) is also considered.

$$\text{Stalls}_{\text{mem}} = \frac{\text{Memory Dependency Stalls}}{\text{All Stalls}} \tag{2}$$

$$\text{Bandwidth}_{\text{mem}} = \frac{\text{Device memory transactions} \times \text{Transaction size}}{\text{Global memory bandwidth}} \tag{3}$$

Hence, in order to perform the power-aware classification, additional metrics that characterize the utilization of the GPU resources are required, namely performance counters that characterize the amount of time the GPU resources are being used. Among the provided set of execution metrics that are nowadays made available in GPU devices, it was selected a subset of metrics related with the total kernel execution time and the core utilization ($\mathbf{Util_{core}}$) which corresponds to the average percent of time over the previous sample period during which one or more kernels was executing on the GPU.

Algorithms 1 and 2 formalize the proposed methodologies to classify GPU applications into classes that characterize their performance and power consump-

tion, respectively, over different frequency levels based on performance counters measured while executing on single core operating frequency.

**Algorithm 1** Classification methodology of GPU applications based on the effects on execution time of DVFS.

**Inputs:** $\text{Bandwidth}_{\text{mem}}$ and $\text{Stalls}_{\text{mem}}$.
**Output:** Benchmark classification.

```
1: if Bandwidth_mem>α then
2:    memory-bound class.
3: else
4:    if Stalls_mem_depend>β then
5:       memory-bound class.
6:    else
7:       compute-bounded class.
8:    end if
9: end if
```

**Algorithm 2** Classification methodology of GPU applications based on the DVFS effects on power consumption.

**Inputs:** Exec_time, $\text{Stalls}_{\text{mem}}$, $\text{Util}_{\text{core}}$.
**Output:** Benchmark classification.

```
1: if Exec_time<γ OR Util_core<δ then
2:    Low Device Utilization class
3: else
4:    if Stalls_mem>η then
5:       High Mem. Utilization class
6:    else
7:       High Core Utilization class
8:    end if
9: end if
```

For the performance-aware classification (Algorithm 1), a given application can be classified by executing it in the target GPU on a single chosen frequency level, during which the two mentioned performance counters are measured. The application under analysis is considered to be memory-bounded if one of the two measured values is higher than the corresponding respective thresholds, $\alpha$ and $\beta$, respectively, whose values were experimentally determined by using a training set to determine the combination of values that would result in the minimum misclassified applications. The power-aware classification (Algorithm 2), is similar to the performance one, with an additional class that selects the applications with low device utilization, according with a set of threshold $\gamma$, $\delta$ and $\eta$ which were determined in the same way as before.

## 3 Experimental Results

To evaluate the proposed methodologies, several CUDA-based application benchmarks from the Parboil [7], Rodinia [8] and Polybench [9] suites (see Table 1) were executed on a NVIDIA Tesla K40c GPU (Kepler microarchitecture), which provides a user-level interface to scale the core operating frequency ($f_{CORE}$) in four non-idle levels, namely 875 MHz, 810 MHz, 745 MHz and 666 MHz.

Each application was executed at the four allowed core frequency levels and their execution time was measured using CUDA events, together with the previously referred performance counters (see Section 2) for the reference (default) frequency level (875 MHz). Finally, in order validate the devised power-aware classification, the power consumption of each application executed on Tesla K40c GPU was obtained using NVML. Such power samples were obtained at a sample interval of 15 ms, and the final power consumption was computed as the average of all the kernels that constitute each application benchmark.

**Table 1:** Summary of the considered application benchmarks.

| Rodinia | | Polybench | | Parboil | |
|---|---|---|---|---|---|
| **Application** | **Size** | **Application** | **Size** | **Application** | **Size** |
| Backprop | 655360 | 2MM | Default | CUTCP | Large |
| CFD | missile.domn.0.2M | CORR | Default | Histo | Large |
| Gaussian | 2048×2048 | COVAR | Default | LBM | Long |
| Hotspot | 1024, 2, 10000 | FDTD-2D | Default | MRI-Gridding | Small |
| K-Means | 3000000_34f.txt | GEMM | 2048×2048 | | |
| Lud | 8192×8192 | GESUMMV | 10240 | | |
| SRAD2 | 4096 | GRAMSCHM | Default | | |
| Streamcluster | Default | SYRK | Default | | |

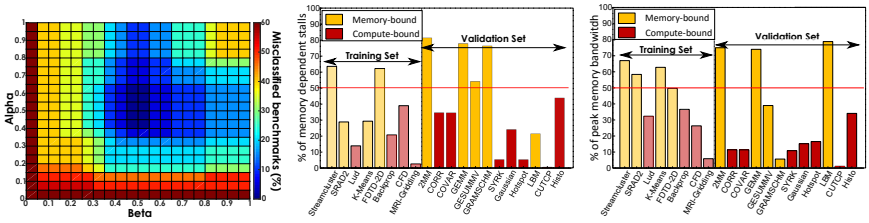
### 3.1 Classification Parameters

The previously proposed performance classification algorithm is dependent on two architecture-related parameters ($\alpha$ and $\beta$), which are determined using a randomly selected training set of applications. To assess such values Figure 3a presents the number of misclassified benchmarks for different values of these parameters for the Tesla K40c GPU, when comparing the classifications resulting from the proposed algorithm with a manual classification of each application (oracle classifications), which confirms that $\alpha = 0.5$ and $\beta = 0.5$ correspond to the optimal setup. The values for the $\gamma$, $\delta$ and $\eta$ parameters used in the proposed power classification methodology were also experimentally determined using the same approach. The obtained values are $\gamma = 170$ms, $\delta = 50\%$ and $\eta = 22\%$.

### 3.2 Performance-Aware Algorithm Evaluation

Figure3b and 3c depict the obtained values for the two metrics considered in the performance classification ($\text{Stalls}_{\text{mem}}$ and $\text{Bandwidth}_{\text{mem}}$), with the GPU cores set to 875 MHz. From these plots it can be observed that some applications have the majority of stalls caused by memory dependencies, while simultaneously achieving low usage of the device memory bandwidth. Some others display the opposite behaviour. Hence, after applying Algorithm 1, the two resulting classes correspond the ones presented in Figure 4. To validate the obtained classification, all considered benchmarks were executed using the four allowed core frequency levels for the considered GPU. The applications classified *memory-bounded* (see Figure 4a) have a variation of their execution time lower than the frequency variation for frequencies above 810 MHz. Hence, at this core frequency the applications start behaving as compute-bound applications and have their execution time scaling approximately linearly with the core frequency. Furthermore, the *compute-bounded* applications (see Figure 4b) always have their execution time scaling approximately linearly with the core frequency.

Again, it is important to stress that this methodology allows the classification of GPU applications into classes that characterize their performance at all frequency levels, by using the information obtained from their execution at a single core frequency in a real hardware device.

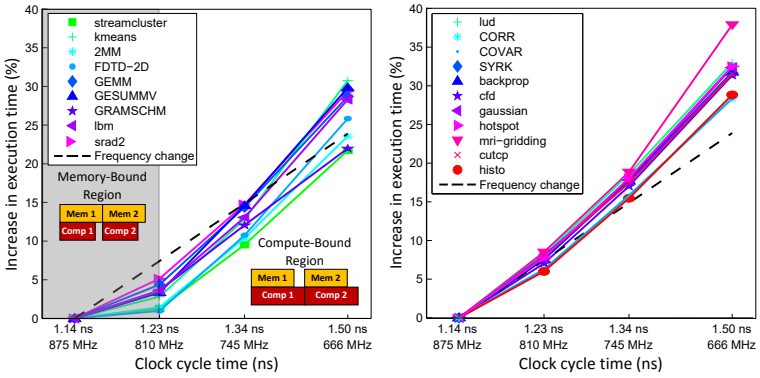



**(a)** Percentage of misclassified applications when changing the parameters $\alpha$ and $\beta$.

**(b)** Memory dependency stalls.

**(c)** Memory8 bandwidth.

**Fig. 3:** Considered metrics for the performance-aware classification on NVIDIA's Tesla K40c.



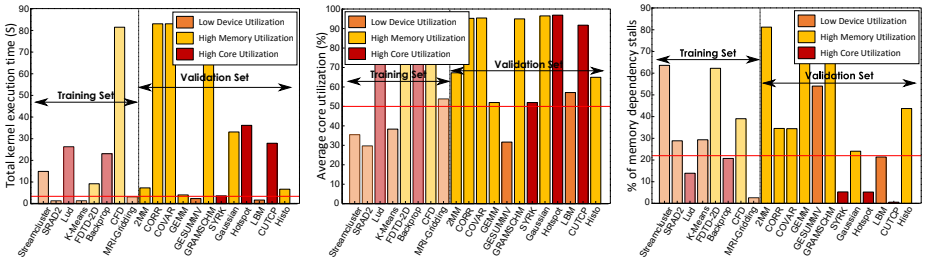

**(a)** Memory-bound class.

**(b)** Compute-bound class.

**Fig. 4:** DVFS effects on the execution time of the considered applications of the two performance classes in NVIDIA's Tesla K40c.

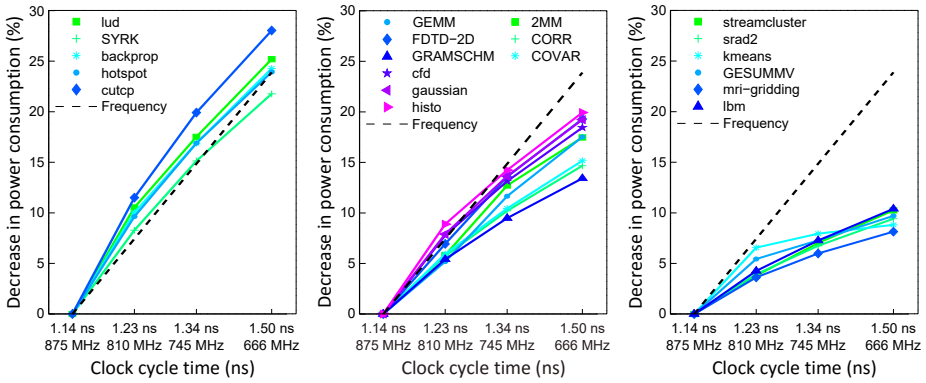
### 3.3 Power-Aware Algorithm Evaluation

Figure 5 presents the metrics considered in the devised power classification methodology (Exec_time, $\text{Util}_{\text{core}}$ and $\text{Stalls}_{\text{mem}}$) for the considered applications. By looking at their execution time (see Figure 5a) it can be observed that there are many applications whose total kernel execution time is below the previously obtained $\gamma$ parameter (170 ms), which will classify them in the *Low Device Utilization* class. By combining the values for these metrics for each of the workloads and applying Algorithm 2, the three classes presented in Figure 6 are obtained. To validate the obtained classification, all considered applications were executed using the four allowed frequency levels for the considered GPU, resulting in distinct power curves. This is particularly noticeable by looking at the value of the power decrease at 666 MHz, since in the *Low Device Occupancy* class all applications have a decrease in power consumption below 10%; in the *High Memory Utilization* class have power savings between 10% and 20%; and finally in the *High Core Utilization* class the applications decrease their power consumption by more than 20%.




**(a)** Execution time. **(b)** Core utilization. **(c)** Memory stalls.

**Fig. 5:** Considered metrics for power classification on NVIDIA's Tesla K40c.




**(a)** High Core Util. **(b)** High Memory Util. **(c)** Low Device Util.

**Fig. 6:** DVFS effects on power consumption of applications in the three power classes in NVIDIA's Tesla K40c.

### 3.4 Energy Clusters

Since the proposed methodology depicts two performance classes and three power classifications for a given set of applications, it is possible to combine all this information, thus obtaining six energy classes, whose results is depicted in Figure 7. Hence, when considering this classification methodology, it is not possible for one application to be simultaneously in the *memory-bounded* and *high core utilization* classes (it would require an application to simultaneously have $Stalls_{mem} > 0.5$ and $Stalls_{mem} < 0.22$).

When validating this classification by executing the applications at the GPU core levels and measuring the consumed energy (see Figure 7), it can be seen that the applications within each class display a similar behaviour in the presence of DVFS. Hence, the result of those classification can be used to select the applications in which DVFS is more likely to generate greater energy-savings.

The proposed approach is considerable more versatile than other related approaches. As an example, Adhinarayanan et al. [6] perform a clustering of GPU applications using 13 performance counters, later reduced using Principal Component Analysis (PCA) into 6 principal components used in the hierarchical clustering stage. However, when their methodology is replicated in Tesla K40c

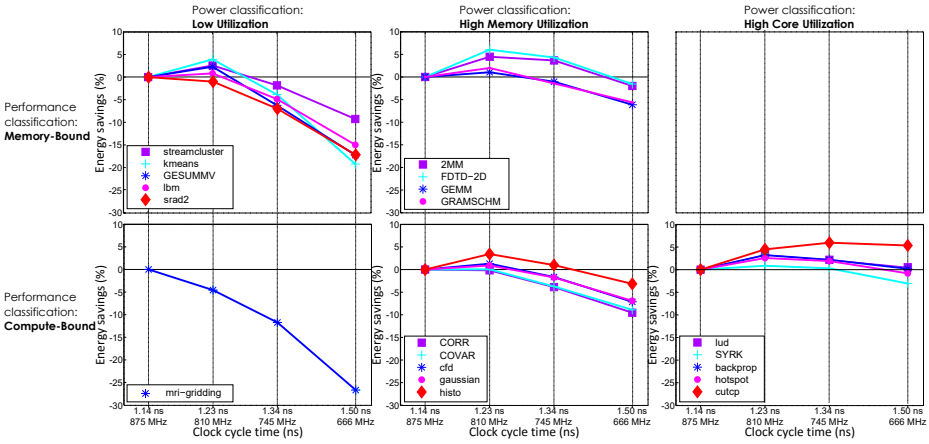



**Fig. 7:** Different energy classes resulting from the performance and power classes obtained for NVIDIA's Tesla K40c.

using the applications benchmarks that were used in this work, and by considering six clusters during the hierarchical clustering stage, the results presented in Figure 8 are achieved. As it can be seen, unlike the results from the proposed methodologies, this approach produces classes of applications that do not exhibit similar characteristics when DVFS is applied. Additionally, by using PCA and hierarchical clustering makes it harder to extract any architectural meaning from the resulting classifications. In particular, it is hard to understand from Figure 8 which of the resulting classes would correspond to the class composed by memory-bounded applications with high memory utilization.

## 4 Conclusion

A new methodology for GPU applications classification based on the resulting effects of DVFS on the application's execution time and power consumption was proposed. Such results from the fact that, existing classification techniques are not targeted for this specific goals, resulting in many wrongly classified applications when performance and power consumption are considered. To circumvent this absence, the proposed algorithms allow the classification of GPU applications into classes that characterize their performance (or power consumption) at all operating frequency levels, by solely using the information obtained from the execution of each application at a single core frequency. The performance and power classes define 6 distinct energy-aware classes of applications that present a similar behaviour in the presence of DVFS.

## Acknowledgment

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT), under the project UID/CEC/50021/2013.

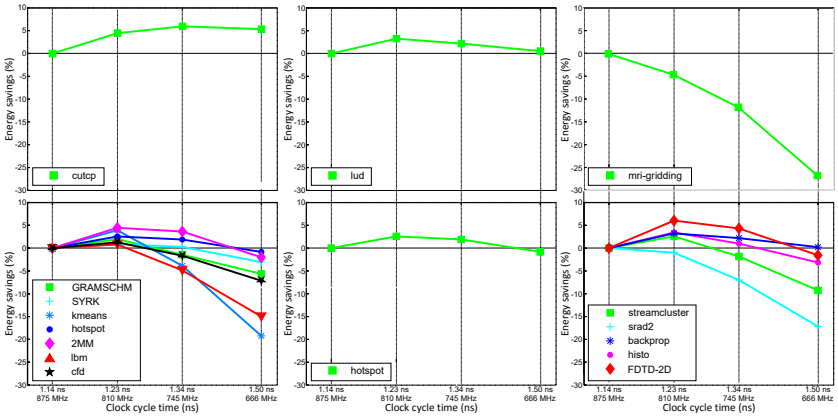



**Fig. 8:** Different classes resulting from the methodology proposed in [6].

## References


1. S. Herbert and D. Marculescu, "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors," in *Low Power Electronics and Design (ISLPED)*. IEEE, 2007.
2. S. Zhuravlev, S. Blagodurov, and A. Fedorova, "Addressing shared resource contention in multicore processors via scheduling," in *ACM SIGARCH Computer Architecture News*. ACM, 2010.
3. J. J. Yi, R. Sendag, L. Eeckhout, A. Joshi, D. J. Lilja, and L. K. John, "Evaluating benchmark subsetting approaches," in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2006.
4. A. Kerr, G. Diamos, and S. Yalamanchili, "A characterization and analysis of ptx kernels," in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2009.
5. S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron, "A characterization of the Rodinia benchmark suite with comparison to contemporary CMP workloads," in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2010.
6. V. Adhinarayanan and W.-c. Feng, "An Automated Framework for Characterizing and Subsetting GPGPU Workloads," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2016.
7. J. A. Stratton, C. Rodrigues, I.-J. Sung, N. Obeid, L.-W. Chang, N. Anssari, G. D. Liu, and W.-m. W. Hwu, "Parboil: A revised benchmark suite for scientific and commercial throughput computing," *Center for Reliable and High-Performance Computing*, 2012.
8. S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *International Symposium on Workload Characterization (IISWC)*. IEEE, 2009.
9. L.-N. Pouchet, "Polybench: The polyhedral benchmark suite," *URL: http://www.cs.ucla.edu/~pouchet/software/polybench/*, [cited March 2016,]. 2012.
10. G. Keramidas, V. Spiliopoulos, and S. Kaxiras, "Interval-based models for run-time DVFS orchestration in superscalar processors," in *Proceedings of the 7th ACM international conference on Computing frontiers*. ACM, 2010, pp. 287–296.