A Study of Hardware Performance Counters Selection for Cross Architectural GPU Power Modeling

Martín Pi Puig[1], Laura de Giusti[1,2], Marcelo Naiouf[1], and Armando De Giusti[1,3]

[1] Instituto de Investigación en Informática LIDI, CEA-CIC, Facultad de Informática, Universidad Nacional de La Plata, La Plata, Argentina.
[2] Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC), La Plata, Argentina.
[3] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), CABA, Argentina.
{mpipuig, ldgiusti, mnaiouf, degiusti}@lidi.info.unlp.edu.ar

Abstract. In the exascale race where huge corporations are spending billions of dollars on designing highly efficient heterogeneous supercomputers, the real need to reduce power envelopes forces current technologies to face crucial challenges as well as it demands the scientific community to evaluate and optimize the performance-power ratio. While energy consumption continues to climb up, the viability of these massive systems becomes a growing concern. In this context, the relevance of specific power-related research works turns into a priority. So we here develop an exhaustive step-by-step process for selecting a comprehensive set of hardware performance counters to serve as an input in an eventual GPU cross-architectural power consumption model. Our experiments show a high power-performance correlation between shared GPU events. Also, we present a set of events that delivers exclusive performance information in order to predict accurately GPU power fluctuations.

Keywords: GPU · Architectures · Performance Counters · Power Consumption · Correlation · Prediction.

1 Introduction

Currently, High Performance Computing (HPC) systems are power-bounded, and this trend is expected to continue for the upcoming future. Then, an exascale computing system (order 10^{18} FLOPS) estimated to arrive in 2021, is expected to work under a rigorous 20-40 megawatts power scope [1][2][3]. To meet this performance (a thousandfold increase over the first petascale computer) in an acceptable power cost will require to exploit improvements and innovations in hardware and software design. The major hurdles for the scientific community include excessive energy consumption, higher fault rate, and porting applications from petascale to exascale computing [4][5].

So as to improve energy efficiency the HPC community has adopted at an impressive rate the use of accelerator-based supercomputers. Furthermore, the NVIDIA Graphics Processing Units (GPUs) have completely outperformed its main competitor, the Intel Xeon Phi, becoming the principal general purpose accelerator for any supercomputer system. Current general purpose computing on GPUs (GPGPU) applications can be found in the HPC area [6] but there are plenty of emerging applications in other domains. Examples are deep learning [7], autonomous vehicles [8], image processing [9][10], encryption [11][12], medical algorithms [13][14], among others.

Together with their performance capabilities, GPU-based accelerators are also popular because of their remarkable energy efficiency [15][16], as portrayed in The Green500 list [17]. The June 2019 ranking shows that 8 of 10 top (performance-per-watt) computer systems incorporate GPU accelerators.

As the usage of GPUs has become dominant, it is increasingly critical to investigate the power consumption profile, to identify eventual power bottlenecks and to find mechanisms to maximize their energy efficiency when running real applications.

Since not all the current GPUs that constitute world's largest HPC centers are equipped with power sensors that can directly assess the power dissipation, the device power is either inspected from a isolated power meter or derived from the examined current and voltage. Nevertheless, variations in power consumption happen so fast that an external measurement reading cannot provide an in time accurate sample. Also such approaches do not represent a practical solution especially on large-scale distributed systems being the fact that they involve additional hardware devices.

In contrast, the design and use of software approaches to predict power consumption is still in progress. Albeit some prior attempts have been made with some constrains like targeting a particular application or architecture, they cannot contribute to the understanding of GPU power consumption considering the lack of analysis in the metrics they can gather.

So an alternative is to directly measure processor events that are causing energy and temperature changes, and to analyze metrics correlation with power consumption. The best proxies for this approach are the hardware performance counters found in all modern computer architectures. Considering a very large number of counters, the task of selecting events that are most representative of the full system power profile is becoming a challenge. Furthermore, the complexity naturally increases when different types of GPU architecture implementations are considered. The matter is further problematic by the fact that each one has a finite number of registers that can be simultaneously recorded reducing the prediction accuracy.

Generally, selected counters and power prediction models vary from one CPU/GPU architecture to the next due to differences in the accessibility and availability of hardware counters and the fact that different subset of counters can reduce the power prediction error in a given microarchitecture.

In this work, we investigate and analyze hardware performance counters on two different GPU architectures to test its correlation with power consumption. Furthermore, we are seeking for a common subset of performance events in order to build a statistical cross-architecture power prediction model.

The rest of this paper is organized as follows. In Section 2 we review some of the related work. Section 3 provides some background information about performance counters, power measurement, the studied GPU architectures and a brief description of the instrumented benchmarks. Section 4 describes the practical methodology while Section 5 covers our experimentation analysis. Finally, Section 6 includes some conclusions of this work.

2 Related Work

Similar previous works have widely focused on using hardware counters to analyze performance and power in CPU architectures. Isci et al. [18] have developed a software tool that provides real time power measurements for Intel Pentium 4 processors using hardware counters. Lim et al. [19] have constructed a power estimation model for an Intel Core i7 system based on hardware performance counters and a statistical regression method. Bellosa [20] has exploited active hardware units information (from performance events) to establish a thread-specific energy accounting. On the GPU side, Song et al. [21] have combined hardware performance counter data with machine learning and advanced analytics to model power-performance efficiency on GPUs. They briefly described the event collection and selection processes.

3 Background

3.1 Performance Counters

Hardware performance counters are a set of special-purpose registers built into today architectures to read and store the counts of low-level hardware activities within the device. They provide high-speed access to a considerable amount of performance information related to the circuit functional units, multiple caches, main and special memories, register banks, etc.

However, the types, meanings and number of hardware counters that we can access heavily depends on the architecture being used due to the aggressive growth of technology processes. Apart from architectural restrictions, the NVIDIA compute capability also limit the number of events we can collect from a particular GPU version. As a result, our Fermi-based accelerator with compute capability 2.0 contains 4 hardware registers and 74 countable events while our Pascal one (capability 6.1) presents 8 registers and 70 events.

So for a given NVIDIA GPU, we can collect the full range of events through three different approaches. The low-level one implies to use CUPTI (CUDA Performance Tools Interface) [22], a dynamic library that enables the creation of profiling and tracing tools. It is generally included in the CUDA Toolkit [23].

Whereas it supports all available platforms, the library needs modifications in the source code.

On the other hand, we can employ the PAPI-CUDA component to quantify GPU performance events [24]. It is a hardware performance counter measurement technology based on CUPTI. The library is distributed with the latest releases of PAPI and it also needs a source code instrumentation.

Finally and on a high level, the NVPROF tool enables the collection and description of profiling data from the command-line [25]. It does not require to modify the application source code and it also includes a Visual Profiler to automatically generate a timeline of application's CPU and GPU activity including events and metrics.

In this work, we make use of NVPROF to gather performance event samples during a given kernel execution.

3.2 Power Measurement

We apply a software measurement approach, where GPU power consumption is monitored using the NVIDIA Management Library (NVML) [26].

NVML is a C-based API for supervising and managing diverse NVIDIA GPUs features like the ability to set/unset ECC (Error Correction Code), or to monitor memory usage, temperature, utilization rates, and more. Moreover, this library provides the ability to query power consumption at runtime through the built-in power sensor.

Therefore, we coded a power measurement tool that queries the GPU sensor via NVML and stores data logs to disk. The sampling frequency is fixed at 62.5Hz (16 ms) [27][28].

3.3 GPU Architectures

This work focuses on Fermi and Pascal NVIDIA GPU architectures. Among these different hardware generations, we examine the Fermi-based Tesla C2075 GPU and the Pascal-based GTX Titan Xp GPU exploring performance relations. The old-fashioned Fermi architecture correspond to the initial GPU generations (after Tesla) while Pascal represents a recent graphics hardware. Table 1 details the specifications of the studied devices.

Table 1. Hardware specifications.

Feature	**Tesla C2075**	**GTX Titan Xp**
Architecture	Fermi	Pascal
Process Size (nm)	40	16
Transistor Count (M)	3000	11800
TDP (Watts)	225	250
CUDA Cores	448	3584
Memory Size (GB)	6	12
Memory Bandwidth (GB/s)	144	547

These GPUs are designed for massive parallel processing employing the SPMD (Single Program Multiple Data) paradigm. They consist of a large number of streaming multiprocessors (SMs), each one containing multiple small hardware units called streaming processors (SPs) for classical processing, load/store (LD/ST) units, special function units (SFUs) to handle particular calculations and a set of high-speed registers for thread local storage. Each SP has one or multiple fully pipelined integer arithmetic logic units and floating point units. The GPUs also include a L1 cache memory for each SM and an unified L2 cache memory shared by all the SMs. More than that, these devices have an on-chip cached texture memory for 2D spatial locality (image and video applications). They also incorporate an off-chip DRAM global memory supporting a fixed GB data capacity. Then, a PCI-Express bus is used to connect the host system to one or multiple GPUs for bidirectional data transferring.

In the software side, a kernel is executed in parallel by an array of threads. These threads are arranged as a grid of thread blocks in which different kernels can have a diverse grid/block configuration. The grid of blocks and the thread blocks are organized in a three-dimensional hierarchy. So, when a CUDA program invokes a kernel grid, the blocks are enumerated and distributed to SMs with available execution capacity. Threads in a thread block execute concurrently on one SM, and multiple thread blocks can execute concurrently on one SM. As thread blocks finishes, new blocks are launched on the vacated SMs. Finally, the basic scheduling unit of GPU execution is the warp. It is a group of threads that are executed simultaneously by an SM. The mapping between warps and thread blocks can affect the kernel performance.

3.4 Benchmarks

To explain our experimental methodology, we tested a wide variety of GPU applications collecting at the same time power and performance counters data. We chose workloads that exploits different computational, communication and synchronization patterns representing a spectrum of application models.

The benchmarks include Rodinia, Polybench and CUDA SDK programs [29][30][31]. In particular, Rodinia is a set of benchmark applications designed for heterogeneous computing environments. To provide a high-level abstraction of common computing, memory access and communication patterns, each application is classified according to the Berkeley's dwarf taxonomy [32]. Polybench is a selection of CUDA codes to test different NVIDIA GPUs. In addition, we also include samples from the CUDA SDK that are incorporated in the CUDA Toolkit.

These benchmarks include programs representing a high number of real world applications like Data Mining, Image and Video Processing, Linear Algebra, Finance, Simulation, Physics, among others.

4 Experimental Methodology

In this section, we explain the proposed experimental methodology.

Firstly, we modified all the involved applications to include a profiling CPU thread using the Pthreads library, that periodically collects live power data from built-in sensors via NVML. Then, the only communication between the power measuring thread and computational threads is a flag variable. Each kernel execution is repeated multiple times for an accurate analysis. The problem sizes for the different applications were chosen to ensure enough power data points. Finally, power readings are stored in a log file for further processing.

Later, we run the compiling process for all the application benchmarks on both GPU architectures. Furthermore, in order to gather a record of performance events the on-chip GPU monitoring counters are periodically sampled. This counters acquiring process occur in a new application execution using NVPROF, which stores a csv file with hardware counter information.

Lastly, we combine hardware counter data with power readings using Python for statistical analysis. Figure 1 resumes the mentioned experimental methodology.

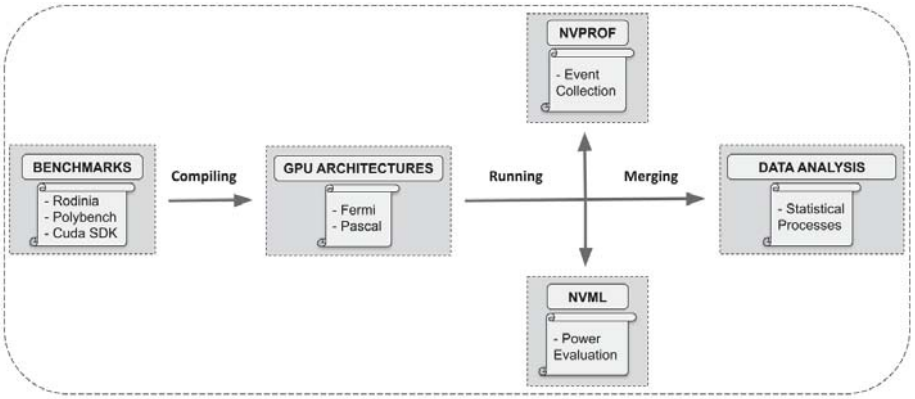

Fig. 1. Experimental framework.

5 Evaluation and Analysis

5.1 Performance Event Support

Since the evaluated GPUs correspond to very distinctive architectures, we got different performance events depending on the accelerator hardware. Figure 2 details the supported events in the examined generations.

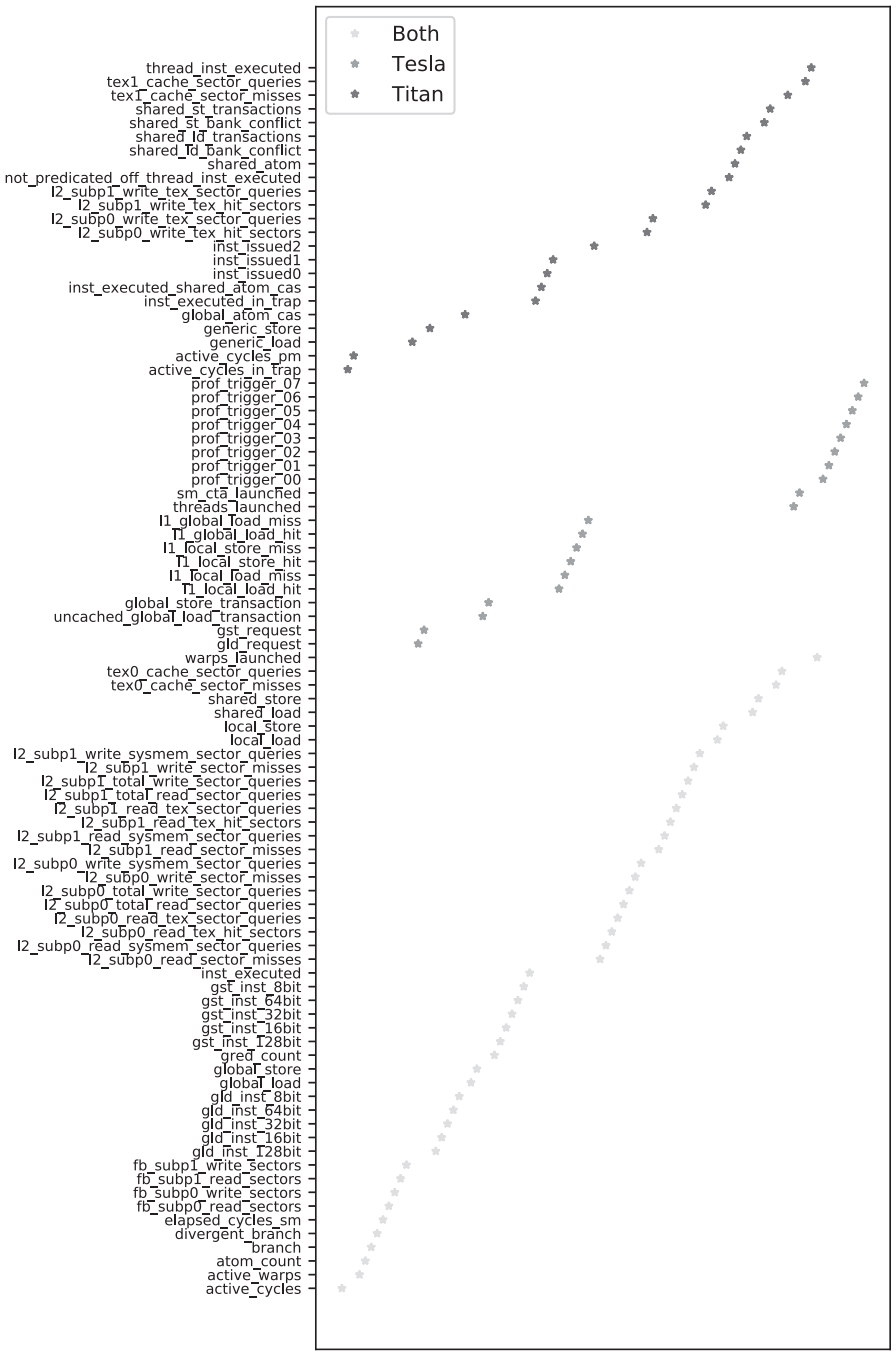

Fig. 2. Supported events.

Only 50% of the collected events are shared by both architectures. As one and the other are very distant on the NVIDIA generation roadmap, their hardware features sorely differ.

Apart from the increase in the number of single precision and double precision cores, number of load/store units, number of special function units, size of register file and memory caches, the Pascal architecture integrates other radical technological improvements with respect to Fermi.

The Titan Xp includes half precision floating point operations for high performance Deep Learning. Also, it supports unified memory, a single memory address space accessible from any processor in a system. This technology allows applications to allocate data that can be read or written from code running on either CPUs or GPUs. While Fermi GPUs featured a configurable shared memory and L1 cache that could split the allocation of memory between L1 and shared memory functions depending on workload, beginning with Maxwell, the cache hierarchy was changed. A Pascal SM has its own dedicated pool of shared memory and an L1 cache that can also serve as a texture cache depending on workload.

These factors lead to considerable differences in available performance attributes across different GPU architectures.

5.2 Performance Event Selection

In this section we would like to restrict our exploration to a small subset of performance events that would have strong impact on GPU power consumption. Since we are searching for an online power prediction model, we limit ourselves to the minimum number of hardware registers that are available on both microarchitectures. In this case, the Tesla GPU has 4 performance event slots so it delimits the number of events that can be simultaneously gathered in real time for the whole application.

The steps involved in our experimentation are specified below:

1. GPU performance events and power samples are collected for each benchmark by running the CUDA applications multiple times.

2. All performance events are normalized based on the total number of elapsed clock cycles per multiprocessor yielding a performance event rate for each counter on every application.

3. Events showing a zero value across all applications are removed in a preprocessing task.

4. Pearson, Spearman and Kendall Tau correlations are computed between performance rates and power consumption for all the benchmarking applications.

5. A filter is created to eliminate those events that are not common among all correlation methods.

6. Two thresholds are established, one for positive correlation and the other for the negative one. These border values are chosen so as to filter a high number of performance events within step 5.

7. Event rates showing an overall correlation below the threshold are discarded.

8. From the remaining set of strong power-performance correlation, the mutual correlation is analyzed. This process identifies redundant information.

9. Select as many events as the architecture support. Start by picking an unique event rate for each hardware functionality.

So, we then attempt to find the correlation between selected performance event rates and GPU power consumption employing different statistical approaches: Pearson, Spearman and Kendall Tau methods. While Pearson correlation depicts linear relationships and Spearman represent monotonic behaviors, the Kendall Tau mechanism usually gives deep statistical properties.

Figure 3 shows the power-performance correlation plots for every method on each accelerator. This plot depicts the reduction from 45 to 16 high correlated event rates.

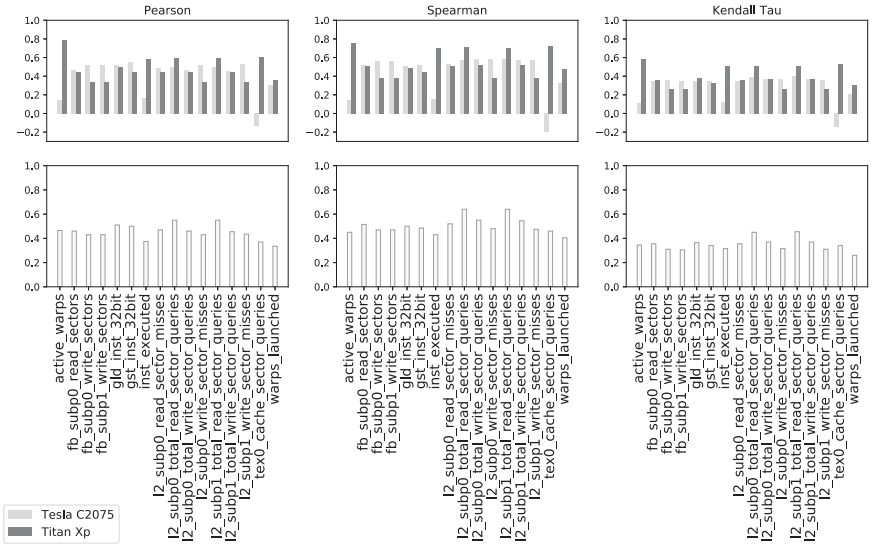

Fig. 3. Pearson, Spearman and Kendall Tau methods for high correlated events.

The first charts row exposes the different statistical correlations for every GPU. On the other hand, second row shows the absolute mean correlation involving the two GPU for each performance rate. This chart gives an insight into high performance-power correlated events through different NVIDIA GPU architectures. It can be seen that these strong correlated events cover the whole accelerator hardware functionality: execution system (*active_warps*, *inst_executed*, *warps_launched*), memory system (*fb_subp0_read_sectors*, *fb_subp0_write_sectors*, *fb_subp1_write_sectors*, *gld_inst_32bit*, *gst_inst_32bit*) and caches system (*l2_subp0_read_sector_misses*, *l2_subp0_total_read_sector_queries*, *l2_subp0_total_write_sector_queries*, *l2_subp0_write_sector_misses*, *l2_subp1_total_read_sector_queries*, *l2_subp1_total_write_sector_queries*, *l2_subp1_write_sector_misses*, *tex0_cache_sector_queries*).

In order to identify redundant information, Figure 4 considers the events in concern, illustrating the two Pearson correlation heatmaps across the selected performance-power correlated events. In this case we use only the Pearson approach because it is more susceptible to outliers, and thus more robust. We also include power data to analyze the strongest correlated event rates.

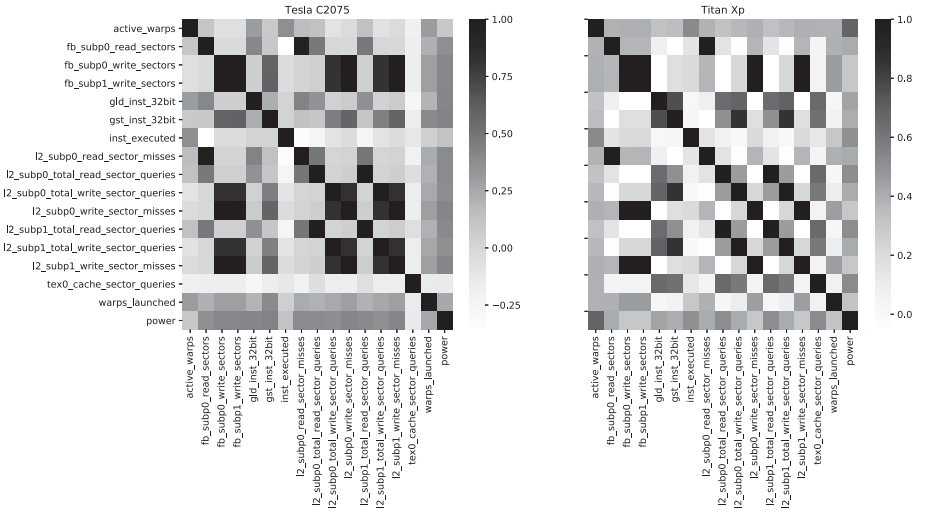

Fig. 4. Correlation heatmap for each GPU.

Strong correlations are expressed by foggy shadows and weak ones are represented by brighter shades. The diagonal terms show that each variable always perfectly correlates with itself. The other components are the Pearsons correlation coefficient between two performance event rates.

Last row of the lower triangular matrix (or last column of the upper one) exposes a high correlation of the selected events with power consumption across different GPU architectures.

As we aim to find the minimum set of hardware performance rates that highly correlate with power, the selection procedure consists on picking an unique event rate for each hardware functionality in the accelerator.

We chose *inst_executed*, the total number of executed instructions per warp, because it has a low correlation coefficient with memory and cache events and it has high correlation with events in the same functional unit (*active_warps* and *warps_launched*) in both GPUs. These considerations allow this event rate to provide non-redundant information to the prediction model.

Furthermore, we pick *gst_inst_32bit*, the total number of global store instructions that are executed by all the threads across all blocks. The event rate has strong correlation with cache performance counters across GPU generations but

it is not restricted to a DRAM memory subpartition since it counts every store instruction in each parallel thread block.

Lastly, we select *l2_subp0_read_sector_misses*, the number of read requests sent to DRAM from slice 0 of L2 cache, due to it has no correlation with *inst_executed* and a weak one with *gst_inst_32bit*. L2 misses/Global transactions correlation exists because of the intrinsic relationship in memory hierarchy in all nowadays computer systems. As the selected event for memory subsystem considers write transactions, we want to complement it by including read operations. L2 cache misses addresses this situation computing DRAM reads from cache failures.

Unlike [21] we found a considerable contribution from L2 performance counters to power consumption (Pearson's coefficients from 0.35 to 0.59).

To sum up *inst_executed*, *gst_inst_32bit* and *l2_subp0_read_sector_misses* with an *elapsed_cycles_sm* normalization are the selected hardware performance counters for an eventual GPU cross-architectural power prediction scheme.

In a last phase, we focus on determining the influence of each selected event rate in an prior power estimation model. Table 2 depicts statistical information from a preliminary Ordinary Least Squares regression model.

Table 2. OLS Linear Regression statistical properties.

Event Rate	**Coefficient**	**P-Value**	**T-Value**
inst_executed_per_cycle	20.153	7.068e-16	9.451
gst_inst_32bit_per_cycle	17.461	5.977e-14	8.605
l2_subp0_read_sector_misses_per_cycle	409.706	5.039e-11	7.289

P-values (probability values) and regression coefficients work together to explain which model relationships are statistically significant and the nature of those links. The coefficients describe the mathematical connections between each event rate and power consumption. Moreover, t-values gives an insight into the coefficient standard error (in power units). Furthermore, the smaller the p-value, the larger the absolute value of the t-value and the greater the evidence of importance for the event.

The prior regression output shows that all the event rates are statistically considerable because their p-values are minimal compared to the usual significance level of 0.05 (less than 5%) and their corresponding t-values are greater than zero.

In particular, *inst_executed_per_cycle* represent beforehand the strongest event rate, while *gst_inst_32bit_per_cycle* and *l2_subp0_read_sector_misses_per_cycle* have a slightly lower contribution to power.

For the results above, we confirm that all the selected performance events are meaningful since variations in the event values are related to changes in power consumption.

6 Conclusions

This paper suggest a specific methodology for selecting hardware performance counters in order to build a real time cross-architectural GPU power model. Our work analyzes performance events on different GPU generations studying performance-power correlation to grant an universal subset of events. This counter selection process is even more difficult than in a classic CPU system due to the complexity and variety of its hardware structure.

Our experimental results reveal that choosing a common high correlated event for every hardware feature allows to generate an unique and robust data input to any power estimation scheme. Consequently, the performance events *inst_executed* (compute), *gst_inst_32bit* (memory), *l2_subp0_read_sector_misses* (cache) and *elapsed_cycles_sm* (cycles rate) can clearly explain power fluctuations in both GPU architectures.

In future work, we will develop a multi-architecture power model using common hardware performance counters to analyze and predict in real time the GPU power consumption.

References

1. M. Lobet, M. Haefele, V. Soni, P. Tamain, J. Derouillat, and A. Beck. High performance computing at exascale: challenges and benefits. In *15me congrs de la Socit Franaise de Physique division Plasma*, pages 1–34, (2018).
2. J. Chen, A. Choudhary, S. Feldman, B. Hendrickson, C. R. Johnson, R. Mount, V. Sarkar, V. White, and D. Williams. Synergistic challenges in data-intensive science and exascale computing. In *DOE ASCAC Data Subcommittee Technical Report. Department of Energy Office of Science*, pages 1–10, (2013).
3. M. J. Schulte, M. Ignatowski, G. H. Loh, B. M. Beckmann, W. C. Brantley, S. Gurumurthi, N. Jayasena, I. Paul, S. K. Reinhardt, and G. Rodgers. Achieving exascale capabilities through heterogeneous computing. In *IEEE Micro*, volume 35, pages 26–36, (2015).
4. R. Springmeyer, C. Still, M. Schulz, J. Ahrens, S. Hemmert, R. Minnich, P. McCormick, L. Ward, and D. Knoll. From petascale to exascale: eight focus areas of r and d challenges for hpc simulation environments. In *Technical Report*, pages 1–11, (2011).
5. S. Fiore, M. Bakhouya, and Waleed W. Smari. On the road to exascale: Advances in high performance computing and simulationsan overview and editorial. In *Future Generation Computer Systems*, volume 82, pages 450–458, (2018).
6. The top 500 ranking, June 2019. Available at `https://www.top500.org/lists/2019/06/`.
7. L. N. Huynh, Y. Lee, and R. K. Balan. Deepmon: Mobile gpu-based deep learning framework for continuous vision applications. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pages 82–95, (2017).
8. M. McNaughton, C. Urmson, J. M. Dolan, and J Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *IEEE International Conference on Robotics and Automation*, pages 4889–4895, (2011).

9. B. Haghighi, N. D. Ellingwood, Y. Yin, E. A. Hoffman, and C. Lin. A gpu-based symmetric non-rigid image registration method in human lung. In *Springer Medical and biological engineering and computing*, volume 56, pages 355–371, (2018).
10. M. Cárcamo, P. E. Román, S. Casassus, V. Moral, and F. R. Rannou. Multi-gpu maximum entropy image synthesis for radio astronomy. In *Astronomy and computing*, volume 22, pages 16–27, (2018).
11. S. Aljawarneh, M. B. Yassein, and W. A. Talafha. A resource-efficient encryption algorithm for multimedia big data. In *Springer Multimedia Tools and Applications*, pages 22703–22724, (2017).
12. A. Saxena, V. Agrawal, R. Chakrabarty, S. Singh, and J. S. Banu. Accelerating image encryption with aes using gpu: A quantitative analysis. In *Springer International Conference on Intelligent Systems Design and Applications*, pages (372–380), (2018).
13. Y. Wang, T. Mazur, O. Green, Y. Hu, H. Li, V. Rodriguez, H. Wooten, D. Yang, T. Zhao, S. Mutic, and H. Li. Thabbra07: Penelopebased gpuaccelerated dose calculation system applied to mriguided radiation therapy. In *Wiley Online Medical physics*, volume 43, pages 3855–3855, (2016).
14. M. Kim, H. Shin, J. Jung, S. Kim, D. Yoon, and T. S. Suh. Development of gpu-based fast reconstruction algorithm for gamma ray imaging with insufficient conditions. In *Wiley Online Medical physics*, volume 44, pages 3147–3147, (2017).
15. M. Pi Puig, L. C. De Giusti, and M. Naiouf. Are gpus non-green computing devices? In *Journal of Computer Science and Technology*, volume 18, pages 153–159, (2018).
16. M. Pi Puig, L. C. De Giusti, M. Naiouf, and A. E. De Giusti. Gpu performance and power consumption analysis: A dct based denoising application. In *XXIII Congreso Argentino de Ciencias de la Computación*, pages 185–195, (2017).
17. The green 500 ranking, June 2019. Available at `https://www.top500.org/green500/lists/2019/06/`.
18. C. Isci and M. Martonosi. Runtime power monitoring in high-end processors:methodology and empirical data. In *IEEE Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, pages 93–105, (2003).
19. M. Y. Lim, A. Porterfield, and R. Fowler. Softpower: Fine-grain power estimations using performance counters. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 308–311, (2010).
20. F. Bellosa. The benefits of eventdriven energy accounting in power-sensitive systems. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, pages 37–42, (2000).
21. S. Song, C. Su, B. Rountree, and K. W. Cameron. A simplified and accurate model of power-performance efficiency on emergent gpu architectures. In *IEEE 27th International Symposium on Parallel and Distributed Processing*, pages 673–686, (2013).
22. Cupti api reference guide, May 2019. Available at `https://docs.nvidia.com/cuda/cupti/index.html`.
23. Cuda library, March 2019. Available at `https://developer.nvidia.com/cuda-zone`.
24. Papi cuda component, April 2019. Available at `https://developer.nvidia.com/papi-cuda-component`.
25. Profiler's user guide, April 2019. Available at `https://docs.nvidia.com/cuda/profiler-users-guide/index.html`.
26. Nvml library, June 2019. Available at `https://developer.nvidia.com/nvidia-management-library-nvml`.

27. K. Kasichayanula, D. Terpstra, P. Luszczek, S. Tomov, S. Moore, and G. D. Peterson. Power aware computing on gpus. In *IEEE Symposium on Application Accelerators in High Performance Computing*, pages 64–73, (2012).
28. V. Weaver, M. Johnson, K. Kasichayanula, J. Ralph, P. Luszczek, D. Terpstra, and S. Moore. Measuring energy and power with papi. In *IEEE International Conference on Parallel Processing Workshops*, pages 262–268, (2012).
29. S. Che, M. Boyer, J. Meng, D. Tarjan, J. Sheaffer, S. Lee, and K. Skadron. Rodinia: A benchmark suite for heterogeneous computing. In *IEEE International Symposium on Workload Characterization*, pages 44–54, (2009).
30. Polybench benchmark, May 2019. Available at `http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/`.
31. Cuda sdk samples, June 2019. Available at `https://docs.nvidia.com/cuda/cuda-samples/index.html`.
32. K. Asanovi, R. Bodik, B.n Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, Lester P. W., J. Shalf, S. Williams, and K. Yelick. The landscape of parallel computing research: A view from berkeley. In *Technical Report, EECS Department, University of California, Berkeley*, pages 56–67, (2006).