Performance/energy aware optimization of parallel applications on GPUs under power capping

Adam Krzywaniak and Paweł Czarnul

Faculty of Electronics, Telecommunications and Informatics
Gdansk University of Technology, Poland
adam.krzywaniak@pg.edu.pl, pczarnul@eti.pg.edu.pl

Abstract. In the paper we present an approach and results from application of the modern power capping mechanism available for NVIDIA GPUs to the benchmarks such as NAS Parallel Benchmarks BT, SP and LU as well as cublasgemm-benchmark which are widely used for assessment of high performance computing systems' performance. Specifically, depending on the benchmarks, various power cap configurations are best for desired trade-off of performance and energy consumption. We present two: both energy savings and performance drops for same power caps as well as a normalized performance-energy consumption product. It is important that optimal configurations are often non-trivial i.e. are obtained for power caps smaller than default and larger than minimal allowed limits. Tests have been performed for two modern GPUs of Pascal and Turing generations i.e. NVIDIA GTX 1070 and NVIDIA RTX 2080 respectively and thus results can be useful for many applications with profiles similar to the benchmarks executed on modern GPU based systems.

Keywords: performance/energy optimization, power capping, GPU, NAS Parallel Benchmarks

1 Introduction

In the paper, we present results of research on performance and energy aware optimization of parallel applications run on modern GPUs under power capping. Power capping has been introduced as a feature available for modern server, desktop and mobile CPUs as well as GPUs through tools such as Intel's RAPL, AMD's APM, IBM's Energyscale for CPUs and NVIDIA's NVML/nvidia-smi for NVIDIA GPUs [4,6].

We provide a follow up of our previous research [10,11] for CPU-based systems aimed at finding interesting performance/energy configurations obtained by setting various power caps. Within this paper, we provide results of running selected and widely considered benchmarks such as: NPB-CUDA which is an implementation of the NAS Parallel Benchmarks (NPB) for Nvidia GPUs in CUDA [5] as well as cublasgemm-benchmark [15]. We show that it is possible to obtain visible savings in energy consumption at the cost of reasonable performance drop, in some cases the performance drop being smaller than energy saving gains, percentage wise. Following work [3], we could also observe gradual and reasonably slow drop of power consumption on a GPU right after application has finished.

2 Related work

Our recent review [4] of energy-aware high performance computing surveys and reveals open areas that still need to be addressed in the field of energy-aware high performance computing. While there have been several works addressing performance and energy awareness of CPU-based systems, the number of papers related to finding optimal energy-aware configurations using GPUs is still relatively limited. For instance, paper [8] looks into finding an optimal GPU configuration in terms of the number of threads per block and the number of blocks. Paper [13] finds best GPU architectures in terms of performance/energy usage.

In paper [12] authors have presented a power model for GPUs and showed averaged absolute error of 9.9% for NVIDIA GTX 480 card as well as 13.4% for NVIDIA Quadro FX5600, using RODINIA and ISPASS benchmarks. Furthermore, they presented that coarse-grained DVFS could achieve energy savings of 13.2% while fine-grained DVFS 14.4% at only 3% loss of performance for workloads with phase behavior, exploiting period of memory operations. SM cluster-level DVFS allowed to decrease energy consumption by 7% for the HRTWL workload – some SMs become idle at a certain point due to load imbalance.

Similarly, in [1] authors showed that by proper management of voltage and frequency levels of GPUs they were able to reduce energy by up to 28% at the cost of only 1% performance drop for the hotspot Rodinia benchmark, by a proper selection of memory, GPU and CPU clocks, with varying performance-energy trade-offs for other applications.

In [7], authors propose a GPU power model and show that they can save on average approximately 10% of runtime energy consumption of memory bandwidth limited applications by using a smaller number of cores. Another model – MEMPower for detailed empirical GPU power memory access modeling is presented in [14].

There are also some survey type works on GPU power-aware computations. Paper [2] presents a survey of modeling and profiling methods for GPU power. Paper [16] discusses techniques that can potentially improve energy efficiency of GPUs, including DVFS, division of workloads among CPUs and GPUs, architectural components of GPUs, exploiting workload variation and application level techniques.

Paper [17] explores trade-off between accuracy of computations and energy consumption that was reduced up to 36% for MPDATA computations using GPU clusters.

There are relatively few works on power capping in CPU-GPU environments. Selected works on load partitioning among CPUs and GPUs are discussed in [16]. Device frequencies and task mapping are used for CPU-GPU environments in [9]. In paper [18], desired frequencies for CPUs and GPUs are obtained with dynamic adjustment for a GPU at application runtime are used for controlling power consumption.

3 Proposed approach

In our previous works [10,11] we investigated the impact on performance and energy consumption while using power capping in modern Intel CPUs. We have used an Intel

Running Average Power Limit (RAPL) driver which allows for monitoring CPU energy consumption and controlling CPU power limits through Model-Specific Registers (MSR). Based on RAPL we have implemented an automatic tool called EnergyProfiler that allows for finding the energy characteristic of a device-application pair in a function of power limit. We evaluated our prototype tool on several Intel CPUs and presented that when we consider some performance impact and accept its drops power capping might result in significant (up to 35% of) energy savings. We were also able to determine such configurations of power caps that energy consumption is minimal for a particular workload and also configurations where energy consumption and execution time product is minimal. The latter metrics allow to find such power limits for which the energy savings are greater than performance loss.

In this paper we adapt our approach to Nvidia GPUs. We use power limiting and power monitoring features available in modern Nvidia graphic cards and extend EnergyProfiler to work on a new device type. Using the extended EnergyProfiler we can investigate the impact of limiting the power on performance and energy consumption on GPUs running HPC workloads.

3.1 Power capping API

The power management API available on Nvidia GPUs is included in Nvidia Management Library (NVML). It is a C-based programmatic interface for monitoring and controlling various states within NVIDIA GPUs. Nvidia also shared a command line utility `nvidia-smi` which is a user friendly wrapper for the features available in NVML.

We based our prototype extension of EnergyProfiler on `nvidia-smi`. For controlling the power limit there is command `nvidia-smi -pl <limit>` available. The limit that we can set must fit into the range between minimal and maximal power limit which are specific for the GPU model. Monitoring and reporting the total energy consumption had to use quite a different approach than in our previous work as the NVML allows for reading the current power consumption while Intel RAPL lets the user to read the counters representing energy consumed. That caused that for evaluating the energy consumption of GPU running our testbed workload we needed to integrate the current power readings gathered while test run. Therefore, we estimated the total energy consumption as a sum of current power readings and sampling period products.

To monitor the power the `nvidia-smi dmon -s p -o T -f <filename>` was used. The parameter `-s p` specifies that we observe power and temperature, the parameter `-o T` adds the time point to each entry reported and the parameter `-f <filename>` stores the output of `dmon` to a file specified by `<filename>`.

The prototype extension of the EnergyProfiler tool runs a given application in parallel with the `dmon`, when the testbed application finishes, the tool analyses the log file and reports the energy consumption and the average power. Unfortunately, the minimal value of a sampling period in `nvidia-smi dmon` is 1 second, which means that the measurements might be inaccurate for the last sample which is taken always after the testbed application has finished.

For the testbed workloads with a really short execution time such an error would be unacceptable. In our experiments we have used workloads for which execution time

varies from 20 to 200 seconds which means that the maximal error of energy consumption readings is less than 5% and the minimal error for the longest test runs is less than 0.5%. The execution time reported is based on precise measurements using `std::chrono::high_resolution_clock` C++11 library.

3.2 Methodology of tests

The methodology of our research is similar as in [11]. We run the testbed application automatically for different values of power limit and read the execution time, the total energy consumption and the average power for the run. For each power limit we execute a series of five test runs and average the result. We observed that when Nvidia GPU temperature raises the energy consumption and average power for the same power limit settings is higher. To eliminate the impact of that phenomenon on our test runs firstly we execute a series of dummy tests which warm the GPU up. This ensures the same conditions for all test runs regardless of the position in a sequence.

In contrast to CPU, where the default settings is no power cap, on the GPU the default power limit is lower than the maximal available power limit. Another difference is that on Intel CPUs it is possible to force the power cap which is lower than the idle processor power request while on Nvidia GPU has the minimal power limit defined relatively high. Therefore, we run a series of tests starting from the maximum power limit with a 5W step until we reach the minimal possible power limit. We refer the results of energy consumption for each result to the values obtained for the power limit set to the default value. For instance, for the Nvidia GeForce RTX 2080 the maximal power limit is 240W, the minimal power limit is 125W and the default power limit is 215W.

4 Experiments

4.1 Testbed GPUs and systems

The experiments have been performed on two testbed systems with modern Nvidia GPUs based on Pascal and Turing architecture. The first tested GPU is Nvidia GeForce GTX 1070 (Pascal architecture) with 1920 Nvidia CUDA cores with 1506 MHz base Core frequency, 8 GB of GDDR5 memory and Power Limit range between 95W and 200W. The other system is equipped with Nvidia GeForce RTX 2080 (Turing) with 2944 Nvidia CUDA cores with 1515 MHz base Core frequency, 8 GB of GDDR6 memory and Power Limit Range between 125W and 240W. Table 1 collects all details including CPU models and CUDA version used in both testbed systems.

4.2 Testbed applications and benchmarks

For the experiments we have selected four representative computational workloads with different computation intensity. Three of the testbed applications were selected from the well known NAS Parallel Benchmark (NPB) collection implemented in CUDA to be used on GPU [5].

Table 1: Testbed environments used in the experiments

System	CPU model	GPU model	RAM	CUDA version	GPU Default Power Limit	GPU Power Limit range
GTX	Intel® Core® i7-7700 (Kaby Lake)	Nvidia GeForce GTX 1070 (Pascal)	16 GB	10.0	190W	95W - 200W
RTX	Intel® Xeon® Gold 6130 (Skylake-X)	Nvidia GeForce RTX 2080 (Turing)	32 GB	10.1	215W	125W - 240W

The kernels that were used for the tests included: Block Tri-diagonal solver (BT), Scalar Penta-diagonal solver (SP) and Lower-Upper Gauss-Seidel solver (LU). Similarly as in the CPU version, GPU NPB may also be run for various input data sizes represented by classes A, B, C, D, E, S and W. Classes A, B, W and S were not preferred in our experiments as the execution times of the kernels with such input data size were too short and - as it was mentioned in the previous section, we could not accept the measurement error caused by minimal sampling resolution equal to 1s. On the other hand classed D and E were not able to allocate enough data space due to GPU memory limitations. We decided to use all of three kernels with the same input data size – class C.

The fourth application used in the experiments was cublasgemm-benchmark [15] implementing General Matrix Multiplication (GEMM) kernel. We have modified the benchmark application to execute a series of 10 matrix multiplication (MM) operations with given square matrix size. To fulfill our requirement regarding long enough testbed application execution times we decided to use the input square matrix sizes of at least 16384x16384. Due to GPU memory limitations for the system with the GTX card the maximal matrix size we were able to run was only 24576x24576. For the RTX system we could run 32768x32768.

The total execution times (t) as well as total energy (E) and average power (P) consumed by each of aforementioned applications run on both testbed systems (GTX and RTX) in the default configuration of Power Limit were collected in Table 2. These values were our baseline for the relative energy savings and performance drop calculations.

4.3 Tests results

The test results presented in the figures are organized in columns which represent the testbed workload type (GEMM for various matrix size on the left and NPB kernels on the right) and in rows which represent the observed physical magnitudes in the order as follows: relative Energy savings evaluated in percent, relative performance drop evaluated in percent, normalized energy-time product and the total average power consumption. Each figure in one column has the same horizontal axis which is the Power Limit level evaluated in Watts. All relative results are compared to the results obtained for the default Power Limit which is 190W for GTX and 215W for RTX system. It is important to note that GEMM and NPB kernels present different power usage profiles.

Table 2: Baseline values of total execution time, total energy and average power consumption for the default Power Limit settings.

System		GTX			RTX		
App		t [s]	E [J]	P [W]	t [s]	E [J]	P [W]
BT		150,1	13493	87,6	73,9	14851	198,0
LU		44,4	3756	87,4	27,35	4972	184,2
SP		24,2	2518	109,5	17,24	3345	196,8
GEMM	16k	34,3	3030	91,8	26,3	3213	123,6
	24k	94,9	8696	89,7	69,2	8767	125,2
	32k	-	-	-	194,7	27295	135,8

While NPB kernels begin the computation right after the application started, GEMM execution has two clear phases differing in power consumption level. The initial phase is a data preparation phase and the latter is actual MM computation phase. Both phases significantly differ in power consumption which is illustrated in Figure 1 where sample series of test runs with different Power Limit have been presented. The sample we present was collected on the RTX system for GEMM size 16384 and NPB BT kernel.

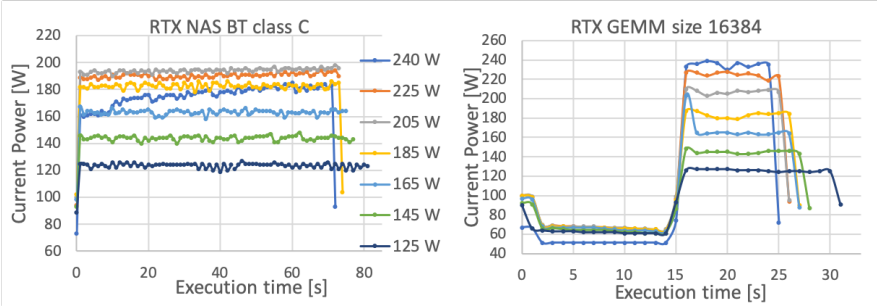

Fig. 1: Sample series of test runs with different Power Limit on RTX system.

Figure 2 presents results obtained for the first testbed system with the GTX 1070 GPU. For GEMM we observed the impact of limiting for the values below 150W. Above 150W the limit is neutral for the computations as the maximum spike power consumed by GEMM on GTX was around 148W. For the lower Power Limit levels we can observe a linear falloff in power consumption. The energy consumption is also decreasing and the maximal energy saving that we could reach for that application-device pair was 15%. That value corresponded with less than 10% of performance drop and was obtained for the power limit value of 110W. Below that value the energy savings are also observable but the benefits of limiting the power consumption are worse as the performance drops even more (up to 20%) while only 10% of energy can be saved.

While the clear energy consumption minimum seem to be found at 110W, the other target metric suggests that a better configuration would be to set the limit between

115W and 120W. With such a scenario we are able to save almost 13% of energy while sacrificing only 5-6% of performance.

The results for NPB kernels obtained on GTX system are less impressive mostly because the power consumption values while executing these testbed workloads were close or even below the minimum Power Limit level possible to be set on GTX 1070. Only for the SP kernel we can observe some interesting level of energy savings up to 17% for the lowest value of limit. What is more interesting is that the energy was saved with no cost as the performance has not dropped. This might mean that the Base Core frequency which is lowered while limiting the power has no meaningful impact on the SP execution time so the boundary in that case was memory speed.

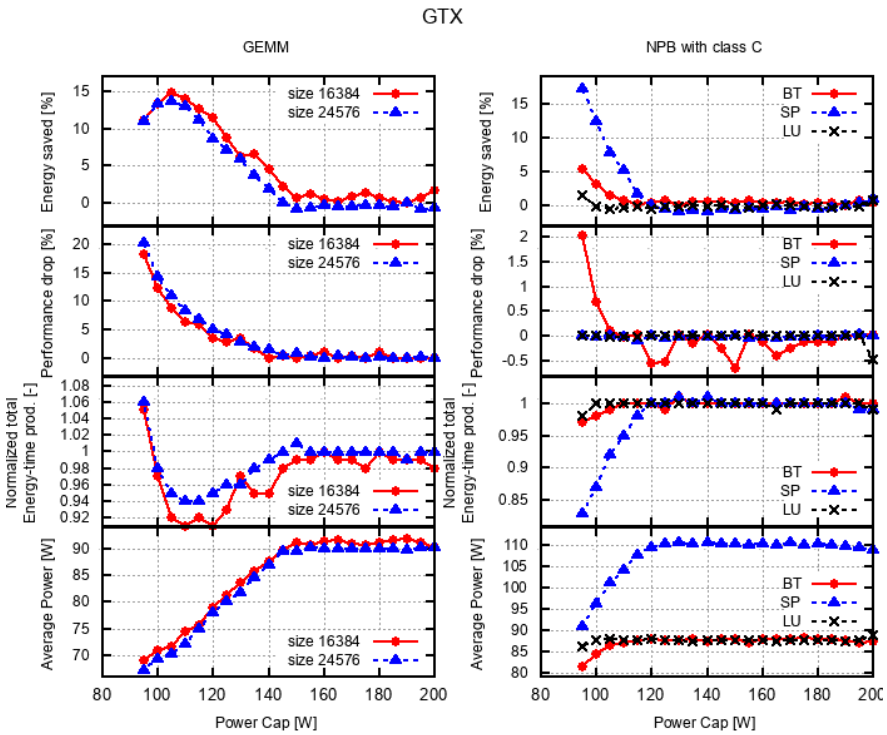

Fig. 2: Results of tests for GTX 1070 for three different problem sizes of GEMM kernel (left charts) and for three NAS Parallel Benchmarks applications run with problem size class C (right charts).

Figure 3 presents results obtained for the testbed system with RTX 2080 GPU. With the same testbed workload RTX system present different energy characteristics than GTX system. Firstly what we observed is that none of the testbed workloads' average power consumption with default settings is below the minimal Power Limit available

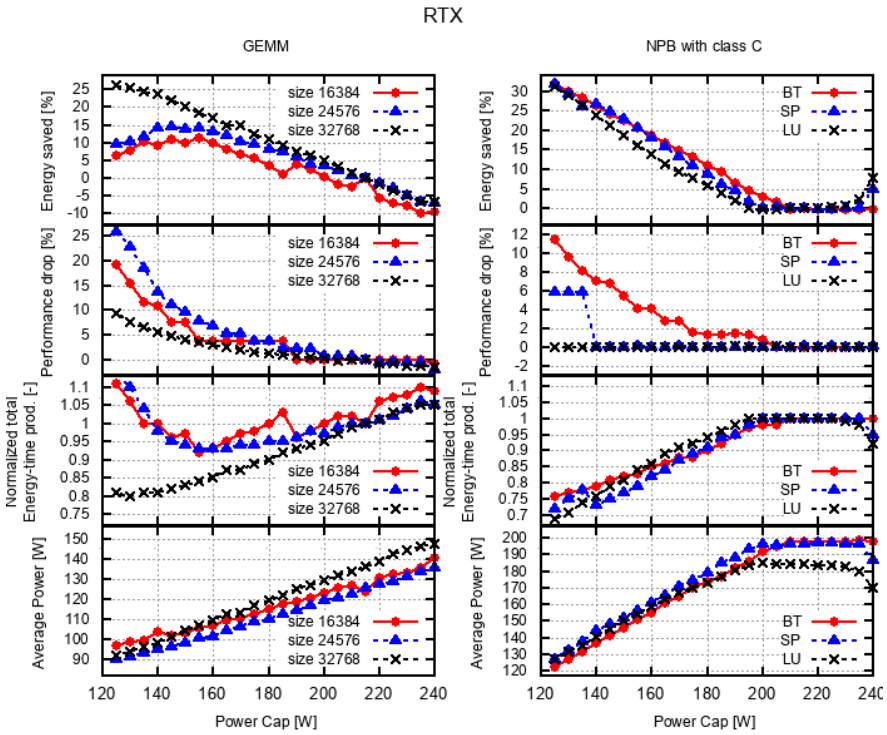

Fig. 3: Results of tests for RTX 2080 for four different problem sizes of GEMM kernel (left charts) and for three NAS Parallel Benchmarks applications run with problem size class C (right charts).

one RTX system (125W). This implies having more abilities to adjust the power and energy consumption level.

For GEMM we observe a linear falloff in power consumption in the whole available power limit range. For the same matrix sizes as was used in GTX system we observe similar energy characteristics with clear energy minimum (up to 15% of energy saved) for the limits in range 140W - 160W and corresponding performance drop less than 10%. On the other hand, the other target metric which is energy-time product suggests its minimum for the Power Limit in range 160W - 170W where we save up to 10% of energy loosing only 5% of performance. More interesting results were observed for the matrix size that was possible to run only on the RTX system which is 32786x32786. When the input data size was increased the energy savings are reaching 25% while the corresponding performance drop is only 10%. We see that the performance characteristics slope is less than for the smaller matrices so the energy savings are more profitable. This may suggest that for big input data when the memory is a bottleneck lowering the power using Power Limits available on the Nvidia GPUs is a really good way to lower the costs of computations.

For all three NPB kernels we observed similar energy characteristics which shows that below the 200W Power Limit we obtain stable reduction of energy consumption with the maximum 30% of energy savings. The performance drop for that type of testbed workloads seem to encourage for limiting the power as its maximal value is only 12% and was observed only for the BT kernel. We see that for SP and LU kernels the performance drops are much smaller, even close to 0%. This confirms the observations from the GTX system which may suggest that the NPB kernels are more memory than computation bound.

5 Conclusions and future work

Our research presented in this paper showed that it is possible and even worth to lower the energy consumption by using software power caps in modern HPC systems. After our first research focused on Intel CPUs [10,11] we tested another popular in HPC manufacturer and devices: Nvidia's GPUs. Using various testbed workloads with different input data size, different computation intensity and also non trivial power consumption profile we tested two modern Nvidia's GPUs.

Our research showed that:

1. depending on the workload type it is possible to reach up to 30% of energy savings using Power Limits available on Nvidia's GPUs while corresponding performance drop evaluated in percent is usually smaller than benefits of lowering the costs,
2. power limiting in order to minimize the costs is a non-trivial task as the Power Limits for which we observed the maximum of energy savings was specific to the application-device pair and was not the minimal possible Power Limit value what could suggest the intuition,
3. the second target metric we used – energy-time product, allows for finding even more profitable configurations of Power Limits as the energy savings are close to maximal possible but the performance drop is significantly lower.

In the future we plan to extend out research with more tests with other types of workloads specific to GPU Deep Learning benchmarks or training. We also aim at a hybrid power capping approach (RAPL + NVML) for hybrid applications run on hybrid (Intel CPU + Nvidia GPU) systems.

References

1. Yuki Abe, Hiroshi Sasaki, Martin Peres, Koji Inoue, Kazuaki Murakami, and Shinpei Kato. Power and performance analysis of gpu-accelerated systems. In *Presented as part of the 2012 Workshop on Power-Aware Computing and Systems*, Hollywood, CA, 2012. USENIX.
2. Robert A. Bridges, Neena Imam, and Tiffany M. Mintz. Understanding gpu power: A survey of profiling, modeling, and simulation methods. *ACM Comput. Surv.*, 49(3):41:1–41:27, September 2016.
3. Emmanuell D Carreño, Adiel S Sarates Jr, and Philippe O A Navaux. A mechanism to reduce energy waste in the post-execution of gpu applications. *Journal of Physics: Conference Series*, 649(1):012002, 2015.

4. Pawel Czarnul, Jerzy Proficz, and Adam Krzywaniak. Energy-aware high-performance computing: Survey of state-of-the-art tools, techniques, and environments. *Scientific Programming*, 2019:8348791:1–8348791:19, 2019.
5. Jörg Dümmler. Npb-cuda. Technische Universitat Chemnitz, https://www.tu-chemnitz.de/informatik/PI/sonstiges/downloads/npb-gpu/index.php.en.
6. R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong. Effects of dynamic voltage and frequency scaling on a k20 gpu. In *2013 42nd International Conference on Parallel Processing*, pages 826–833, Oct 2013.
7. Sunpyo Hong and Hyesoon Kim. An integrated gpu power and performance model. *SIGARCH Comput. Archit. News*, 38(3):280–289, June 2010.
8. S. Huang, S. Xiao, and W. Feng. On the energy efficiency of graphics processing units for scientific computing. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–8, May 2009.
9. T. Komoda, S. Hayashi, T. Nakada, S. Miwa, and H. Nakamura. Power capping of cpu-gpu heterogeneous systems through coordinating dvfs and task mapping. In *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pages 349–356, Oct 2013.
10. A. Krzywaniak, J. Proficz, and P. Czarnul. Analyzing energy/performance trade-offs with power capping for parallel applications on modern multi and many core processors. In *2018 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 339–346, Sep. 2018.
11. Adam Krzywaniak, Pawel Czarnul, and Jerzy Proficz. Extended investigation of performance-energy trade-offs under power capping in hpc environments. accepted for International Conference on High Performance Computing & Simulation (HPCS 2019), Dublin, Ireland, in press.
12. Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M. Aamodt, and Vijay Janapa Reddi. Gpuwattch: Enabling energy optimizations in gpgpus. *SIGARCH Comput. Archit. News*, 41(3):487–498, June 2013.
13. P. Libuschewski, P. Marwedel, D. Siedhoff, and H. Müller. Multi-objective, energy-aware gpgpu design space exploration for medical or industrial applications. In *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, pages 637–644, Nov 2014.
14. Jan Lucas and Ben Juurlink. Mempower: Data-aware gpu memory power model. In Martin Schoeberl, Christian Hochberger, Sascha Uhrig, Jürgen Brehm, and Thilo Pionteck, editors, *Architecture of Computing Systems – ARCS 2019*, pages 195–207, Cham, 2019. Springer International Publishing.
15. He Ma. cublasgemm-benchmark. University of Guelph, Canada, https://github.com/hma02/cublasgemm-benchmark.
16. Sparsh Mittal and Jeffrey S. Vetter. A survey of methods for analyzing and improving GPU energy efficiency. *CoRR*, abs/1404.4629, 2014.
17. Krzysztof Rojek. Machine learning method for energy reduction by utilizing dynamic mixed precision on gpu-based supercomputers. *Concurrency and Computation: Practice and Experience*, 31(6):e4644, 2019. e4644 cpe.4644.
18. K. Tsuzuku and T. Endo. Power capping of cpu-gpu heterogeneous systems using power and performance models. In *2015 International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, pages 1–8, May 2015.