2025 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) | 979-8-3315-2643-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/IPDPSW66978.2025.00132

Improving energy efficiency of HPC applications using unbalanced GPU power capping

Albert d'Aviau de Piolant*, Hayfa Tayeb*†, Berenger Bramas†, Mathieu Faverge*,
Abdou Guermouche*, Amina Guermouche*

* University of Bordeaux, CNRS,
Bordeaux INP, Inria, LaBRI
Talence, France

† ICube Lab.
University of Strasbourg, CNRS, Inria
Strasbourg, France

***Abstract*—Energy efficiency represents a significant challenge in the domain of High-Performance Computing (HPC). One potential key parameter to improve energy efficiency is the use of power capping, a technique for controlling the power limits of a device, such as a CPU or GPU.**

In this paper, we propose to examine the impact of GPU power capping in the context of HPC applications using heterogeneous computing systems. To this end, we first conduct an extensive study of the impact of GPU power capping on a compute intensive kernel, namely matrix multiplication kernel (GEMM), on different Nvidia GPU architectures. Interestingly, such compute-intensive kernels are up to 30 % more energy efficient when the GPU is set to 55-70 % of its Thermal Design Power (TDP). Using the best power capping configuration provided by this study, we investigate how setting different power caps for GPU devices of a heterogeneous computing node can improve the energy efficiency of the running application. We consider dense linear algebra task-based operations, namely matrix multiplication and Cholesky factorization.

We show how the underlying runtime system scheduler can then automatically adapt its decisions to take advantage of the heterogeneous performance capability of each GPU. The results show that for a given platform equipped with four GPU devices, applying a power cap on all GPUs improves the energy efficiency for matrix multiplication up to 24.3 % (resp. 33.78 %) for double (resp. single) precision.

***Index Terms*—Energy Efficiency, Heterogeneous Computing, Power capping**

I. Introduction

High-performance computing (HPC) applications are optimized to run fast. This is achieved through a variety of software and hardware parallelization and optimization techniques, from Single Instruction Multiple Data (SIMD) using vector extensions in CPUs, to multi-core implementations, to heterogeneous computing using both CPUs and accelerators such as GPUs. Task-based runtime systems have also shown great potential in various applications [1], [2]. In this context, the developer expresses the computations through a directed acyclic graph (DAG) of tasks. The runtime system is then responsible for the execution of the DAG, by taking scheduling decisions and moving data across all available resources.

Using heterogeneous resources comes at a high energy cost. As a matter of fact, supercomputers consume a lot of power - tens of megawatts in the case of the largest supercomputers [3]. Energy consumption reduction has often been neglected by the HPC community, which traditionally focuses primarily on raw computational performance.

More recently, the energy consumption of machines and applications became a subject of research in the scientific community, and energy-efficient solutions emerged as key to green computing. Several state-of-the-art techniques have been proposed to reduce power consumption such as Dynamic Voltage and Frequency Scaling (DVFS) [4] and Dynamic Uncore Frequency (UFS) for CPUs, and power capping and undervolting with fault tolerance for GPUs [5].

Power capping is a technique to lower the power budget of a device (CPU or GPU). It allows one to set a power limit that the device will not exceed while being used. Choosing the right power cap to apply can be tedious, especially when considering heterogeneous architectures where power capping can be applied to both CPUs and GPUs. We first observe that on such architectures, the energy efficiency is maximized when applying a power cap on the GPU. This means that introducing a slowdown, even if it is high (19.71 %), improves energy efficiency (23.17 %). However, such a slowdown is not always acceptable. Therefore, we propose to set different power capping values on the GPUs, to study their impact and provide a trade-off between performance and energy efficiency.

To benefit from the different GPUs, we use StarPU, a task-based runtime system that will, thanks to its scheduler, take into account the performance of each GPU (considering their different power capping) to place the computations such that the execution time is minimized. We test our configuration on three different architectures, two different dense linear algebra operations (namely GEMM and POTRF) using single and double precision computations. We show that power capping all GPUs provides the best energy efficiency (24.3 %), but at the cost of a drop in performance (-26.41 %) compared to the default configuration. Applying a power cap on a subset of GPUs can provide a good trade-off for performance (-12.32 %) and for energy efficiency (9.28 %). Finally, we also study the additional impact of CPU power capping. We show that, since the CPU is less used when most computations take place on the GPU, applying a power cap on the CPU increases the energy efficiency by roughly 21 % with no additional impact on performance. This represents a total energy efficiency of 42.45 Gflop/s/Watt, compared to an initial 19.5 Gflop/s/Watt with no CPU or GPU power capping, representing a total improvement of 108.2 %.

The paper is organized as follows. We first provide a motivation to our work (Section II). Then we present the task-based runtime system that we used and the library providing the GEMM and POTRF operations (Section III). After that, we introduce the target architectures and the methodology used to set the power cap and measure the energy consumption (Section IV). Section V presents the impact of power capping on all the studied configurations. Finally we present the existing works related to power capping (Section VI) before concluding.

II. Motivation

This work considers a number of metrics. Performance, represented in flop/s, is the primary objective in HPC. Energy consumption is represented in Joule. Energy efficiency, as defined by the Top 500 [3], is the performance delivered per Watt of electrical power consumed, thus expressed as Gflop/s/Watt. Consequently, the higher the energy efficiency, the more effectively the available power is utilized. In this work, we consider these three metrics together, to have a better understanding of the device behavior. Nevertheless, our goal is focused on the optimization of energy efficiency.

In order to motivate our idea of applying different power capping limits to different GPUs, we initially conducted a study to examine the impact of power capping on the energy efficiency on three different GPU architectures namely one NVIDIA TESLA V100 and two NVIDIA TESLA A100. The NVIDIA TESLA A100 comes in two variations: PCIe, which has a TDP of 250 Watt, and SXM4, which has a TDP of 400 Watt. We used `sgemm` and `dgemm` kernels from cuBLAS version 11.8. Figure 1 depicts the energy efficiency, the performance and the energy consumption on different matrix sizes (one single large tile) on A100-SXM4 with single precision and double precision. Table I summarizes the best results across all architectures with regard to energy efficiency. In all experiments, we vary the power cap from the lowest possible limit on the GPU, to no power capping at all (100 % on Figure 1) with a step of 2 %.

It shows that the best energy efficiency is reached when the power cap is set below the TDP, regardless of the matrix size and the precision. More specifically, the best energy efficiency is reached when the power cap is set to 54 % (resp.

TABLE I: Best configuration for energy efficiency depending on GPU and precision.

GPU	precision	matrix size	power cap (% TDP)	Eff. saving (% default)
A100-SXM4	single	5120	40	27.76
	double	5120	54	28.81
A100-PCIe	single	5760	60	23.17
	double	5760	78	10.92
V100-PCIe	single	5120	58	20.74
	double	5120	60	18.52

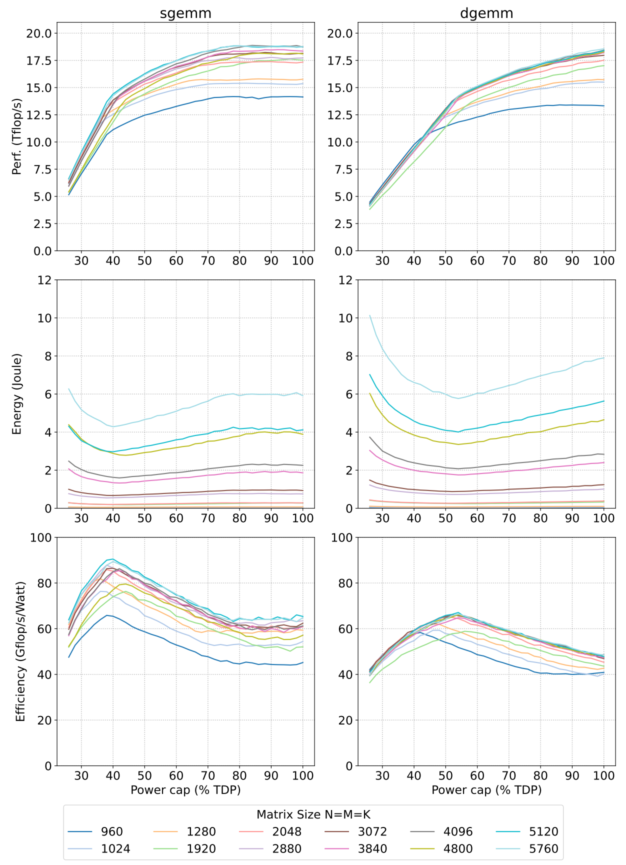


Fig. 1: Power capping impact on energy efficiency, performance and energy consumption considering cuBLAS on different matrix sizes on A100-SXM4. The power cap varies from 104 to 400W.

40 %) of TDP on large matrix sizes for double (resp. single) precision. This represent a saving of 28.81 % (resp. 27.76 %) compared to the energy efficiency obtained when no power cap is applied. Bigger matrix sizes tend to have better energy efficiency due to their higher performance compared to smaller ones. Smaller matrices do not fill the GPU workload enough. As a consequence, the performance is lower, and the consumed power is not low enough to have a positive impact on the energy efficiency.

This observation indicates that even for compute-intensive applications running on GPUs, **faster is not equivalent to being energy efficient**. On the contrary, when energy efficiency is the target, it is better to reduce the power limit. However, as shown in Figure 1, applying a power cap of 54 % of TDP impacts performance with an observed slowdown of 22.93 % (on double precision) compared to no power cap. This implies that if we apply the power cap that provides the best energy efficiency to all available GPUs, then the performance loss may not be acceptable for some users. We rather propose, in this study, to dedicate some GPUs to energy efficiency, and other GPUs to performance. We assume that some computations have a higher priority than others, and delaying them would impact the whole execution performance. As a consequence, these computations must run on the fastest GPUs whenever possible, while other operations, which are less critical, will run on the most energy-efficient GPUs. As such, users would be able to choose the best configuration for their needs.

III. Runtime system and target applications

In this section, we first introduce the runtime system and the underlying scheduling. We then proceed to present the applications that we consider to be use cases.

A. *Task-based Runtime Systems*

Taking advantage of heterogeneous platforms can be a tedious process. Runtime systems –a software layer that abstracts the hardware complexity– represent a promising solution. We are particularly interested in task-based runtime systems, which consider the application as a DAG of tasks and efficiently schedule tasks and data migrations across all available processing units, including CPUs and accelerators. Several runtime systems have emerged [6]–[8]. Among them is StarPU [9], a task-based runtime system designed specifically for parallelizing algorithms on heterogeneous architectures. It provides an efficient interface that manages the application as a DAG of tasks, while accepting different implementations for each task to enable execution on different architectures such as CPU, and GPU, but also other devices such as FPGA which are not considered in this work.

B. *Task Scheduling on Heterogeneous Systems*

The utilization of an entire heterogeneous platform for the execution of our applications presents a multitude of challenges that must be taken into account in our investigation. We use StarPU, which can automatically build performance models to estimate the execution time of tasks on a given device. These estimations are provided by StarPU through a history-based or a regression-based performance model [10], [11]. In addition, StarPU provides a set of predefined scheduling policies that represent the literature.

Among the multiple policies, the Dmdas scheduler [12] considers task execution performance models (provided by StarPU), the time required to transfer data to the processor, and sorts the queues based on the task priority value specified by the application expert. This is why in the upcoming experimental study, we will use Dmdas.

Given that we run the applications on different power configurations, which introduces additional heterogeneity into the machine, it is necessary to assess the capacity of the scheduler to be aware of the change in power levels of the GPUs. Indeed, the performance models are calibrated following each modification to the power capping settings on the machine. Thus, the scheduler is implicitly informed of the changes. For instance, if a GPU is set to the lowest power level, the performance models for tasks on that device will indicate slower execution times than those on devices operating at maximum power. Therefore, when tasks are assigned across the devices, the scheduler will inherently allocate fewer tasks to be executed on the device with reduced processing speed.

C. *Target Applications*

A significant number of problems in areas such as physics, engineering, and data analysis are expressed in terms of linear operations on vectors and matrices, most of which contain non-zero values. Some common examples include solving systems of linear equations, matrix factorization, eigenvalue problems, and matrix multiplication. The research community aims to improve the optimization of these algorithms for high-performance computing, ensuring that they run efficiently on modern hardware. Chameleon [13] is a framework that provides routines for solving a variety of mathematical problems, including dense, general systems of linear equations, symmetric, positive definite systems of linear equations, and linear least squares problems. These routines use LU, Cholesky, QR, and LQ factorizations. Chameleon allows the use of task-based programming models by tiling a dense matrix into multiple data tiles on which tasks operate, thus parallelizing them on different resources. It also allows GPUs to be used by kernels provided by cuBLAS. It defines the user priorities for each task within a given routine. These priorities are optimized offline by experts.

In this paper, we select two widely used operations that have distinct DAGs: matrix multiplication (GEMM) and Cholesky factorization (POTRF). The DAG of GEMM operation contains numerous identical compute-intensive tasks, as well as a high level of parallelism that enables the use of all available resources on a given platform. This is representative of numerous other HPC applications. As the GEMM kernel is highly parallel, GPUs are more appropriate to run it than CPUs. Therefore, the scheduler plays a critical role in ensuring good workload balancing in such operations. The DAG of POTRF operation has $\frac{Nt(Nt+1)(Nt+2)}{6}$ vertices and $\frac{(Nt-1)Nt(Nt+1)}{2}$ edges for $Nt \times Nt$ tiles, in particular, it has $\frac{Nt(Nt-1)(Nt-2)}{6}$ GEMM tasks. The critical path comprises numerous tasks that are executed on the CPU. The scheduling of such DAGs on heterogeneous systems is a challenge. The insight provided by experts on task priorities, as in the case of Chameleon, can inform scheduling decisions.

IV. Experimental Setup and Methodology

In light of the observations presented in Section II, our objective is to study the impact of static power capping of GPUs. In particular, we aim to determine whether limiting the GPU's power to below its maximum can enhance the energy efficiency of compute-intensive HPC applications on heterogeneous computing platforms. The objective is to create further heterogeneous resources and to seek for better energy efficiency in a multi GPU configuration. This section presents an overview of the various modern platforms on which the study is conducted, as well as the experimental protocol that is followed.

A. Platforms Details

Setting power capping on processing units requires root access privileges to the machines, which may not always be granted. Such constraints prevented us from conducting experiments on the most recent Nvidia GPU model, namely the H100. Therefore, the experiments were conducted on nodes from Grid'5000 [14], which is a highly reconfigurable and controllable infrastructure that allows for root access privileges. In the present experimental study, we propose setting static power capping to different Nvidia GPU models, each of which is integrated into a different platform. The hardware specifications of the nodes used for the experiments are summarized below.

- **24-Intel-2-V100**: The architecture is composed of two Intel Xeon Gold 6126 (Skylake-SP) processors, each with 12 cores running at 2.60GHz, 192GB of memory, and two Nvidia Tesla V100-PCIE-32GB GPUs. All the experiments were run on the node "chifflot-7", situated at the Lille Grid'5000 site.
- **64-AMD-2-A100**: The architecture is composed of two AMD Zen2 AMD EPYC 7452 processors, each with 32 cores running at 2.35GHz, 512GB of memory, and two Nvidia A100-PCIE-40GB GPUs. All the experiments were run on the "grouille-1" node, on the Nancy cluster.
- **32-AMD-4-A100**: The architecture is composed of one AMD Zen3 EPYC 7513, with 32 cores running at 2.6GHz, 128GB of memory, and four Nvidia A100-SXM4-40GB GPUs. All the experiments were run on "chuc-1" and is situated at the Lille Grid'5000 site.

The 24-Intel-2-V100 platform have GCC v10.2.1 and CUDA V11.5. The remaining two platforms have GCC v12.2.0 and CUDA V11.8. All platforms have the Intel MKL library v2020.4.304-4. All available computing resources on all platforms are utilized. Hyperthreading and turboboost are disabled on all CPUs.

B. Applications Configuration

In order to leverage the available computing resources, when using Chameleon, the user must set the appropriate tile sizes according to which the dense matrix is divided into equal tiles. We made an exhaustive benchmarking campaign of different tile sizes studied in Section II and selected the most appropriate sizes for each platform. Table II summarizes the used matrix sizes ($\boldsymbol{N} \times \boldsymbol{N}$) and the corresponding tile sizes ($\boldsymbol{nb}$) for each configuration.

C. Experimental Protocol and Energy Measurement

The objective is to present a comparative analysis of distinct configurations of GPUs under power capping. To this end, we introduce the three key states of interest: the minimum power consumption (P_{min}), the best power consumption (P_{best}), and the maximum power consumption (P_{max}). The minimum and maximum power consumption values are provided by the manufacturer of the GPU. The best power consumption is selected from the empirical study on the GEMM kernel presented in Section II and is the one providing the best energy efficiency (the highest point from the energy efficiency). Our choice is justified by the fact that the GEMM kernel is used extensively, in both the task-based GEMM operation and the Cholesky factorization. Table II provides a summary of the power consumption parameters for the two GPU models used in the study. We study different combinations of the states by power capping the GPUs to see how it affects the energy efficiency of the applications. In the following, the P_{min} state is referred to as `L` for the lowest power, the P_{best} state as `B` for best power, and the P_{max} state as `H` for the highest power.

To illustrate, a configuration designated as `HHLL` indicates that the GPUs 0 and 1 are maintained at their highest default power level, whereas GPUs 2 and 3 are limited to the lowest power. In the experimental study, we conducted a comprehensive analysis of all possible configurations. For instance, when four GPUs were employed, the configuration `HHHB` was evaluated, as were the combinations `HHBH`, `HBHH` and `BHHH`. We found that the variation in results was negligible. To simplify the presentation of the results, we limit the configurations to a single one and omit the combinations. Our objective is to demonstrate the results for configurations that have a varying number of GPUs set to different power levels. In other words, we consider a configuration with one GPU at the highest power level (`HBBB`), two GPUs at the highest power level (`HHBB`), and so on.

While varying different configurations as previously detailed, we run the HPC operations and measure the energy consumption of all processing units on a given platform during the total execution. This data is cumulated in a total energy consumption of the parallelized application. It should be noted that the operations are parallelized as a graph of tasks; however, our energy measurement methodology is not at the granularity of the task, but rather considers the entire application. In order to measure CPU energy consumption, we use the Performance API (PAPI) library [15]. PAPI employs Intel's Running Average Power Limit (RAPL) interface, which monitors the hardware counters on the CPU. We utilize the native events encoded by PAPI to obtain the total number of Joules consumed by all cores and the last level cache on the package. Energy consumption is measured on all resources used by the application (CPUs and GPUs) at the start and end of the execution. The values are then subtracted to determine the total energy consumed. For Nvidia GPUs, we utilize the NVML library, and apply the same methodology. A recent work [16] investigated the error of energy measurement provided by NVML on NVIDIA GPUs, and how a non constant power consumption can lead to incorrect results. Our case study does not face such problem as our applications are mainly composed of chained GEMM kernels, maximizing the GPU workload.

V. Experimental results

This section presents the study of the energy efficiency conducted on two task-based operations, namely GEMM and POTRF. As detailed in Section III, the StarPU runtime system with the scheduler Dmdas is utilized for all runs. We set static

TABLE II: Selection of matrix sizes equal to $\boldsymbol{N} \times \boldsymbol{N}$ suitable for each operation and platform with the appropriate tile size denoted by $\boldsymbol{nb}$. GPU power limits: P_{min} is the low hardware limit on GPU, P_{best} is the power which provide the best energy efficiency on GPU, and P_{max} is the high hardware limit on GPU.

Platform	Operation	N	nb	Precision	GPU $P_{best}(B)$	GPU $P_{min}(L)$	GPU $P_{max}(H)$
24-Intel-2-V100	**GEMM**	43200	2880	double	62 %	100 W	250 W
				single	60 %		
	POTRF	96000	1920	double	56 %		
				single	66 %		
64-AMD-2-A100	**GEMM**	69120	5760	double	78 %	150 W	250 W
				single	60 %		
	POTRF	115200	2880	double	78 %		
				single	60 %		
32-AMD-4-A100	**GEMM**	74880	5760	double	54 %	100 W	400 W
				single	40 %		
	POTRF	172800	2880	double	52 %		
				single	38 %		

power limits for the GPUs on the three target platforms (see Table II), resulting in a variety of configurations. We then measure the performance and energy consumption of the HPC operations under these configurations. Our study examines the impact on the previously mentioned metrics.

The results for the three platforms are presented in the following figures. It is important to note that our analysis is conducted on a heterogeneous platform, and we measure the total energy consumption of all devices on the platform (see the details in Section IV-C). In order to investigate the impact of different GPU power configurations on performance and energy consumption, we calculate the percentage increase or decrease relative to the default setting. For the performance, a positive percentage value indicates speedup, while a negative percentage value indicates a slowdown. For energy consumption, a positive percentage value indicates a reduction in energy consumption, whereas a negative percentage value indicates an increase in energy consumption. The results are analyzed in two stages, according to the numerical precision used for the data of the operations.

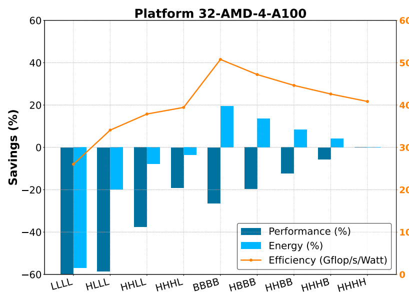
A. Evaluating Energy Efficiency for Task-Based Operations in Double Precision

Figures 2a and 2d present the results on 32-AMD-4-A100. On this platform, there is a notable variation in the available GPU power, with levels ranging from 100 to 400 Watt. We start by studying the impact of setting the GPUs to the lowest possible power cap then gradually increasing the power budget to its highest value. When the two operations are executed on a configuration with four GPUs operating at the lowest power level specified by the TDP (configuration `LLLL`), they both suffer a significant performance degradation, approximately -80 % compared to the default power configurations of the GPUs, namely the configuration `HHHH`. These results are predictable; however, it is noteworthy that excessive slowdown results in significantly higher energy consumption, with a notable increase of approximately 60 % in energy consumption compared to the default configuration. We proceed with our investigation by setting one, two, and three GPUs to their default power levels and maintaining the remaining GPUs at the lowest power levels. The results of this experiment show a gradual improvement in performance, yet the operations remain too slow compared to the default configuration, resulting in a significantly higher energy consumption. The energy efficiency calculated for these cases begins at 26 Gflop/s/Watt (respectively 24 Gflop/s/Watt) for GEMM (respectively POTRF) for the extreme configuration `LLLL`, and increases to 38 Gflop/s/Watt for GEMM (36 Gflop/s/Watt for POTRF) for the configuration `HHHL`. It remains less energy efficient than the default configuration, `HHHH`, which scores at 41 Gflop/s/Watt for GEMM (respectively 38 Gflop/s/Watt for POTRF).

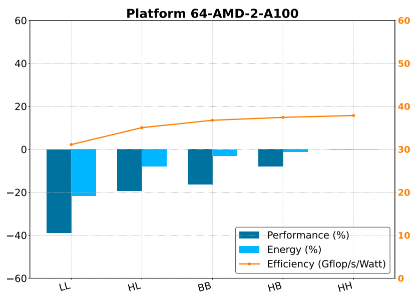
We now proceed by setting the GPUs to the best value selected in the study presented in Table II (P_{best}). In this case, we observe an interesting improvement in the energy efficiency of both operations. The energy efficiency was recorded at 52 Gflop/s/Watt (46 Gflop/s/Watt) for GEMM (POTRF), for the configuration `BBBB`. This indicates that the operations demonstrate approximately 20 % improved energy efficiency when GPUs are constrained to a power level of P_{best} in comparison to the default configuration. The degradation of performance is approximately at 21 %. If the objective is to maximize energy efficiency in terms of the number of floating operations per second per Watt consumed, then a power capping configuration of this type is recommended.

We proceed with our investigation by setting one, two, and three GPUs to their default power levels and maintaining the remaining GPUs at the best power level to determine if we can improve energy efficiency while maintaining performance at an acceptable level. The number of GPUs set to the default power level has a direct impact on the performance of the operations and the energy efficiency. However, even with a single GPU capped at `B` and the remaining GPUs set at the default level (`HHHB`), we observed a 5 % increase in energy efficiency, from 40 to 42 Gflop/s/Watt, for GEMM, and a 1 % improvement for POTRF.

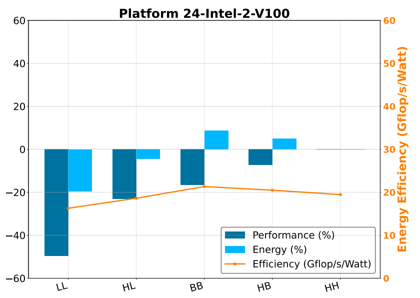
Figures 2b and 2e show the results on the platform 64-AMD-2-A100. The results are not as compelling on this platform. The default configuration remains the most energy-efficient option, outperforming `BB` by 5 % for GEMM and 1 %



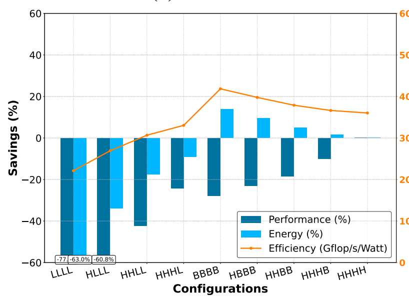
(a) DGEMM



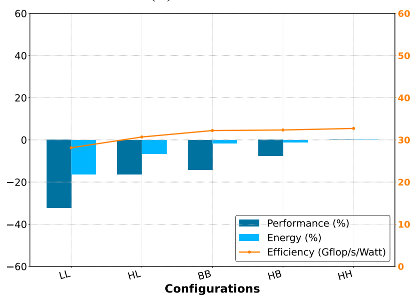
(b) DGEMM



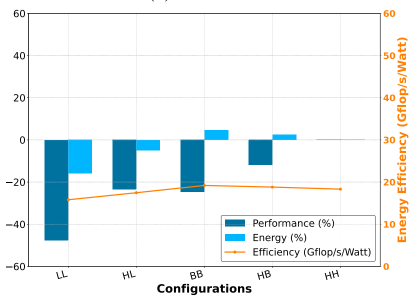
(c) DGEMM



(d) DPOTRF



(e) DPOTRF



(f) DPOTRF

Fig. 2: Performance and energy analysis for GEMM (Top) and POTRF (Bottom) operations on three platforms in **double** precision. The left y-axis (blue bars) shows the percentage change in performance and total energy consumption relative to the default GPU power configuration (H). Positive values indicate a performance gain or energy savings, while negative values represent performance loss or increased energy consumption. The right y-axis (orange line) represents energy efficiency, measured in Gflop/s/Watt. Higher values reflect better trade-offs between performance and energy.

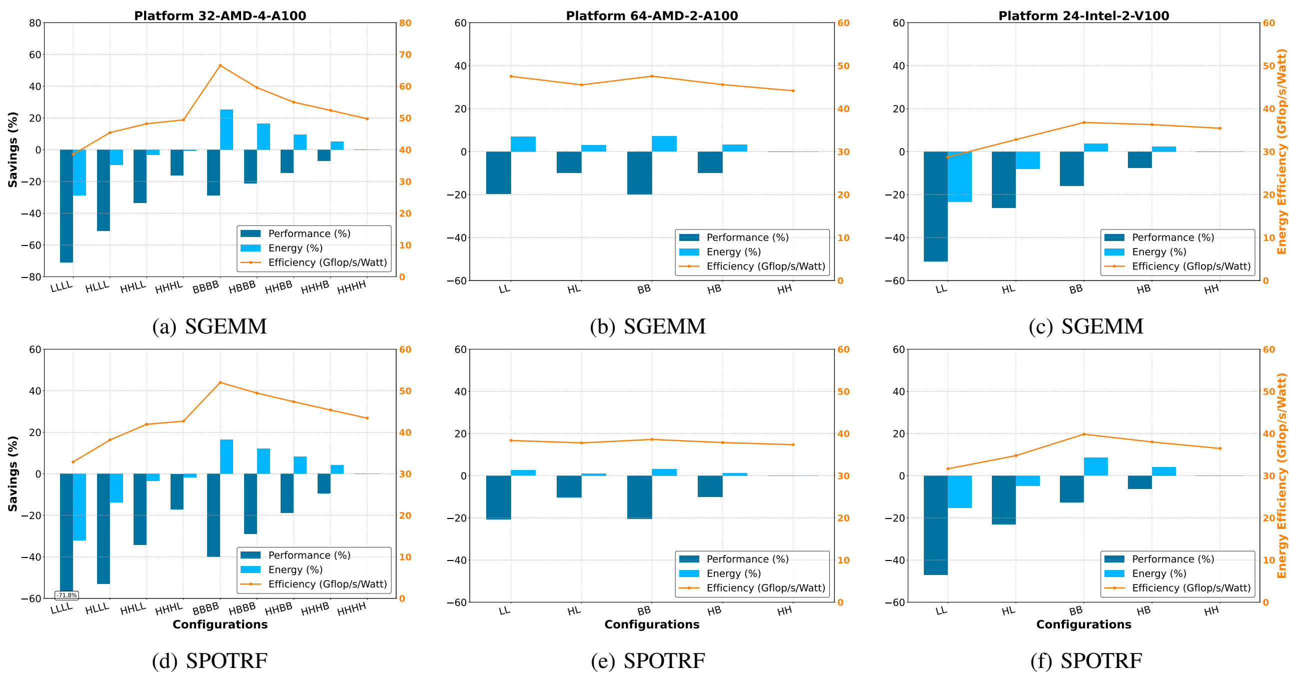
for POTRF. All other configurations demonstrate a notable decline in performance, with a range of 4 % to nearly 40 % for the GEMM for configuration `LL`. For the configuration `HB`, we note a decline in performance of approximately 10 %. It can be observed that the energy efficiency follows a similar trend, with a slight decrease of approximately 1 to 2 Gflop/s/Watt in comparison to the default configuration. It is also worth noting that this platform has two A100 GPUs with power levels that range between 150 and 250 Watt, unlike 32-AMD-4-A100 which has 4 GPUs with the power ranging from 100 to 400 Watt. As a consequence, the best power cap provided by Table II is in fact close to the minimum possible power. To illustrate, in the case of GEMM, the P_{min} is 150 Watt while the P_{best} is 195 Watt. Therefore, there is insufficient power difference between configurations L, B, and H to demonstrate energy efficiency. Furthermore, in contrast to the experiments presented in Section II, where only the GPUs were utilized, in this case, the two CPUs are also employed (as the GPUs are less numerous, and less powerfull). Consequently, the CPU impact on power consumption is significant. For instance, considering GEMM operations, the processes power consumption represents 37 % of the total power consumption on 64-AMD-2-A100 while on 32-AMD-4-A100 it is only 9 %. All of these factors contribute to the difficulty of assessing the impact of the GPU power capping, as it can be cancelled due to the power consumption of both CPUs. A more detailed discussion on the impact of CPU consumption is presented in Section V-C.

Figures 2c and 2f show the results on the platform 24-Intel-2-V100. Energy efficiency shows a similar trend to 32-AMD-4-A100. The peak in `BB` improves energy efficiency from 19.5 to 21.3 Gflop/s/Watt (a 9 % improvement for GEMM and 5 % for POTRF). For `HB`, GEMM (POTRF) saves 5 % (2.6 %) of total energy with a 7 % (11 %) decline in performance. The reason behind this behavior is the same as 64-AMD-2-A100: there are less GPUs that consume less power, thus the impact of CPU power consumption is higher.

B. Evaluating Energy Efficiency for Task-Based Operations in Single Precision

Figure 3 shows the results of the same operations and problem sizes in single precision. The results show that the configurations with GPUs power capped to P_{best} are more energy efficient overall, with less performance degradation than the results with double precision. We can also observe higher energy efficiency when using lower precision. In the single precision case, the arithmetic intensity is higher while the memory footprint is lower than in the double precision case. This reduces the impact of data transfers leading to a better utilization of the GPU devices.

On the platform 32-AMD-4-A100, when power capping all GPUs (`BBBB`), we observed an improvement of 33.78 % in



(a) SGEMM (b) SGEMM (c) SGEMM

(d) SPOTRF (e) SPOTRF (f) SPOTRF

Fig. 3: Performance and energy analysis for GEMM (Top) and POTRF (Bottom) operations on three platforms in **single** precision (refer to the detailed caption in Fig. 2).

energy efficiency for the GEMM, a nearly 25 % reduction in energy consumption with a 28.6 % decline in performance. Including GPUs at `H` power setting offers a trade-off between performance and energy savings. For instance, for GEMM, `HHBB` saves 9 % energy with a 14 % performance drop, improving energy efficiency by 10 % (49.7 to 55 Gflop/s/Watt). Other configurations like `HHHB` shows an even lower slowdown but at the cost of a lower energy efficiency. The results on the 64-AMD-2-A100 platform are noteworthy (Figures 3b and 3e), particularly given that the configurations `LL` and `BB` are at the same level of power – 60 % corresponding to 150 Watt. This is explained by the results presented in Section II, which demonstrate that the cuBLAS GEMM kernel in single precision is more energy efficient at low levels of GPU power. Finally, the results on the platform 24-Intel-2-V100 are promising since the efficiency is improved by 3.8 % for `BB` compared to `HH`. As previously stated in the case of double precision, it is suspected that for such platforms with two CPUs and two GPUs, the impact of GPU power capping can be cancelled by the power consumption of the CPUs.

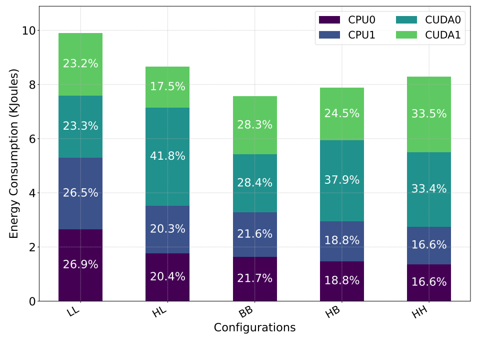
C. Impact of the energy consumption of CPUs

Figure 4 illustrates the proportion of energy consumption across all available devices. We can observe that the consumption of the different processing units varies both relatively and in absolute terms. This is also true for the CPUs, even though we did not change their power limits. This behavior is a side effect of the GPU power capping and scheduling decisions. The scheduling policy favors the fastest processing unit and may even leave slower units idle in some cases. Consequently, during normal execution, the GPU handles significantly more tasks than the CPUs, particularly for GEMM tasks.

Our hypothesis is that the power cap on the CPUs could be beneficial in improving efficiency, given that they are not as crucial for overall performance. As a matter of fact, since the CPU handles less tasks than the GPU (23 % of the total number of tasks for GEMM on 24-Intel-2-V100 on default configuration), reducing its power cap should have a minimal impact on performance. Note that, we are only able to do it on the 24-Intel-2-V100 platform, because we were not able to use power capping on AMD CPUs, which are the only available platforms with A100 GPUs on Grid'5000.

For this part, we want to set a low power cap on the CPU to have an important impact on the results. As such, both CPUs power cap is set at 48 % of the TDP of each CPU (60 Watt over 125). A power limit under this value has shown stability issues. Figure 5 compares the scenario when no CPU power capping is applied with the scenario when CPU power capping is applied (the former is equivalent to the value computed in Figures 2c, 2f, 3c, and 3f).

First of all, power capping CPUs has no impact on performance. As a matter of fact, no overhead was introduced in all configurations. We also observe an overall improvement in energy efficiency across all configurations, regardless of the operation and precision. For instance, when power capping both CPUs, the configuration `BB` provides 21.5 % more energy savings than no CPU power cap for DGEMM. The same behavior is observed when the GPUs are not power


(a) DGEMM

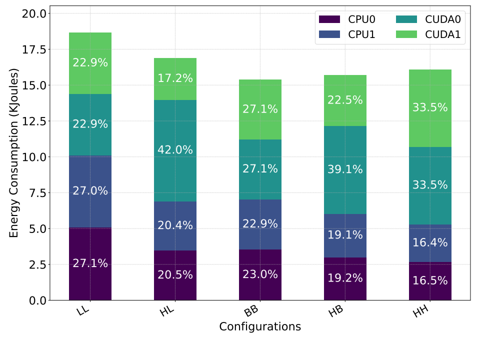

(b) DPOTRF

Fig. 4: Total energy consumptions per configuration on the platform 24-Intel-2-V100 for GEMM (Top) and POTRF (Bottom) in double precision. Detailed energy consumption percentages by device.

capped (HH) with an improvement of 15.2 % for DGEMM. Note that we tried to cap only one CPU but the observed energy efficiency was lower (around 10 % for one CPU). This behavior comes from the fact that 18 % of the tasks are transferred from the CPUs to the GPUs. This demonstrates that, since the scheduler has no knowledge of energy efficiency and therefore does not make decisions with this objective in mind, manually lowering the power of the CPU, as we did, is a way to control the CPU's overhead. However, this process should be automated.

D. *Summary of the results*

Figure 6 summarizes the energy efficiency on all three platforms with additional tile sizes. Note that Figure 6c depicts the efficiency when the CPU is power capped as explained in Section V-C. From figure 6, similar conclusions can be drawn on all different tile sizes. The results show that in most cases, applying a power cap to all GPUs (configuration BBBB) provides the best energy efficiency. Moreover, even if only a subset of GPUs is power capped, we observe an improvement in energy efficiency. Finally, power capping is more beneficial when using lower precision computations due to the higher efficiency of GPUs for these precisions.

The overall conclusions of the experiments are:

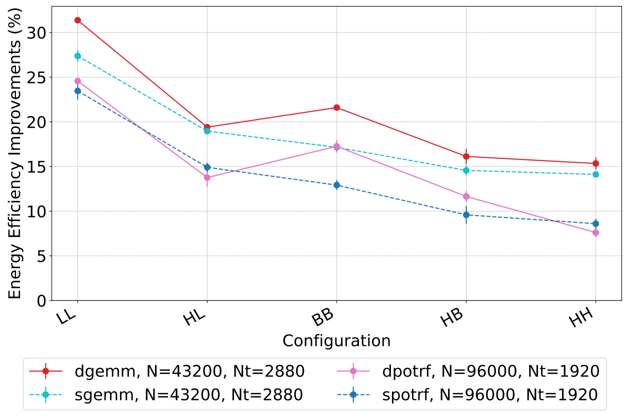


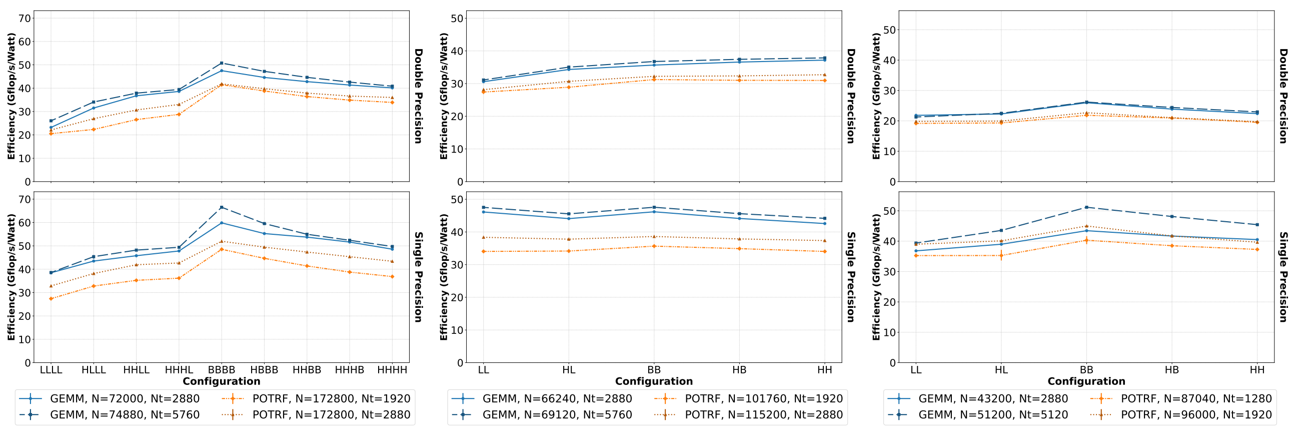
Fig. 5: Energy efficiency improvements with two CPUs power capped for GEMM and POTRF operations on the 24-Intel-2-V100 platform in single and double precision compared with the same configuration when no CPU is power capped (taking previous results as a baseline for each configuration).

- GPU powercapping showed that on GPUs, faster does not mean more energy efficient
- The best energy efficiency is obtained when all GPUs are at their best power capping (best energy efficiency improvement: 24.3 % - observed slowdown : 26.41 %)
- Applying different power caps to GPUs allows for more acceptable slowdown (energy efficiency improvement: 9.28 % - observed slowdown: 12.32 %)
- CPU power capping adds an additional leverage regarding energy efficiency (energy efficiency improvement: 21 % with no performance loss).

VI. Related Work

GPU power capping have been widely studied across several domains. Whether it is under power constraint or by applying it, the goal is always to find better energy efficiency, to have better trade-offs between performance and energy, or just to maximize performance under power capping. Studies like [17]–[19] focus on tuning both power capping and frequency. [20] and [21] propose methods to maximize performance under power capping. [22] and [23] use dynamic power capping based on a monitoring and tuning. [24] focuses on GPU power capping at a cluster level. It especially targets the optimization of training completion time respecting global power budget. These contributions target the same objective as our study. However, they either only consider hardware parameters (like performance, energy consumption and efficiency, the software part is not considered) or rely in dynamic approaches (dynamic power capping). In this paper, we consider task-based applications and the key parameters impacting their behavior (precision and granularity of computations).

Closer to our work, [25] studies the impact of GPU power capping on energy efficiency, temperature, performance, and power draw at the level of a cluster in a context where



(a) Platform 32-AMD-4-A100 (b) Platform 64-AMD-2-A100 (c) Platform 24-Intel-2-V100

Fig. 6: Energy efficiency in Gflop/s/Watt for the GEMM and POTRF operations in double (Top) and single (Bottom) precisions with different tile sizes. The results are presented on the three platforms. Note that on the 24-Intel-2-V100 platform (Right), both CPUs are power capped.

multiple users submit jobs for deep learning model training and inference. [26]–[29] profiles GPU performance and energy efficiency according to power capping at an application level. These studies show similar conclusions to the study in this paper, but in different context. In this work, we further use these observations to schedule and study possible improvement on energy efficiency while limiting the impact on performance.

Most studies focusing on scheduling target job scheduling rather than scheduling within the context of an application using task-based runtime systems. [30] uses machine learning to co-schedule, partition resources and power cap heterogeneous systems (CPU-GPU). [31] provides optimization for job allocation and partitioning under power constrained HPC system. This targets hardware-level GPU partitioning for job allocation. [32] and [33] propose a tool to dynamically power cap CPUs and GPUs on a server in the context of job submission with users sharing the same resources. [34] uses CPU and GPU power capping in the context of co-scheduling independent jobs. Also, this study focuses on integrated GPUs (in the CPU). These studies are complementary to our work since we do not target energy efficiency at the same level.

To our knowledge, our work is the first to propose a study of State-of-the-Art HPC task-based runtime system operating on multi GPUs system under different selected static power caps for energy efficiency.

VII. Conclusion and Future Work

This work presents a detailed study of the impact of GPU power capping on the performance and the energy efficiency of task-based HPC operations. We first showed that on such platforms, applying a power cap actually improves energy efficiency despite the induced slowdown. Based on this observation, we proposed to apply a power cap on only a subset of the available GPUs to benefit from both performance and energy efficiency. Then, we rely on StarPU, a task-based runtime system, to schedule tasks by considering the performance of each computing device.

The results show that power capping a subset of GPUs improves energy efficiency with a cost on performance. The best observed energy savings are reached when all GPUs are power capped (24.3 % improvement). However this comes with a cost on performance (26.41 % slowdown). As a consequence, a trade-off between performance and energy efficiency can be found by capping less GPUs. For instance, we observed that when capping less GPUs, we can reduce the slowdown to 9.28 % but still improve the energy efficiency by 12.32 %. Note that applying a power capping to the CPU as well further improves energy efficiency with no additional impact on performance.

The presented results also showed that the scheduling strategy was not performing well especially for platforms were CPU power capping is used. We plan on designing dynamic scheduling algorithms optimizing energy efficiency in the context of heterogeneous platforms. We also plan to consider on one hand dynamic power capping and its interaction with scheduling decisions, and on the other hand mixed precision computations as a complementary way to find the best trade-off between raw performance and energy efficiency. Finally, we plan to consider complex/irregular scientific applications.

Acknowledgments

This work is supported by the TEXTAROSSA project G.A. n.956831, as part of the EuroHPC initiative. As part of the ”France 2030” initiative, this work has benefited from a national grant managed by the French National Research Agency (Agence Nationale de la Recherche) attributed to the Exa-SofT project of the NumPEx PEPR program, under the reference ANR-22-EXNU-0003. Experiments presented in this paper were carried out using the Grid’5000 experimental testbed, being developed under the INRIA ALADDIN de-

velopment action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

References

[1] E. Agullo, B. Bramas, O. Coulaud, E. Darve, M. Messner, and T. Takahashi, "Task-based FMM for heterogeneous architectures," *Concurrency and Computation: Practice and Experience*, vol. 28, no. 9, pp. 2608–2629, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3723

[2] J. M. C. Carpaye, J. Roman, and P. Brenner, "Design and analysis of a task-based parallelization over a runtime system of an explicit finite-volume CFD code with adaptive time stepping," *Journal of Computational Science*, 2018.

[3] "TOP500 Supercomputer Site," http://www.top500.org.

[4] G. D. Costa and J.-M. Pierson, "DVFS governor for HPC: Higher, faster, greener," in *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*, 2015, pp. 533–540.

[5] H. Zamani, Y. Liu, D. Tripathy, L. Bhuyan, and Z. Chen, "GreenMM: energy efficient GPU matrix multiplication through undervolting," in *Proceedings of the ACM International Conference on Supercomputing*, ser. ICS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 308–318. [Online]. Available: https://doi.org/10.1145/3330345.3330373

[6] G. Bosilca, A. Bouteiller, A. Danalis, M. Faverge, T. Hérault, and J. J. Dongarra, "Parsec: Exploiting heterogeneity to enhance scalability," *Computing in Science & Engineering*, vol. 15, no. 6, pp. 36–45, 2013.

[7] J. Bueno, J. Planas, A. Duran, R. M. Badia, X. Martorell, E. Ayguadé, and J. Labarta, "Productive programming of GPU clusters with OmpSs," in *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, 2012, pp. 557–568.

[8] J. Kim, S. Lee, B. Johnston, and J. S. Vetter, "IRIS: A portable runtime system exploiting multiple heterogeneous programming systems," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*, 2021, pp. 1–8.

[9] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "StarPU: a unified platform for task scheduling on heterogeneous multicore architectures," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 2, pp. 187–198, 2011. [Online]. Available: https://hal.inria.fr/inria-00550877

[10] C. Augonnet, S. Thibault, and R. Namyst, "Automatic calibration of performance models on heterogeneous multicore architectures," in *Euro-Par 2009 – Parallel Processing Workshops*, H.-X. Lin, M. Alexander, M. Forsell, A. Knüpfer, R. Prodan, L. Sousa, and A. Streit, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 56–65.

[11] E. Agullo, B. Bramas, O. Coulaud, L. Stanisic, and S. Thibault, "Modeling Irregular Kernels of Task-based codes: Illustration with the Fast Multipole Method," INRIA Bordeaux, Research Report RR-9036, Feb. 2017. [Online]. Available: https://inria.hal.science/hal-01474556

[12] C. Augonnet, J. Clet-Ortega, S. Thibault, and R. Namyst, "Data-aware task scheduling on multi-accelerator based platforms," in *2010 IEEE 16th International Conference on Parallel and Distributed Systems*, 2010, pp. 291–298.

[13] E. Agullo, C. Augonnet, J. Dongarra, H. Ltaief, R. Namyst, S. Thibault, and S. Tomov, "A Hybridization Methodology for High-Performance Linear Algebra Software for GPUs," in *GPU Computing Gems Jade Edition*, ser. Applications of GPU Computing Series, W. mei W. Hwu, Ed. Boston: Morgan Kaufmann, 2012, pp. 473–484. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123859631000344

[14] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard, "Grid'5000: a large scale and highly reconfigurable grid experimental testbed," in *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.*, 2005, pp. 8 pp.–.

[15] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *Int. J. High Perform. Comput. Appl.*, vol. 14, no. 3, p. 189–204, aug 2000. [Online]. Available: https://doi.org/10.1177/109434200001400303

[16] Z. Yang, K. Adamek, and W. Armour, "Accurate and convenient energy measurements for GPUs: A detailed study of NVIDIA GPU's built-in power sensor," in *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2024, pp. 1–17.

[17] R. Schoonhoven, B. Veenboer, B. Van Werkhoven, and K. J. Batenburg, "Going green: optimizing GPUs for energy efficiency through model-steered auto-tuning," in *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 2022, pp. 48–59.

[18] 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)*. IEEE, 2013, pp. 349–356.

[19] 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)*. IEEE, 2015, pp. 1–8.

[20] K. Straube, J. Lowe-Power, C. Nitta, M. Farrens, and V. Akella, "Improving provisioned power efficiency in HPC systems with GPU-CAPP," in *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. IEEE, 2018, pp. 112–122.

[21] T. Allen, X. Feng, and R. Ge, "Performance optimization in power capped GPU computing."

[22] A. Krzywaniak, P. Czarnul, and J. Proficz, "Dynamic GPU power capping with online performance tracing for energy efficient GPU computing using DEPO tool," *Future Generation Computer Systems*, vol. 145, pp. 396–414, 2023.

[23] ——, "DEPO: A dynamic energy-performance optimizer tool for automatic power capping for energy efficient high-performance computing," *Software: Practice and Experience*, vol. 52, no. 12, pp. 2598–2634, 2022.

[24] D.-K. Kang, Y.-G. Ha, L. Peng, and C.-H. Youn, "Cooperative distributed gpu power capping for deep learning clusters," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 7, pp. 7244–7254, 2021.

[25] D. Zhao, S. Samsi, J. McDonald, B. Li, D. Bestor, M. Jones, D. Tiwari, and V. Gadepally, "Sustainable supercomputing for AI: GPU power capping at HPC scale," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, 2023, pp. 588–596.

[26] A. Krzywaniak and P. Czarnul, "Performance/energy aware optimization of parallel applications on gpus under power capping," in *Parallel Processing and Applied Mathematics: 13th International Conference, PPAM 2019, Bialystok, Poland, September 8–11, 2019, Revised Selected Papers, Part II 13*. Springer, 2020, pp. 123–133.

[27] T. Patki, Z. Frye, H. Bhatia, F. Di Natale, J. Glosli, H. Ingolfsson, and B. Rountree, "Comparing gpu power and frequency capping: A case study with the mummi workflow," in *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*. IEEE, 2019, pp. 31–39.

[28] Y. Abe, H. Sasaki, M. Peres, K. Inoue, K. Murakami, and S. Kato, "Power and performance analysis of GPU-Accelerated systems," in *2012 Workshop on Power-Aware Computing and Systems (HotPower 12)*, 2012.

[29] A. Krzywaniak, P. Czarnul, and J. Proficz, "GPU power capping for energy-performance trade-offs in training of deep convolutional neural networks for image recognition," in *International conference on computational science*. Springer, 2022, pp. 667–681.

[30] I. Saba, E. Arima, D. Liu, and M. Schulz, "Orchestrated co-scheduling, resource partitioning, and power capping on CPU-GPU heterogeneous systems via machine learning," in *International Conference on Architecture of Computing Systems*. Springer, 2022, pp. 51–67.

[31] E. Arima, M. Kang, I. Saba, J. Weidendorfer, C. Trinitis, and M. Schulz, "Optimizing hardware resource partitioning and job allocations on modern GPUs under power caps," in *Workshop Proceedings of the 51st International Conference on Parallel Processing*, 2022, pp. 1–10.

[32] R. Azimi, C. Jing, and S. Reda, "PowerCoord: A coordinated power capping controller for multi-cpu/gpu servers," in *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*. IEEE, 2018, pp. 1–9.

[33] ——, "PowerCoord: Power capping coordination for multi-CPU/GPU servers using reinforcement learning," *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100412, 2020.

[34] Q. Zhu, B. Wu, X. Shen, L. Shen, and Z. Wang, "Co-run scheduling with power cap on integrated CPU-GPU systems," in *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2017, pp. 967–977.