Improving energy efficiency of HPC applications using unbalanced GPU power capping

Albert d'Aviau de Piolant, Hayfa Tayeb, Bérenger Bramas, Mathieu Faverge, Abdou Guermouche, Amina Guermouche

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. The goal is to find a trade-off between performance and energy consumption using static GPU power capping.

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) without losing too much from the performance perspective. 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 obtained results show that, for a given platform equipped with 4 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. simple) precision.

***Index Terms*—**

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 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 schedules decisions and moving data across all available resources. Using such heterogeneous resources come at a high energy cost. A supercomputer consumes a lot of power - tens of megawatts in the case of the largest supercomputers [3]. Energy reduction has often been neglected by the HPC community, which focuses primarily on raw computational performance.

More recently, the energy consumption of machines and applications is the subject of research in the scientific community, and energy-efficient solutions are the key to green computing. Several state-of-the-art techniques have been proposed to reduce power consumption, which can be divided into two groups. The first group consists of hardware-level techniques 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]. The second group includes software-level techniques. These include scheduling algorithms on heterogeneous resources that minimize multi-objective functions by including energy consumption as a criterion [6]–[8]. In this paper we focus on hardware-level techniques, more specifically on power capping.

Power capping is a technique to lower 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. As a matter of fact, a low power cap can provide large power savings, but the performance loss may be too large to provide energy savings. On the other hand, a high power cap can have little to no impact on power consumption, and no energy savings are observed as well. In this work, we focus on GPU power capping. 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 consumption.

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 different 3 different architectures, 2 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 8 % with no additional impact on performance. This represents a total energy efficiency of 42.45 Gflop/s/Watt.

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 architecture 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. We also have energy consumption represented in Joule, and the compromise energy efficiency, represented in flop/s/Watt. Energy efficiency is defined as the ratio of the amount of performance achieved to the amount of power consumed. 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.

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. Figure 1 depicts the energy efficiency, the performance and the energy consumption when running cuBLAS GEMM on different matrix sizes (one single large tile) on A100-SXM4-40GB with single precision and double precision. Table I summarizes the best results accros all architectures in regard to energy efficiency. In all experiments, we vary the power cap from lowest possible limit on the GPU, to no power capping (100 % on Figure 1) at all with a step of 2 %.

Figure 1 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. 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, therefore better energy utilization. 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 would be too important, and may not be acceptable for all 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. In the next section (section III), we will introduce the applications and the task-based runtime system that will allow us to schedule the computations differently.

III. Target applications and runtime system

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

We situate our study of energy efficiency using GPU power capping in the context of heterogeneous platforms. HPC applications are frequently parallelized across heterogeneous systems, thus leveraging the multiple available computing resources. Taking advantage of heterogeneous platforms can be a tedious process. That is why HPC experts are working hard to bridge the gap between domain expert implementations and the use of such platforms. Runtime systems –a software layer that abstracts the hardware complexity– represent a promising solution. We are particularly interested in task-based

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

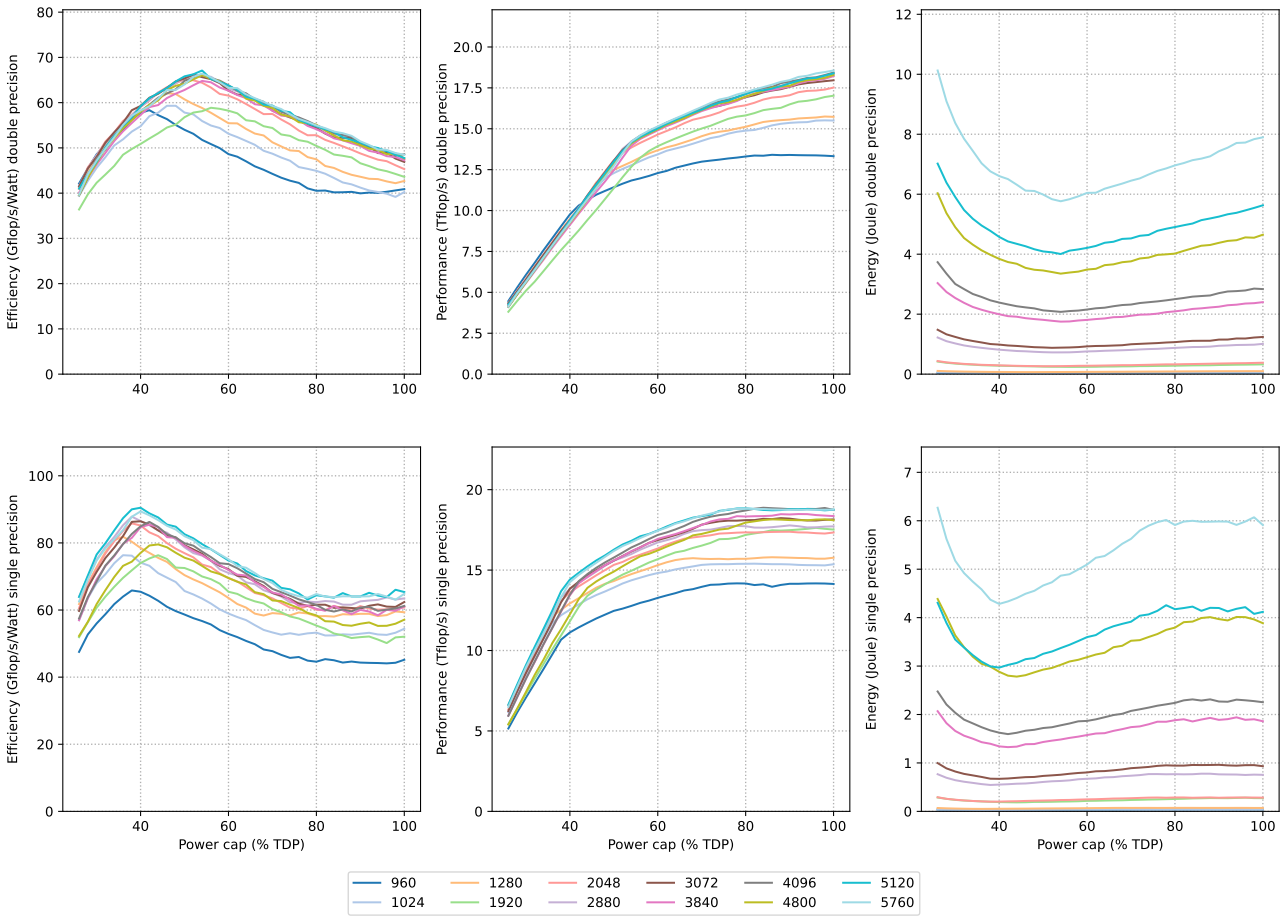


Fig. 1: Power capping impact on energy efficiency, performance and energy consumption considering cuBLAS DGEMM on different matrix sizes on A100-SXM4-40GB. The power cap varies from 104 to 400W.

runtime systems, which consider the application as a DAG of tasks and efficiently schedule tasks and data migrations across all available processing units, including central processing units (CPUs) and accelerators. Several runtime systems have emerged [9]–[11]. Among them is StarPU [12], 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, GPU, and FPGA.

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. Our work is centered on the study of energy efficiency in task-based parallelized applications. The dynamic scheduling of tasks on heterogeneous resources represents a significant challenge with the potential to impact the performance of the application. To illustrate, if the scheduler decides to assign a fewer number of tasks of a specific type to CPUs, given their relatively slow processing speed, and instead assigns the majority of them to GPUs, which are capable of processing tasks more rapidly, the overall execution time will be reduced. However, if the scheduler decides incorrectly and assigns more tasks to CPUs, the execution will be longer and the energy consumption could be higher. Therefore, in our study of energy efficiency, it is essential to use fine-tuned scheduling strategies that have been proven effective for our use case operations.

As previously stated, we use StarPU –a task-based runtime system– to orchestrate tasks across heterogeneous resources. StarPU 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 [13], [14]. It requires a few calibration runs to collect some data points that will allow the model to be built. In addition, StarPU provides a set of predefined scheduling policies that represent the literature. These include a work stealing policy, a random scheduling policy, and the Heterogeneous Earliest Finish Time (HEFT) strategy.

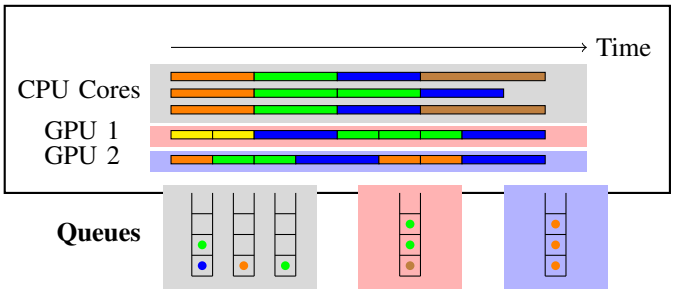


Fig. 2: The dequeue model strategy relies on performance models to schedule tasks to the resource with the best expected completion time.

The dequeue model (dm) scheduler family [15] (also called heft-tm-pr) considers task execution performance models to implement a scheduling strategy similar to HEFT that selects the best resource queue to assign the task to. The scheduler estimates and selects the best expected completion time of the task on each processing unit (see Fig. 2). This evaluation is based on the performance models provided by StarPU. The Dmda (dequeue model data-aware) scheduler (also called heft-tmdp-pr) goes a step further by also considering the time required to transfer data to the processor. The Dmdas (dequeue model data aware sorted) variant sorts the queues based on the task priority values specified by the application expert. For tasks with equal priorities, it prioritizes those whose data buffers are already available on the target device, making it more sensitive to data locality. In the upcoming experimental study, we will use Dmdas, a fine-tuned scheduler, that is adequate for the purposes of our study.

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 [16] 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 model 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. The GEMM kernel is approximately 20 times faster on GPUs than on CPUs. Therefore, the scheduler plays a critical role in ensuring good workload balancing in such operations. The DAG of POTRF operation has $\frac{N(N+1)(N+2)}{6}$ vertices and $\frac{(N-1)N(N+1)}{2}$ edges for $N \times N$ tile matrix, in particular, it has $\frac{N(N-1)(N-2)}{6}$ GEMM tasks, which represents approximately half of all 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 investigate the trade-off between performance and energy consumption, with regard to the energy efficiency metric. 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. To this end, experiments were conducted on nodes from Grid5000 [17], 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 (V100-PCIE-32GB, A100-PCIE-40GB, and A100-SXM4-40GB), each of which is integrated into a different platform, as detailed 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" and is 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 "grouille-1" and is situated at the Nancy Grid'5000 site.
- **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.

24-Intel-2-V100 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. On all GPUs, tensor cores are used on double precision, but not on simple precision.

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 recapitulates the matrix sizes equal to $\boldsymbol{N} \times \boldsymbol{N}$ and the corresponding tile sizes denoted by $\boldsymbol{Nt}$ for GEMM and Cholesky factorization. In the following, all results presented use these values.

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. The highest point from the energy efficiency data set was selected. 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, in which half of the tasks are GEMM tasks. 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, for a configuration designated as `HHLL`, this 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 [18]. 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 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.

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 statically power capping levels 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 and offers insight into the trade-off between performance and energy efficiency.

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 the energy

TABLE II: Selection of matrix sizes equal to $N \times N$ suitable for each operation and platform with the appropriate tile size denoted by Nt dividing the dense matrix into equal tiles. 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	Nt	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%		

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.

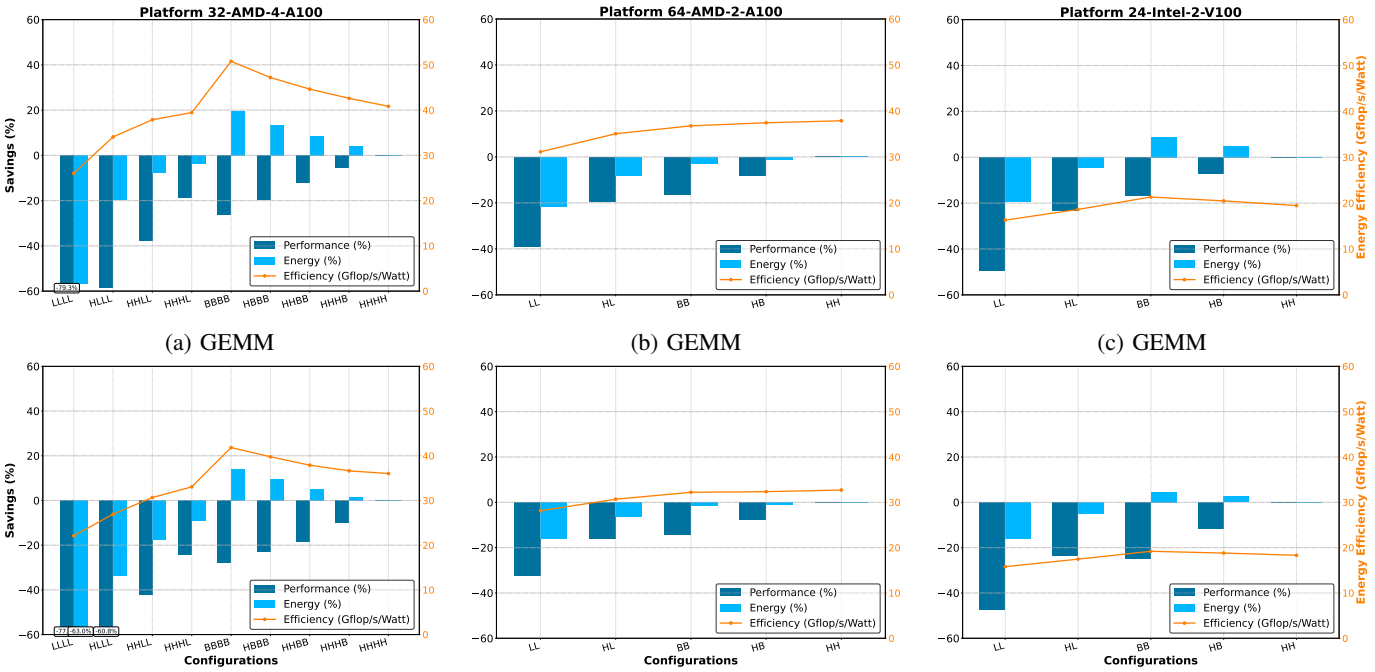
A. Evaluating Energy Efficiency for Task-Based Operations in Double Precision

Figures 3a and 3d 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. The two operations were executed on a configuration with four GPUs operating at the lowest power level specified by the TDP. This configuration is designated as `LLLL`. Under this extreme configuration, both operations 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 to determine if we can reduce energy consumption by keeping some GPUs at their lowest power levels. The results of this experiment prove that the previous hypothesis is incorrect. Indeed, there is a gradual improvement in performance, yet the operation remains 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`. The energy efficiency increases to 38 Gflop/s/Watt (respectively 36 Gflop/s/Watt) for GEMM (respectively POTRF) for the configuration `HHHL`. It remains less energy efficient than the default configuration, `HHHH`, which scores 41 Gflop/s/Watt (respectively 38 Gflop/s/Watt) for GEMM (respectively POTRF).

When the power cap of all the GPUs is set to the best value selected in the study presented in Table II (P_{best}), 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), respectively, 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 energy savings achieved are approximately 20%, with a corresponding degradation of performance of approximately 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 reduce energy consumption 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 amount of energy consumed. However, even with a single GPU capped at `B` and the remaining GPUs set at the default level (`HHHB`), we observed a 4% reduction in total energy consumption for the GEMM compared to the platform's default configuration. This resulted in an improvement in energy efficiency from 40 Gflop/s/Watt to 42 Gflop/s/Watt (5 % improvement).

Figures 3b and 3e 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, with a 5% improvement in efficiency compared to `BB` configuration. 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`, although there is a decline in performance of approximately 10%, the total energy consumption is slightly greater than that of the default configuration. 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-



(a) GEMM (b) GEMM (c) GEMM

(d) POTRF (e) POTRF (f) POTRF

Fig. 3: Performance and energy analysis for GEMM and POTRF 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.

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 CPUs are also employed, with two CPUs present in the platform 64-AMD-2-A100. The impact of the CPUs on power consumption is taken into account, and the TDP of each CPU is 125 Watt, representing half the maximum power of the GPUs. Consequently, the CPU impact on power consumption is significant. 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 energy consumption of both CPUs. A more detailed discussion on the impact of CPU consumption is presented in Section V-C.

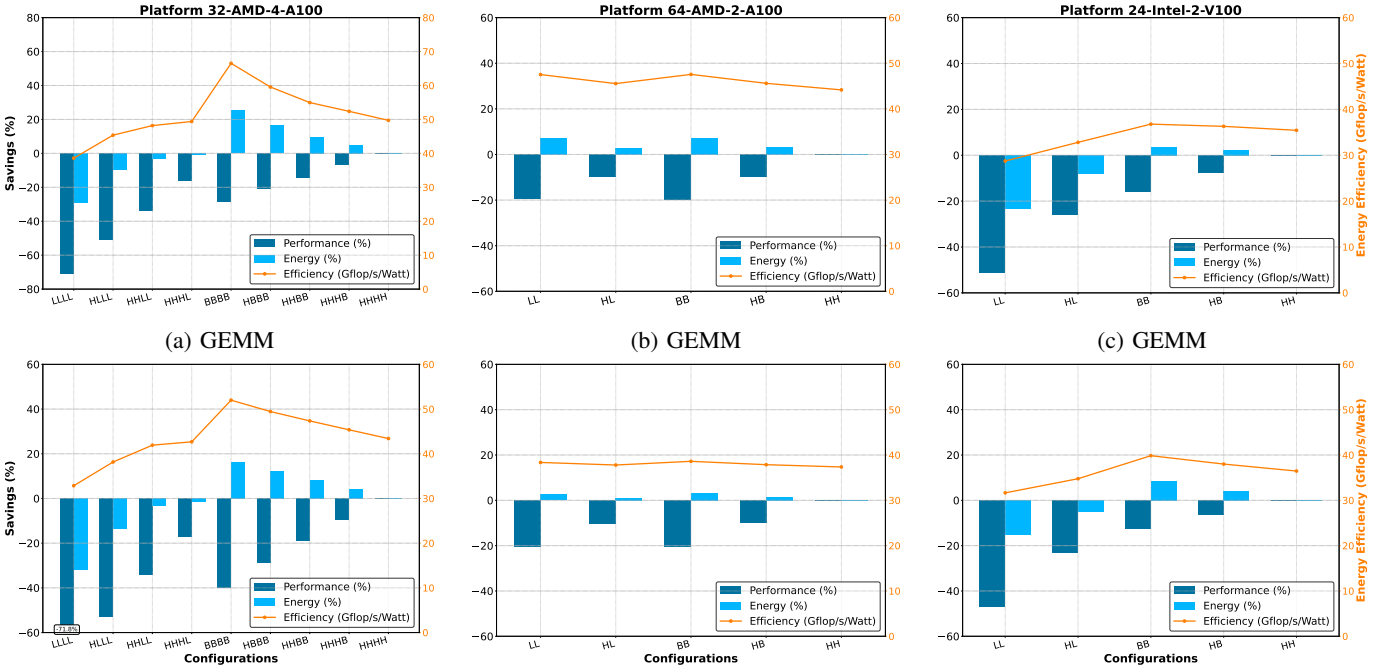
Figures 3c and 3f show the results on the platform 24-Intel-2-V100. The energy efficiency plot shows a similar trend to 32-AMD-4-A100. The peak in `BB` indicates an improvement in energy efficiency, from 19.5 Gflop/s/Watt for the default configuration to 21.3 Gflop/s/Watt (9.23 % improvement). For the GEMM in `HB`, we save 5% of total energy with a 7% decline in performance. We suspect that the energy savings may be underestimated, as the consumption of two CPUs is included in the total, and CPUs may not be optimal for such compute-intensive operations.

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

Figure 4 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 energy efficiency for the GEMM, a nearly 25% reduction in energy consumption with a 28.6% decline in performance for the POTRF. The configuration `HHBB` demonstrates a trade-off between performance and energy savings, with approximately 9.53% energy savings for both operations and a 14.59% decline in performance, leading to an improvement of energy efficiency of 54.9 Gflop/s/Watt compared to 49.73 Gflop/s/Watt for the default configuration (roughly 10 %). Other configurations like `HHHB` shows an even lower slow-



(a) GEMM (b) GEMM (c) GEMM

(d) POTRF (e) POTRF (f) POTRF

Fig. 4: Performance and energy analysis for GEMM and POTRF operations on three platforms in **single** precision (refer to the detailed caption in Fig. 3).

down but at the cost of a lower energy efficiency. The results on the 64-AMD-2-A100 platform are noteworthy (Figures 4b and 4e), 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 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, particularly given that the TDP of these specific GPUs is only 125 Watt higher than those of the CPUs.

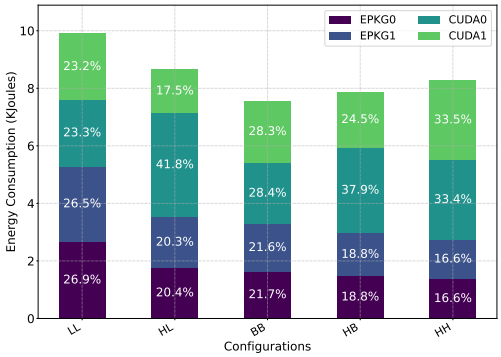
C. Impact of the energy consumption of CPUs

Figure 5 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.

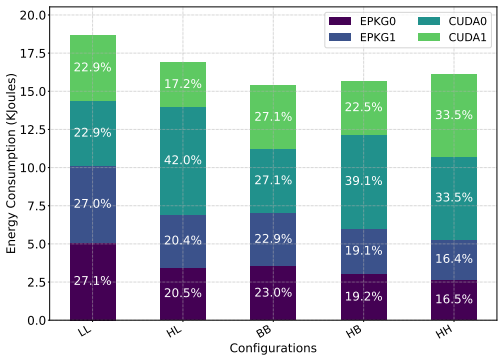
However, when we impose power caps on the GPUs, the ratio of tasks computed by the CPUs relative to those computed by the GPUs increases. Since CPUs are much less energy efficient than GPUs, having them compute more tasks negatively impacts overall energy efficiency. This explains why, when GPUs are power capped at their lower limit (`LL` configuration), part of the energy savings from GPU power capping is offset by the increased energy consumption from the CPUs, resulting in higher overall energy usage.

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. However, 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 architectures, 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, we chose to power cap the second CPU of the node at 48 % of its TDP (60 Watt over 125). A power limit under this value have shown stability issues. Figure 6 compares the scenario when no CPU power capping is applied with the scenario when CPU power capping is applied (the latter is equivalent to the value computed in Figures 3c, 3f, 4c and 4f). In this specific case, CPU power capping improves energy efficiency by more than 10 %, reaching 14 % for GEMM. Moreover, we observe an overall improvement in energy efficiency across all configurations, regardless of the operation and precision. This demonstrates that capping the power of one CPU does not delay critical tasks while reducing the negative impact of the CPUs' poor energy efficiency on the overall system. It also shows that, because the scheduler has no knowledge of



(a) GEMM



(b) POTRF

Fig. 5: Total energy consumptions per configuration on the platform 24-Intel-2-V100 for both operations in double precision. Detailed energy consumption percentages by device.

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 7 summarizes the energy efficiency on all three platforms but on additional tile sizes. Note that figure 7c depicts the efficiency when the CPU is power capped as explained in Section V-C. From figure 7, one can see 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, we can observe that 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:

- 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%)

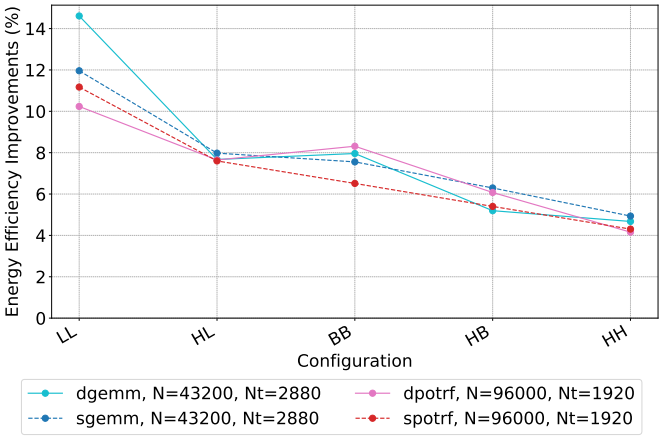


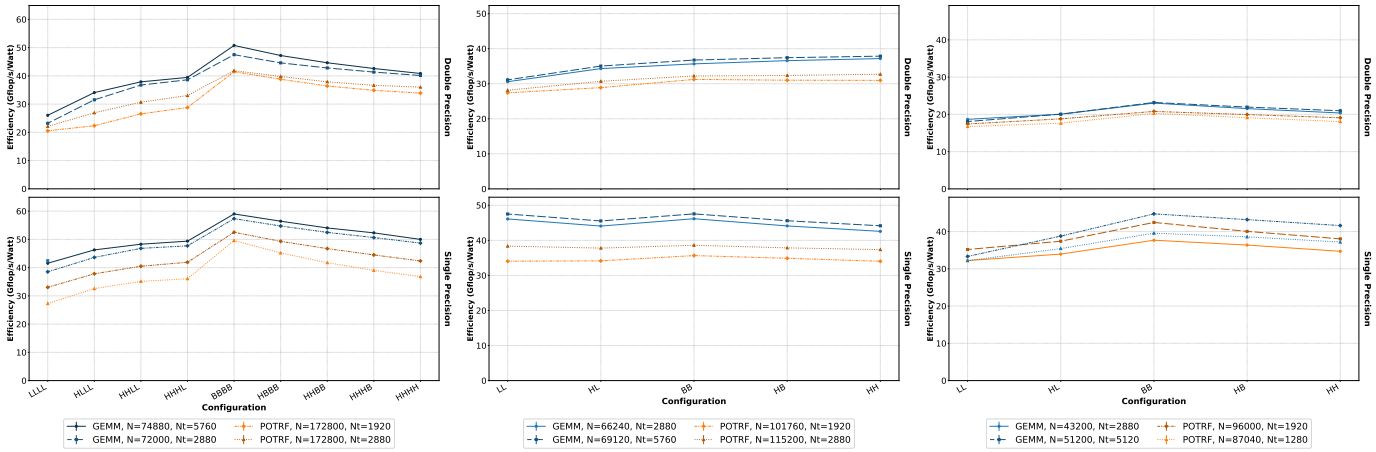
Fig. 6: Energy efficiency improvements with one CPU power capped for GEMM and POTRF operations on the 24-Intel-2-V100 platform in single and double precision compared the same configuration when no CPU is power capped.

- If the user cannot afford high slowdown, applying different power caps to GPUs allows for more acceptable trade-off (energy efficiency improvement: 9.28% - observed slowdown: $12.32\ \%$)
- CPU power capping adds an additional leverage regarding energy efficiency (energy efficiency improvement: $8\ \%$ 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 [19]–[21] focus on tuning both power capping and frequency. [22] and [23] propose methods to maximize performance under power capping. [24] and [25] use dynamic power capping based on a monitoring and tuning. [26] 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 or rely dynamic approaches. In this paper, we consider task-based applications and the key parameters impacting their behavior (precision and granularity of computations).

Closer to our work, [27] 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 multiple users submit jobs for deep learning model training and inference. [28]–[31] 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 to schedule and study possible trade-offs regarding performance and energy efficiency.



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

Fig. 7: Energy efficiency in Gflop/s/Watt for the GEMM and POTRF operations in single and double precisions with different tile sizes. The results are on three platforms. On platform 24-Intel-2-V100, one CPU is power capped.

Most studies focusing on scheduling target job scheduling rather than scheduling within the context on an application using task-based runtime systems. [32] uses machine learning to co-schedule, partition resources and power cap heterogeneous systems (CPU-GPU). [33] provides optimization for job allocation and partitioning under power constrained HPC system. This targets hardware-level GPU partitioning for job allocation. [34] and [35] 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. [36] 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. For that purpose, we rely on StarPU, a task-based runtime system which can schedule tasks by considering the performance of each computing device. The results show that applying a power cap to some GPUs provides a good trade-off between performance and energy efficiency. Moreover, we showed that applying CPU power capping provides additional energy savings with no impact on performance.

The presented results 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 consumption. 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. Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development 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] K. Huang, M. Jing, X. Jiang, S. Chen, X. Li, W. Tao, D. Xiong, and Z. Liu, "Task-level aware scheduling of energy-constrained applications on heterogeneous multi-core system," *Electronics*, vol. 9, no. 12, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/12/2077

[7] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 257–271, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1630214X

[8] J. Chen, M. Manivannan, M. Abduljabbar, and M. Pericàs, "Erase: Energy efficient task mapping and resource management for work stealing runtimes," *ACM Trans. Archit. Code Optim.*, vol. 19, no. 2, Mar. 2022. [Online]. Available: https://doi.org/10.1145/3510422

[9] 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.

[10] 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.

[11] 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.

[12] 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

[13] 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.

[14] 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

[15] 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.

[16] 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

[17] 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.–.

[18] 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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] ——, "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.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] ——, "Powercoord: Power capping coordination for multi-cpu/gpu servers using reinforcement learning," *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100412, 2020.

[36] 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.