Future Generation Computer Systems 149 (2023) 71–88

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An automated and portable method for selecting an optimal GPU frequency

Ghazanfar Ali [a,*], Mert Side [a], Sridutt Bhalachandra [b], Nicholas J. Wright [b], Yong Chen [a]

[a] Texas Tech University, 2500 Broadway, Lubbock, 79409, TX, USA
[b] Lawrence Berkeley National Laboratory, 1 Cyclotron Rd, Berkeley, 94720, CA, USA

ARTICLE INFO

Article history:
Received 22 October 2022
Received in revised form 3 July 2023
Accepted 7 July 2023
Available online 17 July 2023

Keywords:
GPU frequency selection
DVFS
GPU power modeling
GPU performance modeling
Energy delay product
Multi-objective function
Energy efficiency

ABSTRACT

Power consumption poses a significant challenge in current and emerging graphics processing unit (GPU) enabled high-performance computing systems. In modern GPUs, dynamic voltage frequency scaling (DVFS) appears to be a reliable control to regulate power consumption and performance. However, the DVFS design space is large - hence, brute-force approaches are infeasible to select the optimal frequency. Furthermore, no single frequency can be universally optimal for applications with varying computational intensities. Thus, the application's complexity and the availability of a wide range of frequency settings are a challenge in selecting the optimal frequency configuration for a given GPU workload. To that end, this paper proposes a systematic approach that consists of three steps. The feature characterization study identifies the fine-grain GPU utilization metrics that influence the power consumption and execution time of a given workload. To understand the performance, power, and energy consumption behaviors of a workload across GPU's DVFS design space, we derived analytical power and performance models using the identified fine-grain features. It is shown that the same set of GPU utilization metrics can estimate both the power consumption and execution time while being agnostic of changes to frequency and input sizes. Applying a power control with the single objective of reducing power may cause performance degradation, leading to more energy consumption. A multi-objective approach is proposed to select the optimal GPU DVFS configuration for a workload that reduces power consumption with negligible degradation in performance. The evaluation was conducted using SPEC ACCEL benchmarks and three real applications - NAMD LAMMPS, and LSTM on NVIDIA GV100, GA100, and AMD MI210 GPUs. On average, real applications showed 29.6% energy savings with a performance loss of 5.2% on GA100 and 22.6% energy savings with a performance loss of 4.7% on GV100. Moreover, the proposed models are portable to real applications, GPU architectures, and vendors, and require metric collection at only the default frequency rather than all supported DVFS configurations. Additionally, we conducted a comparison between our models and the GPU assembly instructions (PTX)-based static models. The results revealed a significant reduction in the average error rates, with a decrease from 19.7% to 3.1% for power models and from 29.4% to 5.2% for performance models.

1. Introduction

In the new era of post-Moore's law, GPUs are likely to be crucial in accelerating computing capacity for current and future high-performance computing (HPC) systems. While GPUs are performant, they increasingly consume a significant amount of power. For example, today, a single advanced GPU consumes power up to 500 W [1] which is close to a traditional HPC node [2]. As such, the power consumption of HPC systems built with GPUs is limited by power. An exascale system built with current generation GPUs expects to consume more than the desired 20 MW power budget [3] (e.g., the Frontier [4]), even without considering the infrastructure and cooling overheads. Furthermore, HPC data centers have been more concerned about performance historically; however, in more recent times due to the "dark silicon" phenomenon [5], there has been a paradigm shift toward striking a balance between power and execution time [6]. For example, literature [7] estimated that a 5% decrease in power consumption of the Summit supercomputer could generate savings of around 1 million dollars. Therefore, it is increasingly critical to develop GPU power management strategies that can lower power consumption with a minimum impact on execution time.

There are several challenges to designing efficient power management strategies for GPUs. First, the complexity of GPU work-

* Corresponding author.
E-mail address: ghazanfar.ali@ttu.edu (G. Ali).

https://doi.org/10.1016/j.future.2023.07.011
0167-739X/

loads in terms of their utilization of computational resources can lead to diverse power consumption needs. Second, GPUs offer a wide array of power consumption controls, and understanding the impact of these power controls on power consumption and performance is non-trivial. For example, the NVIDIA GA100 (Ampere) and GV100 (Volta) GPUs provide 81 core DVFS configurations in the range of 210–1410 MHz and 167 DVFS configurations in the range of 135–1380 MHz, respectively. While this flexibility is certainly favorable for saving power, it also makes the GPU's DVFS design space more complex in selecting a DVFS configuration that provides optimal power consumption and execution time simultaneously. Given the complexity of different workloads and power controls, it is not realistic for HPC system architects and operators to select the optimal GPU frequency manually.

Limitations of state-of-art approaches: Many studies have explored to improve GPU power, performance, and energy efficiency [8–14]. The major research areas include DVFS space exploration, optimal frequency determination, analytical and machine learning (ML) based models using utilization metrics, and static code analysis. However, the existing approaches have some caveats: (1) features derived using static code analysis or utilization metric are not always best representative of a workload (often workload or architecture-specific), and (2) multi-objective functions provide a range of best frequencies rather a definitive optimal frequency [7,15].

Experimental methodology and artifact availability: To address these challenges, the *Optimal GPU Frequency Selection* [16] has been proposed to automate the selection of the optimal DVFS configuration for a workload that requires three steps. First, characterization and identification of the GPU features that directly influence power and performance. We used the mutual information technique to prune the features most relevant to power and performance. Second, modeling of power and performance behaviors across DVFS design space to enable model-based estimation of a workload's power and execution time using the workload's utilization requirements. Third, the determination of the optimal DVFS configuration based on the estimated power and execution time profiles across all DVFS configurations. Flexible optimal frequency selection techniques were devised using multi-objective functions. These techniques included energy-delay product (EDP) [6,17–19] and energy-delay-square product (ED^2P). EDP takes the optimality of both energy and execution time (delay) into consideration simultaneously while selecting the optimal frequency. ED^2P provides double-weight to the execution time.

Although the *Optimal Frequency Selection* has been previously described, a methodology to make it portable across different GPU architectures and real applications have remained undeveloped. In this study, we approached this by performing data collection for real applications on new GPU architecture, several inter-architectural analyses, and an extension of the power model. In particular, we analyze the portability of the features selected in study [16] on the NVIDIA GA100 GPU. The power model proposed in [16] is extended to mitigate inter-architectural power consumption variations. The application-level portability is evaluated by estimating optimal frequencies of real applications using the models developed with micro-benchmarks. The GPU architecture-level portability is evaluated by estimating the optimal frequencies of real applications on GA100 using the models developed with micro-benchmarks on GV100. We provided more evaluation data (selected frequencies, energy savings, changes in performance), useful insights, and example usage of our methodology in a production environment. The source codes, including data collection, power controls, data analysis, and implementation of analytical models, are publicly available [20].

Key insights and contributions: Overall, this study makes the following contributions.

1. **Features Portability:** The initial characterization of features using micro-benchmarks in study [16] confirms the impact of GPU utilization features on power usage, energy, and execution time. In this study, we evaluate the portability of features in terms of different input sizes, other GPU architectures, and vendors. We observe that the selected features are portable across architectures and vendors.
2. **Models portability:** Based on the characterization study, analytical models for execution time and power were proposed in the study [16]. We evaluated the portability of the models using real-world HPC and machine-learning workloads (application-level portability), NVIDIA GA100 GPUs (architecture-level portability), and AMD MI210 GPUs (vendor-level portability). The metric collection is required only at the GPU's maximum DVFS configuration for a given workload. These metrics are used to estimate a workload's power and performance for the remaining DVFS configurations using the proposed models. We evaluated the portability of the proposed methodology in study [16] for real-world applications. On NVIDIA GV100, using real-world applications, these models estimated power and performance up to 95.2% and 96.9%, respectively. Furthermore, we have evaluated the portability of the models across different GPU architectures and vendors. The power and performance models, developed using GV100's data (thermal design power (TDP) of 250 W), estimated power and execution time of real applications on GA100 (TDP of 500 W) with accuracies of up to 97.9% and 98.2%, respectively. To evaluate vendor-level portability, we have mapped the feature set utilized in constructing the models from NVIDIA to a corresponding feature set available in AMD. The power and performance models, utilizing data from GV100, accurately estimated the power consumption and execution time of real applications on the AMD Instinct MI210 GPU, achieving accuracies of up to 96.1% and 99%, respectively.
3. **Energy-performance trade-offs:** The efficacy of the multi-objective optimal functions is evaluated. The energy profiles chosen by the ED^2P-based optimal frequency achieved an energy saving of up to 29.6% with a performance loss of 5.2% for real applications on GA100.
4. **Comparison with state-of-the-art models:** We conducted a comparison between our models and the GPU assembly instructions (PTX)-based static models [7]. The results revealed a significant reduction in the average error rates, with a decrease from 19.7% to 3.1% for power models and from 29.4% to 5.2% for performance models.

Limitations of the proposed approach: The models require a given workload to be run at the maximum frequency to acquire utilization metrics. The models can only be used in association with DVFS. Other power controls, like power capping, are beyond the current scope of this work.

This paper is organized as follows. Section 2 provides the background and motivation of this research. Section 3 describes the experimental setup. Section 4 presents an overview of the methodology, data collection, feature analysis, analytical modeling, and explains the multi-objective algorithm for selecting the optimal frequency. Sections 5 and 6 present the evaluation results. Section 7 provides a comparison of models with state-of-the-art research. We discuss other related work and comparison in Section 8. Section 9 provides sample deployment options and Section 10 discusses concluding remarks.

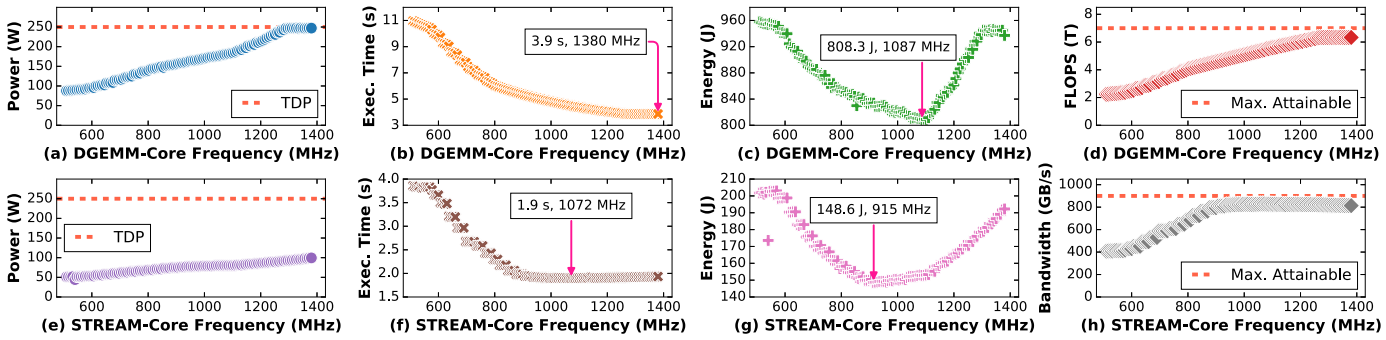


Fig. 1. Power, execution time, energy, and FLOPS variations across different frequency configurations for DGEMM (upper) and STREAM (lower), respectively.

2. Motivation

This section discusses the impact of DVFS on performance, power, and energy patterns on compute- and memory-intensive workloads. It also explains why multi-objective optimal functions are needed to select the optimal DVFS configuration.

2.1. Impact of DVFS on compute-intensive workload

DVFS technique is one of the widely used techniques to regulate power and performance by clocking the GPU core to different frequency configurations. Several previous works [9,10] observed that the impact of DVFS on power and execution time depends on GPU architecture and application intensity. Hence, GPU workloads with different computational intensities show different power and execution time behaviors for a given core DVFS configuration. As a preliminary step, we tested DGEMM and STREAM [21,22] GPU micro-benchmarks to understand the power and execution time characteristics of compute- and memory-intensive applications. Even though we have tested all supported GPU configurations, configurations below 510 MHz showed high performance penalties leading to a higher power and thus are infeasible. Hence, we only use configurations in the range of 510–1410 MHz (61 configurations) for GA100 and 510–1380 MHz (117 configurations) for GV100. It is worth noting that unlike some previous GPU architectures, which provide multiple memory frequency configurations, GA100 and GV100 support a single high bandwidth memory (HBM) frequency, i.e., 1593 MHz and 877 MHz, respectively.

Fig. 1(a) to (d) show variations in power, execution time, energy, and execution time (floating-point operations per second (FLOPS)) across 117 DVFS configurations for the compute-intensive workload (i.e., DGEMM). It is observed that power is approximately a direct linear function of GPU core frequency. Performance degradation of up to 3x was observed when the GPU core frequency was changed from the maximum to the minimum frequency. We also noted that performance degradation is negligible for the frequencies in the range of 1250–1380 MHz. These frequencies can potentially be a viable opportunity for energy-performance trade-offs. Overall, it shows that for compute-intensive applications, DVFS is an effective technique to scale power. The power behavior can be scaled down to less than half of the GPU's TDP at the lower configuration (e.g., 510 MHz). On the other hand, the power can be ramped up to its TDP limit at the maximum frequency.

The execution time exhibits an indirect nonlinear relationship with DVFS configurations, as shown in Fig. 1(b). Performance degradation of up to ∼3x was observed when the DVFS configuration was swayed from the maximum to the minimum configuration. We also noted that the performance degradation is negligible in the ∼1250–1380 MHz frequency range. This frequency range can potentially be viable options for energy and performance trade-offs for compute-intensive workloads.

Fig. 1(c) shows that energy is a parabolic (i.e., quadratic relationship) function of DVFS configuration. The energy metric for each DVFS configuration was computed as a product of power (a) and execution time (b). In general, the global minimum energy point across DVFS configurations is considered the optimal frequency where the compute-intensive DGEMM can save energy up to ∼15.8%.

Fig. 1(d) shows nearly a direct linear relationship between FLOPS and DVFS configurations. Like the execution time, the increment in FLOPS after 1250 MHz is insignificant.

To summarize, we can infer two main corollaries. First, the power consumption is highly dependent on the DVFS configuration. Second, an application's performance (both time and FLOPS) does not improve after reaching a particular DVFS configuration. Hence, any further increase in frequency causes increased in power without noticeable performance gain.

2.2. Impact of DVFS on memory-intensive workload

Fig. 1 (e) to (h) show variations in power, execution time, energy, and bandwidth across supported DVFS configurations for STREAM. Like DGEMM, power for STREAM is nearly linear with DVFS configuration as shown in Fig. 1(e). The power at the maximum DVFS configuration (1380 MHz) is ∼100 W and can be reduced up to ∼50 W at the minimum DVFS configuration (510 MHz) used in this study. Fig. 1(f) shows an indirect nonlinear relationship between execution time and DVFS configurations. It is worth noting that the execution time does not change for over 800 MHz. Thus, this configuration is optimal for the execution time. This phenomenon is also reflected in Fig. 1(h) showing that the bandwidth does not improve after ∼ 800 MHz. Fig. 1(g) depicts the quadratic relationship between the frequency and energy. It is worth noting that the frequency providing the lowest energy point for STREAM (with energy savings of ∼33%) is not the same frequency as DGEMM, suggesting that the optimal frequency of an application is driven by its computational intensity.

The HBM data rate is nearly a direct linear function of DVFS configuration as demonstrated in Fig. 1(h). The increase in GPU frequency also speeds up the data rate of the HBM. In alignment with (f), bandwidth does not improve after a DVFS configuration.

Two key takeaways: First, the execution time, power, and energy patterns of compute- and memory-intensive workloads indicate - (a) the change in GPU frequency effectively changes the execution time, power, and energy metrics. (b) The intensity of the change in these metrics is highly dependent on the workload's computational intensity. Second, the lowest DVFS configuration consumes the lowest power. However, the same configuration

Table 1
The real applications used in our evaluations.

Benchmark	Language	Domain
NAMD	C++/Charm++	Parallel molecular dynamics code for large biomolecular systems
LAMMPS	C++	Large Atomic Simulations, Molecular Simulations
LSTM	Python	Binary classification, Sentiment Analysis

degrades performance at maximum. Furthermore, the maximum configuration can provide maximum performance; on the other hand, it may not be optimal for power and energy saving.

2.3. Can one DVFS configuration fit all?

The optimal DVFS configuration for an application refers to a GPU operating frequency that reduces the power at the cost of no performance degradation (ideally) or achieves the best trade-off between performance degradation and reduction in power and energy. However, empirical results in Figs. 1 (b) and (c) show that the optimal execution time and optimal power consumption are exhibited by different DVFS configurations for an application. Furthermore, these configurations are not portable across applications (Fig. 1(b) and (f) or (c) and (g)). Comparatively, the optimal execution time was achieved at higher frequencies than the frequencies that delivered the optimal energy, and optimizing one objective can adversely affect the other. Thus, selecting the optimal frequency automatically for an arbitrary application is not a trivial task due to conflicting criteria of high performance and low power and energy. This observation supports a need for a multi-objective solution that simultaneously considers both execution time and power consumption for an application to determine the optimal DVFS configuration, which is the ultimate objective of this work.

3. Experimental setup

3.1. Target applications

In this study, we used three real applications, two micro-benchmarks, and 19 industry benchmark applications in the SPEC ACCEL suite.

3.1.1. Real applications

In this study, we used three GPU-enabled real-world applications, including: (1) Nanoscale Molecular Dynamics (NAMD) [23,24], a large biomolecular systems simulation program; (2) Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [25,26], a particle simulator that models solid-state, soft matter, and coarse-grained materials; and (3) Long short-term memory (LSTM) [27] algorithm, a TensorFlow-based [28] implementation of binary sentiment classification of large movie review dataset [29]. The domains for these applications are shown in Table 1

3.1.2. Benchmark applications

The proposed models were validated using the SPEC ACCEL® benchmark suite [30]. The application domains for the benchmarks in the SPEC ACCEL are shown in Table 2.

Table 2
The SPEC ACCEL benchmarks suite containing 19 OpenCL enabled benchmarks.

Benchmark	Language	Domain
`tpacf`	C++	Astrophysics
`stencil`	C++	Thermodynamics
`lbm`	C++	Fluid Dynamics
`fft`	C	Signal processing
`spmv`	C++	Sparse Linear Algebra
`mriq`	C	Medicine
`histo`	C	Silicon Wafer Verification
`bfs`	C	Electronic Design Automation, Graph Traversals
`cutcp`	C	Molecular Dynamics
`kmeans`	C++	Dense Linear Algebra, Data Mining
`lavamd`	C	N-Body, Molecular Dynamics
`cfd`	C++	Unstructured Grid, Fluid Dynamics
`nw`	C++	Dynamic Programming, Bioinformatics
`hotspot`	C	Structured Grid, Physics Simulation
`lud`	C++	Dense Linear Algebra, Linear Algebra
`ge`	C++	Dense Linear Algebra, Linear Algebra
`srad`	C	Structured Grid, Image Processing
`heartwall`	C	Structured Grid, Medical Imaging
`bplustree`	C	Graph Traversal, Search

3.2. Target systems

In this study, we collected the utilization metrics for SPEC ACCEL, DGEMM, and STREAM, real applications (LAMMPS, NAMD, and LSTM) using NVIDIA Ampere A100 GPU node at the National Science Foundation (NSF)'s Chameleon CHI@UC site [31], AMD Instinct MI210 node at AMD site, and Volta V100 GPU node at High Performance Computing Center of Texas Tech University, managed by the Slurm Scheduler [32]. Table 3 lists the configurations of these systems. To avoid any interference from other jobs, all our experiments were run on nodes that were exclusive. All experiments were performed using an NVIDIA GV100 with CUDA version 11.2 and driver version 450, and GA100 with CUDA version 11.5 and driver version 465. For MI210, we used ROCm 5.4, rocprof 2.0, and rocm-smi 5.4. Data analysis and modeling was performed using Python 3.10.1 64-bit.

4. Methodology

This section introduces the overall methodology, data collection process, feature analysis, power modeling, performance modeling, and the multi-objective approach to selecting the optimal frequency.

4.1. Overview

Our methodology consists of two phases: (1) building analytical models for power and execution time and (2) selecting the optimal frequency selection for a given workload using multi-objective optimal functions based on estimated power and execution time using analytical models.

Fig. 2(a) shows the process of building analytical models for power and execution time, which are built using workloads' GPU utilization metrics across GPU's DVFS design space. The following functions were involved in developing power and performance models: workload execution across GPU's DVFS design space, metric collection, feature analysis, and model construction. For developing models, we used utilization metrics of only DGEMM (representative of compute-intensive workloads) and STREAM (representative of memory-intensive workloads) micro-benchmarks. The GPU's utilization metrics were collected across the GPU's DVFS design space for the entire execution duration at the sampling interval of 20 ms. To mitigate statistical errors such as run-to-run variations, these benchmarks were run three times for each frequency. As demonstrated in Fig. 1, extensive

Table 3
Platforms used for our evaluations.

Site	Platform	CPU	Memory	OS	GPU	GPU memory	GPU TDP
Chameleon@UC	Dell PowerEdge XE8545	2 × 64 cores × AMD EPYC 7763	512 GB	CentOS 8	GA100 SXM4	80 GB HBM2e	500 W
Chameleon@UC	Dell PowerEdge C4140	2 × 24 cores × Intel Xeon Gold 6230	128 GB	CentOS 7	GV100 PCIe	32GB HBM2	250 W
HPCC@TTU	Dell PowerEdge R740	2 × 20 cores × Intel Xeon Gold 6242	384 GB	CentOS 8	GV100 PCIe	32GB HBM2	250 W
Test Server@AMD	SUPERMICRO AS-4124GS-TNR	2 × 64 cores × AMD EPYC 7742	528 GB	Ubuntu 18.04	MI210 PCIe	64GB HBM2e	300 W

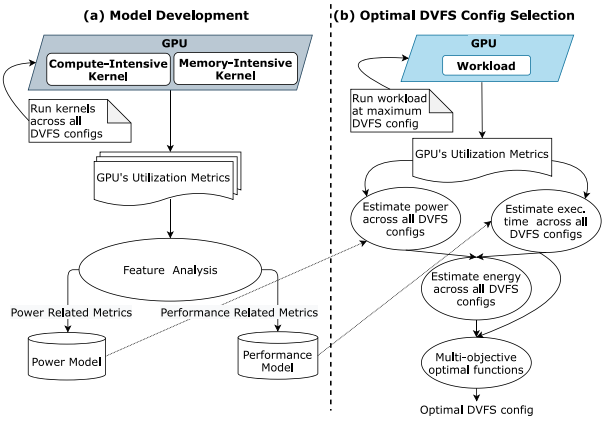


Fig. 2. (a) Functions related to power and performance models development. (b) Functions related to the selection of the optimal frequency for a workload using the proposed models and multi-objective techniques.

analyses were performed to understand the impact of different frequencies on power, execution time, energy, bandwidth, and FLOPS. The metrics were characterized to find their relationship with power and execution time, as shown in Section 4.3. Finally, power and performance models were constructed empirically using the features which showed the highest correlations with power and execution time, respectively. One of the main objectives of these models is the portability and applicability of this methodology to a wide variety of applications and other GPU architectures. We modeled GPU architectural characteristics, which are instrumental in mitigating the changes in power and execution time for a target GPU architecture. These models do not require readjustment based on the target GPU architecture.

As shown in Fig. 2(b), to determine the optimal GPU DVFS configuration, the following steps are involved. First, a workload was run three times for data collection. As performance is paramount to HPC workloads, we collected GPU's utilization metrics at the GPU's maximum DVFS configuration. Furthermore, the power and performance profile at the maximum configuration was used as a reference point for the power and performance profile at the selected DVFS configuration. GPU's metrics were collected at a sampling interval of 20 ms for each run of the workload. Second, the workload's power and execution time were estimated via the proposed power and performance models, respectively. These estimations were performed for each GPU DVFS configuration using the workload's utilization metrics acquired at the maximum frequency. This model-based estimation of power and execution time across a GPU's supported DVFS configurations eliminates the need for the execution of a workload across these different DVFS configurations. The energy for a workload was computed using the estimated power and execution time for each DVFS configuration. Finally, multi-objective functions were used to determine the optimal frequency among the GPU's supported frequencies. These multi-objective functions use EDP and ED^2P, which establish energy-performance trade-offs by simultaneously taking energy savings and performance degradation into account. The EDP function computes the score for each frequency by multiplying the energy and execution time of the DVFS configuration. The DVFS configuration with the lowest score is determined as the optimal frequency. The ED^2P function is similar to EDP; however, ED^2P applies more weight to the execution time. The ED^2P always selects a higher DVFS configuration than the EDP for a given workload. Thus, it is useful in enabling performance-centric energy-saving trade-offs.

4.2. Data collection

We collected 12 GPU utilization metrics (seemingly relevant to power and performance) for DGEMM, STREAM, SPEC ACCEL benchmarks, and three real applications (LAMMPS, NAMD, and LSTM) across 117 DVFS configurations on the NVIDIA GV100. The same metrics were collected for real applications across 61 DVFS configurations on the NVIDIA GA100. We used the state-of-the-art NVIDIA Data Center GPU Manager interface (DCGMI) [33] interface for metric acquisition. The same interface was used to change the DVFS configuration of the GPU.

Table 4 provides the description of the collected metrics. As described above, metrics related to DGEMM and STREAM were used to build the power and performance models. The metrics related to the SPEC ACCEL and real applications were used as the measured data in the model validation demonstrated in Sections 5 and 6. Section 6 also evaluates the inter-architectural portability of the proposed models and the selection of the optimal frequency mechanism using the real applications.

4.3. Feature engineering

In this section, we discuss the process of selecting fine-grained features which directly impact power and execution time. Furthermore, we analyze the impact of different DVFS configurations and input sizes on the selected features, and the portability of these features across GPU architectures.

4.3.1. Selection of the fine-grain features

Feature analysis was performed to choose features that directly impact an application's power usage and execution time. These features are critical to developing accurate, reliable, and scalable analytical models for power and execution time estimation. We used the Mutual Information (MI) technique [34–36] to identify the features correlated with power and execution time. MI estimates distances using nonparametric k-nearest neighbors algorithm. This approach shows an unbiased correlation, which is more effective than the correlation (often algorithm-specific) shown by a machine learning algorithm. As a representative of compute-intensive and memory-intensive applications, the feature analysis used the dataset for DGEMM and STREAM benchmark applications only. Fig. 3 shows the dependency between *power_usage* and *run_time*, and other GPU utilization features. The feature with a higher mutual correlation value (close to 1) is indicative of a higher dependency. Out of these features, we observed that *fp_active*, *sm_app_clock*, and *dram_active* are the most prominent features that influence both power usage and execution time.

The *fp_active* and *dram_active* are instrumental in understanding the computational intensity of a workload. In general, compute-intensive applications show higher *fp_active* than the memory-intensive applications as depicted in Fig. 4 where floating-point

Table 4
Feature description.

Feature	Description
power_usage	Last measured power draw for the entire board. [From DCGMI.]
dram_active	Fraction of cycles where data was sent to or received from device memory. It reports a value between 0 and 1 that represents an average activity over a time interval. For example, an activity of 0.2 indicates that 20% of the cycles read from or write to device memory over the time interval. [DCGMI.]
fp64_active	Fraction of cycles where the FP64 (double precision) pipe was active. It reports a value between 0 and 1 that represents an average over a time interval. [DCGMI.]
fp32_active	Fraction of cycles where the FP32 (single precision) pipe was active. The value is defined similarly to fp64_active feature. [DCGMI.]
gr_engine_active	Overall graphics engine activity. The value (between 0 and 1 represents an average over a time interval. [DCGMI.]
sm_app_clock	Application level SM clock frequency (MHz). [DCGMI.]
sm_active	Fraction of time at least one warp was active on a multiprocessor, averaged over all multiprocessors. Warps both performing actively computing and waiting on memory requests are considered active. The value [0:1] represents an average over a time interval. Usually, a value of 0.8 or higher indicates effective usage of GPU. [DCGMI.]
sm_occupancy	Fraction of resident warps on a multiprocessor, relative to the maximum number of concurrent warps supported on a multiprocessor. The value [0:1] represents an average over a time interval. The higher occupancy does not always represent optimum GPU usage. [DCGMI.]
pcie_tx_bytes	Bytes sent by PCIe. [DCGMI.]
pcie_rx_bytes	Bytes received by PCIe. [DCGMI.]
gpu_utilization	Fraction of time the compute pipe was busy. The value represents an average over a time interval. [DCGMI.]
run_time	Execution time of a specific benchmark kernel. Sourced as *wall time*.

activity (fp_active) for DGEMM is higher than STREAM. While DGEMM is also shown to have considerable memory activity (dram_active), the value for STREAM is much higher. Moreover, we observed that the inclusion of *gpu_utilization* does not improve the prediction accuracy, and the *pcie_** metrics did not provide any significant improvement for our models either.

The *sm_app_clock* is used to scale the power and execution time of an application. For both benchmarks, power decreases (as depicted in Fig. 1-(a) and (e)) with *sm_app_clock* while the execution time increases (as illustrated in Fig. 1(b) and (f)). It shows that a change in GPU frequency changes the power and execution time in a computational intensity-aware way. Therefore, it can be deduced that the GPU metrics *fp_active*, *dram_active*, and *sm_app_clock* are reliable features for controlling an application's power usage and time.

4.3.2. Impact of DVFS on computational activities

We further investigated the impact of changes in DVFS configurations on the computational activities (i.e., fp_active and dram_active) of memory- and compute-intensive applications.

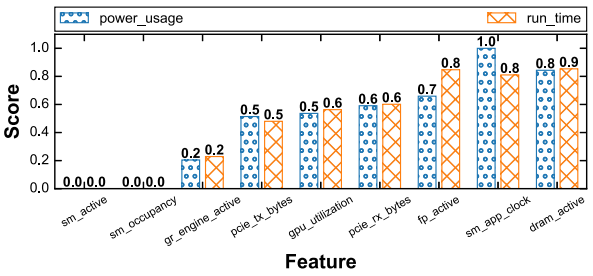


Fig. 3. Dependency between GPU's utilization metrics, power and time.

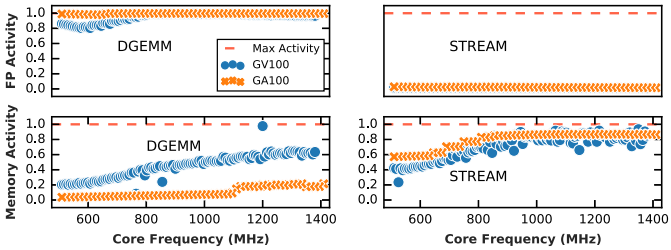


Fig. 4. Impact of DVFS on the computational activities (i.e., *fp_active* and *dram_active*) of memory- and compute-intensive applications.

DGEMM and STREAM were tested by changing the DVFS configurations at maximum input sizes on GA100 and GV100 architectures. As demonstrated in Fig. 4, the floating-point activity is almost unaffected by the change of DVFS configurations for both compute- and memory-intensive applications; however, memory-activity shows variations for both applications. We also observed that DGEMM exhibited different memory usage behaviors across GV100 and GA100. We will explain the rationale behind this deviation later.

4.3.3. Impact of input size on computational activities

We investigated the impact of changes in input sizes on the computational activities of memory- and compute-intensive applications. DGEMM and STREAM were tested using different input sizes at the maximum core frequency on GA100 and GV100 architectures, as depicted in Fig. 5.

As in the case of changes in frequency, we observed similar patterns concerning the change of input sizes on computational activities. The floating-point activity is approximately unaffected by the change of input sizes for both applications on both GPU architectures. The memory activity of DGEMM showed variations within and across both architectures. Unlike in the previous case, the memory activity of STREAM was observed to be mostly unaffected by the change in input sizes on both architectures. In addition, our preliminary analyses confirm that a change in input sizes of memory and compute-intensive applications does not change their power signature [37].

4.3.4. Features portability across GPU architectures

We analyzed the portability of fp_active and dram_active reported by memory-intensive (STREAM) and compute-intensive (DGEMM) kernels across GV100 and GA100 architectures. Figs. 4 and 5 corroborate five findings concerning to portability of these features across GPU architectures: (1) floating-point activity for memory- and compute-intensive kernels was reported the same on both architectures and were unaffected by the change in DVFS configuration and the change in input size; (2) memory activity is nearly unimpacted by the change in input size for a memory-intensive kernel; (3) memory activity to some extent showed variation with the change in DVFS configuration for both kernels;

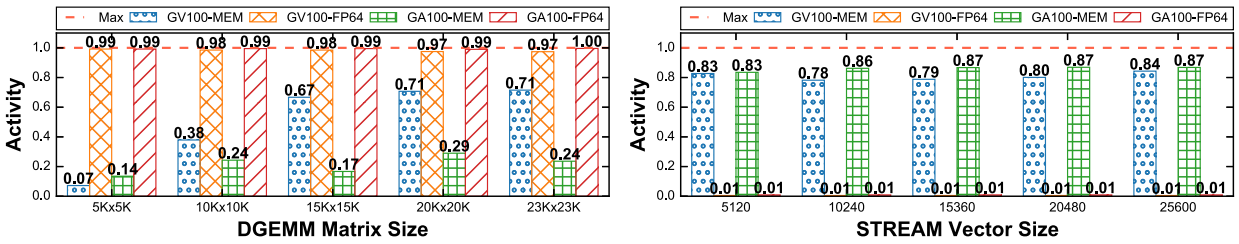


Fig. 5. Impact of different input sizes on the computational activities (i.e., *fp_active* and *dram_active*) of memory- and compute-intensive applications.

and (4) non-uniform memory activity patterns on GV100 and GA100 for DGEMM benchmark.

While GA100 memory frequency (i.e., 1593 MHz) is significantly higher than the GV100 memory frequency (i.e., 877 MHz), results showed comparatively low memory activity on GA100 for DGEMM. We investigated this deviation by looking into the architectural characteristics of both GPUs. We found that GA100 is enhanced with double-precision tensor cores, which support double-precision matrix multiply-accumulate (DMMA) instruction. A single DMMA instruction (on GA100) is equal to eight traditional FP64 instructions (on GV100) [38]. This architectural enhancement enables GA100 to save significant memory space and bandwidth. The reduction in memory activity for DGEMM on GA100 (Fig. 4) is due to its support for Double-Precision Tensor Cores capability.

Summary: MI technique confirms `fp_active`, `dram_active`, `sm_app_clock` as the top three features exhibiting a strong relationship with power and execution time. sm_app_clock (DVFS configuration) is a hardware feature of the target GPU. fp_active is unaffected by the change in sm_app_clock, the change in the input size, and the change of GPU architecture. dram_active is slightly affected by the change in sm_app_clock, the change in the input size, and the change in GPU architecture. Overall, fp_active and dram_active of an application can uniquely identify power and execution time signature for a given GPU sm_app_clock.

4.4. Power modeling

To develop a power model, it is essential to consider the aspects of applications and architectures that directly influence power. Our empirical analysis indicates that the floating-point and memory activities directly impact the (dynamic) power at a given core frequency. This implies that the floating-point and memory activities are reliable features to identify an application's power signature. The power is shown to increase approximately in a linear manner up to the GPU's TDP, depending upon the application's activity. With these underlying basics, we use floating-point activity (FP_{act}), memory activity($DRAM_{act}$), and core frequency (f) to model activity-driven power (P_f) behavior of an application as shown in Eq. (1).

$$P_f = \alpha \cdot FP_{act} + \beta \cdot DRAM_{act} + \gamma \cdot f + C \pm \lambda \tag{1}$$

where α, β, and γ represent regression coefficients for floating-point activity, memory activity, and core frequency, respectively, and C is a constant. These coefficients are estimated using metrics data from DGEMM and STREAM benchmarks. λ is a constant factor that essentially scales up or down power for other GPU architectures. Its value is a ratio of the target GPU's core count to the base GPU's core count. When the core count of the target GPU is *more* than the core count of the base GPU, the resultant value is calculated by adding this value. However, when the cores count of the target GPU is *less* than the cores count of the base GPU, the resultant value is estimated by subtracting this value. Moreover, computing the coefficients for the models has no noticeable overhead. In our evaluations, the estimation of power and execution time, along with the selection of the optimal frequency, took less than a second.

4.5. Performance modeling

While the execution time of individual kernels is predictable based on their computational activities, repetitive tasks and different data input sizes involved in real-world applications make execution time estimation complicated. The execution time depends on the input size, and literature [39] confirms our observations. The proposed performance model requires the execution time of a workload at the maximum-frequency, and then our model scales the execution time for other frequencies. Another key point in designing a DVFS-based performance model is to consider the impact of frequency scaling on time. Based on our observations, the execution time exhibits nonlinear inverse relation with GPU's core frequencies, as shown in Fig. 1-(b) and (f). To address this challenge, researchers use the application's execution time at maximum core frequency as the application's default execution time and linearly estimate the variations in the execution time for the remaining core frequencies. For example, recent literature [39] tried to estimate the change in execution time in relation to a change in core frequency by using the application's default execution time as an input execution time. Our evaluation of [39] shows two fundamental shortcomings. First, the execution time estimation is limited to compute-intensive applications. Second, the change in the estimated execution time when the frequency is changed from the GPU's highest frequency is estimated in linear rather than the desired nonlinear fashion. Therefore, we model these nonlinear (nearly parabolic) behaviors demonstrated in Fig. 1-(b) and (f) as a second-degree polynomial function of floating-point activity (FP_{act}) and change in frequency (Δf). The performance model is derived using Eqs. (2), (3), and (4). The performance model is intended to estimate nonlinear variations in the application's execution time between the highest core frequency and the remaining core frequencies.

$$T_f = T_{f_{max}} + T_{f_\Delta} \tag{2}$$

where T_f denotes the execution time at frequency f, $T_{f_{max}}$ represents the execution time at the highest frequency, and T_{f_Δ} refers to the change in execution time from the maximum core frequency to the given core frequency f, which is determined using Eq. (3).

$$T_{f_\Delta} = \beta_1 \cdot FP_{act} + \beta_2 \cdot \Delta f + \beta_3 \cdot {FP_{act}}^2 + \beta_4 \cdot FP_{act} \cdot \Delta f + \beta_5 \cdot \Delta f^2 \tag{3}$$

where FP_{act} refers to the application's FP activity at maximum frequency and Δf denotes the change in frequency from maximum to the given frequency as shown in Eq. (4).

$$\Delta f = f_{max} - f \tag{4}$$

The β_1, β_2, β_3, β_4, and β_5 are polynomial coefficients, which are estimated using variations in execution time corresponding to changes in frequency configurations and application's FP activity. These estimations were empirically computed using metrics data from DGEMM and STREAM benchmarks. The inclusion of FP activity is critical because it reflects the application's computational activity (see Fig. 1(b) and (f)).

Table 5
Power and performance estimation accuracy for SPEC ACCEL benchmark applications.

	tpacf	stencil	lbm	fft	spmv	mriq	histo	bfs	cutcp	kmeans	lavamd	cfd	nw	hotspot	lud	ge	srad	heartwall	bplustree
Power(%)	98.5	86.1	94.5	90.8	89.6	97.8	83.2	98.3	93.7	83	98.4	95.1	94.2	90	90	96.3	90.8	99.1	94.6
Time (%)	91.9	92.2	97.8	86.3	97.5	81.9	98.8	98.7	85.5	97.6	80.6	96.2	98.3	87.2	94.1	98.4	94.8	96.2	93

4.6. Optimal frequency selection

As already discussed in Section 2.3, the optimal frequency is the one that reduces the power with no performance degradation (ideally) or achieves the best trade-off between execution time and power. The optimal frequency for an application is selected using a multi-objective approach including EDP [6,17–19] and ED^{2}P. These approaches require energy and execution time estimations. The energy is computed for each frequency (f) using Eq. (5) based on the power usage and execution time estimated via the proposed power and performance models.

$$E_{f_{estimated}} = P_{f_{estimated}} \times T_{f_{estimated}} \tag{5}$$

Algorithm 1 Optimal frequency determination using ED^{2}P

Require: $E_1 \ldots E_N, T_1 \ldots T_N, F_1 \ldots F_N$ ▷ list of energies, run times, and frequencies
Ensure: f ▷ optimal frequency

```
function OPTIMAL(E[ ], T[ ], F[ ])
    EDP ← E × T^2                      ▷ compute list of EDP scores
    min ← 0
    index ← 0
    N ← length(ED^2P)
    for k = 1 to N do                  ▷ find the minimum EDP score
        if ED^2P_k < min then
            min ← ED^2P_k
            index ← K
    f ← F_index                        ▷ optimal frequency
```

The algorithm for selecting the optimal frequency among supported DVFS configurations is straightforward and shown in Algorithm 1. This algorithm takes three lists, including energy (`E`), execution time (`T`), and frequency (`F`) as input. It outputs the optimal `f` setting based on the ED^{2}P score. The algorithm involves two major steps: First, the ED^{2}P score for each set of energy and time is computed by multiplying the energy with the square of execution time. Second, the lowest score decides the optimal energy-delay profile out of the given sets of energy and time for the given workload. The frequency (`f`) corresponding to the lowest score is the optimal frequency and will be selected as the optimal frequency. The optimal frequency selection using EDP is similar to this algorithm. The only difference is that the EDP score is calculated instead of the ED^{2}P score, where the energy is multiplied by the execution time (i.e., energy and time are given equal weights).

5. Evaluation with SPEC ACCEL benchmarks

This section provides evaluation results for 19 benchmark applications in the SPEC ACCEL suite (see Table 2). Their utilization metrics were unseen by our proposed models.

5.1. Estimation of power and performance

The power usage and execution time were estimated for the SPEC ACCEL benchmarks across 117 DVFS configurations on GV100 using the proposed power and performance models.

An application's power was estimated using the frequency along with the FP and DRAM activities acquired at the maximum frequency. Fig. 6 compares the estimation power generated by the proposed power model and measured power for each benchmark in the SPEC ACCEL. We used the mean absolute percentage error (MAPE) metric to understand the accuracy of the proposed models. As shown in Table 5, the proposed power model estimated power usage for 15 of the benchmarks in the SPEC ACCEL with an accuracy of over 90% (and up to 99.1%). However, the model slightly overestimated or underestimated power usage for the benchmarks with significantly low or high computational activities. For example, `hist` (FP=0.0005, DRAM=0.0235) and `kmean` (FP=0.0243, DRAM=0.3197) overestimated power usage. Conversely, `stencil` (FP=0.2781, DRAM=0.7301) underestimated power usage.

For estimating the execution time, only frequency and FP activity were used. Fig. 7 compares the execution time estimated by the proposed performance model and measured execution time for each benchmark in the SPEC ACCEL. The execution time was estimated with an accuracy of more than 90% (and up to 98.8%) for 15 benchmarks, as shown in Table 5. We did not observe any underestimation of execution time. However, the proposed model is likely to slightly overestimate execution time for a benchmark exhibiting higher FP activity (e.g., `lavamd`, `mriq`).

Table 6
The optimal DVFS at GV100 selected with measured-EDP, estimated-EDP, measured-ED^{2}P and estimated-ED^{2}P for SPEC ACCEL benchmark.

Benchmark	Optimal frequency (MHz)			
	EDP		ED^{2}P	
	Measured	Estimated	Measured	Estimated
tpacf	907	1020	997	1110
stencil	1102	1020	1102	1102
lbm	907	982	997	1065
fft	1102	1065	1102	1155
spmv	1102	990	1102	1072
mriq	960	1117	997	1207
histo	1050	982	1102	1057
bfs	1200	982	1200	1057
cutcp	907	1072	997	1162
kmeans	1072	990	1072	1065
lavamd	990	1132	990	1230
cfd	1072	997	1102	1080
nw	997	982	997	1065
hotspot	907	1057	907	1147
lud	1222	1005	1222	1095
ge	997	982	997	1065
srad	997	1005	1102	1087
heartwall	997	997	997	1080
bplustree	907	1012	997	1102

5.2. Optimal frequency selection

The (measured) M-EDP and M-ED^{2}P optimal frequencies refer to the optimal frequencies selected via EDP and ED^{2}P approaches using *measured* energy and execution time metrics. Similarly, (estimated) E-EDP and E-ED^{2}P optimal frequencies refer to the optimal frequencies selected via EDP and ED^{2}P approaches using energy and execution time metrics *estimated* by the proposed models. Fig. 8 shows the optimal frequencies selected via M-EDP, E-EDP, M-ED^{2}P, and E-ED^{2}P approaches for each benchmark in the SPEC ACCEL on GV100. Table 6 lists M-EDP, E-EDP, M-ED^{2}P, and E-ED^{2}P optimal frequencies for each SPEC ACCEL benchmark on GV100.

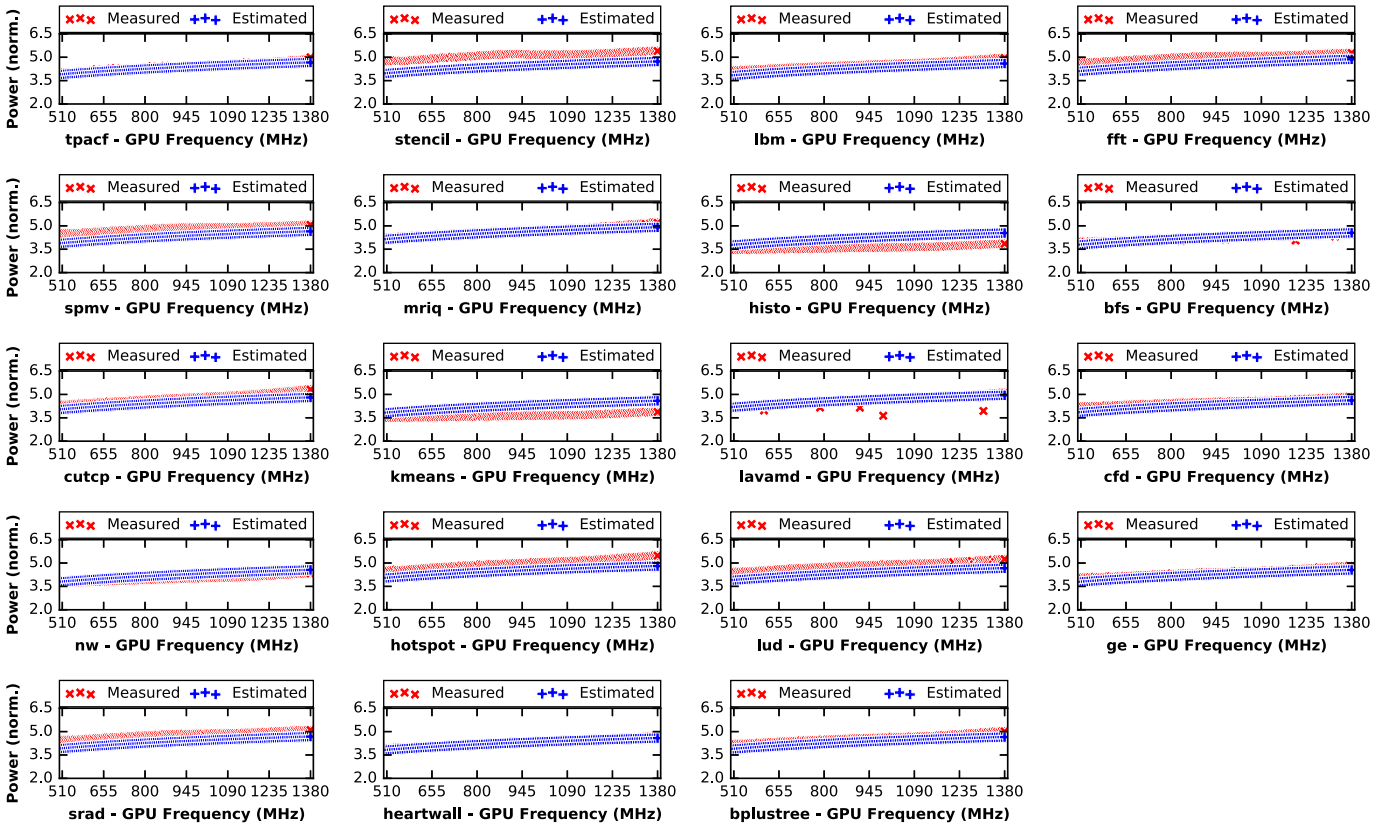


Fig. 6. Comparisons between evaluated and estimated power for each benchmark in the SPEC ACCEL.

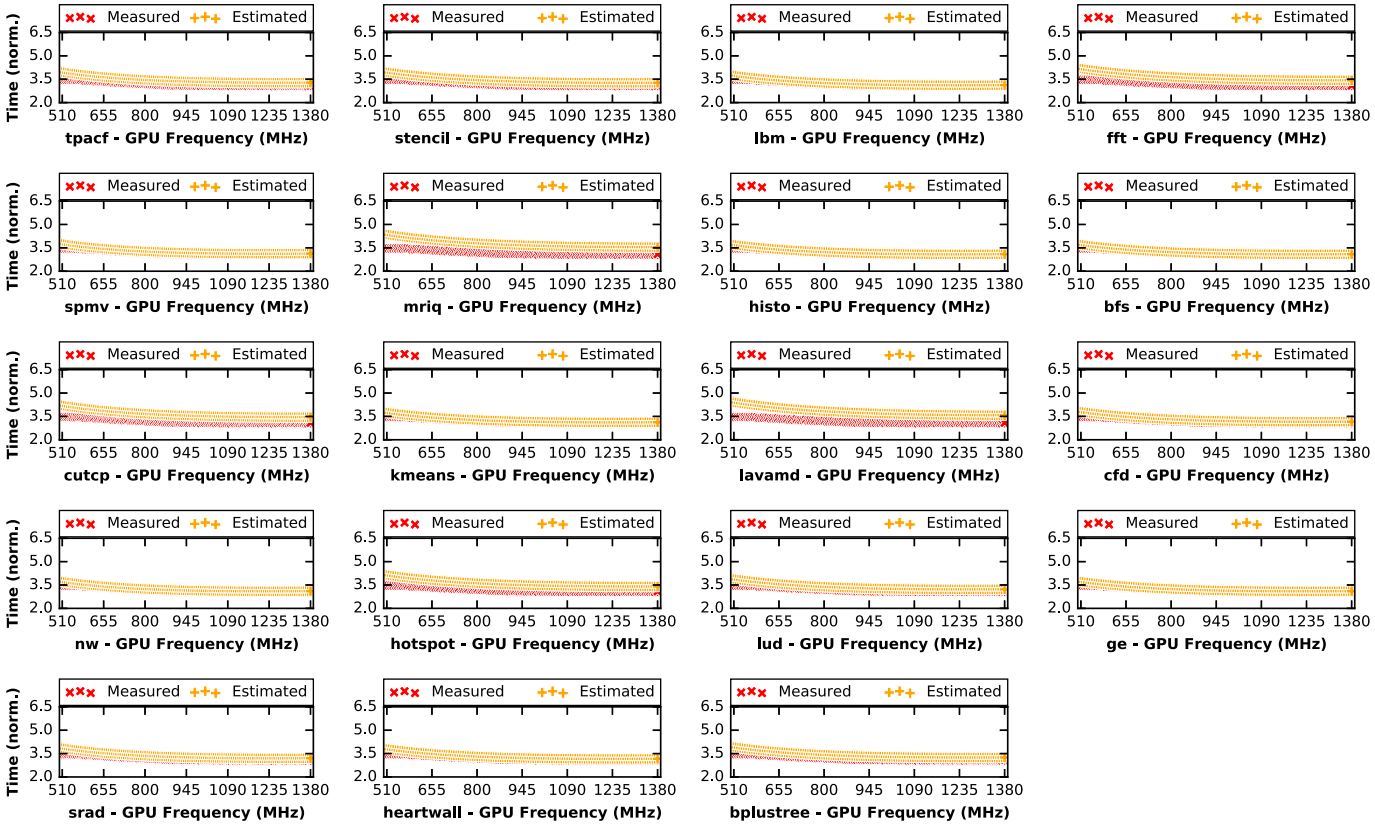


Fig. 7. Comparisons between evaluated and estimated execution time for each benchmark in the SPEC ACCEL.

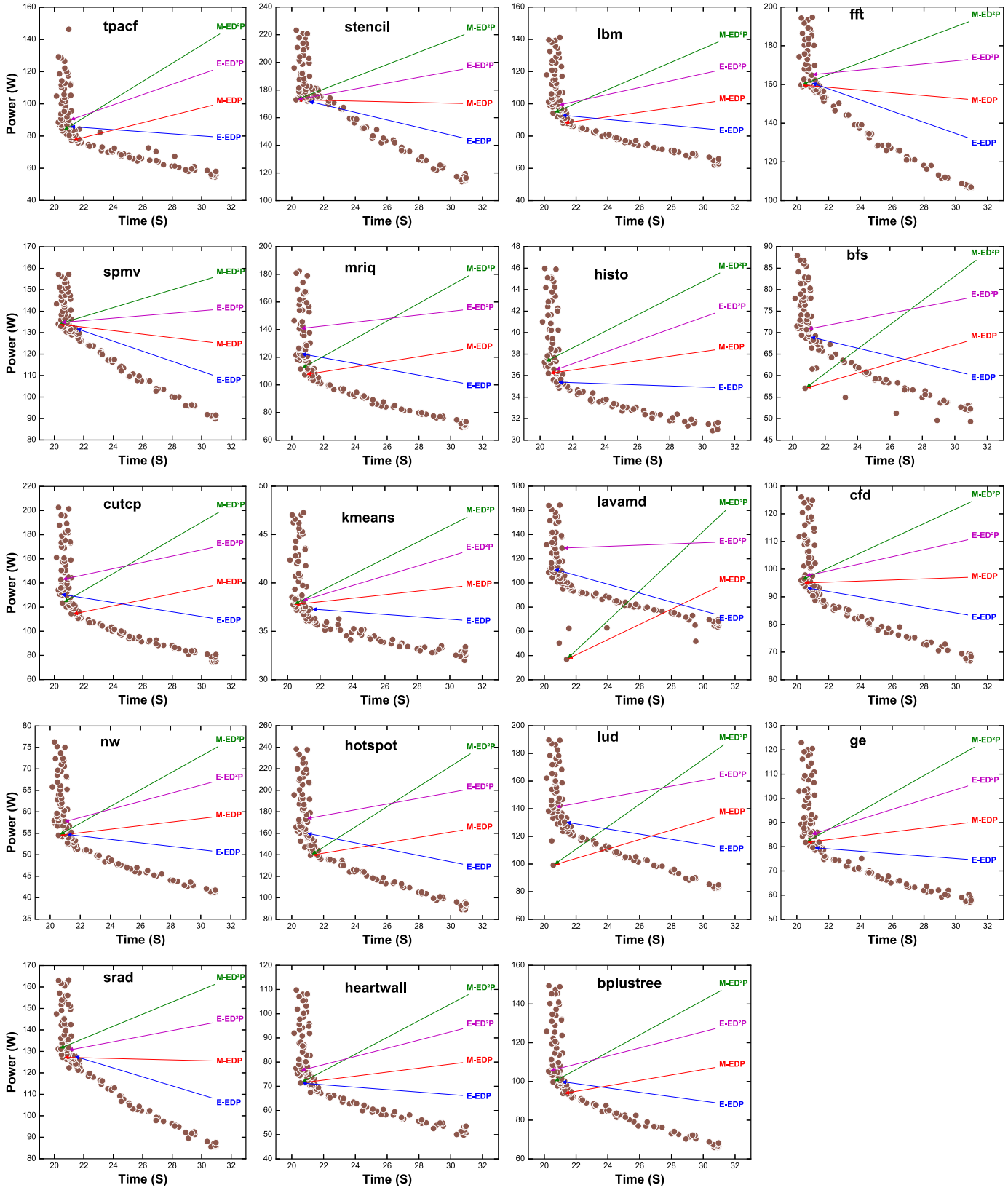
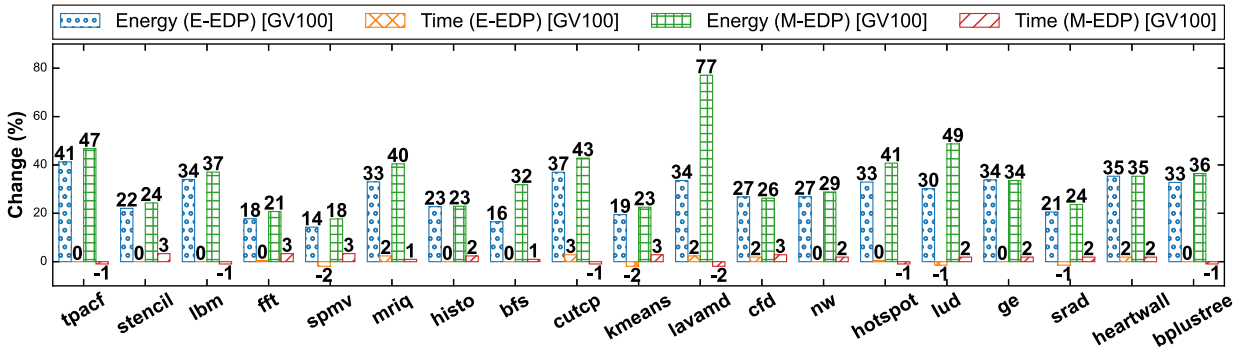
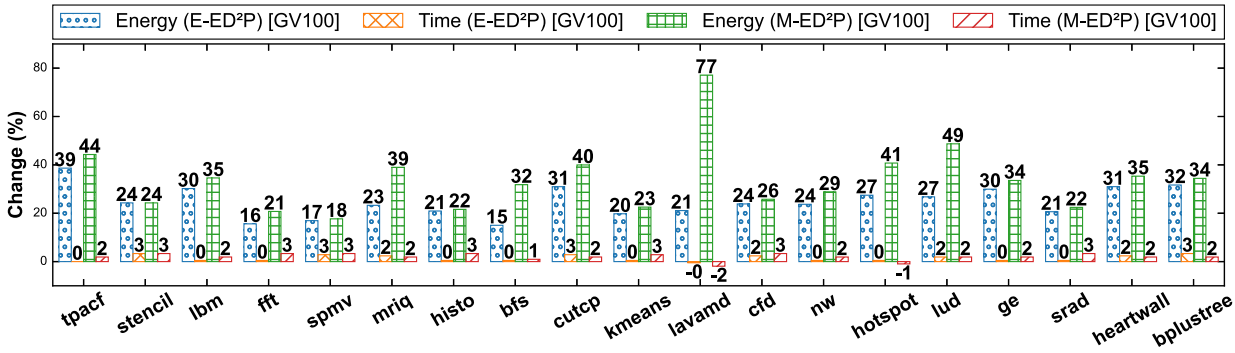


Fig. 8. The optimal DVFS profiles at GV100 selected with measured-EDP, estimated-EDP, measured-ED^{2}P and estimated-ED^{2}P for each benchmark in the SPEC ACCEL shown along with the power and execution time for each supported DVFS configurations.

In general, the M-EDP, E-EDP, M-ED^{2}P, and E-ED^{2}P optimal frequencies for each benchmark were less than the GPU's maximum frequency. This observation confirms our hypothesis that the GPU's *maximum frequency is not always optimal*. Further, E-ED^{2}P optimal frequency selected for each benchmark was always higher than the E-EDP optimal frequency. This outcome affirms



(a) Energy savings and change in execution time achieved with measured and estimated EDP optimal frequencies for each SPEC ACCEL benchmark application.



(b) Energy savings and change in execution time achieved with measured and estimated ED^{2}P optimal frequencies for each SPEC ACCEL benchmark application.

Fig. 9. Energy savings and change in execution time achieved with (a) measured and estimated EDP optimal frequencies and (b) measured and estimated ED^{2}P optimal frequencies for each SPEC ACCEL benchmark application.

our assumption that ED^{2}P approach is useful in defining more performant trade-offs. We also observed a symbiotic relationship between models accuracy, and P-EDP and E-ED^{2}P optimal frequencies: (1) A higher accuracy in estimation of power usage and execution time for a benchmark lead to the selection of more accurate E-EDP and E-ED^{2}P optimal frequencies (e.g., `ge`, `nw`); (2) an overestimated power lead to the selection of comparatively lower E-EDP and E-ED^{2}P optimal frequencies (e.g., `kmean`, `histo`); and (3) an overestimated execution time lead to selection of comparatively higher E-EDP and E-ED^{2}P optimal frequencies (e.g., `lavamd`, `mriq`). The actual energy-performance trade-offs are evaluated below.

5.3. Energy and performance evaluation

The effectiveness of the optimal frequency is measured by its ability to save energy with minimal performance degradation. The change in execution time and energy savings of the optimal frequency are calculated with reference to the GPU's highest frequency. The change in execution time can be computed using Eq. (6):

$$T\_Change(\%) = 100 \cdot \left(\frac{T_{maximum} - T_{optimal}}{T_{maximum}} \right) \tag{6}$$

where $T_{maximum}$ and $T_{optimal}$ are the measured execution times for the application at maximum and optimal frequencies, respectively. The *T_Change* can be either positive or negative. A positive value indicates performance gain, and a negative value suggests performance degradation using the optimal frequency. The energy savings can be computed using Eq. (7):

$$E\_Savings(\%) = 100 \cdot \left(\frac{E_{maximum} - E_{optimal}}{E_{maximum}} \right) \tag{7}$$

where $E_{maximum}$ and $E_{optimal}$ are energy consumed as measured at the maximum and optimal frequencies, respectively.

Fig. 9 shows energy savings and changes in execution time achieved with (a) measured and estimated EDP optimal frequencies and (b) measured and estimated ED^{2}P optimal frequencies for each SPEC ACCEL benchmark application. On average, M-EDP and E-EDP (Fig. 9(a)) saved energies 36% and 28.6% with performance gains 0.3% and 1.2%, respectively. Similarly, M-ED^{2}P and E-ED^{2}P (Fig. 9(b)) collectively saved energies 35.2% and 25.2% with performance gains 1.3% and 2%, respectively. We noted that energy savings attained with estimated optimal frequencies are less than the energy savings attained with the measured optimal frequencies. The reason behind this minor undersaving is overestimating the power usage of some benchmarks (as concluded in the previous section). In addition, the measured and estimated energy savings are similar for the benchmarks having higher accuracy in the estimation of their power and execution time (e.g., `ge`, `cfd`). In conclusion, our approach saves energy by one-fourth without performance degradation for the SPEC ACCEL 19 benchmarks. These observations confirm the effectiveness of our approach.

6. Portability evaluation

This section evaluates the portability of the proposed approach from two perspectives: (1) portability with real-world applications, including NAMD, LAMMPS, and LSTM (see Table 1 for more details about the applications), and (2) portability with the state-of-the-art NVIDIA Ampere GPU and AMD Instinct MI210 GPUs (see Table 3 for more details about GPUs). These evaluations confirm the suitability and applicability of the approach across various architectures and vendors.

To evaluate real applications, we used the following configurations. For NAMD, we performed an experiment using the

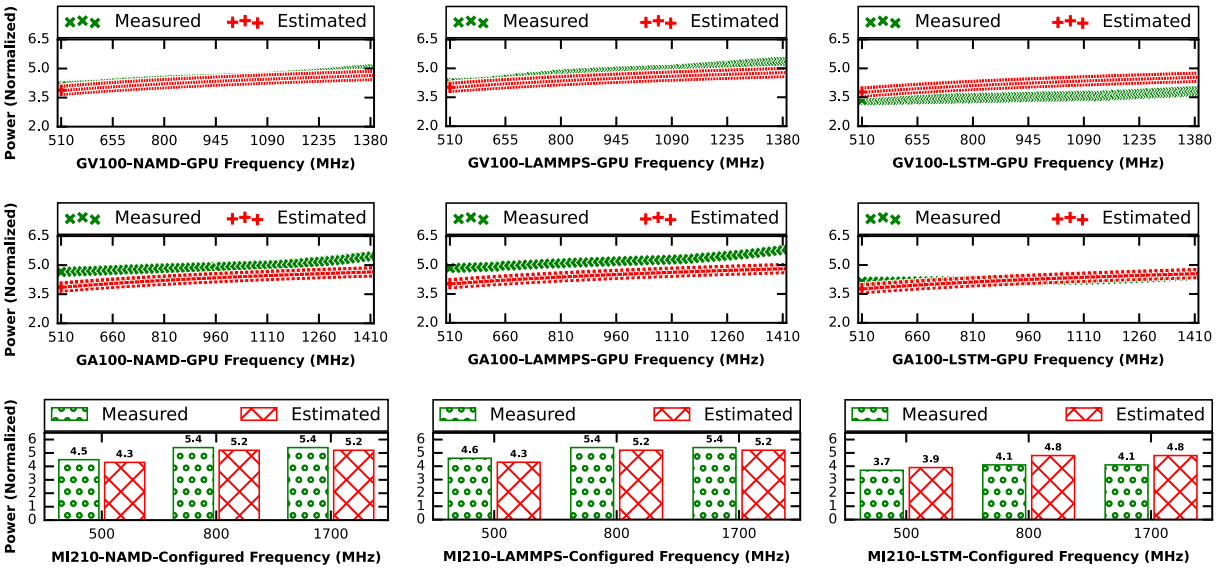


Fig. 10. Comparison of power estimated by the proposed power model and measured power for applications on NVIDIA GV100, GA100, and AMD MI210.

Table 7
Feature mapping between NVIDIA DCGM and AMD rocprof.

NVIDIA DCGM	AMD rocprof
FP_ACTIVE (sum of FP64_ACTIVE, FP32_ACTIVE, FP16_ACTIVE, and TENSOR_ACTIVE)	sum of VALUBusy and SALUBusy
DRAM_ACTIVE	MemUnitBusy

Table 8
Accuracy of power and performance models.

GPU	Application	Power model	Performance model
NVIDIA GV100	NAMD	95.2%	96.9%
	LAMMPS	94.4%	85.5%
	LSTM	80.8%	95.9%
NVIDIA GA100	NAMD	96%	98.2%
	LAMMPS	97.9%	91.4%
	LSTM	80.8%	96.4%
AMD MI210	NAMD	96.1%	97.2%
	LAMMPS	94.5%	93.9%
	LSTM	86.5%	99%

standard Apolipoprotein A1 (ApoA1) dataset, which comprised 92,224 atoms of lipid, protein, and water [40]. ApoA1 simulates a bloodstream lipoprotein. For LAMMPS, we performed a standard Lennard-Jones 3D melt experiment. For LSTM, we used a dataset of 50000 movie reviews for binary sentiment classification; 50% of the movie reviews were used for training and the remaining 50% for testing. In contrast to the benchmarks executed only on the GPU, the real applications run on both CPU and GPU. However, only the corresponding GPU metrics are used in our evaluation.

6.1. Feature mapping

The proposed approach requires floating point and memory activities of an application at the GPU's default frequency, along with the set of supported frequency configurations, in order to estimate power and performance at each frequency. We acquired these features using the NVIDIA DCGM interface for the GV100 GPU. Since the DCGM interface is supported by NVIDIA GA100, these features are inherently portable to NVIDIA GA100. However, for the AMD MI210 GPU, which supports different metrics and interfaces, it posed a challenge to identify equivalent features and related interfaces for data acquisition. After exploring various interfaces and metrics, we confirmed that the AMD rocprof interface provides the necessary low-level architectural features that are equivalent to the features used in our approach. Table 7 illustrates the mapping between the NVIDIA DCGM and AMD rocprof interfaces for the features employed in our approach. We utilized the AMD rocm-smi interface to obtain a list of supported frequency configurations and power consumption data. The same interface was also used to modify the GPU core frequency. For measuring the execution time, we used wall-clock time.

6.2. Estimation of power and performance

In this section, we not only evaluate the portability of the models to real applications but also evaluate the portability to unseen GPU architectures. Real applications, like SPEC ACCEL, are unseen by the power and performance models. The model features for the MI210 were calculated using the feature mapping shown in Table 7. For the real applications, the power and execution time are estimated similarly to that for SPEC ACCEL presented in Section 5.

Fig. 10 provides a comparison of power consumption estimated by the proposed power model and measured power usage for NAMD, LAMMPS, and LSTM on NVIDIA GV100, GA100, and AMD Instinct MI210. The accuracy of the power models for NAMD, LAMMPS, and LSTM on these GPUs are shown in Table 8. On average, power usage estimation for HPC applications was achieved with accuracy $\geq$ 95% on both GPU architectures. As we observed in SPEC ACCEL benchmarks, LSTM overestimated power consumption due to its lower floating-point and memory activities.

Fig. 11 compares execution time estimated by the proposed performance model and measured execution time for NAMD, LAMMPS, and LSTM on NVIDIA GV100, GA100, and AMD Instinct MI210. The accuracy of the performance models on these GPUs is shown in Table 8. These applications showed $\geq$ 91% accuracy except LAMMPS, which showed comparatively low accuracy due to overestimation. We selected the OpenCL-based SPEC ACCEL benchmark suite, which is a standard application suite for measuring GPU performance. Compared to OpenCL, CUDA

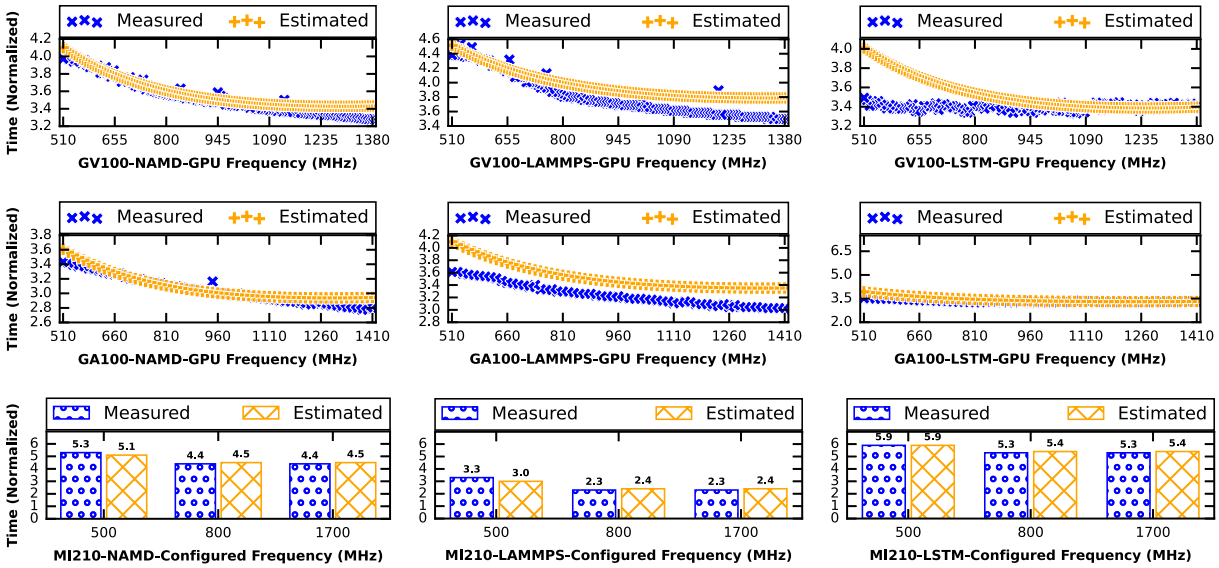


Fig. 11. Comparison of time estimated by our performance model and measured time for applications on NVIDIA GV100, NVIDIA GA100, and AMD MI210.

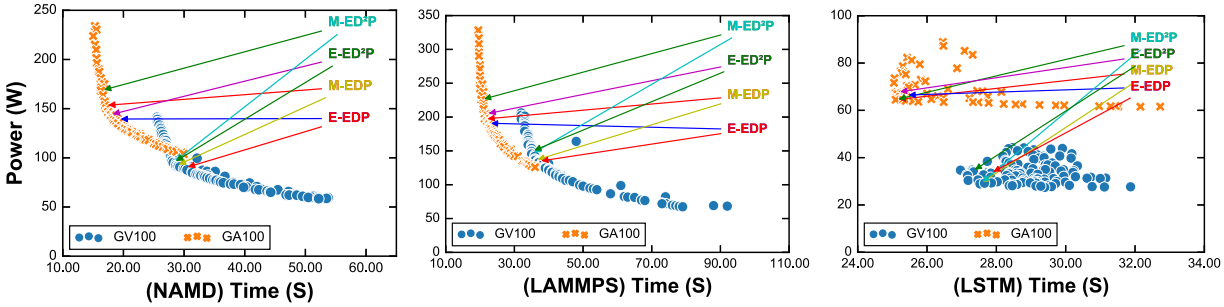


Fig. 12. The optimal DVFS profiles selected with measured(M)-EDP, estimated(E)-EDP, measured(M)-ED^{2}P and estimated(E)-ED^{2}P for NAMD, LAMMPS, and LSTM shown along with the power and execution time for each supported DVFS configurations on GV100 and GA100.

Table 9
The measured EDP, estimated EDP, measured ED^{2}P, and estimated ED^{2}P optimal frequencies for real applications.

GPU	Application	M-EDP	E-EDP	M-ED^{2}P	E-ED^{2}P
GV100	NAMD	1072	1012	1095	1095
(Optimal	LAMMPS	1072	1050	1125	1140
Frequency (MHz))	LSTM	652	975	652	1057
GA100	NAMD	1155	1020	1215	1095
(Optimal	LAMMPS	1110	1065	1215	1155
Frequency (MHz))	LSTM	810	975	810	1065

benchmarks are more optimized for NVIDIA GPUs; however, our selected features (floating-point activity, memory activity, and frequency) are agnostic of OpenCL, CUDA, or HIP. In addition to SPEC ACCEL, our approach predicted power and performance with accuracy between 80% and 99% for CUDA and HIP-based real HPC applications (i.e., LAMMPS, NAMD).

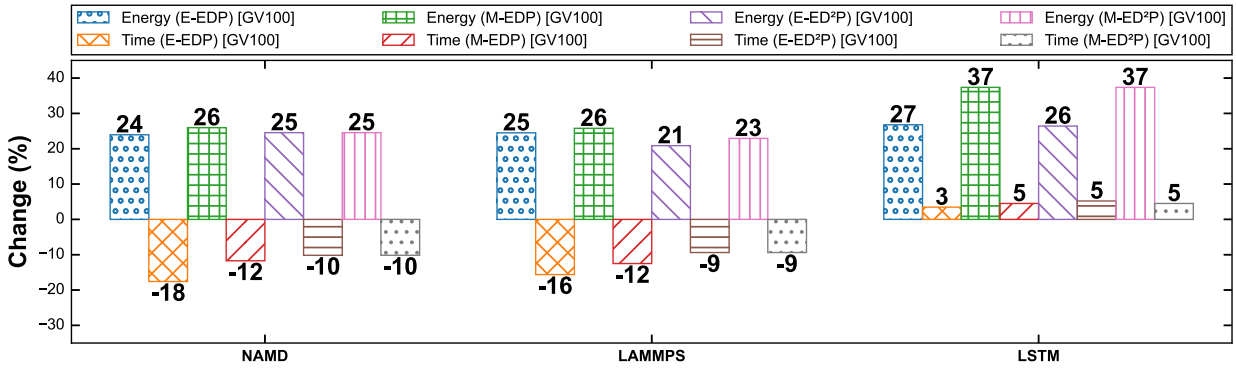
Summary: These results confirm that the inter-architecture and inter-vendor accuracy difference is negligible. The overall accuracy also *confirms the feasibility* of the selected features in estimating power and time on a new GPU architecture (same vendor) and a GPU architecture from another vendor.

6.3. Optimal frequency selection

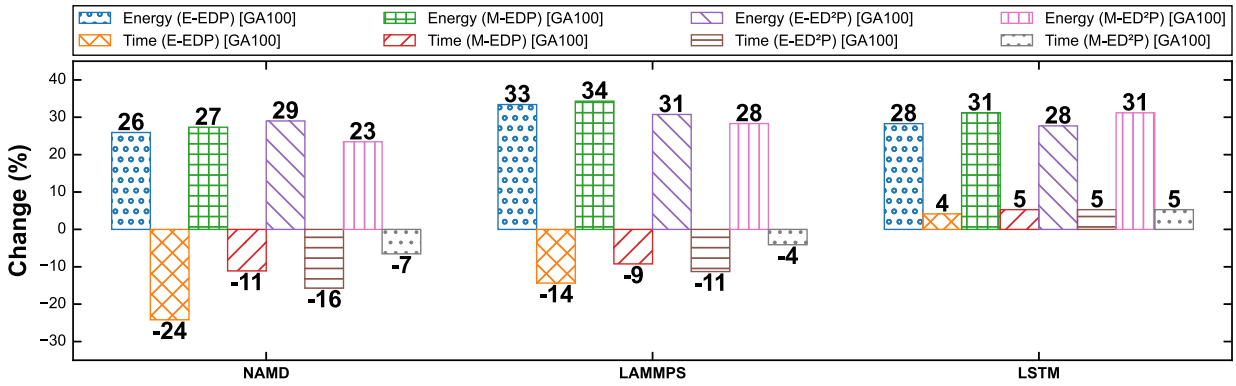
6.3.1. Optimal frequency for NVIDIA GPUs

The optimal frequencies selected for real applications were in alignment with the optimal frequencies selected for the SPEC ACCEL benchmark applications. Fig. 12 shows DVFS profiles selected with measured(M)-EDP, estimated(E)-EDP, M-ED^{2}P, and E-ED^{2}P for NAMD, LAMMPS, and LSTM, along with the power and execution time for each supported DVFS configuration. Table 9 provides a list of estimated and measured optimal frequencies for these applications. For each application, as expected, estimated optimal frequencies using ED^{2}P were *always higher than the optimal frequencies* using EDP. The selection of higher frequencies leads to lower performance degradation. Also, all selected optimal frequencies for real applications were always less than the GPU maximum frequency. Due to a slight overestimation of power usage, the estimated optimal frequencies for LAMMPS and NAMD were slightly less than or close to their measured optimal frequencies. As a consequence of the overestimation of the execution time, the estimated optimal frequencies for LSTM were observed to be higher than its measured optimal frequencies. This observation was consistent with SPEC ACCEL benchmarks.

Concerning the inter-architectural portability of the selected optimal frequency, the difference between the optimal frequencies selected via ED^{2}P for an application on GV100 and GA100 is minimal ($< 1\%$). For example, E-ED^{2}P selected the same optimal frequency (i.e., 1095 MHz) for NAMD on both GV100 and GA100. It *validates the portability of the selection of the optimal frequency technique across* different GPU architectures. It is pertinent to mention that the optimal frequency is kernel or algorithm-specific. Therefore, an application involving multiple kernels may lead to the selection of different optimal frequencies throughout the application's execution lifecycle.



(a) Energy savings and change in execution time achieved with measured and estimated optimal frequencies achieved using EDP and ED^{2}P for applications on GV100.



(b) Energy savings and change in execution time achieved with measured and estimated optimal frequencies obtained with EDP and ED^{2}P for applications on GA100.

Fig. 13. Energy savings and change in execution time achieved with (a) measured and estimated optimal frequencies achieved using EDP and ED^{2}P approaches on GV100 and (b) measured and estimated optimal frequencies achieved using EDP and ED^{2}P approaches on GA100 for NAMD, LAMMPS, and LSTM applications.

Table 10
Average energy saving and change in execution time for real applications using measured and estimated optimal frequencies achieved via EDP and ED^{2}P approaches on NVIDIA GV100 and GA100.

GPU	Approach	Energy saving	Performance
GA100	M-ED^{2}P	↑ 27.4%	↓ −0.6%
	E-ED^{2}P	↑ 29.6%	↓ −5.2%
	M-EDP	↑ 31.7%	↓ −3.4%
	E-EDP	↑ 30.2%	↓ −8.8%
GV100	M-ED^{2}P	↑ 25%	↓ −4.9%
	E-ED^{2}P	↑ 22.6%	↓ −4.7%
	M-EDP	↑ 27.1%	↓ −6.6%
	E-EDP	↑ 24.6%	↓ −9.8%

6.3.2. Optimal frequency for AMD MI210

The AMD MI210 GPU offers three frequency configurations: Minimum (500 MHz), Auto (800 MHz), and Maximum (1700 MHz). The Minimum and Maximum frequency configurations are allowed to be set in `manual` performance mode, while the Auto configuration is applied in `auto` performance mode. In the Auto mode, the frequency starts at 800 MHz as a baseline and dynamically increases based on the workload's computational intensity and GPU's TDP limit.

During our evaluation of real applications on the MI210, we observed that the Minimum configuration resulted in reduced power consumption (see Fig. 10 for MI210 power evaluation). However; it also led to significant performance degradation (see Fig. 11 for MI210 performance evaluation). On the other hand, both Auto and Maximum frequencies exhibited similar power and performance behaviors. This behavior was consistent across memory- and compute-intensive benchmarks as well. Consequently, the frequency configurations available for the MI210 GPU are not suitable for balancing power and performance trade-offs effectively.

Furthermore, we noted a distinction between AMD and NVIDIA GPUs in terms of frequency and voltage configurations. While AMD GPUs offer three frequency options and allow users to adjust multiple voltage configurations, NVIDIA GPUs only allow frequency adjustments and manage voltage internally without user configurability. Based on this observation, we hypothesize that a combination of frequency and voltage configurations specific to a workload could potentially yield improved energy-performance trade-offs for AMD GPUs. In future research, we plan to investigate and study the selection of optimal voltage and frequency configurations for maximizing performance while minimizing energy consumption.

6.4. Energy and performance evaluation

In this section, we evaluate the energy savings and change in execution time in regard to real applications across GV100 and GA100 architectures. Fig. 13 shows energy savings and change in execution time for NAMD, LAMMPS, and LSTM applications achieved with (a) measured and estimated optimal frequencies achieved using EDP and ED^{2}P approaches on GV100 and (b) measured and estimated optimal frequencies achieved using EDP and ED^{2}P approaches on GA100. Overall, the real applications' energy saving and change in execution time are listed in Table 10. In contrast to SPEC ACCEL benchmarks, NAMD and LAMMPS applications showed performance degradation even with the measured optimal frequencies. For NAMD, the measured and estimated optimal frequencies selected via ED^{2}P approach showed exactly the same energy saving (i.e., 25%) and performance loss (i.e., 10%) on GV100. For LAMMPS, the measured and estimated optimal

frequencies selected via ED^{2}P approach showed slightly variable energy savings (i.e., 23% and 21%) at the same performance loss (i.e., 9%) on GV100. On the other hand, LSTM saved 28% and 26% of energy on GA100 and GV100, respectively, with no performance loss. It indicates that an application with *higher computational activities* (e.g., NAMD, LAMMPS) is likely to save energy at the *cost of some performance loss* compared to an application with lower computational activity (e.g., LSTM). In other words, applications with higher computational activity are less likely to have sweet spots of DVFS configurations, reducing energy without any performance penalty. More adaptive approaches could take ED^{2}P execution time as a baseline and scale the execution time to the desired level. These performance-centric approaches would ensure minimal performance degradation while saving energy.

Another important observation is that the proposed approach is able to determine optimal frequency even when there was lower accuracy in estimating power or execution time. For example, LSTM with comparatively lower accuracy (i.e., 80.8%) in the estimation of power consumption on both GPU architectures showed significant energy savings with no performance loss. Based on the energy savings and performance degradation for SPEC ACCEL benchmarks and real applications, ED^{2}P is shown to be a better choice as it offers better power and execution time trade-offs. These energy savings for real-world applications across GPU architectures further confirm the effectiveness of our approach.

7. Comparison to state of the art

7.1. Overview

We compared our approach with Guerreiro et al. [7], which is a state-of-the-art research and method. Guerreiro et al. proposed GPU predictive models, which were trained using a dataset created by sequencing the GPU assembly (i.e., PTX) instructions of the workloads. These models intend to predict changes in execution time, power, and energy consumption and select the minimum-energy frequency configuration.

Among the real workloads used in our evaluations 6.2, only executables of LAMMPS and NAMD support CUDA PTX code. We used the `cuobjdump` tool to acquire the PTX code of LAMMPS and NAMD using the same CUDA executables which were used in collecting the utilization metrics. These executables were built on NVIDIA GA100 GPU. Finally, the `gpuPTXParser` [7] tool was used to parse the PTX code and to generate the dataset for LAMMPS and NAMD.

We used the `gpuPTXModel` [7] tool to train power, time, and energy models using 126 benchmarks for training and 14 benchmarks on GTX Titan X. We used LAMMPS and NAMD for testing using their statistical PTX dataset and performance counters on GA100, including one memory and 61 core frequency configurations along with power, energy, and time across these frequency configurations. The PTX dataset was used for predicting the power, energy, and time of real applications on GA100 using the models trained on GTX Titan X.

7.2. Model comparison

We compared the efficacy of the models proposed in [7] and models proposed in this study. In particular, we compared the prediction accuracy, quality of optimal frequency, and modeling complexity and cost.

The models trained with 126 benchmarks on the GTX Titan X GPU showed errors of 18.9%, 16.7%, and 16.5% for predicting power, performance, and energy, respectively, for the 14 validation benchmarks on the same GPU. We predicted the execution time, power, and energy of real applications on the GA100 using the same models trained on the GTX Titan X. The error rates are shown in Table 11. Our power and performance models showed on average ∼6X improvements compared to the state-of-the-art model in [7]. This prediction accuracy is crucial for enabling real savings. For example, a 5% decrease in power consumption at the scale of the Summit supercomputer could generate savings of ∼1 million dollars [7].

Table 11
Comparison of our models with the state of the art.

Model	App			
	LAMMPS		NAMD	
	Our Work	Guerreiro et al. [7]	Our Work	Guerreiro et al. [7]
Power	2.1%	21.8%	4%	17.6%
Time	8.6%	24.4%	1.8%	34.4%

7.3. Optimal frequency comparison

Guerreiro et al. [7] and our approach use different methods in determining the optimal frequency for a given workload. Guerreiro et al. manually configured a lower-bound optimal frequency (e.g., 80% of the supported maximum frequency). In their study, the maximum frequency is implicitly considered as an upper-bound optimal frequency. A Pareto-optimal is defined using the energy consumption of the frequencies within the lower- and upper-bounds. Their study considers a frequency configuration with the lowest power consumption as an optimal frequency. However, we observed the following caveats in their selection of optimal frequency. First, manually picking a (lower-bound) frequency for all workloads may potentially obstruct the selection of an energy-efficient optimal frequency, especially for memory-bound workloads. Second, our observations indicate that a frequency showing the lowest energy is not necessarily a performant frequency (see Fig. 1 - b, c, f, g).

In our study, as explained in Section 4.6, we addressed these concerns as follows. First, our method does not require manual lower-bound optimal frequency. It rather searches the entire DVFS design space for the optimal frequency. Second, we do not select an optimal frequency, merely exhibiting the lowest energy. Instead, our algorithm selects the optimal frequency that shows minimum energy *with little to no performance degradation*.

8. Other related work and comparison

Existing **analytical models** depend on acquiring some features (e.g., voltage) that involve complex and costly procedures. Moreover, these features do not necessarily influence power and performance effectively across different computational intensities; therefore, the applicability and accuracy of these models are often limited [41–46].

Static code-based models analyze GPU assembly instructions and try to establish their relationship with performance, power, and energy [7,15]. Braun et al. [13] characterized PTX code and attempted to predict execution time and power; however, their study does not explore DVFS configurations.

DVFS is the preferred control for scaling power and performance. Several state-of-the-art studies target only performance or power consumption across different DVFS configurations. Wang et al. [12] proposed a DVFS-based model. However, its scope is limited to the application's performance. Nabavinejad et al. [47] implemented *batchDVFS* approach, which leveraged the batch size of the DNN inference and DVFS technique to control the power and performance. Nevertheless, the scope of this work is

Table 12
Comparison of this study against the state-of-the-art.

Study	Analytical	Static	ML	Real apps	Cross-GPU	Multi-Objective
Guerreiro et al. [7]	✗	✓	✓	✗	✓	✗
Fan et al. [15]	✗	✓	✓	✗	✗	✗
Wu et al. [8]	✗	✗	✓	✗	✗	✗
Our Work	✓	✗	✗	✓	✓	✓

limited to DNN workloads. Guerreiro et al. [48] proposed DVFS-based power and performance models; however, they do not offer an optimal DVFS configuration. Similarly, Dutta et al. [11] provided a DVFS-based ensemble machine learning framework that only predicts power usage for a target GPU frequency. Wu et al. [8] attempted to define clusters of kernels exhibiting similar power and performance patterns across GPU's DVFS design space. Then, it used machine learning techniques to map a new kernel to one of the clusters. As the models are based on coarse-grained performance counters; thus, prediction accuracy is limited. Such models will result in the selection of a sub-optimal GPU profile. In contrast, our work proposes a fine-grain and comprehensive approach to predict power, execution time, and energy with better accuracy and to select a performance-aware optimal frequency.

Numerous studies attempt to develop **machine learning-based models** to predict the application's power and execution time. However, the portability of the ML-based model is a concern. For example, a real HPC application's execution time is unbounded and can vary based on input sizes. Therefore, an ML-based model trained with a particular input size cannot effectively predict the application's execution time with different input sizes or an unseen application.

Multi-objective solutions mainly involve two approaches. First, the Pareto-optimal uses a set of solutions that any member of the solution set does not dominate. Second, optimal decision-making techniques narrow down to a single solution from the available set. EDP [6], ED^2P, and MCDM [49–51] are the prominent decision-making techniques. Guerreiro et al. [7], and Fan et al. [15] use PTX-based assembly code to extract features related to GPU performance, power, and energy across DVFS configurations using ML-based models. These studies leverage the Pareto-optimal mechanism to find the optimal set of DVFS configurations. While PTX-based code modeling is promising as it does not require prior application execution, it is challenging to determine utilization, especially for memory access patterns.

Summary: This research differentiates itself from the previous studies on several aspects as highlighted in Table 12. First, as the low-level application's utilization metrics and architecture's scaling features are used in developing the analytical models, no static source code analysis and ML-based modeling are required. Second, the multi-objective algorithm can use EDP, ED^2P, or potentially other functions to determine the optimal frequency. We have observed that changing the objective function has no impact on the underlying analytical models. On average, these frequencies save one-fifth of the energy with little to no performance loss. Third, it is one of the first studies that evaluate the portability and feasibility of optimal frequency selection with the state-of-the-art NVIDIA GA100 GPU. On GA100, our models showed accuracy up to 98% and saved one-fourth of the energy consumption of real applications. Finally, unlike the majority of the prior works, we use two HPC applications and one real ML application in our evaluation to further demonstrate our proposed approach's effectiveness.

9. Use in HPC production environment

The proposed techniques can be integrated into the HPC production environment via two methods. The first is the offline method that involves executing the desired application at the maximum frequency to acquire the utilization metrics (i.e., floating-point and memory activities) to estimate the application's power and execution time across the DVFS configurations supported on the target GPU. Afterward, the optimal DVFS configuration is determined and stored in a database. When the same application is scheduled for execution, the workload manager (e.g., Slurm) fetches the application's optimal configuration (as a part of the Slurm Prolog mechanism) and interjects the optimal frequency into the job script. The optimal frequency is applied to the target GPU at the application's execution time. This method requires the user to submit the job using an identifiable job name. Furthermore, the DVFS configuration cannot be readjusted during the execution lifecycle of the application. This method is useful for applications with uniform computational activity. We have a template that automates selecting the optimal frequency for a given application with minimal user input.

The second method is the online method which does not require any prior execution or identity of the application. The application's utilization metrics are collected periodically during its execution. When the application's utilization metrics change, power and execution time are estimated across the GPU's DVFS configurations, and the new optimal DVFS configuration is determined accordingly. The optimal configuration is enforced directly using GPU's native interface (e.g., dcgmi, nvidia-smi). This method is suitable for applications comprising multiple kernels (or phases).

10. Conclusions and future work

Power consumption presents an increasingly critical challenge in current and emerging GPU-enabled HPC systems and is the dominant constraint for exascale systems and beyond. Arguably, it is imperative to develop effective GPU power management approaches to lower power while maintaining minimal impact on execution time. The DVFS is a reliable control for regulating power and execution time; however, the DVFS design space for GPU is large; therefore, brute-force approaches are infeasible in selecting the optimal power and execution time. The problem is further compounded by the fact that it is impractical to actually measure power and execution time across all DVFS configurations in the GPU's DVFS design space. Furthermore, the selection of a DVFS configuration (among the DVFS design space) that is optimal in terms of both energy and performance is non-trivial. To address these challenges in a more systematic manner, we came up with an approach that involves three key steps: (1) identification of GPU utilization metrics that influence both the power and execution time of a given workload; (2) development of analytical models to estimate power and execution time across GPU's DVFS design; and (3) selection of optimal frequency using multi-objective optimal functions. To evaluate the efficacy of the proposed approach, we acquired metrics using the state-of-the-art NVIDIA DCGM interface for 24 workloads including two HPC applications, one real ML application, 19 benchmark applications from SPEC ACCEL, and two micro-benchmarks.

Through feature characterization, we have identified key features that directly cause power consumption and change in the execution time of applications with different computational intensities. We empirically developed reliable and scalable models using the identified feature set. The multi-objective approach took optimal performance and optimal energy into consideration simultaneously and selected an energy-efficient optimal DVFS configuration. The accuracy of analytics-based power and performance models for estimating SPEC ACCEL benchmark applications were up to 99% and 98% for the estimation of power and execution time, respectively. On average, the energy savings

for SPEC ACCEL benchmark applications were over 25%, with no performance degradation on GV100. Similarly, the real applications showed over 22.6% energy savings with a performance degradation of 4.7% on GV100.

We validated the portability of the selected feature set, analytical models, and multi-objective approaches on the state-of-the-art HPC-grade NVIDIA GA100 and AMD MI210 GPUs using real applications. The power and performance models developed on GV100 can be used on GA100 with accuracy up to 97.9%. The same models estimated power and performance on MI210 GPU with accuracy up to 96.1% and 99%, respectively. The evaluation showed 29.6% energy savings with a performance loss of 5.2%. Additionally, we conducted a comparison between our models and PTX-based static models. The results revealed a significant reduction in the average error rates, with a decrease from 19.7% to 3.1% for power models and from 29.4% to 5.2% for performance models.

The curated feature set, power and performance estimation models, and systematic determination of the optimal DVFS profile using the multi-objective approach are together a novel attempt toward building an energy-efficient HPC system. In the future, we would like to extend our work to create a solution that encompasses both CPU and GPU to optimize the power draw of the entire node.

CRediT authorship contribution statement

Ghazanfar Ali: Conceived and designed the analysis, Collected the data, Contributed data or analysis tools, Performed the analysis, Writing – original draft. **Mert Side:** Collected the data, Contributed data or analysis tools, Writing – original draft. **Sridutt Bhalachandra:** Conceived and designed the analysis, Writing – original draft. **Nicholas J. Wright:** Conceived and designed the analysis. **Yong Chen:** Conceived and designed the analysis, Writing – original draft.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yong Chen reports equipment, drugs, or supplies, statistical analysis, and writing assistance were provided by E O Lawrence Berkeley National Laboratory.

Data availability

Data will be made available on request.

Acknowledgments

The National Energy Research Scientific Computing Center (NERSC) is a U.S. Department of Energy Office of Science User Facility operated under Contract No. DEAC02-05CH11231. Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. This research is supported in part by the National Science Foundation, United States under grants CNS-1817094, OAC-1835892, and CNS-1939140 (A U.S. National Science Foundation Industry-University Cooperative Research Center on Cloud and Autonomic Computing). The authors thank Mathew Colgrove (NVIDIA) and Max Katz (NVIDIA) for their assistance in SPEC ACCEL and GPU metrics, Thomas Brown (TTU) for reviewing, and Victor Sheng (TTU) and Abdul Serwadda (TTU) for their comments on modeling.

References

[1] NVIDIA A100 GPU datasheet, 2022, https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-nvidia-us-2188504-web.pdf. (Accessed 8 April 2022).
[2] Y. Jiao, et al., Power and performance characterization of computational kernels on the GPU, in: 2010 IEEE/ACM Int'L Conference on Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing, IEEE, 2010, pp. 221–228.
[3] K. Bergman, et al., Exascale computing study: technology challenges in achieving exascale systems, vol. 15, in: Tech. Rep., Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO, 2008.
[4] Top500, Top500, June 2022 Ranking, 2022, TOP500 https://www.top500.org/lists/top500/2022/06/.
[5] M. Shafique, S. Garg, Computing in the dark silicon era: Current trends and research challenges, IEEE Des. Test 34 (2) (2016) 8–23.
[6] J.H. Laros III, et al., Energy delay product, in: Energy-Efficient High Performance Computing, Springer, 2013, pp. 51–55.
[7] J. Guerreiro, et al., GPU static modeling using PTX and deep structured learning, IEEE Access 7 (2019) 159150–159161.
[8] G. Wu, et al., GPGPU performance and power estimation using machine learning, in: 21st International Symposium on High Performance Computer Architecture, IEEE, 2015, pp. 564–576.
[9] R.A. Bridges, et al., Understanding GPU power: A survey of profiling, modeling, and simulation methods, ACM Comput. Surv. 49 (3) (2016) 1–27.
[10] X. Mei, et al., A survey and measurement study of GPU DVFS on energy conservation, Digit. Commun. Netw. 3 (2) (2017) 89–100.
[11] B. Dutta, et al., GPU power prediction via ensemble machine learning for DVFS space exploration, in: Proceedings of the 15th ACM International Conference on Computing Frontiers, 2018, pp. 240–243.
[12] Q. Wang, X. Chu, GPGPU performance estimation with core and memory frequency scaling, IEEE Trans. Parallel Distrib. Syst. 31 (12) (2020) 2865–2881.
[13] L. Braun, et al., A simple model for portable and fast prediction of execution time and power consumption of GPU kernels, ACM Trans. Archit. Code Optim. (TACO) 18 (1) (2020) 1–25.
[14] A. Majumdar, et al., A taxonomy of GPGPU performance scaling, in: 2015 IEEE International Symposium on Workload Characterization, 2015, pp. 118–119, http://dx.doi.org/10.1109/IISWC.2015.22.
[15] K. Fan, et al., Predictable GPUs frequency scaling for energy and performance, in: Proceedings of the 48th International Conference on Parallel Processing, 2019, pp. 1–10.
[16] G. Ali, S. Bhalachandra, N. Wright, M. Side, Y. Chen, Optimal GPU frequency selection using multi-objective approaches for HPC systems, in: 2022 IEEE High Performance Extreme Computing Conference, HPEC, IEEE, 2022.
[17] J. Park, J.A. Abraham, A fast, accurate and simple critical path monitor for improving energy-delay product in dvs systems, in: IEEE/ACM International Symposium on Low Power Electronics and Design, IEEE, 2011, pp. 391–396.
[18] R. Gonzalez, M. Horowitz, Energy dissipation in general purpose microprocessors, IEEE J. Solid-State Circuits 31 (9) (1996) 1277–1284.
[19] V. Mishra, S. Akashe, Calculation of power delay product and energy delay product in 4-bit FinFET based priority encoder, in: Advances in Optical Science and Engineering, Springer, 2015, pp. 283–289.
[20] G. Ali, M. Side, Power analysis and prediction model for GPU architectures in HPC systems, 2020, GitHub repository, GitHub, https://github.com/nsfcac/gpupowermodel.
[21] NVIDIA Corporation, CUDA samples, 2013, NVIDIA Documentation Center, URL https://docs.nvidia.com/cuda/cuda-samples/index.html#matrix-multiplication--cublas-.
[22] T. Deakin, et al., GPU-STREAM v2. 0: Benchmarking the achievable memory bandwidth of many-core processors across diverse parallel programming models, in: International Conference on High Performance Computing, Springer, 2016, pp. 489–507.
[23] NVIDIA Corporation, NGC NAMD container, 2020, UIUC, https://ngc.nvidia.com/catalog/containers/hpc:namd.
[24] J.C. Phillips, et al., Scalable molecular dynamics on CPU and GPU architectures with NAMD, J. Chem. Phys. 153 (4) (2020) 044130.
[25] NVIDIA Corporation, NGC LAMMPS Container, 2021, Sandia National Lab, https://ngc.nvidia.com/catalog/containers/hpc:lammps.
[26] A.P. Thompson, et al., LAMMPS - A flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales, Comput. Phys. Comm. (2021) 108171.
[27] TensorFlow, Long Short-Term Memory layer - Hochreiter 1997, 2021, https://www.tensorflow.org/api_docs/python/tf/keras/layers/LSTM.
[28] Martín Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015, Software available from tensorflow.org. URL https://www.tensorflow.org/.
[29] A.L. Maas, et al., Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Portland, Oregon, USA, 2011, pp. 142–150, URL http://www.aclweb.org/anthology/P11-1015.

[30] G. Juckeland, et al., Spec accel: A standard application suite for measuring hardware accelerator performance, in: International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems, Springer, 2014, pp. 46–67.
[31] K. Keahey, et al., Lessons learned from the chameleon testbed, in: Proceedings of the 2020 USENIX Conference on Usenix Annual Technical Conference, USENIX Association, USA, 2020, pp. 219—233.
[32] HPCC, High Performance Computing Center, 2020, URL http://www.depts.ttu.edu/hpcc.
[33] NVIDIA Corporation, NVIDIA DCGM, 2021, NVIDIA Developer URL https://developer.nvidia.com/dcgm.
[34] F. Pedregosa, et al., Scikit-learn: Machine learning in Python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[35] B.C. Ross, Mutual information between discrete and continuous data sets, PLoS One 9 (2) (2014) e87357.
[36] A. Kraskov, et al., Estimating mutual information, Phys. Rev. E 69 (6) (2004) 066138.
[37] G. Ali, et al., Evaluation of power controls and counters on general-purpose Graphics Processing Units (GPUs), in: SC Research Poster, 2020.
[38] G. Gupta, What is a double-precision tensor core? 2020, NVIDIA Blog URL https://blogs.nvidia.com/blog/2020/05/14/double-precision-tensor-cores/.
[39] S. Ramesh, et al., Understanding the impact of dynamic power capping on application progress, in: 2019 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2019, pp. 793–804.
[40] N. Kashyap, HPC Benchmarks and Applications Performance Study on Broadwell-EP 4S Processor, 2016, Dell URL https://downloads.dell.com/manuals/all-products/esuprt_software/esuprt_it_ops_datcentr_mgmt/high-computing-solution-resources_white-papers59_en-us.pdf.
[41] J. Guerreiro, et al., GPGPU power modeling for multi-domain voltage-frequency scaling, in: 2018 IEEE International Symposium on High Performance Computer Architecture, HPCA, IEEE, 2018, pp. 789–800.
[42] V. Adhinarayanan, et al., Online power estimation of graphics processing units, in: 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid, IEEE, 2016, pp. 245–254.
[43] J. Lim, et al., Power modeling for GPU architectures using McPAT, ACM Trans. Des. Autom. Electron. Syst. 19 (3) (2014) 1–24.
[44] S. Ghosh, et al., Statistical modeling of power/energy of scientific kernels on a multi-GPU system, in: 2013 International Green Computing Conference Proceedings, IEEE, 2013, pp. 1–6.
[45] J. Chen, et al., Statistical GPU power analysis using tree-based methods, in: 2011 International Green Computing Conference and Workshops, IEEE, 2011, pp. 1–6.
[46] X. Ma, et al., Statistical power consumption analysis and modeling for GPU-based computing, in: Proceeding of ACM SOSP Workshop on Power Aware Computing and Systems, Vol. 1, HotPower, 2009.
[47] S.M. Nabavinejad, et al., Coordinated batching and DVFS for DNN inference on GPU accelerators, IEEE Trans. Parallel Distrib. Syst. 33 (10) (2022) 2496–2508.
[48] J. Guerreiro, et al., DVFS-aware application classification to improve GPGPUs energy efficiency, Parallel Comput. 83 (2019) 93–117, http://dx.doi.org/10.1016/j.parco.2018.02.001.
[49] F. Xiao, A multiple-criteria decision-making method based on D numbers and belief entropy, Int. J. Fuzzy Syst. 21 (4) (2019) 1144–1153.
[50] G.-H. Tzeng, K.-Y. Shen, New Concepts and Trends of Hybrid Multiple Criteria Decision Making, CRC Press, 2017.
[51] T. Florindo, et al., Application of the multiple criteria decision-making (MCDM) approach in the identification of carbon footprint reduction actions in the Brazilian beef production chain, J. Clean. Prod. 196 (2018) 1379–1389.

Ghazanfar is a Ph.D. scholar at the Department of Computer Science, Texas Tech University, Lubbock, Texas, USA. His overall research goals are to develop new methodologies to make HPC systems more performant, power- and energy-efficient, automated, predictable, and responsive to evolving computing needs. Prior to his Ph.D. studies, Ghazanfar has been representing ZTE Corporation at various standards development organizations (SDOs) and has been an active researcher, contributor and editor in the development of Information and Communications Technologies (ICT) related specifications (e.g., cloud computing platform, service delivery platforms and communication service enablers) at International Telecommunications Union (ITU), Open Mobile Alliance (OMA) and Distributed Management Task Force (DMTF). He also represented ZTE as a Chair of DMTF Cloud Management Working Group (CMWG) and worked in the area of standardization of cloud computing technologies to promote ZTE cloud strategic interests in international SDOs. He attended approx. 60 face-to-face international meetings and delivered about 300 proposals and technical editor of several standards at DMTF (CIMI, OVF, DNS service management profile), OMA (Converged IP Messaging (CPM) enabler), and ITU-T (Y.2025, Y.2240, Y.2214, Q.3610, Q.3611). He received his M.Sc. degree in Computer Science in 2003 from Quad-e-Azam University (QAU), Islamabad (Pakistan). His M.Sc. thesis was on the design and implementation of an Internet Protocol-Private Branch eXchange (IP-PBX).

Mert Şide is a Ph.D. Student at Texas Tech University, under the supervision of Prof. Yong Chen. He earned a bachelor's degree in Computer Engineering and a bachelor's degree (double major) in Industrial Engineering, both from Istanbul Okan University. He has been involved in microarchitectural security projects on GPUs. He is currently participating in a research project that offers a scalable global address space extension for High-Performance Computing (HPC) environments. His research interests include computer architecture and HPCs.

Sridutt Bhalachandra is a staff member in the Advanced Technologies Group (ATG) [NERSC] at the Lawrence Berkeley National Laboratory. He received his Ph.D. from the Computer Science department at the University of North Carolina-Chapel Hill in 2018, where he was a research assistant at Renaissance Computing Institute (RENCI). Before joining Berkeley Lab, he was a postdoctoral appointee in the Mathematics and Computer Science Division at Argonne National Laboratory.

Nicholas J. Wright is the chief architect and the advanced technologies group lead at the National Energy Research Scientific Computing (NERSC) center. Most recently, he led the effort to optimize the architecture of the Perlmutter machine, the first NERSC platform designed to meet needs of both large scale simulation and data analysis from experimental facilities. His research interests are in performance analysis of HPC applications and architectures and he has published more than 40 papers in these areas. Nicholas has a Ph.D. from the University of Durham in computational Chemistry and has been with NERSC since 2009

Yong Chen a Professor in the Computer Science Department of the Texas Tech University (TTU) in Lubbock, Texas. He is the founding Director of the Data-Intensive Scalable Computing Laboratory (DISCL). He is also the Site Director of the Cloud and Autonomic Computing Center at TTU (CAC@TTU) sponsored by the National Science Foundation IUCRC (Industry-University Cooperative Research Centers) Program. His research focuses on data-intensive computing, high-performance computing, parallel and distributed computing, cloud computing, computer architectures, and systems software support for high-performance scientific computing/high-end enterprise computing.