HeShare: Energy-Aware and Efficient Multi-Task GPU Sharing in Heterogeneous GPU-based Computing Systems

Zhuolong Jiang, Zinuo Cai, Hongyu Zhao, Baoheng Zhang, Tianqi Wu, Yiming Qiang, Ruhui Ma†, *Member, IEEE*, Haibing Guan, Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—With the rapid growth of artificial intelligence and large-scale model computing, the demand for GPUs in data-centers continues to increase, especially for large-scale training and inference tasks. Heterogeneous multi-GPU systems, which integrate GPUs with varying types and computational capabilities, have become critical computing resources. This leads to two main challenges. First, due to the differences in GPU performance and power consumption, task scheduling involves a complex multi-objective optimization to balance energy efficiency and performance. More importantly, the lack of coordinated mechanisms for multi-task sharing and energy-efficient resource management across heterogeneous GPUs can result in GPU overload or underutilization, leading to wasted resources and potential system risks.**

To address these challenges, we propose HESHARE, an energy-aware and efficient heterogeneous GPU framework for datacenters. First, we design an energy-aware task scheduling strategy that optimizes task allocation across different GPUs to achieve a balance between energy consumption and performance. Second, we introduce a GPU sharing optimization mechanism that adaptively configures MPS and DVFS settings for each GPU, enhancing resource utilization, reducing overall energy consumption, and ensuring task performance. Compared to the state-of-the-art framework, we reduce average energy costs by 26% and improve job completion time by 31%, achieving a balance between energy efficiency and performance.

***Index Terms*—Heterogeneous GPU System, DVFS technology, GPU Sharing**

I. INTRODUCTION

In recent years, GPUs have become the core acceleration hardware for compute-intensive applications such as deep learning [5]–[7], scientific simulation [8], intelligent transportation [9], and image processing [10], [11], due to their powerful parallel computing capabilities and high throughput. To further improve computational performance and energy efficiency, heterogeneous GPU systems integrate high-performance GPUs to form diverse device combinations, enabling matching of performance and power consumption requirements for different tasks and demonstrating great potential in cloud and edge computing. However, heterogeneous GPUs exhibit significant differences in both computing capability and power consumption. Assigning tasks to high-performance GPUs can substantially reduce the execution time of individual tasks, but their high power consumption often leads to energy efficiency improvements that are not proportional to performance gains. In contrast, allocating tasks to low-power GPUs can reduce instantaneous power usage, but due to limited computational capabilities, the prolonged execution time may result in increased overall energy consumption. Furthermore, executing multiple tasks concurrently can improve resource utilization, but also introduces resource contention, which further exacerbates the trade-off between performance and energy efficiency. Therefore, how to effectively manage energy consumption and performance for multi-task execution in heterogeneous GPU systems has become an important problem that urgently needs to be addressed.

TABLE I
A COMPARISON OF HESHARE AND EXISTING SOLUTIONS THAT OPTIMIZE BOTH ENERGY CONSUMPTION AND PERFORMANCE HESHARE

Solution	GPU Frequency Adjustment	Multi-Task Parallelism	Heterogeneous
BatchDVFS [1]	✓	✗	✗
PWR [2]	✗	✗	✓
μ-Serve [3]	✓	✗	✗
Morak [4]	✓	✓	✗
HESHARE	✓	✓	✓

As shown in Table I, various methods have been proposed in recent years to optimize both energy consumption and performance. Among them, dynamic voltage and frequency scaling (DVFS) has been widely adopted to reduce GPU power consumption by dynamically adjusting the core voltage and frequency to balance power and performance. BatchDVFS [1] focuses on a single task on homogeneous GPUs and optimizes inference throughput and power consumption by dynamically adjusting the batch size and DVFS configuration. μ-Serve [3] time-shares the execution of different parts of

Zhuolong Jiang, Zinuo Cai, Hongyu Zhao, Ruhui Ma, Haibing Guan are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: {jzz19961996, kingczn1314, sjtu-zhy, ruhuima, hbguan}@sjtu.edu.cn).

Baoheng Zhang is with Aerospace System Engineering Shanghai, Shanghai 201109, China. (email: beowulfbhz@126.com).

Tianqi Wu and Yiming Qiang are with China Ship Scientific Research Center (e-mail: wutianqi@cssrc.com.cn,715436657@qq.com).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia (e-mail: rbuyya@unimelb.edu.au).

†This work is supported by Shanghai Key Laboratory of Scalable Computing and Systems, the Eighth Research Institute of China Aerospace Science and Technology Group Company, Ltd., under Grant USCAST2023-17 and Grant USCAST2023-21, and National Key Laboratory of Ship Structural Safety. Corresponding author is Ruhui Ma (ruhuima@sjtu.edu.cn).

multiple models on homogeneous GPUs and dynamically adjusts DVFS settings to ensure task performance. PWR [2] considers heterogeneous scenarios in datacenters, but it does not consider resource contention among multiple tasks and energy optimization based on DVFS. These methods either focus solely on single-task scheduling across heterogeneous GPUs or only support DVFS-based frequency adjustment on homogeneous GPUs, making them difficult to apply in datacenters. To improve GPU utilization, NVIDIA's Multi-Process Service (MPS)* allows multiple processes to share the same GPU and execute tasks concurrently by multiplexing the GPU context, thereby enabling parallel computation of tasks. Morak [4] combines MPS with DVFS to optimize energy consumption and improve GPU utilization simultaneously, but it only considers homogeneous GPUs. Therefore, implementing an efficient heterogeneous GPU system is necessary and faces the following two challenges:

- **Challenge 1: The complexity of multi-objective optimization**. Datacenters typically need to execute a large number of tasks to fully utilize GPUs. However, the significant differences in GPU computing capabilities and power characteristics, along with task diversity, make finding an optimal solution that meets performance requirements while minimizing energy consumption a typical NP-hard multi-objective optimization problem.
- **Challenge 2: Energy and performance configuration for heterogeneous GPUs**. The computing capabilities and energy consumption of heterogeneous GPUs vary significantly (see §II-B). Improper task allocation can cause GPU overload or system crashes, and may also result in low utilization leading to resource wastage(see §II-A). Moreover, parallel execution of multiple tasks further amplifies differences in energy consumption and performance, making it difficult to achieve an optimal balance without coordinated optimization(see §II-C).

To address these challenges, we propose HESHARE, an effective heterogeneous GPU architecture designed to reduce energy consumption while ensuring performance in heterogeneous GPU systems. HESHARE can efficiently solve the multi-objective problem based on task requirements and initially allocate tasks to appropriate GPUs to achieve a balance between performance and energy consumption. HESHARE utilizes DVFS and MPS to provide tasks with optimal computational resources and GPU frequencies, further optimizing energy consumption and improving GPU utilization.

Specifically, to address Challenge 1, we propose an energy-aware task scheduling strategy for heterogeneous GPUs. This strategy employs a multi-objective task scheduling algorithm to generate high-quality initialization schemes based on the energy consumption and performance distribution of tasks across different GPUs. During the task allocation process, each scheme undergoes local evaluation to select the corresponding optimal allocation. Subsequently, a priority-based mutation is applied to each scheme according to task characteristics, in order to avoid falling into local optima. Finally, this method produces a Pareto solution set that balances energy consumption and performance.

*https://docs.nvidia.com/deploy/mps/index.html

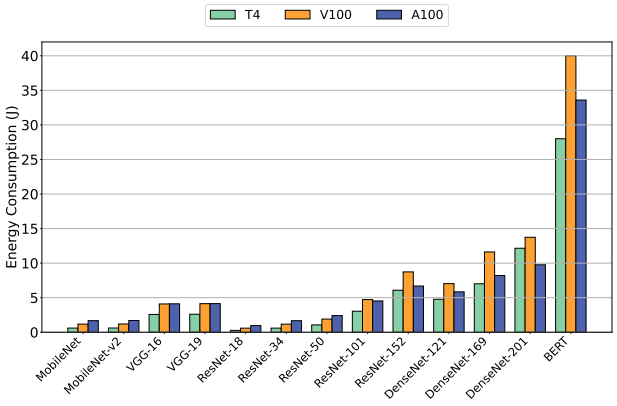


Fig. 1. Power consumption of different tasks on NVIDIA Tesla T4, NVIDIA Tesla V100, and NVIDIA A100.

To solve Challenge 2, we propose a GPU sharing optimization mechanism based on MPS and DVFS, which aims to further configure energy consumption and resource allocation for the generated Pareto solution set. This mechanism first generates the optimal task execution batches for each solution in the Pareto set. Then, it adaptively adjusts the computational resource configuration and GPU operating frequency for each task in each solution by leveraging MPS and DVFS. This enables fine-grained control of resource usage, reduces performance degradation caused by resource contention, and further improves GPU resource utilization while reducing energy consumption.

In summary, this paper makes the following contributions:

- We propose HESHARE, an energy-aware and efficient multi-GPU task scheduling framework for heterogeneous GPU systems, aiming to optimize energy efficiency and overall performance.
- We develop an efficient energy-aware task scheduling strategy that generates high-quality task allocation schemes, yielding Pareto-optimal solutions to balance energy consumption and performance.
- We design a GPU sharing optimization mechanism that leverages MPS and DVFS to adaptively adjust GPU resource configurations and operating frequencies, enabling fine-grained resource control and further improving GPU utilization while reducing energy consumption.
- Extensive evaluations on heterogeneous GPUs show that HESHARE achieves 26% lower average energy consumption and 31% lower job completion time than the state-of-the-art framework

II. Background And System Model

A. Task Power Consumption on Heterogeneous GPUs

Devices with varying computational capabilities exhibit different power consumption [5], [12]. Fig. 1 shows the power consumption of various models on NVIDIA T4, V100, and A100 GPUs. We measure power consumption using the NVIDIA Management Library (NVML) API. During task

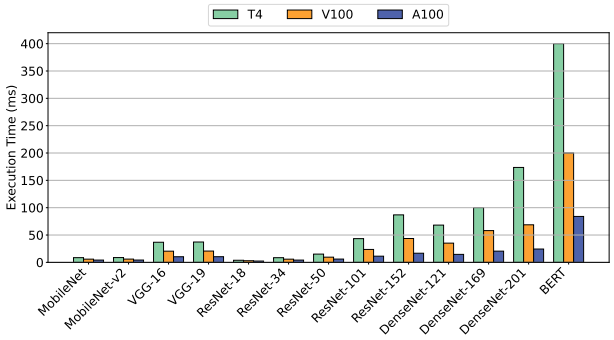


Fig. 2. Inference performance of different tasks on NVIDIA Tesla T4, NVIDIA Tesla V100, and NVIDIA A100.

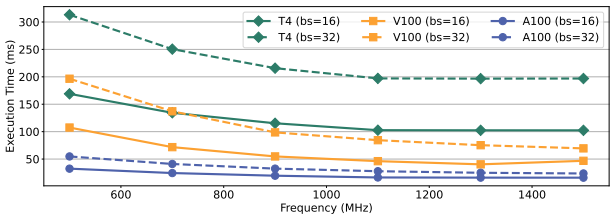


Fig. 3. Execution time of ResNet-152 with batch size (bs) 16 and 32 across different GPUs and their corresponding frequency settings.

execution, a service process samples power data using a 50μs time window. The instantaneous power sampled within each window is treated as the average power for that interval, based on which the energy consumption of the window is computed. By accumulating the energy consumption across all such windows throughout the execution period, we obtain the total energy consumption of the model inference.

For models with relatively low computational loads, such as the MobileNet series, the energy consumption differences across GPUs are small. However, for models with higher computational complexity, such as DenseNet-201 and BERT, the differences become significant. This indicates that as task workload increases, the impact of GPU computational capability on power consumption becomes more pronounced, making the selection of appropriate GPUs critical for energy efficiency optimization.

B. Differences in Task Execution Performance on Heterogeneous GPUs

GPUs with different computational capabilities have a significant impact on the execution performance of various tasks, making the selection of GPUs critical for task execution. Fig. 2 shows the execution times of multiple models on different GPUs. For models with relatively low computational loads, such as MobileNet, the execution time differences across GPUs are relatively small due to their low resource demands. However, as task computational complexity increases, the performance differences between GPUs become more pronounced. For example, for ResNet-50, the execution time on the NVIDIA T4 GPU is 2.5× higher than on the A100 GPU. For the more computationally intensive BERT, the execution time on the NVIDIA T4 GPU is over 5× higher than on the NVIDIA A100 GPU. This indicates that as task scale and computational demands increase, the impact of GPU computational capability on execution performance becomes more significant. Moreover, Fig. 3 further demonstrates that different GPU frequencies have varying impacts on inference performance across different GPUs.

C. Performance Variations of Collocation on Heterogeneous GPUs

In heterogeneous GPU systems, parallel task execution leads to varying degrees of performance degradation. Fig. 4 shows the comparison of average completion times for multiple pairs of tasks executed individually and in parallel on NVIDIA T4, V100, and A100 GPUs. The results show that the average completion time for all GPUs is higher during parallel execution compared to individual execution, indicating that parallelism causes resource contention and performance degradation. Additionally, the degree of degradation differs between GPUs. The NVIDIA T4 GPU, with the lowest computational capability, experiences the most severe performance degradation, with the largest increase in task execution time. The NVIDIA V100 GPU shows a moderate level of degradation. The NVIDIA A100 GPU, with the highest computational capability, has the least degradation, and its parallel execution times are close to those of individual executions.

III. System Design

We propose HeShare, an energy-aware and efficient multi-GPU scheduling framework designed to optimize the energy efficiency and overall computational performance of heterogeneous GPU systems. The system framework is shown in Fig. 5, which consists of multi-objective prediction, energy-aware task scheduling strategy, and GPU sharing optimization. The multi-objective prediction predicts task performance and GPU energy consumption based on task and GPU resource characteristics. The energy-aware task scheduling strategy generates task allocation schemes that balance performance and energy efficiency according to task characteristics and their energy and performance distribution across GPUs. The GPU sharing optimization further refines energy consumption and resource allocation by dynamically setting GPU frequencies and configuring optimal computational resources for each task to reduce energy consumption while maximizing performance.

When a set of tasks arrives ❶, HeShare first performs energy and performance modeling and predicts task performance based on resource characteristics ❷. It then generates task allocation schemes through energy-aware task scheduling, assigning tasks to appropriate heterogeneous GPUs to balance energy consumption and performance ❸. Each GPU forms an independent sub-task queue based on the allocation results. To further reduce energy consumption and improve task performance, HeShare employs a batch generation method to create optimal task execution combinations for the task queues of each heterogeneous GPU. Then configures computational resources and sets GPU frequencies for each task ❹.

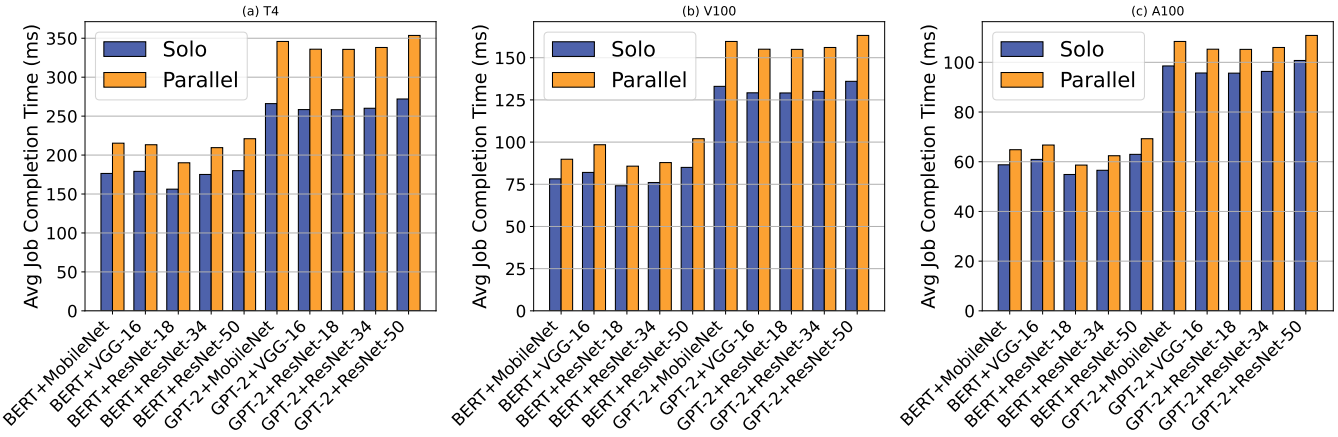


Fig. 4. The performance of multi-task simultaneously scheduled on NVIDIA Tesla T4, NVIDIA Tesla V100, and NVIDIA A100. Multi-task exhibits varying performance characteristics on different devices.

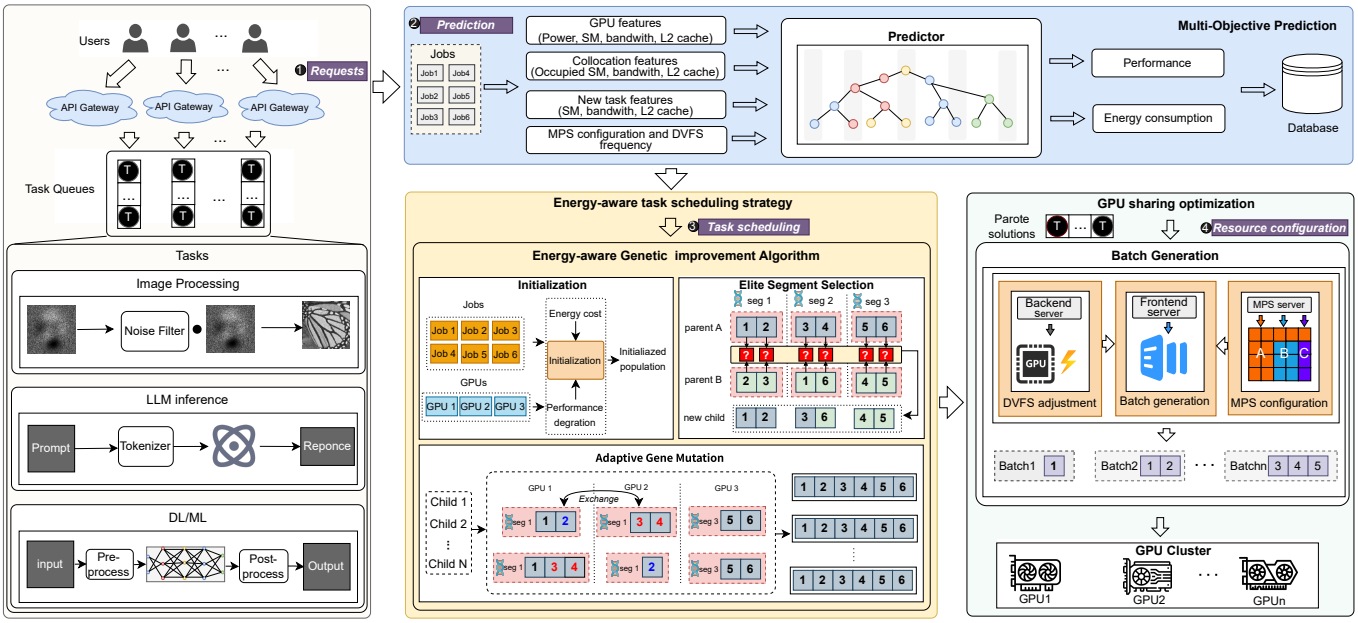


Fig. 5. System framework.

A. Multi-Objective Prediction

Due to the dynamic changes in GPU energy consumption with task loads and the fact that GPUs have DVFS mechanisms, accurate energy consumption prediction is necessary to reduce overall energy usage. In addition to improving the task parallelism performance of GPUs, efficiently utilizing MPS and accurately obtaining the performance of collocation under MPS are also key factors in optimizing GPU sharing performance.

We adopt Random Forest (RF)[†] as our prediction model because of its ability to effectively handle nonlinear feature relationships, making it suitable for complex multi-dimensional input and output tasks. We use the RF to simultaneously predict both energy consumption and performance. Specifically, the inputs include task features, single-task MPS configuration, frequency, current GPU resource utilization features, and current GPU energy consumption, and the outputs are the execution time of the new incoming task under sharing, and the updated overall GPU energy consumption.

For energy consumption prediction, we establish a mapping relationship between task features, GPU core frequency, current GPU resource utilization features, and current overall energy consumption and the overall GPU energy consumption after the arrival of a new task. These features describe the operational state of the GPU under the current load and the task requirements, reflecting the dynamic changes in power consumption under DVFS adjustments, as well as the interaction between task and system resource usage. This enables an effective mapping to predict the energy consumption level of the GPU in the time segment after the arrival of a new task.

For performance prediction, we use task features, MPS

[†]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

configuration, current GPU resource utilization features, and available GPU resources as inputs to implicitly model the mapping relationship between task resource characteristics and performance under resource contention. This enables effective estimation of task performance degradation and outputs the task throughput. All features can be found in Table II

Model Training. To train the RF, we collect performance and energy data by profiling all workloads in Table IV under different MPS configurations and GPU DVFS frequencies on T4, V100, and A100 GPUs. For each configuration, GPU power is sampled at 50μs intervals. Each sampled value is treated as the average power over the corresponding time window, and total energy consumption is calculated by summing the energy of all windows over the task execution period. We further augment the data by varying batch sizes and task combinations to improve model generalization. In total, 1000 training samples are collected, each representing a distinct combination of workload, MPS setting, frequency, and GPU type. We use 75% of the data for training and 25% for validation, optimizing the model with mean absolute error (MAE) as the loss function and the Adam optimizer for parameter updates.

B. Energy-Aware Task Scheduling

Energy-aware task scheduling includes the energy consumption model and the multi-objective task scheduling algorithm. The task scheduling model is used to describe the energy consumption in multi-GPU task scheduling. The multi-objective task scheduling algorithm optimizes energy consumption and performance by accelerating the optimization process and improving global search efficiency.

1) Energy Consumption Model: Suppose there are n tasks and k heterogeneous GPUs. Let the task set be $T = \{t_1, t_2, t_3, \ldots, t_n\}$, and the GPU set be $G = \{\text{GPU}_1, \text{GPU}_2, \ldots, \text{GPU}_k\}$.

A scheduling scheme S is defined as:

$$S = \{T_1, T_2, \ldots, T_K\} \tag{1}$$

where $T_k \subseteq T$ represents the set of tasks assigned to GPU k, satisfying:

$$\bigcup_{k=1}^{K} T_k = T, \quad T_i \cap T_j = \emptyset \quad (\forall i \neq j). \tag{2}$$

We further represent each task set T_k as:

$$T_k = \{t_i \mid i \in \mathcal{I}_k\}, \quad \mathcal{I}_k \subseteq \{1, 2, \ldots, N\}, \quad |\mathcal{I}_k| = m_k \tag{3}$$

where $\mathcal{I}_k$ denotes the index set of tasks assigned to GPU g_k, and $m_k = |T_k|$ is the number of tasks on GPU g_k.

The total power consumption of GPU m is the sum of static, dynamic, and idle power consumption, expressed as:

$$P^m = P^m_{\text{static}} + P^m_{\text{dynamic}} + P^m_{\text{idle}}. \tag{4}$$

To simplify the model and improve prediction efficiency, we model the total power consumption using the predictor:

$$P^m = f(\text{features}), \tag{5}$$

TABLE II
FEATURES USED TO TRAIN THE PREDICTOR

Name	Description
Task features	SM utilization, memory bandwidth utilization, L2 cache utilization, execution time.
MPS configuration	Assigned SM percentage for the task.
DVFS frequency	Configuration for GPU frequency.
Collocation features	Occupied SM utilization, memory bandwidth utilization, L2 cache utilization, energy consumption in the current GPU.
GPU features	GPU characteristics including SM count, memory bandwidth, L2 cache size, and basic power.

where features are the inputs to the predictor f and P^m is the total power consumption output by the predictor.

The total energy consumption of GPU m is calculated as:

$$E^m = P^m. \tag{6}$$

The total energy consumption of all GPUs is:

$$E_{\text{total}} = \sum_{m=1}^{k} E^m. \tag{7}$$

Moreover, we use the average completion time to model task performance. The completion time of the i-th task is:

$$T_i = W_i + E_i \tag{8}$$

where W_i and E_i represent the waiting time and execution time of the i-th task respectively.

The average completion time T_{avg} for all tasks is expressed as:

$$T_{\text{avg}} = \frac{1}{n} \sum_{i=1}^{n} (W_i + E_i). \tag{9}$$

The objective is to simultaneously minimize the total energy consumption and the average job completion time. This is formulated as a multi-objective optimization problem:

$$\min \{E_{\text{total}}, T_{\text{avg}}\}. \tag{10}$$

2) Multi-Objective Task Scheduling Algorithm: To solve the multi-objective optimization problem, we propose a multi-objective task scheduling algorithm based on the genetic algorithm to generate high-quality Pareto solutions. Compared to the traditional genetic algorithm, the adaptive optimized genetic algorithm includes the following three improvements: 1) multi-objective initialization; 2) multi-objective elitism selection; 3) priority mutation. Next, we introduce these in detail.

a) Multi-Objective Initialization: We propose a multi-objective initialization method to accelerate the convergence of the multi-objective task scheduling in terms of both energy consumption and execution time. Specifically, we first calculate the execution energy consumption and execution time of each task on different GPUs to obtain their respective distributions across all available GPUs. Then, for each task, we adopt a ranking-based trade-off strategy: we separately rank the GPUs based on energy consumption and execution time, compute the average rank of each GPU across the two rankings, and select the GPU with the smallest average rank as the initial allocation device for that task. If multiple

GPUs share the same average rank, one of them is randomly selected for task assignment. By initialization, the quality of the initial population is significantly improved, providing a diverse and high-quality starting point for subsequent elitist selection and mutation, thereby accelerating the convergence of the algorithm.

b) Multi-Objective Elitism Selection: To maximize the utilization of high-quality task allocation structures from parent individuals while considering both energy consumption and average job completion time, we propose a multi-objective elitist selection strategy to generate high-quality solutions. Specifically, we calculate each GPU subset's total energy consumption and average job completion time in both parent individuals. During the elitism selection process, we retain the optimal segments concerning both objectives and inherit these segments into the offspring, ensuring that the superior task configurations of both parents on different GPUs are directly preserved. This multi-objective inheritance strategy allows the offspring to achieve optimized performance across multiple objectives. In cases where segment inheritance conflicts occur, we compare the conflicting inheritance schemes and select the one that performs better in the targeted optimization objectives.

c) Priority Mutation: To avoid being trapped in local optima, we design a multi-objective mutation method that combines task migration and task recovery, consisting of two main steps: the task migration stage and the task recovery stage.

In the task migration stage, after the crossover operation generates offspring individuals, for each task, we calculate the target GPU with the minimum energy consumption and the target GPU with the minimum performance degradation based on its energy and degradation distributions across all available GPUs. The task is then migrated to these two target GPUs, respectively, generating two offspring with different optimization preferences, which enhances the diversity of the solution set.

In the task recovery stage, migrating tasks to the target GPU may lead to resource overload, making it unable to accommodate the incoming tasks. To ensure consistent GPU resource utilization before and after migration, we select one or more tasks from the target GPU whose resource usage is most similar to the incoming task as the tasks to be migrated out, and remove them from the target GPU set.

C. GPU Sharing Optimization Based on MPS and DVFS

To further optimize the Pareto solutions obtained in the first stage, we propose a GPU sharing optimization strategy. Specifically, the tasks assigned to each GPU in the solution are further partitioned into execution batches to explore optimal task execution combinations. Then, we leverage MPS and DVFS to perform optimal resource configuration for the tasks, aiming to optimize multi-objective performance.

1) Batch Model: Due to resource contention among tasks, excessive contention leads to performance degradation. Therefore, instead of executing all tasks simultaneously, we divide them into multiple batches for sequential execution to effectively reduce contention and improve overall performance and energy efficiency.

Assume there are N tasks, denoted as $T = \{t_1, t_2, ..., t_N\}$. Each task t_i has two features: execution time tp_i and memory requirement mem_i, where tp_i represents the execution time of task i on the current GPU, and mem_i represents its memory requirement.

We partition the N tasks into B batches:

$$Batch = \{B_1, B_2, ..., B_z\} \tag{11}$$

where each batch $B_k \subseteq T$ represents a set of tasks executed simultaneously on the GPU, satisfying:

$$\bigcup_{k=1}^{B} B_k = T, \quad B_i \cap B_j = \emptyset, \quad \forall i \neq j \tag{12}$$

The completion time of task t_i is defined as the sum of execution times of all preceding batches plus its own execution time within its batch:

$$JCT_i = \sum_{j=1}^{k-1} \tau_{B_j} + \tau_i, \quad t_i \in B_k \tag{13}$$

where τ_{B_j} denotes the execution time of batch j, typically taken as the maximum execution time among tasks in the batch.

In addition, the total energy consumption of the GPU, E_{total}, is defined as the sum of energy consumed during the execution of all batches:

$$E_{total} = \sum_{k=1}^{B} E_{B_k} \tag{14}$$

where E_{B_k} represents the energy consumption of batch B_k, which can be estimated by a predictor based on the task features within the batch and GPU configuration.

The average job completion time of all tasks is defined as:

$$T_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} JCT_i \tag{15}$$

In the Batch Generation stage, our optimization objective is, under a fixed task assignment scheme, to minimize the GPU's total energy consumption and average task completion time by adjusting MPS parallel configurations and GPU DVFS frequency settings. The mathematical formulation is:

$$\min_{MPS,DVFS} \left(E^{(2)}, T^{(2)} \right) \tag{16}$$

where $E^{(2)}$ and $T^{(2)}$ denote the total energy consumption and average task completion time of all batches after configuration optimization, respectively.

2) Resource Configuration Algorithm: As shown in Algorithm 1, we propose the resource configuration algorithm with MPS and DVFS to further enhance the quality of the Pareto solution set generated in the first stage. We take the Pareto solution set PS generated in the first stage as input, where each solution S contains the task-to-GPU assignment scheme. Next, we first determine the optimal batch grouping. Tasks with shorter execution times are given higher priority for scheduling. Whether a batch should be split depends on whether the division can reduce the overall average job

completion time. If splitting the batch results in a lower average job completion time, a new batch is created (lines 3-5). Then, we iterate over the tasks in each batch and select the MPS configuration that minimizes the average job completion time of each task (lines 6-10), thereby improving the overall execution efficiency of the batch. After the MPS configuration adjustment, we compute the updated energy consumption and average job completion time (line 11). Subsequently, we iterate over all possible GPU frequencies (line 12). For each frequency configuration, we calculate the total energy consumption and average job completion time across all batches (line 13). If the current configuration is non-dominated and improves energy consumption or performance compared to the current solution, it is added to the Pareto solution set as a new solution (lines 14-18). Then, any solutions dominated by this new solution are removed to maintain the non-dominated property of the set (line 19). Finally, the updated Pareto solution set PS' is output (line 23).

Algorithm 1: Resource Configuration Algorithm

Input : Pareto solution set PS, MPS configuration set $MPS\_configs$, GPU frequency set GPU_freqs
Output: Updated Pareto solution set PS'

1 Initialize $PS' \leftarrow PS$;
2 **foreach** *solution* S *in* PS **do**
3 Retrieve task-to-GPU assignment from S;
4 Prioritize tasks: tasks with shorter execution times assigned higher priority;
5 Group tasks into batches based on whether splitting reduces overall average job completion time;
6 **foreach** *batch* B **do**
7 **foreach** *task* t *in* B **do**
8 Select MPS configuration in $MPS\_configs$ that minimizes average job completion time of t;
9 **end**
10 **end**
11 Compute updated energy and average job completion time after MPS adjustment;
12 **foreach** *GPU freq* f *in* GPU_freqs **do**
13 Compute total energy and average job completion time under current MPS and f;
14 **if** *energy or avg job completion time improve over current* S **then**
15 Form new solution S_{new} with updated assignment, MPS, and GPU frequency config;
16 **if** S_{new} *is not dominated by any solution in* PS' **then**
17 Add S_{new} to PS';
18 **end**
19 Remove solutions in PS' dominated by S_{new};
20 **end**
21 **end**
22 **end**
23 **return** PS';

IV. METHODOLOGY

System Setup: We evaluate HESHARE on multiple hardware platforms using popular deep learning applications and Table IV summarizes the test applications used. As shown in Table III, our experimental platform consists of 4 NVIDIA Tesla T4 GPUs (70W power consumption, 2560 CUDA cores, 16GB GDDR6 memory), 2 NVIDIA Tesla V100 GPUs (250W power consumption, 5120 CUDA cores, 32GB HBM2 memory), and 2 NVIDIA A100 GPUs (400W power consumption, 6912 CUDA cores, 80GB HBM2e memory). These GPUs provide a heterogeneous computing environment for high-performance large-scale AI inference workloads.

We follow the workloads settings of MISO [21] and generate a job trace based on publicly available GPU trace Helios [22] to evaluate HESHARE on the real-world Helios trace. For the testbed evaluations, we create 60 inference jobs from Table IV.

Performance Metrics: This section introduces several classic performance metrics to assess the effectiveness of our work. We identify the following metrics.

1) **Average Job Completion Time (Average JCT):** JCT represents the average end-to-end completion time of a job, which includes the sum of the queue waiting time and the job execution duration. Lower JCT means shorter task completion time, indicating better performance.
2) **Energy Consumption:** Energy consumption represents the total energy consumed by the GPU during task execution.

Baselines: Since the proposed method is a multi-objective optimization algorithm, we evaluate its performance against the following five approaches: 1) Random denotes a non-optimized scheduling method that assigns tasks randomly; 2) Sched-only performs multi-objective task scheduling to balance energy consumption and performance without applying any resource configuration for performance and energy optimization; 3) Trade-off represents a trade-off method that incorporates MPS and DVFS-based resource configuration; 4) En-opt is a method that prioritizes the optimization of energy consumption, while JCT-opt prioritizes the optimization of JCT; 5) Morak [4] and μ-Serve [3] both implement heterogeneous GPU task scheduling based on random assignment strategies.

TABLE III
GPU HARDWARE CONFIGURATION.

GPU Platform	SM Count	CUDA Cores	Power (TDP, W)	Memory (GB)
NVIDIA Tesla T4	40	2560	70	16
NVIDIA Tesla V100	80	5120	250	32
NVIDIA A100	108	6912	400	80

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2025.3628924

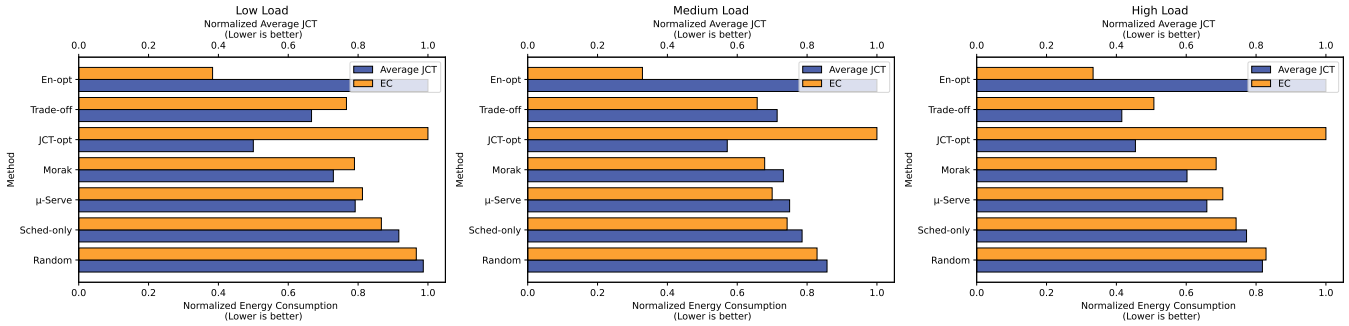


Fig. 6. Performance comparison.

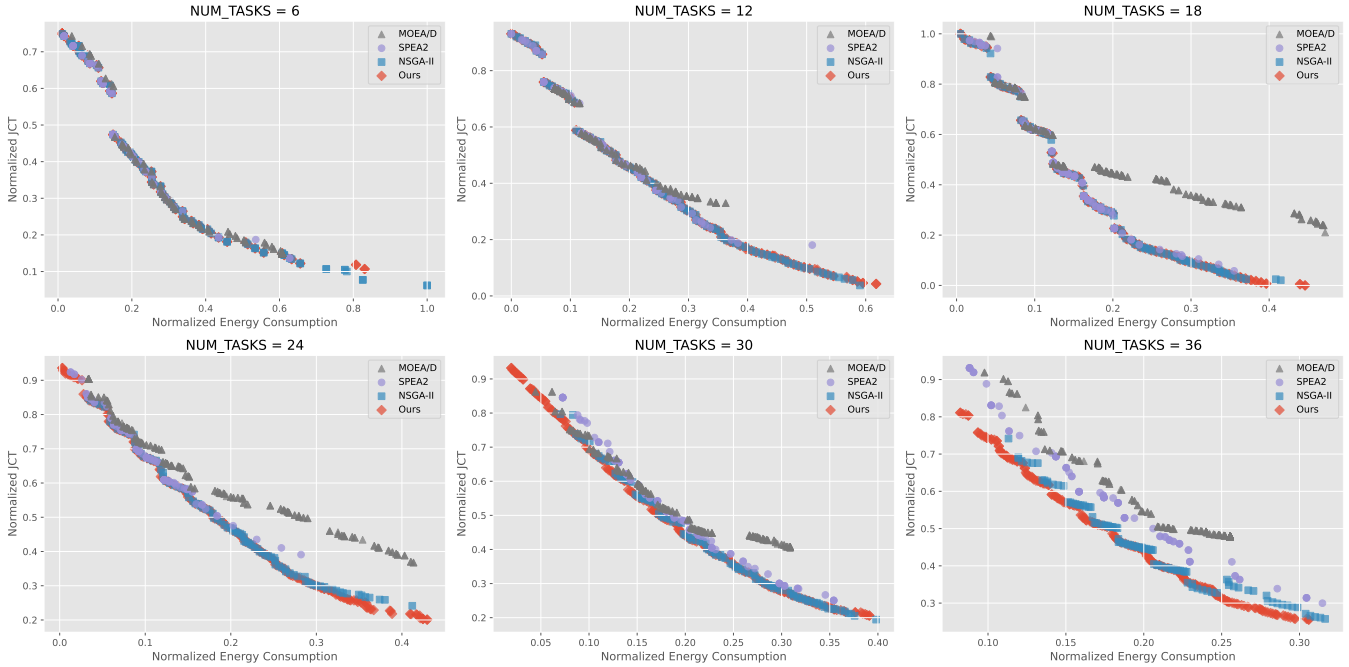


Fig. 7. PFs comparison of different schemes.

V. PERFORMANCE EVALUATION

A. Performance Comparision

Fig. 6 illustrates the performance comparison among five approaches in terms of average JCT and energy consumption. Trade-off, which combines MPS and DVFS for resource configuration, improves energy consumption by 26% and JCT by 31% compared to Morak and by 28% and 36% compared to μ-Serve. Morak integrates MPS and DVFS but does not support heterogeneous GPU scheduling and does not consider task execution priority, while μ-Serve does not support heterogeneous multi-task parallelism. Random assigns tasks without any optimization and therefore delivers the poorest performance and energy efficiency. En-opt focuses solely on reducing energy, achieving a 30% reduction compared to Random and a 25% reduction compared to JCT-opt, but sacrifices performance. JCT-opt minimizes JCT, achieving improvements of 48% and 20% compared to Random and En-opt, respectively, but at the cost of higher energy consumption. Sched-only performs multi-objective scheduling without resource configuration and therefore fails to achieve optimal results in either metric.

To evaluate the quality of the Pareto solutions, we conducted an in-depth comparison with the Pareto fronts (PF) obtained by three classical multi-objective algorithms: Strength Pareto Evolutionary Algorithm 2 (SPEA2) [23], Nondominated Sorting Genetic Algorithm II (NSGA-II) [24], and Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [25]. Based on existing method [26], we set the population size to 50, the mutation rate to 0.1, the selection strategy to binary tournament, and the number of generations to 500 for the above algorithms.

As shown in Fig. 7, SPEA2 suffers from poor convergence, leading to insufficient exploration of the solution space and failing to cover the complete Pareto front. NSGA-II performs well in certain regions but tends to fall into local optima, resulting in insufficient solution diversity and making it difficult to simultaneously meet performance and energy consumption requirements. MOEA/D optimizes by decomposing

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2025.3628924

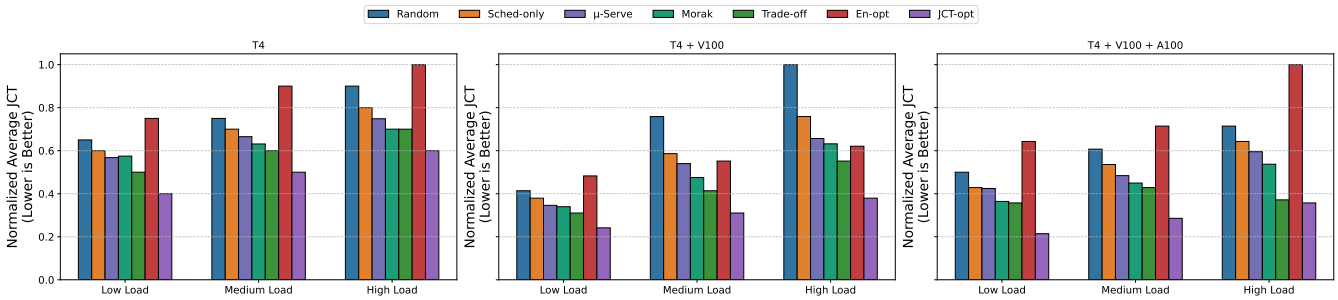


Fig. 8. JCT comparison of different hardware platforms.

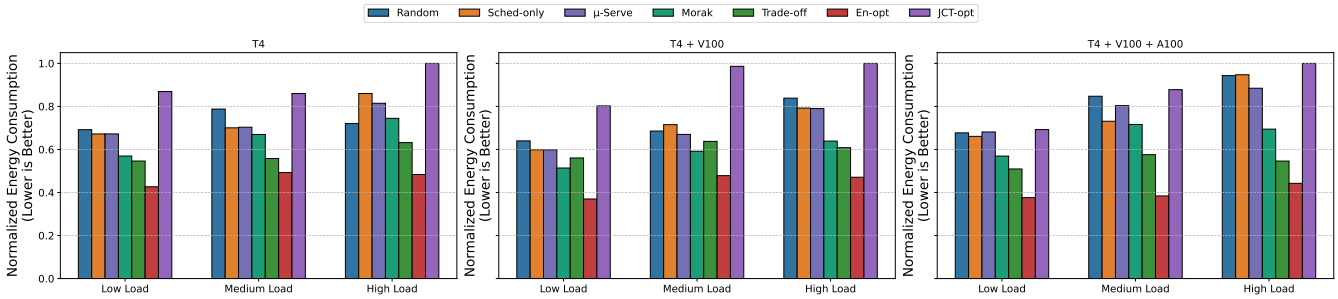


Fig. 9. Energy consumption comparison of different hardware platforms.

TABLE IV
WORKLOADS USED TO EVALUATE HESHARE

Name	Description & Source
ResNet-152 [13]	Image classification
ResNet-101 [13]	Image classification
ResNet-50 [13]	Image classification
ResNet-34 [13]	Image classification
ResNet-18 [13]	Image classification
DenseNet-121 [14]	Deep dense connection
DenseNet-161 [14]	Deep dense connection
DenseNet-201 [14]	Deep dense connection
VGG-19 [15]	Feature extraction
VGG-16 [15]	Feature extraction
MobileNet [16]	Mobile network
MobileNet V2 [17]	Mobile network
BERT Large [18]	NLP model, text classification and QA
Transformer [19]	Sequence-to-sequence model, translation
GPT-2 [20]	Large-scale language generation model

the problem, but its solutions lack distribution uniformity and tend to concentrate on specific subproblem regions, resulting in inadequate coverage of the global Pareto front. Our method improves the search starting point through high-quality initialization, prevents the loss of superior solutions via an optimal structure retention mechanism, enhances search diversity through priority-based mutation, and achieves Pareto-optimal solutions by finely configuring GPU resources with MPS and DVFS.

B. Evaluation of Different GPUs

As shown in Fig. 8, we evaluate different GPU combinations under varying loads: high-load scenarios include large models like BERT and GPT, medium-load scenarios include ResNet152 and DenseNet, and low-load scenarios involve lighter tasks. Trade-off achieves a 26% improvement in performance over Morak and a 35% improvement over μ-Serve. Compared to these two methods, Trade-off fully supports heterogeneous multi-task scheduling and resource optimization. Morak, due to its lack of heterogeneous GPU support, cannot effectively schedule tasks as the diversity of GPU types increases. μ-Serve, limited by its lack of heterogeneous and parallel task execution, fails to utilize the computing potential of different GPUs fully. In contrast, as GPU types increase, our Trade-off method can efficiently scale task scheduling across devices with varying computational capabilities, while providing optimal resource configurations through coordinated MPS and DVFS adjustments, ensuring both performance and energy efficiency.

JCT-opt minimizes task completion time by prioritizing GPUs with higher computing power, but causes excessive energy consumption. En-opt focuses solely on reducing energy consumption by assigning tasks to low-power GPUs, leading to degraded execution performance. Sched-only performs basic scheduling without resource configuration, and Random assigns tasks blindly, both resulting in suboptimal performance and energy results.

As shown in Fig. 9, Trade-off reduces energy consumption by 20% compared to Morak and 24% compared to μ-Serve. Compared to JCT-opt and En-opt, it achieves energy savings of 12% and 18% under low load, 10% and 15% under medium load, and 8% and 12% under high load. These results verify that Trade-off can achieve balanced optimization of

Fig. 10. Average JCT Comparison on Homogeneous GPUs.

Fig. 11. Energy Consumption Comparison on Homogeneous GPUs.

energy efficiency and task performance across varying load conditions.

C. Homogeneous GPU Performance Comparison

We further conduct a performance analysis of the proposed methods on homogeneous GPU platforms to verify their effectiveness. As shown in Fig. 10, in terms of average JCT, JCT-opt reduces average JCT on the T4 by 30% compared to Morak, and by 27.8% compared to μ-Serve as the number of tasks increases. Morak, although combining MPS and DVFS, fails to consider task execution priorities and cannot provide an optimal scheduling strategy. μ-Serve, relying solely on DVFS-based frequency adjustment for energy control, lacks the parallelism optimization offered by MPS, resulting in longer completion times. In contrast, JCT-opt fully leverages the coordinated configuration of MPS and DVFS to optimize performance. On the V100 and A100, Trade-off also outperforms Morak and μ-Serve in average JCT, achieving improvements of 26% and 24%, respectively. This improvement is attributed to its dynamic resource adjustment mechanism based on MPS and DVFS, which enhances parallelism while effectively avoiding resource contention.

In Fig. 11, En-opt consistently achieves the lowest energy consumption across all platforms. As the number of tasks increases, En-opt reduces energy consumption by 28% compared to Morak on the T4, and by 22% and 24% on the V100 and A100, respectively. This is due to En-opt employing coordinated optimization of MPS and DVFS to provide optimal energy control. Trade-off also achieves lower energy consumption than Morak and μ-Serve, reducing energy usage by approximately 21.3% compared to Morak. This result benefits from the joint optimization of energy consumption through MPS and DVFS in Trade-off, which effectively controls energy consumption while maintaining task performance.

TABLE V
PREDICTION ACCURACY OF DIFFERENT ML TECHNIQUES.

Model	Absolute Error	Relative Error
SVR	26.73%	29.12%
DT	24.51%	23.86%
LR	23.78%	21.54%
MLP	12.12%	5.13%
RF	3.10%	3.91%

D. Performance Prediction Accuracy

We compare five representative machine learning models: support vector regression (SVR)[‡], decision tree (DT)[§], linear regression (LR)[¶]and multilayer perception (MLP)[‖]. As shown in As shown in Table V, RF achieves the best prediction performance, with absolute error of 3.10% and relative error of 3.91%. The absolute error is approximately 4 $\times$ lower and the relative error is about 1.3 $\times$ lower than MLP, and both are significantly better than the other models whose errors all exceed 20%. In addition, RF also achieves better performance compared to DT and SVR. Because RF can effectively handle multi-dimensional and complex input features while maintaining strong generalization ability.

VI. RELATED WORK

This paper focuses on energy management, heterogeneous multi-GPU systems, resource management.

Energy Management. DVFS has been widely applied for power management and is commonly used to improve energy efficiency. BatchDVFS [1] combines DVFS with batch size

[‡]https://scikit-learn.org/stable/api/sklearn.svm.html
[§]https://scikit-learn.org/stable/modules/tree.html
[¶]https://scikit-learn.org/stable/api/sklearn.linear_model.html
[‖]https://scikit-learn.org/stable/api/sklearn.neural_network.html

adjustment to regulate GPU frequency and enhance throughput, but it does not consider multi-tasking or heterogeneous GPUs. Morak [4] integrates DVFS with MPS to improve the throughput of multi-model inference simultaneously, yet it only supports homogeneous GPUs. μ-Serve [3] leverages DVFS along with operator-level frequency adjustments to reduce energy consumption but lacks support for heterogeneous systems. Recently, other researchers [27], [28] have set adaptive DVFS frequencies at the DNN operator level, which is orthogonal to our optimization granularity, but both approaches only support homogeneous GPUs.

Heterogeneous Infrastructures. Existing research has thoroughly investigated the application of heterogeneous computing architectures, with a particular focus on the integration of GPUs. Cai *et al.* [29] propose a mechanism of two-layer GPU sharing to boost the effectiveness of GPU usage. Furthermore, the integration of Machine Learning techniques into resource allocation strategies for heterogeneous devices has gained traction. In parallel, recent studies have placed a significant emphasis on achieving energy efficiency in GPU-accelerated computing. Mittal *et al.* [30] offer comprehensive methods for analyzing and improving GPU energy efficiency, contributing valuable insights to the broader discourse on sustainable and optimized GPU computing practices. Unlike prior works, we focus on optimizing both energy consumption and performance through collocation-based resource scheduling in heterogeneous GPU systems.

GPU Sharing. Gandiva [31] proposes a cluster scheduler that enhances latency and efficiency through GPU time-slicing and job migration. Harmony [32] optimizes resource sharing by reducing job interference, improving performance in distributed ML workloads. Our task-sharing mechanism maximizes GPU utilization with task batching, reducing resource contention.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we propose an energy-aware scheduling framework designed to optimize both energy consumption and performance in heterogeneous GPU systems for datacenter, enabling optimal configuration of energy and performance tradeoffs. First, we desgin an energy-aware task scheduling strategy to generate a Pareto set that balances energy consumption and performance. Second, a GPU sharing optimization mechanism based on MPS and DVFS is introduced to adaptively configure GPU resources and frequencies for each task, improving utilization and reducing energy consumption. Experiments show that compared to the state-of-the-art framework, HESHARE achieves 26% lower average energy consumption and 31% lower job completion time, achieving a good balance between energy efficiency and performance.

In the future, we will focus on providing more fine-grained resource scheduling at the kernel level for heterogeneous GPU systems.

REFERENCES

[1] S. M. Nabavinejad, S. Reda, and M. Ebrahimi, "Coordinated batching and dvfs for dnn inference on gpu accelerators," *IEEE transactions on parallel and distributed systems*, vol. 33, no. 10, pp. 2496–2508, 2022.

[2] F. Lettich, E. Carlini, F. M. Nardini, R. Perego, and S. Trani, "Power- and fragmentation-aware online scheduling for gpu datacenters," in *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. IEEE, 2025, pp. 43–52.

[3] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. Kalbarczyk, T. Başar, and R. K. Iyer, "Power-aware deep learning model serving with {μ-Serve}," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 75–93.

[4] D. Liu, Z. Ma, A. Zhang, and K. Zheng, "Efficient gpu resource management under latency and power constraints for deep learning inference," in *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*. IEEE, 2023, pp. 548–556.

[5] S. Chaudhary, R. Ramjee, M. Sivathanu, N. Kwatra, and S. Viswanatha, "Balancing efficiency and fairness in heterogeneous gpu clusters for deep learning," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

[6] R. Zhang, T. Zhang, Z. Cai, D. Li, and R. Ma, "Memorianova: Optimizing memory-aware model inference for edge computing," *ACM Transactions on Architecture and Code Optimization*.

[7] P. Jiang, H. Wang, Z. Cai, L. Gao, W. Zhang, R. Ma, and X. Zhou, "Slob: Suboptimal load balancing scheduling in local heterogeneous gpu clusters for large language model inference," *IEEE Transactions on Computational Social Systems*, 2024.

[8] J. Ames, D. F. Puleri, P. Balogh, J. Gounley, E. W. Draeger, and A. Randles, "Multi-gpu immersed boundary method hemodynamics simulations," *Journal of computational science*, vol. 44, p. 101153, 2020.

[9] Z. Cai, Z. Chen, Z. Liu, Q. Xie, R. Ma, and H. Guan, "Ridic: Real-time intelligent transportation system with dispersed computing," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[10] W. Auccahuasi, P. Castro, E. Flores, F. Sernaque, A. Garzon, and E. Oré, "Processing of fused optical satellite images through parallel processing techniques in multi gpu," *Procedia Computer Science*, vol. 167, pp. 2545–2553, 2020.

[11] K. Sun, T.-H. Tran, J. Guhathakurta, and S. Simon, "Fl-misr: fast large-scale multi-image super-resolution for computed tomography based on multi-gpu acceleration," *Journal of Real-Time Image Processing*, vol. 19, no. 2, pp. 331–344, 2022.

[12] L. Braun, S. Nikas, C. Song, V. Heuveline, and H. Fröning, "A simple model for portable and fast prediction of execution time and power consumption of gpu kernels," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 18, no. 1, pp. 1–25, 2020.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[18] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of Association for Computational Linguistics: Human Language Technologies*, 2019.

[19] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

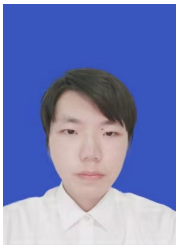
[20] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[21] B. Li, T. Patel, S. Samsi, V. Gadepally, and D. Tiwari, "Miso: exploiting multi-instance gpu capability on multi-tenant gpu clusters," in *Proceedings of Symposium on Cloud Computing*, 2022.

[22] Q. Hu, P. Sun, S. Yan, Y. Wen, and T. Zhang, "Characterization and prediction of deep learning workloads in large-scale gpu datacenters," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021.

[23] S. Jiang and S. Yang, "A strength pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 329–346, 2017.

[24] G. Zhang, Z. Su, M. Li, F. Yue, J. Jiang, and X. Yao, "Constraint handling in nsga-ii for solving optimal testing resource allocation problems," *IEEE Transactions on Reliability*, vol. 66, no. 4, pp. 1193–1212, 2017.
[25] R. Tanabe and H. Ishibuchi, "A framework to handle multimodal multi-objective optimization in decomposition-based evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 720–734, 2019.
[26] Q. Jiang, H. Wang, Q. Kong, Y. Zhang, and B. Chen, "On-orbit remote sensing image processing complex task scheduling model based on heterogeneous multiprocessor," *IEEE Transactions on Geoscience and Remote Sensing*, 2023.
[27] J. Geng, Z. Zhu, W. Liu, X. Zhou, and B. Li, "Powerlens: An adaptive dvfs framework for optimizing energy efficiency in deep neural networks," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.
[28] Z. Wang, Y. Zhang, F. Wei, B. Wang, Y. Liu, Z. Hu, J. Zhang, X. Xu, J. He, X. Wang *et al.*, "Using analytical performance/power model and fine-grained dvfs to enhance ai accelerator energy efficiency," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, 2025, pp. 1118–1132.
[29] Z. Cai, Z. Chen, R. Ma, and H. Guan, "Smss: Stateful model serving in metaverse with serverless computing and gpu sharing," *IEEE Journal on Selected Areas in Communications*, 2023.
[30] S. Mittal and J. S. Vetter, "A survey of methods for analyzing and improving gpu energy efficiency," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1–23, 2014.
[31] W. Xiao, R. Bhardwaj, R. Ramjee, M. Sivathanu, N. Kwatra, Z. Han, P. Patel, X. Peng, H. Zhao, Q. Zhang *et al.*, "Gandiva: Introspective cluster scheduling for deep learning," 2018.
[32] Y. Bao, Y. Peng, and C. Wu, "Deep learning-based job placement in distributed machine learning clusters," 2019.



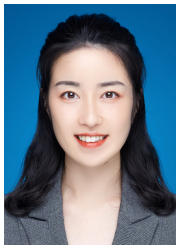
Zhuolong Jiang is currently a doctoral student in Information Security at Shanghai Jiao Tong University, China. His research interests are focused on serverless GPU optimization.

Zinuo Cai is currently a graduate student in Computer Science at Shanghai Jiao Tong University, China. He obtained the bachelor's degree in Software Engineering at Shanghai Jiao Tong University. His research interests are focused on resource schedule and system security in cloud computing.

Hongyu Zhao is currently an undergraduate student in Information Security at Shanghai Jiao Tong University, China. His research interests are focused on resource schedule in cloud computing.

Baoheng Zhang is currently a senior engineer at the Aerospace System Engineering Shanghai, who was awarded a master's degree in Aircraft Design from Northwestern Polytechnical University. He mainly engages in information processing and motion simulation.

Tianqi Wu is an engineer in China Ship Scientific Research Center with research focus on data science in ship and marine engineering.



Yiming Qiang is an engineer in China Ship Scientific Research Center with research focus on artificial intelligence and data analysis in ship and marine engineering.

Ruhui Ma is currently an associate professor in the Department of Computer Science and Engineering at Shanghai Jiao Tong University. He received his Ph.D. degree in computer science from Shanghai Jiao Tong University. His research interests include cloud computing systems, AI systems, and machine learning.

Haibing Guan (Member, IEEE) received the PhD degree in computer science from Tongji University, Shanghai, China, in 1999. He is currently a full professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, China. He is a member of ACM and CCF. His research interests include computer architecture, distrusted computing, virtualization, cloud computing, and system security

Rajkumar Buyya (Fellow, IEEE) is a Redmond Barry distinguished professor and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He has authored more than 625 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide.