IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 July 2026, accepted 14 August 2026, date of publication 18 August 2026, date of current version 31 August 2026.

Digital Object Identifier 10.1109/ACCESS.2026.3725140

RESEARCH ARTICLE

DVFS-Aware Energy-Efficient Implementation of DNNs on Heterogeneous Hybrid Parallel Computers

HAMIDREZA KHALEGHZADEH[1], ATEFEH KHAZAEI[1], AND ALEXEY LASTOVETSKY[2], (Member, IEEE)
[1]School of Computing, University of Portsmouth, PO1 3HE Portsmouth, U.K.
[2]School of Computer Science, University College Dublin, Dublin D04 V1W8, Ireland

Corresponding author: Alexey Lastovetsky (alexey.lastovetsky@ucd.ie)

This work was supported in part by the Taighde Éireann–Research Ireland under Grant 20/FFP-P/8683; and in part by the Faculty of Technology, University of Portsmouth, under the Development Projects Funding (DPF) Scheme 2024/25.

ABSTRACT High-performance heterogeneous computing has emerged as a key resource for training increasingly Deep Neural Networks (DNNs). Achieving optimal performance and energy efficiency on such platforms, however, requires careful resource optimisation at both application and platform levels. While state-of-the-art DNN optimisation approaches typically focus on only one of these dimensions or are designed for homogeneous platforms, this paper considers both application- and platform-level configurations for the bi-objective optimisation of DNN training, to minimise execution time and energy consumption on single-node heterogeneous computing platforms. After formulating the bi-objective optimisation problem, we propose a methodology to model the performance and dynamic energy profiles of DNN applications, expressed as functions of the number of batches and Dynamic Voltage and Frequency Scaling (DVFS) states. These models are then exploited by a model-based algorithm that employs batch distribution (application-level) and DVFS states (platform-level) as decision variables to solve the bi-objective optimisation problem. The algorithm produces execution time-energy consumption trade-off solutions in the form of Pareto fronts. The proposed approach is evaluated on two heterogeneous hybrid servers comprising multicore CPUs and many-core GPUs. Experimental results demonstrate that solutions incorporating integrated application- and platform-level configurations outperform those that consider only a single dimension, in terms of both performance and energy efficiency. Furthermore, comparing our approach with TensorFlow data parallelism and homogeneous distribution strategies demonstrates its superiority. A comprehensive analysis of the results reveals that DNNs achieve greater improvements in energy efficiency than in performance.

INDEX TERMS Bi-objective optimisation, DNNs, DVFS, energy optimisation, energy sustainability, parallel computing, performance optimisation, workload distribution.

I. INTRODUCTION

Owing to their effectiveness in handling complex modelling, Deep Neural Networks (DNNs) are increasingly utilised in a wide range of applications, including image recognition, medical science, textual document processing, and autonomous vehicles. DNN training is computationally intensive as it emulates the complex processes of the human brain, and is time- and energy-consuming.

Modern machine learning frameworks, such as TensorFlow and Keras, leverage parallel processing on High-Performance Computing (HPC) platforms to address the increasing demand for faster modelling speed. However, the rapid growth of DNN applications has raised significant concerns about the tremendous energy (electricity) consumption of deep learning and its future environmental impact [1], [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhengmao Li.

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

In recent years, modern HPC platforms have become highly heterogeneous, via the tight integration of multicore CPUs with accelerators such as GPUs, FPGAs, and Intel Xeon Phis. These heterogeneous architectures are among the most promising solutions for meeting the growing demand for faster and more energy-efficient systems, capable of handling the increasing demand for DNN training. However, their effectiveness depends on optimising data-parallel DNN applications and efficient hardware utilisation to significantly enhance performance and reduce energy consumption.

DNN applications can be optimised for performance and energy at two different levels: `Application-level` and `Platform-level`. Application-level optimisation focuses only on adjusting decision variables within the application to achieve performance and energy efficiency. In contrast, platform-level optimisation targets the execution environment by tuning system configurations.

State-of-the-art techniques for the DNN application optimisation for performance and/or energy typically focus on a single domain: either optimising the application by modifying its configuration while keeping the platform fixed or adjusting platform-level decision variables independently of the application. However, existing solutions do not employ both domains simultaneously to deliver holistic bi-objective optimisation strategies.

In addition, while state-of-the-art deep learning packages support training on different types of processing devices, they are primarily designed for homogeneous platforms, assuming identical devices within a cluster, and do not consider in-depth platform-level parameters—such as DVFS states—when partitioning training workloads. For example, TensorFlow provides data parallelism on single-node clusters through two primary distribution strategies: `MirroredStrategy` for GPUs and `TPUStrategy` for TPUs. Under these strategies, training workloads are distributed *evenly* across devices. Applying such homogeneous, platform-oblivious strategies to heterogeneous systems leads to suboptimal training execution, resulting in higher energy consumption, increased carbon emissions, and reduced efficiency.

To address this gap, the main objective of this paper is to integrate heterogeneity-aware workload distribution, as an application-level configuration, and DVFS state, as a platform-level configuration, to optimise the training of fully connected multi-layer perceptron (MLP) DNNs on heterogeneous hybrid servers. The optimisation simultaneously targets two objectives: minimising execution time and reducing energy consumption. Unlike existing methods, our approach jointly optimises application-level and platform-level configurations to generate full performance–energy trade-off solutions.

The proposed approach first models the performance and dynamic energy consumption of DNN training as functions of two input variables: the number of batches and DVFS states. An optimisation algorithm then utilises these models to obtain full performance–energy trade-off solutions in the form of Pareto fronts, with both batch distribution and DVFS states as decision variables. We experimentally demonstrate that considering both application-level and platform-level parameters in DNN training optimisation yields superior solutions in both performance and energy efficiency compared with approaches limited to application-level configurations, with energy savings exceeding performance improvements.

To better illustrate the problem, we present a case study that practically demonstrates the impact of application-level (batch distribution) and platform-level (DVFS state) configurations on the performance and energy consumption of DNN training. In this case study, a fully connected MLP DNN—whose implementation and topology are detailed in Sections IV and VI-B—is trained on the MNIST dataset [3], which contains 60,000 handwritten digit images. We use a batch size of 300 for training, resulting in a total workload of 200 batches.

Figure 1 presents the execution time and dynamic energy consumption of the parallel DNN application for one epoch, executed under three different application and/or computational platform configurations. This case study is conducted on a heterogeneous system comprising an Intel Platinum multicore CPU and an Nvidia A40 GPU.

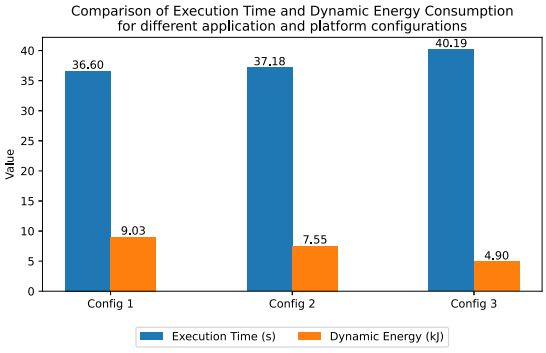


FIGURE 1. Comparison of execution time and dynamic energy consumption for three configurations differing in batch distribution and DVFS states.

The first configuration allocates 15 of the 200 batches to the CPU operating at a DVFS state of 2.2 GHz, with the remaining batches assigned to the GPU whose SM cores operate at a DVFS state of 1,740 MHz. This configuration is denoted as $\{(15, 2.2), (185, 1740)\}$. The second configuration, $\{(15, 2.2), (185, 720)\}$, maintains the same batch distribution as the first but reduces the GPU SM frequency from 1,740 MHz to 720 MHz. This change results in a 1.6% increase in execution time alongside an approximate 16.4% reduction in dynamic energy consumption. The final configuration, $\{(0, -), (200, 720)\}$, allocates the entire workload to the GPU operating at 720 MHz. Compared to the first configuration, this one yields a 9.8% longer execution time but achieves a 45.7% reduction in dynamic energy consumption.

From this case study, it can be concluded that both application-level and platform-level configurations have a significant impact on the two conflicting objectives of execution time and energy consumption in data-parallel DNN training on heterogeneous hybrid platforms. In addition, transitioning from single-decision-variable optimisation—focusing solely on either batch distribution or DVFS state—to dual-decision-variable optimisation, which considers both heterogeneous batch distribution and DVFS states simultaneously, offers greater potential for performance and energy sustainability.

Thus, the main contributions of this work are as follows:

- The formulation of a bi-objective optimisation problem aimed at minimising both execution time and energy consumption during the parallel training of fully connected multilayer perceptron DNNs on single-node heterogeneous servers, achieved through the optimal combination of application- and platform-level configurations.
- A modelling approach for representing performance and dynamic energy consumption of DNN training as functions of two input variables: the number of batches and DVFS states.
- A model-based algorithm that utilises the modelled performance and energy functions to solve the bi-objective DNN optimisation problem, incorporating both application-level and platform-level decision variables—batch distribution and DVFS states—to compute globally optimal Pareto fronts for execution time and dynamic energy consumption.
- An experimental evaluation of the proposed optimisation approach on two heterogeneous hybrid servers, comprising various Intel multicore CPUs and different types and numbers of Nvidia GPUs.
- An in-depth analysis of the experimental results to demonstrate that DNN training is more responsive to energy optimisation than to performance optimisation.

The paper is organised as follows. Section II reviews related work. The bi-objective optimisation problem, considering both application and platform configurations, is formulated in Section III. Section IV presents the design and implementation of a parallel hybrid DNN application for heterogeneous HPC platforms, followed by our model-based solution in Section V. Our methodology for modelling and constructing the performance and dynamic energy consumption profiles of DNN applications, along with experimental results on various heterogeneous platforms, is presented in Section VI. Finally, we conclude the paper in Section VII.

II. RELATED WORKS

This section presents a review of state-of-the-art research on bi-objective optimisation of performance and energy consumption in HPC platforms. The reviewed methods are classified into two primary categories: (i) Application-level approaches, and (ii) Platform-level approaches.

We further categorise the reviewed algorithms according to the strategies they employ for bi-objective optimisation and for handling constraints.

A. APPLICATION-LEVEL OPTIMIZATION

Application-level methods focus on modifying the application itself to optimise performance and energy efficiency. These approaches typically rely on models that predict an application's performance and energy consumption.

Subramaniam and Feng [4] propose a multi-variable regression algorithm to obtain partial trade-off solutions between performance and energy consumption. Their approach is based on four input parameters: problem size, block size, the number of process rows, and the number of processor columns in the process grid.

Song et al. [5] maximise HPC performance under strict power constraints by identifying efficient ways to scale processor count, CPU power/frequency, workload size, and network bandwidth.

Analytical models are proposed in [6] to predict the performance and energy consumption of applications based on workload partitioning between CPU and GPU. The proposed methodology returns energy-efficient configurations with minimal performance loss. Similarly, Chakrabarti et al. [7] propose a bi-objective optimisation algorithm based on workload distribution on heterogeneous platforms. Performance is formulated as a linear function of workload size, while energy consumption is predicted using historical data tables.

Tarplee et al. [8] formulate the bi-objective problem between energy consumption and makespan in heterogeneous computing systems and employ linear programming techniques to determine Pareto-optimal task allocations.

Gabaldon et al. [9] introduce a genetic algorithm aimed at balancing execution time and energy consumption in parallel applications. The primary decision variable is the scheduling or mapping of tasks. Energy consumption is represented by a mathematical model that accounts for both idle and active energy use, as well as the corresponding execution times for computing and idle states.

Manumachu and Lastovetsky [10] propose algorithms to address the bi-objective optimisation problem for performance and energy consumption on homogeneous clusters of multicore CPUs. These algorithms take as input real-life execution time and energy consumption CPU profiles as discreet functions of workload size, employing workload distribution as the sole decision variable.

Khaleghzadeh et al. [11], [12] developed an algorithm to solve discrete bi-objective optimisation problems on heterogeneous platforms, targeting both performance and dynamic energy, as well as execution time and total energy. The algorithm takes as input the discrete performance and dynamic energy profiles of each processor as functions of workload size and outputs the complete set of Pareto-optimal solutions. The key decision variable is workload distribution. Subsequently, they introduced another workload-distribution

algorithm to address a continuous bi-objective optimisation problem on heterogeneous HPC platforms, where the performance functions are continuous and strictly increasing, and the energy functions increase linearly with the increase of workload size [13]. Later, they designed and implemented a hybrid DNN application and optimised its performance and energy consumption, again treating workload distribution as the sole decision variable [14].

B. PLATFORM-LEVEL OPTIMIZATION

Platform-level methods reconfigure system parameters to achieve optimal performance and energy consumption.

Lim et al. [15] propose an adaptive DVFS technique for reducing energy consumption during communication phases, exploiting idle or low-utilisation periods to save energy.

Bi-objective optimisation methods proposed in [16] and [17] utilise optimal DVFS state selection and dynamic voltage/frequency adjustments as the decision variables to minimise energy consumption while respecting performance constraints.

Rountree et al. [18] present a runtime system for bounding the energy consumption of MPI applications. Their method dynamically adjusts CPU frequency and voltage through DVFS, where execution remains within a specified power budget while minimising performance loss.

Just-in-time Dynamic Voltage Scaling techniques are proposed in [19] and [20] to reduce energy consumption in MPI programs during slack periods (idle or underutilised intervals).

A genetic algorithm is presented in [21] addressing the performance and energy consumption bi-objective optimisation problem in cloud infrastructures. It utilises task computation cost and communication cost between tasks as input, while the processor supply voltage is the sole decision variable.

A four-objective optimisation algorithm is presented in [22] that considers performance, economic cost, energy consumption, and reliability for optimising scientific workflows in heterogeneous computing platforms. The proposed algorithm considers the task assignment as a decision variable.

In [23], a heuristic-based approach is proposed to optimise energy efficiency and Quality of Service (QoS) in data centres. The set of decision variables includes clock frequencies and the number of virtual machines (VMs).

Kessaci et al. [24] propose a genetic algorithm aimed at minimising energy consumption and CO_2 emissions while maximising the profit generated by a cloud computing infrastructure. The decision variable in this approach is the application arrival rate.

Power–performance trade-offs on power-constrained HPC systems are examined in [25] and [26]. The studies develop quantitative models and dynamic tuning methodologies that optimise execution time while adhering to power limits by adjusting CPU frequency and job scheduling.

Multi-objective genetic algorithms are implemented in [27] to address the bi-objective optimisation of performance and energy in clusters and cloud environments. The decision variable in this approach is the DVFS state. Rossi et al. [28] propose a performance-aware, energy-efficient framework for cloud data centres. The approach dynamically allocates virtual machines and adjusts resource utilisation to minimise energy consumption while maintaining service performance.

In [29] and [30], energy consumption in large-scale systems is minimised by selectively shutting down idle resources in web servers. The proposed approaches utilise load distribution combined with multi-constraint shutdown policies to achieve energy savings without compromising performance.

Jiang et al. [31] present BytePS, a unified architecture that improves the CPU-GPU communication during GPU-based DNN training via exploiting spare CPU and network bandwidth resources in clusters.

Panda et al. [32] propose a Reinforcement Learning-based algorithm for managing processor DVFS levels to achieve energy-efficient computing. The approach aims to minimise energy consumption while maintaining acceptable performance levels.

Zidar et al. [33] utilise DVFS to reduce energy consumption in embedded systems. They demonstrate that adjusting the supply voltage and clock frequency can effectively balance the trade-off between energy efficiency and performance.

C. CLASSIFICATION BY OPTIMIZATION STRATEGY AND CONSTRAINT HANDLING

The bi-objective optimisation methods can be classified into three categories. The first category includes approaches that optimise performance within a given power cap (or energy budget) or minimise energy consumption under a specified execution time constraint [4], [5], [17], [18], [25], [26], [28], [29], [30]. These methods generate either a single performance–energy trade-off solution or a partial Pareto front by enforcing the power or time constraint and then selecting the optimal configuration based on a user-defined objective.

The second category addresses bi-objective optimisation of performance and energy without imposing execution time or energy constraints, aiming to construct the full Pareto front [6], [7], [8], [9], [10], [11], [13], [24], [27], [34].

The third category includes methods that optimise application execution by leveraging DVFS states during slack phases, while keeping the application configuration fixed [15], [16], [19], [20], [33].

III. APPLICATION- AND PLATFORM-LEVEL DNN BI-OBJECTIVE OPTIMIZATION

In Section I, we experimentally studied the significant impact of application-level (batch distribution) and platform-level (processor DVFS states) configurations on both

execution time and dynamic energy consumption during DNN training. The results also revealed that the two optimisation objectives are conflicting, such that improving one typically degrades the other. This observation highlights the need for a novel bi-objective optimisation method that simultaneously tunes both application-level and platform-level configurations to optimise DNN training.

This section formulates the optimisation problem. Consider a DNN training workload using a dataset of $S \in \mathbb{N}^+$ samples with a batch size $b \in \mathbb{N}^+$. The total number of batches for a single-epoch DNN training execution is then given by $B = \frac{S}{b}$. Suppose the training process is executed on a platform with $p \in \mathbb{N}^+$ heterogeneous processors. Each processor $P_i \in \{P_0, P_1, \ldots, P_{p-1}\}$ supports a set of $j \in \mathbb{N}^+$ distinct DVFS states, denoted by $C_i = \{c_0, c_1, \ldots, c_{j-1}\}$. Let $\mathcal{C} = \{C_0, C_1, \ldots, C_{p-1}\}$ denote the set of all DVFS state sets across the p processors.

We define the bi-objective optimisation problem as a minimisation problem with two decision vectors (variables). The first decision vector, representing the application-level configuration, is $X = \{x_0, x_1, \ldots, x_{p-1}\}$, which determines the distribution of the B batches among the p processors such that $B = \sum_{i=0}^{p-1} x_i$, where $x_i \in \mathbb{N}$ denotes the number of batches assigned to processor P_i. The second decision vector, representing the platform-level configuration, is $\Theta = \{\theta_0, \theta_1, \ldots, \theta_{p-1}\}$, where each $\theta_i \in C_i$ denotes the DVFS state set for processor P_i.

There are two sets of platform-specific functions, known as execution time and energy functions, defined below:

$$F(X, \Theta) = \{f_0(x_0, \theta_0), f_1(x_1, \theta_1), \ldots, f_{p-1}(x_{p-1}, \theta_{p-1})\},$$
and
$$G(X, \Theta) = \{g_0(x_0, \theta_0), g_1(x_1, \theta_1), \ldots, g_{p-1}(x_{p-1}, \theta_{p-1})\} \quad (1)$$

where $f_i(x, \theta) \in \mathbb{R}$ and $g_i(x, \theta) \in \mathbb{R}$ denote the execution time and dynamic energy consumption, respectively, of processing x batches of size b on processor P_i operating at DVFS state θ. Each function models the DNN application characteristics using two input variables: the number of batches, which is application-related, and the DVFS state, which determines the platform-level characteristics.

Equation 2 formulates the bi-objective optimisation problem for DNN training, considering both execution time and dynamic energy consumption objectives. The parallel execution time of the training task is defined as $T(X, \Theta) = \max_{i=0}^{p-1} f_i(x_i, \theta_i)$, while the dynamic energy consumption is expressed as $E(X, \Theta) = \sum_{i=0}^{p-1} g_i(x_i, \theta_i)$.

$$\underset{X,\Theta}{\text{minimise}} \quad \left\{ \max_{i=0}^{p-1} f_i(x_i, \theta_i), \sum_{i=0}^{p-1} g_i(x_i, \theta_i) \right\}$$
$$\text{s.t.} \quad X = \{x_i\}_{i=0}^{p-1}, \ \Theta = \{\theta_i\}_{i=0}^{p-1}, \ B = \sum_{i=0}^{p-1} x_i. \quad (2)$$

Due to the conflicting nature of the two objectives in Equation 2, there is no single pair of workload distribution (numbers of batches) and platform configuration (DVFS states) that minimises both simultaneously. In Section V, we introduce our algorithm to solve this problem by obtaining trade-off solutions for execution time and energy consumption in the form of Pareto fronts.

As the computation process is similar across training epochs, the optimisation problem is formulated for a single epoch. This does not reduce the generality of the proposed method nor its ability to optimise the entire training process over an arbitrary number of epochs.

IV. HYBRID DNN IMPLEMENTATION

This section presents a parallel and portable implementation of fully connected MLP DNN applications, designed for training on heterogeneous hybrid servers. We utilise this methodology to implement the DNN applications used in our experimental evaluation. The implementation employs the Gradient Descent (GD) iterative algorithm, widely utilised for training DNN weights. To enable simultaneous parallel training across different types of computing devices, it is developed using a hybrid programming model featuring two levels of parallelism: (i) kernel-level and (ii) application-level.

At the kernel-level parallelism, we implement two kernels—`Training_Kernel_CPU` and `Training_Kernel_ACC`—to enable parallel GD training on multicore CPUs and manycore accelerators. These kernels leverage multi-threaded and directive-based programming models, including OpenMP [35] and OpenACC [36]. A comprehensive discussion of the kernel design is presented in [14] and the supplementary material.

Algorithm 1 illustrates the application-level parallelism. It employs Pthreads [37] to create CPU threads that simultaneously invoke the designed parallel kernels (`Training_Kernel_CPU` and `Training_Kernel_ACC`) across multiple devices on a heterogeneous platform, consisting of p devices: one multicore CPU and $p-1$ accelerators. This implementation integrates all training kernels, including those for the CPU and accelerators, into a unified hybrid parallel DNN application, enabling training computations to use all computing devices of the heterogeneous platform in parallel.

Algorithm 1 invokes the `distribute_batches_dvfs` function in line 4. It determines the batch distribution and the DVFS settings for all the computing devices involved. The function returns three outputs:

- Batch distribution vector (X): It decides how to divide the entire set of training batches among the available p devices, as the first decision vector.
- DVFS setting vector (Θ): It determines the DVFS state for each device, as the second decision vector.
- Created batches (*batches*): It creates the output vector *batches*, where `batches[i]` is a subset

Algorithm 1 Hybrid DNN Training Application

```
1  Function Hybrid_DNN_Training(input_data,
   batch_size, p, F, G, weights, biases);
   Input :
     • input_data: A 1D array of training labelled samples.
     • batch_size: The number of samples in each batch.
     • p: The number of devices, including 1 CPU and p-1
       accelerators, for parallel run.
     • F(x,Θ): Execution time profiles, as defined in Equation 1.
     • G(x,Θ): Dynamic energy profiles, as defined in Equation 1.
   Output:
     • weights: A 1D array of 2D matrices, where the i-th matrix
       (0 ≤ i < num_layers) contains the trained network's weights
       for the i-th layer.
     • biases: A 2D array, where the i-th column (0 ≤ i < num_layers)
       contains the trained network's biases for the i-th layer.
2  begin
       // Initialise weights and biases matrices
3      init_weights_biases(weights, biases);
       // Create batches, distribute them between
          CPU and accelerators,
       // and determine processors' DVFS states
4      (X, Θ, batches) = distribute_batches_dvfs (input_data,
       batch_size, p, F, G);
       // Set the operational frequency of each
          device
5      set_devices_frequencies(p, Θ);
       // Loop for epochs
6      for e = 0; e < num_epochs; e++ do
           // Create local copies of global weights
              and biases arrays
7          weights_local[0] = weights; biases_local[0] = biases;
           // Execute CPU kernel on a multicore CPU
8          pthread_create(thread_id[0],
           Training_Kernel_CPU(batches[0], X[0], batch_size,
           weights_local[0], biases_local[0]));
           // Execute accelerator kernels on
              p − 1 devices
9          for d = 1; d < p; d++ do
               // Create local copies of global
                  weights and biases arrays
10             weights_local[d] = weights; biases_local[d] = biases;
11             pthread_create(thread_id[d],
               Training_Kernel_ACC(batches[d], X[d], batch_size,
               weights_local[d], biases_local[d]));
12         end
           // Wait for all p threads to finish
13         for d = 0; d < p; d++ do
14             pthread_join(thread_id[d]);
15         end
           // Average all local weights and biases
              to obtain global weights and biases
16         weights = average_local_weights(weights_local);
17         biases = average_local_biases(biases_local);
           // Calculate error using averaged
              weights and biases
18         calculate_error(...);
19     end
20     return (weights, biases);
21 end
```

of `input_data` containing the batches allocated to the i-th device for training.

In Section V, we will explain how the `distribute_batches_dvfs` function determines application-level and platform-level configurations using performance and energy functions of the DNN application.

After setting the operating frequencies of all p devices according to Θ (line 5), the training kernels are invoked to process their allocated workloads in parallel.

The main reasons for using Pthreads instead of OpenMP at this level of parallelism are to overcome certain compiler limitations, such as those of the Nvidia HPC C (NVC) compiler, which struggles with the compilation of nested parallelism in OpenMP code, and to provide better control over the mapping of CPU threads to cores compared with OpenMP.

V. HEPOPT_MERGE: A BI-OBJECTIVE OPTIMIZATION ALGORITHM WITH DUAL DECISION VECTORS

As discussed in Section III, the formulated bi-objective optimisation problem yields a set of trade-off solutions, referred to as Pareto-optimal solutions, between execution time and dynamic energy consumption. Each solution has a specific execution time and energy consumption, determined by two decision vectors: the batch distribution (X) and the platform configuration (Θ). Solving this problem requires an algorithm capable of incorporating both application-level and platform-level decision variables to compute globally optimal Pareto fronts for execution time and dynamic energy consumption.

In [11], an application-level bi-objective optimisation algorithm called `HEPOPTA` (Heterogeneous Energy Performance Optimisation Algorithm) was introduced. This algorithm optimises data-parallel applications by considering workload distribution as the only decision variable. It takes as input the execution time and dynamic energy profiles of processors, where each profile is modelled as a discrete function of workload size. This optimiser does not make any specific assumptions about the shape of its input profiles, and therefore it is applicable to both linear and non-linear profiles. Using dynamic programming and branch-and-bound techniques, HEPOPTA efficiently computes discrete Pareto-optimal solutions for a given workload size with polynomial time complexity. Each solution defines a specific workload distribution across all processors.

We propose `HEPOPTA_Merge`, an extended version of the original HEPOPTA algorithm, designed to solve the bi-objective optimisation problem described in Section III. Similar to HEPOPTA, the proposed method leverages the arbitrary-shaped execution time and dynamic energy profiles of DNN applications to compute Pareto-optimal solutions, while jointly considering the two decision variables: batch distribution and DVFS states.

The implementation of HEPOPTA_Merge is presented in Algorithm 2.

Algorithm 2 iterates over all possible DVFS configurations in $C_0 \times C_1 \times \cdots \times C_{p-1}$ for the p processors, and for each configuration it invokes the HEPOPTA algorithm to compute the corresponding Pareto front (lines 4–7), which consists of all Pareto-optimal solutions specific to that configuration Θ.

Algorithm 2 Implementation of `HEPOPTA_Merge`

```
1  Function HEPOPTA_Merge(B, p, F, G, C);
   Input :
     • B: Total number of batches.
     • p: Number of processors.
     • F: Execution time profiles, as defined in Equation 1.
     • G: Dynamic energy profiles, as defined in Equation 1.
     • C: Set C_0 × C_1 × ··· × C_{p−1} of DVFS configurations for the p processors, where C_i is a set of all DVFS states of the i-th processor.
   Output:
     • PF_global: Global Pareto-optimal set of solutions (Pareto front).
2  begin
       // Initialise a superset of all candidate solutions for inclusion in PF_global
3      PF_superset ← ∅;
       // Iterate over all DVFS configurations to construct the superset of the global
       // Pareto front
4      foreach Θ ∈ C_0 × C_1 × ··· × C_{p−1} do
           // Run HEPOPTA to find all Pareto-optimal solutions for configuration Θ
5          PF_Θ ← HEPOPTA(B, p, F(·, Θ), G(·, Θ));
           // Add the found solutions to the superset
6          PF_superset ← PF_superset ∪ PF_Θ;
7      end
       // Sort solutions in the superset by execution time
8      PF↑_superset ← SortByETime(PF_superset);
       // Filter sorted solutions to retain only non-dominated solutions
9      PF_global ← FilterNonDominated(PF↑_superset);
       // Return the global Pareto front
10     return PF_global;
11 end
```

In this step, HEPOPTA is passed $F(\cdot, \Theta)$ and $G(\cdot, \Theta)$, which are the subsets of the execution time and energy functions F and G, restricted to the specific DVFS configuration Θ. The set $PF_{\texttt{superset}}$ accumulates all resulting Pareto-optimal solutions for different DVFS configurations, including dominant and non-dominant ones. Each solution is represented as a vector $\langle \texttt{eTime}, \texttt{DEng}, X, \Theta \rangle$, where `eTime` and `DEng` denote the execution time and dynamic energy consumption of DNN training using batch distribution X in the platform configuration Θ. To construct the global Pareto front ($\texttt{PF}_{\texttt{global}}$), the solutions in $PF_{\texttt{superset}}$ are first sorted by execution time (line 8), and then scanned to remove dominated entries (line 9). As explained in the supplementary material, HEPOPTA_Merge is scalable and exhibits polynomial time complexity.

Notably, the function `distribute_batches_dvfs` in Algorithm 1 employs the proposed HEPOPTA_Merge to compute the full Pareto front. From this set of Pareto-optimal solutions, one point — characterised by a specific batch distribution and corresponding DVFS states — is selected to configure and execute the DNN training. The selection is made according to the user's preferences regarding performance and energy efficiency.

VI. DNN APPLICATION MODELING AND EXPERIMENTAL EVALUATION

This section evaluates the efficiency of our bi-objective optimisation approach, which jointly optimises application- and platform-level configurations, on two heterogeneous platforms, HCLServer02 and HCLServer03. The first server consists of an Intel Xeon Gold 6152 CPU connected to an NVIDIA P100 PCIe GPU, and the second features an Intel Xeon Platinum 8362 CPU paired with two NVIDIA A40 GPUs. The specifications of these platforms are provided in the supplemental material. We compare the optimality of DNN training achieved using the proposed optimisation algorithm with dual-decision variable against standard Linux frequency governors, which consider workload distribution as the sole decision variable for optimisation. The proposed approach is also compared with TensorFlow's data parallelism distribution strategy.

We use the MNIST dataset, which contains 60,000 grayscale images of handwritten digits, each of size 28 × 28 pixels. The implemented neural network comprises 784 input neurons (representing pixel values), connected to five fully connected hidden layers of sizes {2500, 2000, 1500, 1000, 500}. The output layer contains one neuron. According to [38], this architecture is one of the best-performing topologies for MNIST handwritten digit recognition. This choice is motivated by the fact that MNIST represents a broad class of structured classification problems characterised by sparse, spatially localised patterns, and provides a prototypical dataset–model setting that captures key properties of many neural network-based classification tasks.

In the rest of this section, we describe our methodology for modelling the DNN application characteristics using execution time functions (F) and dynamic energy functions (G) on heterogeneous hybrid servers. The constructed functions are then provided to our model-based HEPOPTA_Merge algorithm to obtain Pareto-optimal solutions for the DNN application on the two heterogeneous servers. We will also statistically analyse Pareto fronts obtained under different batch distribution and DVFS configurations to evaluate their effectiveness in enhancing performance or reducing energy consumption.

A. CONSTRUCTING APPLICATION- AND PLATFORM-AWARE FUNCTIONS FOR PERFORMANCE AND DYNAMIC ENERGY

The HEPOPTA_Merge algorithm requires execution time (performance) and dynamic energy profiles of the DNN application, where each profile is expressed as a function of two variables: the number of batches (application-level configuration) and the DVFS state (platform-level configuration). To construct these functions for a given batch size, we execute the DNN application on HCLServer02 and HCLServer03 for varying numbers of batches and different frequency governors or DVFS configurations.

There are three issues to be addressed in constructing the functions: (i) resource sharing in heterogeneous platforms, (ii) configuring the DVFS states of processors, and (iii) measuring execution time and dynamic energy. These are discussed in the following sections.

1) RESOURCE SHARING IN HETEROGENEOUS PLATFORMS

In hybrid servers such as HCLServer02 and HCLServer03, computational training kernels share various hardware resources, including CPU cores, memory hierarchies, and PCIe links, when executed concurrently. To minimise resource contention and improve the reproducibility of the measured functions, we partition the hybrid platform into *abstract processors* such that inter-processor resource sharing is negligible. By configuring the number of threads and applying thread-binding techniques, each kernel is explicitly mapped to its designated abstract processor for execution.

For example, HCLServer03 comprises three abstract processors: `CPU_Platinum`, `GPU0_a40` and `GPU1_a40`. The `CPU_Platinum` abstract processor consists of 30 out of the 32 physical cores of the Intel Xeon Platinum processor. Each A40-based abstract processor (`GPU0_a40` and `GPU1_a40`) comprises one host CPU core from the remaining two cores, the corresponding Nvidia A40 computational cores, and the PCIe communication link connecting the GPU to its host CPU core. To execute DNN training using the HCLServer03 abstract processors, we implement a three-kernel variant of the `Hybrid_DNN_Training()` function presented in Algorithm 1 (Section IV). The algorithm spawns three threads using Pthreads, where each thread hosts one of the three loosely coupled abstract processors. The first host thread invokes the `Training_Kernel_CPU()` function, which internally creates a team of 30 OpenMP threads with the host thread as the master thread. Ensuring one-to-one mapping, we bind these 30 threads to 30 CPU cores, which collectively implement the `CPU_Platinum` abstract processor. The remaining two host threads are bound to the two remaining CPU cores. Each of these two threads invokes an instance of the `Training_Kernel_ACC()` function, which internally creates and executes GPU threads on the cores of the hosted GPU. In this way, the `GPU0_a40` and `GPU1_a40` abstract processors that execute GPU kernels are implemented. All CPU threads—both Pthreads and OpenMP threads— are bound to their corresponding physical cores in a way that ensures minimal resource sharing between the abstract processors. In addition, the reduced abstract processor core count has been applied to the CPU abstract processors even when no workload is allocated to the GPUs.

2) CONFIGURING THE DVFS STATES

To incorporate DVFS states into the profiles, we need to measure the DNN application characteristics for different DVFS states supported by each abstract processor.

A DVFS state of a processor is defined as a set of voltage/frequency pairs, representing the operating voltage and frequency of the independently powered components of the processor. The standard and reliable practice for modifying a DVFS state is to adjust the frequency, since direct voltage manipulation may compromise system stability. Furthermore, application developers typically have access only to the frequency settings of DVFS states. Once a frequency is specified, the corresponding voltage is automatically set by the hardware, which maintains a fixed internal mapping between operating voltages and frequencies. Our methodology for configuring DVFS states of CPU and GPU cores—using the *on-demand* and *userspace* governors—is detailed in the supplemental.

3) EXECUTION TIME AND DYNAMIC ENERGY MEASUREMENT

Precise measurement of the execution time and dynamic energy consumption of each loosely coupled training kernel (i.e., each abstract processor) is essential for constructing accurate profiles.

The execution time of each kernel is measured by recording the current time at the start and end of its execution, and then computing the difference between these two timestamps to obtain the kernel's elapsed time. This method ensures that the measured duration reflects both computation and PCIe communication time. To incorporate the impact of minor resource sharing between kernels (abstract processors), all kernels in the hybrid training application are executed *simultaneously* when measuring the execution time for each data point.

To measure the energy consumption of applications, three mainstream approaches generally exist: on-chip power sensors, software-based energy predictive models, and physical measurement using external power meters. As the first two approaches are typically less accurate than physical measurement with external power meters [39], we adopt the latter to create the dynamic energy profiles of our hybrid DNN training application. This approach relies on the additivity principle, which states that the total energy consumption of a hybrid application is equal to the sum of the energy consumed by each of its comprising kernels when executed separately.

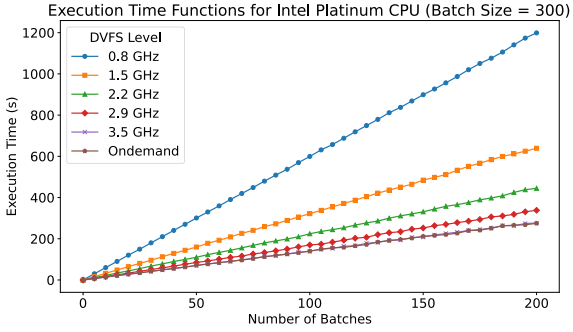
Therefore, to construct the dynamic energy profile of each kernel in our hybrid DNN application executed on a given heterogeneous platform, we execute each kernel *separately* and measure its total energy consumption, including both computation and PCIe communication, using an external power meter connected to the platform. The dynamic energy consumption is then obtained by subtracting the server's base (static) energy from the total measured energy. The base energy is calculated as the product of the platform's idle power consumption (in watts) and the kernel's execution time. We consider only dynamic energy, as it reflects the energy directly consumed by the application, whereas static energy is incurred regardless of whether any application is running. Further details on the scalability and accuracy of this approach can be found in [39].

4) EXECUTION TIME AND DYNAMIC ENERGY FUNCTIONS

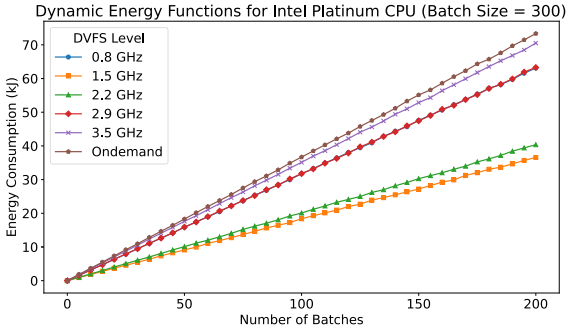
Figures 2–5 present the execution time and dynamic energy functions of the DNN application's kernels for different numbers of batches of size 300 on HCLServer02 and HCLServer03, with a step size of 5. For each data point in these functions, measurements are repeated until the sample means fall within a 95% confidence interval.

For each computing device, we select various userspace DVFS states ranging from the lowest to the highest frequency supported by that device. The highest frequencies for the Intel Platinum and Gold CPUs are 3.6 GHz and 2.1 GHz, respectively; however, we avoid fixing the frequency at the highest DVFS state to prevent power or thermal throttling, which could trigger hardware intervention that would automatically reduce the operating frequency below the specified maximum. In contrast to Intel CPUs, no dynamic frequency adjustment is observed on the Nvidia GPUs when their SM cores are fixed to the highest supported frequencies of 1,740 MHz for A40 and 1,328 MHz for P100, respectively. The A40 GPU supports two memory frequencies, 7,251 MHz and 405 MHz, of which only the maximum is considered. The sole supported main memory frequency for the Nvidia P100 PCIe is 715 MHz.

We also include the execution time and dynamic energy consumption of the application when executed under the on-demand Linux frequency scaling governor.

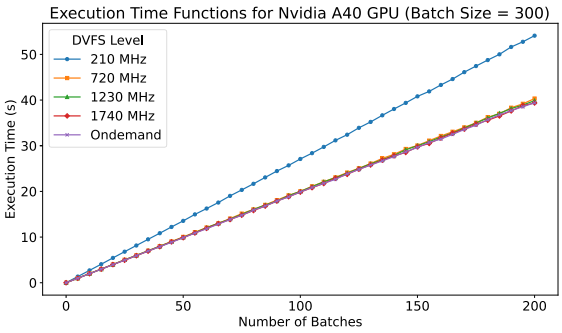


(a) Execution Time Functions

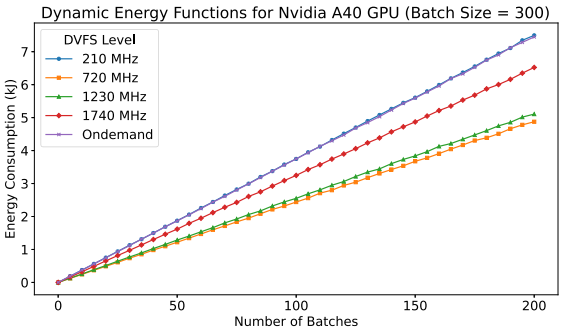


(b) Dynamic Energy Functions

FIGURE 2. Functions for intel platinum CPU at different DVFS states for batch size of 300.

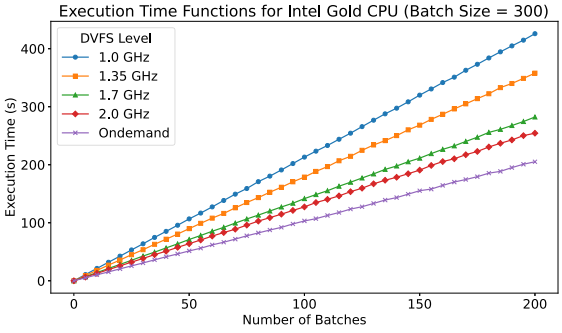


(a) Execution Time Functions

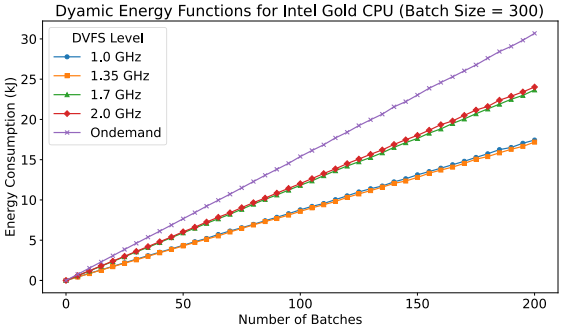


(b) Dynamic Energy Functions

FIGURE 3. Functions for NVidia A40 GPU at different DVFS states for batch size of 300.

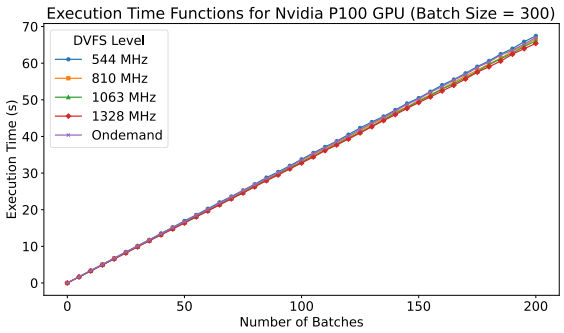


(a) Execution Time Functions

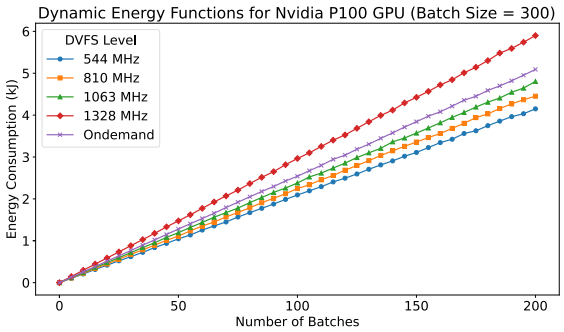


(b) Dynamic Energy Functions

FIGURE 4. Functions for intel gold CPU at different DVFS states for batch size of 300.



(a) Execution Time Functions



(b) Dynamic Energy Functions

FIGURE 5. Functions for Nvidia P100 PCIe CPU at different DVFS states for batch size of 300.

Considering the presented performance and dynamic energy profiles, the on-demand configuration exhibits execution speeds comparable to the highest DVFS setting considered on HCLServer03 for both the Intel Xeon Platinum CPU and the Nvidia A40 GPU; however, it has higher dynamic energy consumption than the highest DVFS setting. In addition, except for the 210 MHz configuration, the remaining DVFS states of the A40 GPU exhibit comparable execution times but a wide range of dynamic energy consumption across different frequency settings. For the Intel Xeon Gold CPU, the on-demand configuration is significantly faster than all other DVFS configurations but has substantially higher dynamic energy consumption. This behaviour can be attributed to the ability of the on-demand Linux governor to dynamically adjust processor frequencies based on real-time workload demands, thereby reducing delays at synchronisation points. In addition, the on-demand governor can utilise the maximum available frequency when required—something deliberately avoided in our fixed DVFS settings to avoid potential throttling issues. Moreover, similar to the Nvidia A40 GPU, the Nvidia P100 GPU exhibits comparable execution times for all considered DVFS states but shows a wide variation in dynamic energy consumption across different frequency settings.

We constructed the DNN performance and dynamic energy profiles for different batch sizes and in all cases observed the following trends:

- the functions exhibit an approximately linear increasing trend with the number of batches
- the execution time and dynamic energy consumption change almost proportionally with the batch size; for example, doubling the batch size approximately doubles both execution time and energy consumption.

This indicates a near-linear relationship between the number of batches and batch sizes with the two measured objectives—execution time and dynamic energy. The reason for these observations is the uniform computational workload across all batches.

Based on this observation, we explain how to construct the complete profiles of the DNN application via fast estimates. Consider a given reference batch size b_{ref} and its *physically measured* execution time (t_{ref}) and energy consumption (e_{ref}). To estimate the execution time ($t_b(n)$) and energy consumption ($e_b(n)$) for processing n batches of an arbitrary batch size b, the reference values can be linearly scaled with the number of batches (n) and the ratio $\frac{b}{b_{ref}}$, as shown in Equation 3.

$$t_b(n) = n \times \frac{b}{b_{ref}} \times t_{ref}, \quad e_b(n) = n \times \frac{b}{b_{ref}} \times e_{ref} \tag{3}$$

This approach allows constructing full DNN profiles for any batch size and any number of points (batches) needed without requiring extra physical measurements. Specifically, because the performance and energy functions are approximately linear and pass through the origin of the coordinate system, each function for a given DVFS configuration can be accurately estimated from a single physical measurement. Therefore, the overhead of constructing the functions is negligible.

However, it is noteworthy that if factors such as memory paging or excessive memory bandwidth utilisation introduce non-linear behaviour, the performance and energy functions can instead be obtained through direct physical measurements or approximated using more accurate linear or non-linear models. Since HEPOPTA_Merge does not assume any specific shape for the input profiles, it can solve the bi-objective optimisation problem for both linear and non-linear performance and energy profiles.

B. HYBRID DNN EVALUATION WITH DVFS-AWARE CONFIGURATIONS

In this section, we study the performance and energy efficiency of the Hybrid DNN application on heterogeneous hybrid servers. We compare the optimality of our dual-decision-variable approach against both standard Linux CPU frequency governors and TensorFlow's MirroredStrategy, a widely used distribution method for training DNNs on single-node servers.

Figure 6 presents two discrete Pareto fronts for HCLServer02: on-demand and DVFS. The on-demand Pareto front illustrates the trade-off between performance and energy consumption when batch distribution is the sole decision variable for optimisation. Each Pareto-optimal point

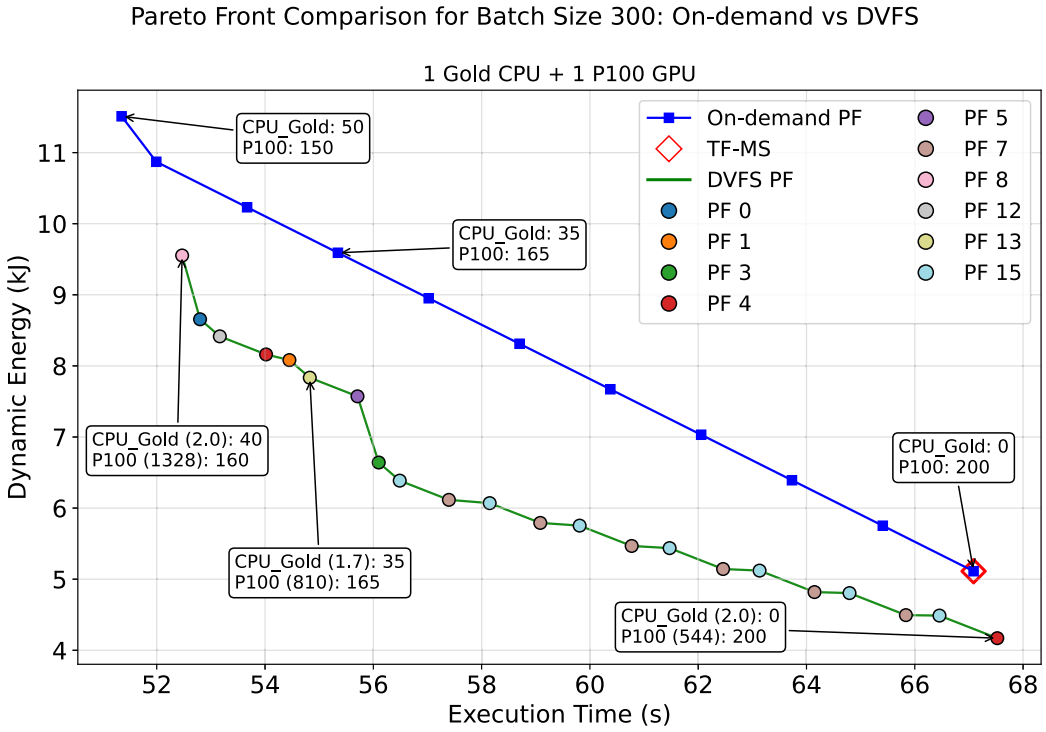


FIGURE 6. Pareto-optimal solutions for training the Hybrid DNN using the MNIST dataset with a batch size of 300, on HCLServer02. The plot compares Pareto fronts for On-demand and DVFS strategies. Annotations indicate the distribution of 200 batches between two abstract processors and their DVFS configurations for selected Pareto-optimal points.

on this front specifies the optimal allocation of 200 batches, each of size 300, between the two abstract processors operating under the on-demand Linux governor. To obtain the global on-demand Pareto-optimal solutions for performance and energy consumption, we input the on-demand profiles, as described in Section VI-A, to the HEPOPTA algorithm.

Another Pareto front, denoted as the DVFS Pareto front, is located below the on-demand front in Figure 6. In contrast to the on-demand front, the DVFS Pareto front is obtained by optimising *both* decision variables: the batch distribution and the DVFS configuration of the two abstract processors. Each Pareto-optimal point on this DVFS Pareto front represents the best trade-off between execution time and dynamic energy consumption, achievable by distributing the 200 batches of size 300 between the two processors, each operating under a selected userspace DVFS setting.

To construct the DVFS Pareto front, the discrete linear performance and energy functions with a step size of five, presented in Section VI-A, are input into our optimiser, HEPOPTA_Merge. As described in Section V, the algorithm first generates individual Pareto fronts for all possible DVFS configurations of the two abstract processors using the HEPOPTA algorithm, optimising only the batch distribution. These individual fronts are then merged to form the global DVFS Pareto front, thereby incorporating DVFS settings as the second decision variable in the optimisation problem. Figure 7 depicts how HEPOPTA_Merge constructs the merged DVFS Pareto front from 16 Pareto fronts (PF 0 to PF 15).

In general, each Pareto front has two endpoints. The leftmost point corresponds to the solution that minimises execution time by balancing the training workload of 200 batches of size 300 between the two abstract processors CPU_Gold and GPU_P100. As annotated on the on-demand Pareto front, unlike homogeneous distribution strategies, the left-most solution *unevenly* distributes a total of 200 batches between the CPU_Gold and GPU_P100 as 35 and 165 batches, respectively, achieving an execution time of 51.4 s with a dynamic energy consumption of 11.5 kJ. This solution is denoted as {(50, on-demand), (150, on-demand)}. Although the performance-optimal point on the DVFS Pareto front is approximately 1.9% slower—taking around 1 second longer than the on-demand optimum-it achieves significant energy efficiency, reducing energy consumption by about 20% to 9.6 kJ. This is due to the on-demand execution time function being significantly faster than the fixed DVFS configurations for CPU_Gold. As discussed earlier, the faster on-demand execution times compared with the DVFS-fixed configurations are achieved because the highest supported processor frequency is excluded from the DVFS settings to avoid potential power and thermal throttling.

The rightmost point on each Pareto front corresponds to the energy-optimal solution, obtained by allocating the workload to the most energy-efficient processor(s).

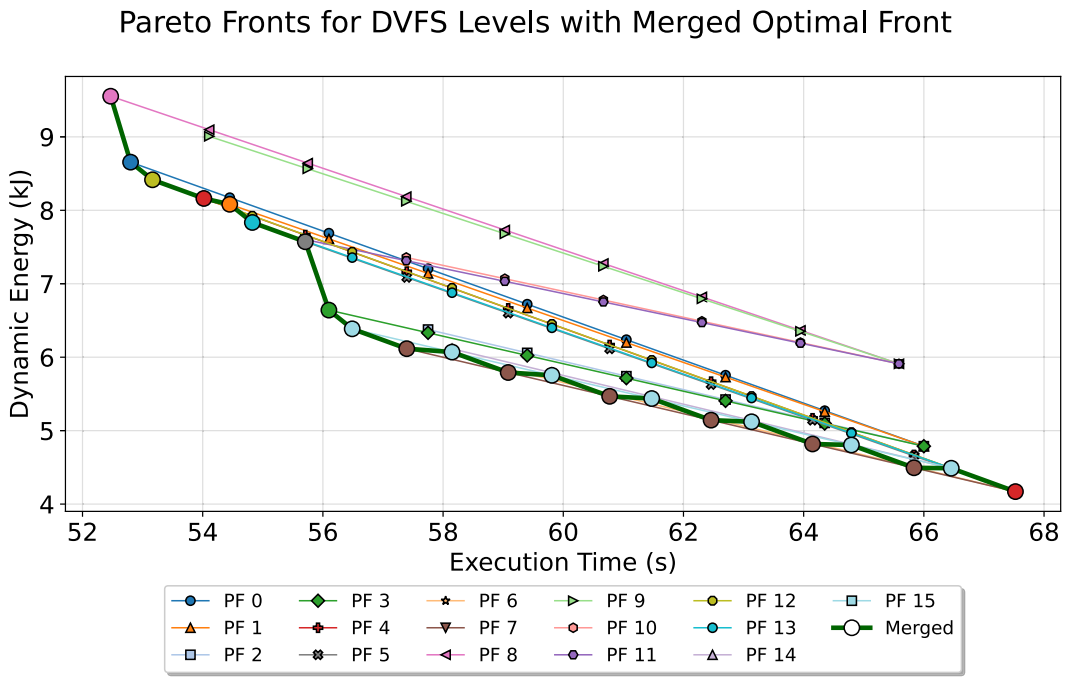


FIGURE 7. Illustration of HEPOPTA_Merge to construct the global DVFS Pareto front by using all possible individual Pareto fronts for different DVFS configurations.

On HCLServer02, GPU_P100 is more energy-efficient than CPU_Gold, which is why all 200 batches are assigned to the GPU to achieve the minimum dynamic energy. Similar to the performance-optimal case, the DVFS energy-optimal point consumes less energy than its on-demand counterpart but is slightly slower. In the Pareto fronts shown in this figure, shifting from the left-most solution with optimal performance to the right-most point with minimum energy consumption results in a performance degradation of about 29%, while achieving an outstanding energy saving of approximately 129%. This occurs because the Pareto front is steep, leading to much larger variations in energy consumption compared to execution time. This indicates that DNN training benefits more significantly from energy optimisation than from performance-oriented optimisation.

Several intermediate Pareto points between the end-points represent trade-off solutions between execution time and dynamic energy consumption for the DNN training workload on HCLServer02. For example, the fourth point {(35, on-demand), (165, on-demand} from the left on the on-demand Pareto front achieves an execution time of 55.3 s and a dynamic energy consumption of 9.6 kJ. This solution involves only one decision variable, allocating a total of 200 batches between the CPU_Gold and GPU_P100 as 35 and 165 batches, respectively, operating under the on-demand Linux governor.

Consider the sixth Pareto solution {(35, 1.7), (165, 810)} from the left on the DVFS Pareto front, which has an execution time closest to that of the fourth on-demand Pareto point. Originating from PF 13, this solution allocates 35 batches to CPU_Gold at a 1.7 GHz DVFS state and 165 batches to GPU_P100 at an 810 MHz DVFS configuration. Executing the DNN application under this configuration yields an execution time of 54.8 s and a dynamic energy consumption of 7.8 kJ. Although its batch distribution and execution time are comparable to the fourth on-demand Pareto point, its dynamic energy consumption is around 23% lower, owing to the inclusion of DVFS state as a second optimisation decision variable.

We can observe the same trend across the other solutions on these Pareto fronts, where the DVFS Pareto solutions are located beneath the corresponding on-demand solutions, indicating that optimising both batch distribution and DVFS states yields solutions that dominate the on-demand configuration in terms of execution time and energy consumption. It is also noteworthy that all intermediate solutions in both Pareto fronts assign the workload unevenly across processing devices. This implies that homogeneous workload distribution strategies yield sub-optimal outcomes on heterogeneous systems.

In addition, Figure 6 highlights the execution time and dynamic energy consumption of the hybrid DNN application when executed under TensorFlow MirroredStrategy (TF-MS). Under this configuration, the whole training workload is allocated to the single P100 GPU available on the server. This solution is 27.9% slower than the DVFS solution with the minimum execution time, and it consumes 22.6% more energy than the DVFS solution with optimal energy consumption.

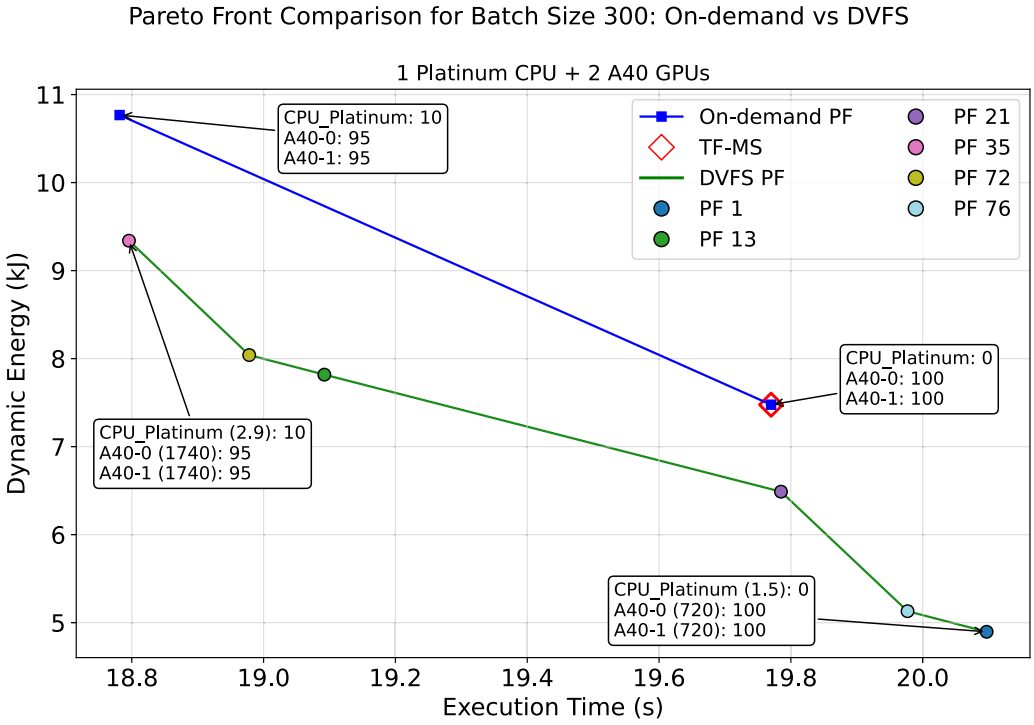


FIGURE 8. Pareto-optimal solutions for training the Hybrid DNN using the MNIST dataset with a batch size of 300, on HCLServer03. The plot compares Pareto-fronts for On-demand and DVFS strategies. Annotations indicate the distribution of 200 batches between two abstract processors and their DVFS configurations for selected Pareto-optimal points.

Figure 8 displays the Pareto fronts for executing the hybrid DNN application on HCLServer03. Consistent with the results in Figure 6, the Pareto fronts are steep, indicating that the application is more responsive to energy optimisation than to performance optimisation, and all DVFS Pareto points lie below the on-demand solutions. In addition, the Pareto fronts include optimal solutions utilising both CPU and GPU. Furthermore, the performance-optimal DVFS solution achieves a shorter execution time, and the energy-optimal DVFS solution yields lower dynamic energy consumption, compared with the TensorFlow solution (TF-MS).

We examined several Pareto fronts for different batch sizes on both servers and observed the same trends, as displayed in Figures 6 and 8. The key findings are summarised below:

- DVFS Pareto superiority: All solutions in the DVFS Pareto set outperform their on-demand counterparts, demonstrating superiority in both execution time and dynamic energy consumption. This indicates that DVFS-based configurations not only enhance energy efficiency but also, in most cases, deliver improved performance for DNN training. These results confirm that dual-decision-variable optimisation achieves greater energy savings than optimising batch distribution alone.
- Static DVFS vs. dynamic on-demand: None of the DVFS solutions appear in the corresponding on-demand Pareto fronts. This indicates that although dynamic DVFS adjustment via the on-demand governor is widely used, static DVFS configuration yields superior solutions in terms of energy efficiency. Therefore, only relying on dynamic on-demand settings may limit optimisation potential.
- Energy sensitivity of DNN training: DNN training on hybrid platforms exhibits steep Pareto fronts. It indicates that small performance degradations can result in substantial energy savings. This places DNN training in the category of applications that respond better to energy optimisation than to execution time optimisation.
- CPUs as valuable resources for DNN training: Contrary to the common belief that CPUs are not suitable for training machine learning models such as MLP DNNs, there are Pareto fronts that include optimal solutions utilising both CPU and GPU. This highlights that all computing devices in heterogeneous hybrid platforms should be regarded as valuable resources for machine learning training tasks.
- Heterogeneous vs homogeneous workload distribution: All solutions in the Pareto fronts assign the workload unevenly across heterogeneous processing devices. This indicates that homogeneous workload distribution strategies lead to sub-optimal results on heterogeneous systems.

C. STATISTICAL ANALYSIS OF PARETO-OPTIMAL SOLUTIONS

This section presents a statistical analysis of the Pareto-optimal solutions for an experimental dataset, consisting

of all batch sizes between 50 and 6,000, which evenly divide the total 60,000 samples. After deriving the corresponding execution time and dynamic energy functions, the HEPOPTA_Merge algorithm is applied to compute the Pareto-optimal solutions for each batch size on HCLServer02 and HCLServer03. For comparison, the on-demand function is also used as input to the `HEPOPTA` algorithm for obtaining the corresponding on-demand Pareto fronts.

The minimum, average, and maximum cardinalities of the Pareto fronts obtained by HEPOPTA_Merge on HCLServer02 are 4, 23, and 69, respectively, while, for HCLServer03, they are 3, 7, and 21. Smaller batch sizes yield a larger number of Pareto-optimal solutions, as they result in more batches available for distribution. Furthermore, HCLServer03 exhibits fewer Pareto-optimal solutions than HCLServer02 due to the presence of two identical A40 GPUs, which reduce the diversity of trade-off configurations, and the structure of the input functions, which limits variability in performance–energy trade-offs across the computing devices.

The minimum, average, and maximum numbers of trade-off solutions in the on-demand Pareto fronts are 1, 15, and 60 on HCLServer02, and 1, 3, and 8 on HCLServer03, respectively. As expected, the cardinalities of the on-demand Pareto fronts are smaller than those of the DVFS fronts.

We use the experimental dataset to investigate how performance optimisation affects dynamic energy consumption and vice versa. The performance gain (%) is calculated as $\frac{t_{e_{opt}} - t_{opt}}{t_{opt}} \times 100$, where t_{opt} is the optimal execution time and $t_{e_{opt}}$ is the execution time of the solution with optimal dynamic energy consumption. The dynamic energy savings (%) is computed as $\frac{e_{t_{opt}} - e_{opt}}{e_{opt}} \times 100$, where e_{opt} is the optimal dynamic energy consumption and $e_{t_{opt}}$ is the energy consumption of the minimum execution-time solution. Since the energy saving is expressed relative to the energy-optimal solution, values greater than 100% are possible and indicate that the solution with minimum execution time consumes more dynamic energy than the energy-optimal solution. On HCLServer02, the minimum, average, and maximum performance gains are 2.95%, 25.1%, and 29.4%, respectively; on HCLServer03, these values are 1.6%, 5.75%, and 14.7%. The minimum, average, and maximum dynamic energy savings are 41.7%, 117.7%, and 131.3% for HCLServer02, and 19.6%, 70.0%, and 114.2% for HCLServer03.

We statistically analyse the performance gain and dynamic energy savings of the on-demand Pareto fronts. On HCLServer02, the minimum, average, and maximum performance gains are 0.0%, 28.1%, and 32.6%, respectively; on HCLServer03, these values are 0.0%, 4.1%, and 13.1%. The minimum, average, and maximum dynamic energy savings are 0.0%, 110.1%, and 125.2% for HCLServer02, and 0.0%, 31.7%, and 58.7% for HCLServer03. These results indicate that DVFS-aware optimisation provides a wider range of improvements in both execution time and dynamic energy savings compared with the on-demand configuration on both platforms. The statistical results for performance gain and energy saving reinforce our observation that DNN training exhibits a stronger response to energy optimisation than to performance optimisation, as discussed in Section VI-B.

We also examine the extent to which performance can be enhanced by allowing up to a 10% increase in dynamic energy consumption over the optimal solution, and conversely, how much dynamic energy can be reduced when accepting a 10% degradation in performance over the optimal solution. The percentage execution time improvement is calculated as $\frac{t_{e_{opt}} - t_{e_{opt \times 1.1}}}{t_{e_{opt}}} \times 100$, where $t_{e_{opt \times 1.1}}$ is the execution time of the solution consuming 10% more dynamic energy than the optimal energy configuration. Similarly, the percentage dynamic energy saving is calculated as $\frac{e_{t_{opt}} - e_{t_{opt \times 1.1}}}{e_{t_{opt}}} \times 100$, where $e_{t_{opt \times 1.1}}$ is the dynamic energy consumption of a solution with a 10% increase in execution time over the optimal execution time. Based on HEPOPTA_Merge results, the average and maximum performance improvements are 2.1% and 3.2% for HCLServer02, and 0.8% and 2.6% for HCLServer03. The corresponding average and maximum dynamic energy savings are 33.1% and 54.5% for HCLServer02, and 37.0% and 53.3% for HCLServer03. It indicates that allowing a 10% relaxation in one of the objectives can lead to substantial improvements, particularly in dynamic energy savings, without significant compromises in execution time. The impact of these trade-offs is determined by the slope of the line segments on the Pareto fronts, where a steeper slope signifies a greater energy saving with only a slight increase in execution time, whereas a gentler slope denotes a more marginal change between the two objectives.

In Section VI-B, we examined Pareto fronts for DNN training with a batch size of 300 on HCLServer02 and HCLServer03, highlighting how integrating batch distribution and DVFS states improves overall system efficiency, particularly in dynamic energy savings. After creating both on-demand and DVFS Pareto fronts for all batch sizes in the experimental dataset, the results consistently show that DVFS solutions are more energy-efficient than on-demand ones for any batch size. To quantify the differences between DVFS and on-demand solutions per batch size, we define the performance efficiency of a DVFS solution over an on-demand one as $\frac{t_{\text{on-demand}} - t_{\text{dvfs}}}{t_{\text{dvfs}}} \times 100$, where $t_{\text{on-demand}}$ and t_{dvfs} denote the execution times of the respective solutions. Similarly, the dynamic energy efficiency percentage is calculated as $\frac{e_{\text{on-demand}} - e_{\text{dvfs}}}{e_{\text{dvfs}}} \times 100$, where $e_{\text{on-demand}}$ and e_{dvfs} are the corresponding dynamic energy consumptions. Since it is impractical to compute these metrics for every Pareto solution pair, we evaluate them only for the endpoints: the performance-optimal (left-most) and dynamic-energy-optimal (right-most) solutions for each batch size. For performance-optimal solutions, the average performance efficiency is −2.8% on HCLServer02 and -0.1% on HCLServer03, with corresponding energy efficiencies

TABLE 1. Comparison of the average statistical results on HCLServer02 and HCLServer03.

Metric	HCLServer02		HCLServer03	
	DVFS	On-Demand	DVFS	On-Demand
Average Cardinality	23	15	7	3
Performance Gain (%)	25.1	28.1	5.75	4.1
Dynamic Energy Savings (%)	117.7	110.1	70.0	31.7
Performance Improvement (10% Energy Relaxation)	2.1	–	0.8	–
Energy Savings (10% Performance Relaxation)	33.1	–	37.0	–
DVFS vs. On-Demand Endpoints Efficiency	HCLServer02		HCLServer03	
	Perf-Optimal	Energy-Optimal	Perf-Optimal	Energy-Optimal
Performance Efficiency (%)	-2.8	-0.6	-0.1	-1.6
Energy Efficiency (%)	18.2	22.6	18.5	52.7
DVFS Optimal vs. MirroredStrategy	HCLServer02		HCLServer03	
Performance Gap (%)	25.1		5.7	
Energy Gap (%)	117.7		67.0	

of 18.2% and 18.5%. For dynamic-energy-optimal solutions, the performance efficiencies are -0.6% and -1.6%, while the energy efficiencies are 22.6% and 52.7% for HCLServer02 and HCLServer03, respectively. These results confirm that DVFS solutions consistently deliver substantial energy savings over on-demand solutions, a trend expected to hold across the full range of trade-off points in the Pareto fronts. These statistical results reinforce our experimental observation of DVFS Pareto superiority over on-demand configurations, and further support the advantage of static DVFS settings compared to dynamic on-demand approaches, as discussed in Section VI-B.

The experimental dataset is also used to compare the efficiency of the proposed dual-decision-variable method with TensorFlow's MirroredStrategy workload distribution. We obtain the performance gap percentage as $\frac{t_{TF}-t_{opt}}{t_{opt}} \times 100$, where t_{TF} denotes the execution time of MirroredStrategy and t_{opt} represents the optimal DVFS execution time. Similarly, the dynamic energy gap percentage is calculated as $\frac{e_{TF}-e_{opt}}{e_{opt}} \times 100$, where e_{TF} is the dynamic energy consumption under MirroredStrategy, and e_{opt} denotes the optimal dynamic energy consumption. The results demonstrate average and maximum performance gaps of 25.1% and 29.4% on HCLServer02 and 5.7% and 14.7% on HCLServer03, respectively, while the corresponding energy gaps are 117.7% and 131.3% on HCLServer02 and 67% and 114.2% on HCLServer03.

Table 1 summarises the average statistical results for the two servers presented in this section:

We conducted all evaluations in this work on MLPs, but the proposed method can be applied to other DNN architectures that exhibit a similar batch-parallel computational structure.

VII. CONCLUSION

This paper addressed the bi-objective optimisation of training fully connected multilayer perceptron DNNs for two conflicting objectives: execution time and dynamic energy consumption. The optimisation considered both application-level and platform-level decision variables, where batch distribution represented the application-level and DVFS states represented the platform-level configurations.

The experimental results on two heterogeneous hybrid servers demonstrated that the joint optimisation of batch distribution and DVFS states consistently outperformed the single-decision-variable optimisation, where workload distribution is considered as the only decision variable when all processors operate under the on-demand governor. On average, the dual-decision-variable optimisation achieved average energy savings of 52.7% compared with single-decision-variable optimisation. In addition, we observe that Pareto-optimal solutions distribute the workload unevenly across processing devices, indicating that homogeneous workload distribution strategies lead to sub-optimal performance and energy efficiency on heterogeneous systems. The obtained results also indicate that DNN training is more responsive to energy optimisation than to performance optimisation, allowing substantial energy savings and reductions in carbon emissions with only minor performance degradation. Furthermore, although it is commonly assumed that CPUs are not suitable for training MLP DNNs, several Pareto-optimal solutions were obtained that utilise both CPU and GPU.

As the proposed framework is neither platform- nor topology-specific, it can be extended to other multilayer perceptron architectures that have data-independence properties similar to those of fully connected DNNs.

These architectures include convolutional neural networks (CNNs), Transformers, autoencoders, and variational autoencoders (VAEs). In these architectures, as in fully connected DNNs, individual training samples can be processed independently.

As future work, we plan to enhance the framework by incorporating additional parameters for performance and energy modelling, including the thermal behaviour of computing devices and system load fluctuations. Furthermore, we aim to extend the framework beyond single-node systems to support multi-node clusters.

DECLARATION OF AI-ASSISTED TECHNOLOGIES

During the preparation of this work, the authors used OpenAI ChatGPT to improve the manuscript's readability and language. The authors reviewed and edited the output as needed and take full responsibility for the final content.

REFERENCES

[1] A. de Vries, "The growing energy footprint of artificial intelligence," *Joule*, vol. 7, no. 10, pp. 2191–2194, Oct. 2023.

[2] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," 2021, *arXiv:2104.10350*.

[3] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.

[4] B. Subramaniam and W.-C. Feng, "Statistical power and performance modeling for optimizing the energy efficiency of scientific computing," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun. Int. Conf. Cyber, Phys. Social Comput.*, Dec. 2010, pp. 139–146.

[5] S. Song, C.-Y. Su, R. Ge, A. Vishnu, and K. W. Cameron, "Iso-energy-efficiency: An approach to power-constrained parallel computation," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, May 2011, pp. 128–139.

[6] J. Lang and G. Rünger, "An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration," *J. Parallel Distrib. Comput.*, vol. 74, no. 9, pp. 2884–2897, Sep. 2014.

[7] A. Chakrabarti, S. Parthasarathy, and C. Stewart, "A Pareto framework for data analytics on heterogeneous systems: Implications for green energy usage and performance," in *Proc. 46th Int. Conf. Parallel Process. (ICPP)*, Aug. 2017, pp. 533–542.

[8] K. M. Tarplee, R. Friese, A. A. Maciejewski, H. J. Siegel, and E. K. P. Chong, "Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 6, pp. 1633–1646, Jun. 2016.

[9] E. Gabaldon, J. L. Lerida, F. Guirado, and J. Planes, "Blacklist muti-objective genetic algorithm for energy saving in heterogeneous environments," *J. Supercomput.*, vol. 73, no. 1, pp. 354–369, Jan. 2017.

[10] R. R. Manumachu and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 160–177, Feb. 2018.

[11] H. Khaleghzadeh, M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on heterogeneous HPC platforms for performance and energy through workload distribution," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 543–560, Mar. 2021.

[12] R. R. Manumachu, H. Khaleghzadeh, and A. Lastovetsky, "Acceleration of bi-objective optimization of data-parallel applications for performance and energy on heterogeneous hybrid platforms," *IEEE Access*, vol. 11, pp. 27226–27245, 2023.

[13] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, "Efficient exact algorithms for continuous bi-objective performance-energy optimization of applications with linear energy and monotonically increasing performance profiles on heterogeneous high performance computing platforms," *Concurrency Comput., Pract. Exper.*, vol. 35, no. 20, p. 7285, Sep. 2023.

[14] H. Khaleghzadeh, A. Khazaei, and A. Lastovetsky, "Parallel DNNs on hybrid heterogeneous platforms: Design, implementation, and optimisation for performance and energy," *IEEE Access*, vol. 14, pp. 48723–48740, 2026.

[15] M. Y. Lim, V. W. Freeh, and D. K. Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs," in *Proc. ACM/IEEE Conf. Supercomputing*, 2006, p. 107.

[16] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan, "Reducing power with performance constraints for parallel sparse applications," in *Proc. 19th IEEE Int. Parallel Distrib. Process. Symp.*, Apr. 2005, p. 8.

[17] V. W. Freeh, D. K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B. L. Rountree, and M. E. Femal, "Analyzing the energy-time trade-off in high-performance computing applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 18, no. 6, pp. 835–848, Jun. 2007.

[18] B. Rountree, D. K. Lowenthal, S. Funk, V. W. Freeh, B. R. de Supinski, and M. Schulz, "Bounding energy consumption in large-scale MPI programs," in *Proc. ACM/IEEE Conf. Supercomputing*, Nov. 2007, pp. 1–9.

[19] V. W. Freeh, N. Kappiah, D. K. Lowenthal, and T. K. Bletsch, "Just-in-time dynamic voltage scaling: Exploiting inter-node slack to save energy in MPI programs," *J. Parallel Distrib. Comput.*, vol. 68, no. 9, pp. 1175–1185, Sep. 2008.

[20] B. Rountree, D. K. Lownenthal, B. R. de Supinski, M. Schulz, V. W. Freeh, and T. Bletsch, "Adagio: Making DVS practical for complex HPC applications," in *Proc. 23rd Int. Conf. Supercomputing*, Jun. 2009, pp. 460–469.

[21] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E.-G. Talbi, A. Y. Zomaya, and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, Nov. 2011.

[22] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGrid)*, May 2012, pp. 300–309.

[23] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012.

[24] Y. Kessaci, N. Melab, and E.-G. Talbi, "A Pareto-based metaheuristic for scheduling HPC applications on a geographically distributed cloud federation," *Cluster Comput.*, vol. 16, no. 3, pp. 451–468, Sep. 2013.

[25] L. Yu, Z. Zhou, S. Wallace, M. E. Papka, and Z. Lan, "Quantitative modeling of power performance tradeoffs on extreme scale systems," *J. Parallel Distrib. Comput.*, vol. 84, pp. 1–14, Oct. 2015.

[26] N. Gholkar, F. Mueller, and B. Rountree, "Power tuning HPC jobs on power-constrained systems," in *Proc. Int. Conf. Parallel Architectures Compilation*, Sep. 2016, pp. 179–191.

[27] J. Kołodziej, S. U. Khan, L. Wang, and A. Y. Zomaya, "Energy efficient genetic-based schedulers in computational grids," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 4, pp. 809–829, Mar. 2015.

[28] F. D. Rossi, M. G. Xavier, C. A. F. De Rose, R. N. Calheiros, and R. Buyya, "E-eco: Performance-aware energy-efficient cloud data center orchestration," *J. Netw. Comput. Appl.*, vol. 78, pp. 83–96, Jan. 2017.

[29] K. Chen, J. Lenhardt, and W. Schiffmann, "Improving energy efficiency of web servers by using a load distribution algorithm and shutting down idle nodes," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 745–748.

[30] A. Benoit, L. Lefèvre, A.-C. Orgerie, and I. Raïs, "Reducing the energy consumption of large-scale computing systems through combined shutdown policies with multiple constraints," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 1, pp. 176–188, Jan. 2018.

[31] Y. Jiang, Y. Zhu, C. Lan, B. Yi, Y. Cui, and C. Guo, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *Proc. 14th USENIX Symp. Operating Syst. Design Implement. (OSDI 20)*, 2020, pp. 463–479.

[32] P. Panda, A. Tripathy, and K. C. Bhuyan, "Reinforcement learning-based dynamic voltage and frequency scaling for energy-efficient computing," in *Proc. 3rd Int. Conf. Distrib. Comput. Electr. Circuits Electron. (ICDCECE)*, Apr. 2024, pp. 1–6.

[33] J. Zidar, T. Matić, I. Aleksi, and Ž. Hocenski, "Dynamic voltage and frequency scaling as a method for reducing energy consumption in ultra-low-power embedded systems," *Electronics*, vol. 13, no. 5, p. 826, Feb. 2024.

[34] A. Lastovetsky and R. R. Manumachu, "New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1119–1133, Apr. 2017.
[35] L. Dagum and R. Menon, "OpenMP: An industry standard API for shared-memory programming," *IEEE Comput. Sci. Eng.*, vol. 5, no. 1, pp. 46–55, Jan. 1998.
[36] S. Wienke, P. Springer, C. Terboven, and D. A. Mey, "Openacc first experiences with real-world applications," in *Proc. 18th Int. Conf., Euro-Par Euro-Par Parallel Process.*, Rhodes Island, Greece. Cham, Switzerland: Springer, 2012, pp. 859–870.
[37] B. Nichols, D. Buttlar, and J. Farrell, *Pthreads Programming: A POSIX Standard for Better Multiprocessing*. O'Reilly Media, Sebastopol, CA, USA, 1996.
[38] Y. LeCun. (1998). *The Mnist Database of Handwritten Digits*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[39] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, p. 2204, Jun. 2019.



HAMIDREZA KHALEGHZADEH received the B.Sc. and M.Sc. degrees in computer engineering (software), in 2007 and 2011, respectively, and the Ph.D. degree from the School of Computer Science, University College Dublin, Ireland, in 2019. He is currently a Senior Lecturer with the School of Computing, University of Portsmouth, U.K. His research focuses on optimizing performance and energy consumption in massively heterogeneous systems, high-performance heterogeneous systems, energy efficiency, and big data applications.



ATEFEH KHAZAEI received the B.Sc. degree (Hons.) in computer engineering from Imam Reza International University, Mashhad, Iran, in 2009, the M.Sc. degree in software engineering from Shiraz University, Shiraz, Iran, in 2011, and the Ph.D. degree from the School of Computer Science, University College Dublin (UCD), in 2025. She is currently with the School of Computing, University of Portsmouth, U.K. Her research interests include data analysis, overlapping clustering, text mining, high-performance computing, and software security.



ALEXEY LASTOVETSKY (Member, IEEE) received the Ph.D. degree from Moscow Aviation Institute, in 1986, and the Doctor of Sciences (Habilitation) degree from Russian Academy of Sciences, in 1997. He authored the monographs *Parallel Computing on Heterogeneous Networks* (Wiley, 2003) and *High Performance Heterogeneous Computing* (Wiley, 2009). His main research interests include heterogeneous parallel computing and energy-efficient parallel computing. He is an Inaugural ScholarGPS Highly Ranked Scholar (Lifetime), ranked within 0.05% of all scholars in Parallel Computing.