Minimization of Execution Time and Energy Consumption of Parallel Applications on Heterogeneous Hybrid Parallel Computers Through Optimal Combination of Workload Distribution and Operating Voltage and Frequency

Hesam Nejati Sharifaldin, *Member, IEEE*, Sourodip Ghoshdastidar, Ravi Reddy Manumachu, *Member, IEEE*, and Alexey Lastovetsky, *Member, IEEE*

***Abstract*—The paper addresses the problem of optimization for performance and energy of parallel applications on heterogeneous hybrid parallel computers. State-of-the-art solutions optimize only either the configuration of the platform or the configuration of the application but not both. In this work, we propose a holistic solution of this problem. We mathematically formulate the problem as a bi-objective minimization problem for execution time and energy consumption of a given workload on p heterogeneous processors, where each processor is modeled by a pair of functions of two variables, processor configuration and workload size, returning the execution time and dynamic energy consumption, correspondingly. The problem has two decision variables: a vector of p processor configurations (representing the platform configuration) and a vector of p workload sizes (workload distribution representing the application configuration). We then propose exact algorithms that solve this optimization problem. We apply this solution to a heterogeneous hybrid server comprising a multicore CPU and GPUs, where the processor configuration is represented by a set of (voltage, frequency) pairs for independently powered processor components. The crucial problem in the practical application of the proposed algorithms is the high cost of construction of the input execution time and dynamic energy functions for all platform configurations. To address this problem, we propose and validate a circuit-level Dennard scaling power model and use it to derive scaling models of the execution time and dynamic energy as functions of operating voltages and frequencies of processors consisting of one and two independently powered components. These models are employed in our scaling method, which only requires one and two pairs of experimental time and energy profiles to build the complete set of profiles for the processor, which consists of one and two independently powered components, respectively. Finally, we demonstrate the efficiency and accuracy of our solution in optimizing 2D fast Fourier transform and mixed-precision GEMM hybrid parallel applications on a heterogeneous platform comprising an Intel Icelake multicore CPU and two Nvidia A40 GPUs.**

***Index Terms*—Parallel computing, performance, energy, mathematical optimization, workload distribution, DVFS, bi-objective optimization, heterogeneous hybrid parallel computers.**

Received 14 February 2025; revised 7 May 2026; accepted 5 July 2026. Date of publication 8 July 2026; date of current version 11 August 2026. This work was supported by Taighde Éireann – Research Ireland under Grant 20/FFP-P/8683. Recommended for acceptance by S. Kaxiras. *(Corresponding author: Ravi Reddy Manumachu.)*

The authors are with the School of Computer Science, University College Dublin, D04V1W8 Belfield, Ireland (e-mail: hesam.nejatisharifaldin@ucdconnect.ie; sourodip.ghoshdastidar@ucdconnect.ie; ravi.manumachu@ucd.ie; alexey.lastovetsky@ucd.ie).

This article has supplementary downloadable material available at https://doi.org/10.1109/TC.2026.3711149, provided by the authors.

Digital Object Identifier 10.1109/TC.2026.3711149

I. Introduction

ACHIEVING the energy efficiency objectives and meeting the performance or response time quality-of-service (QoS) requirements is crucial to the commercial viability of modern high-performance computing (HPC) platforms, computational grids, data centers, and cloud computing infrastructures. Therefore, performance and energy are vital to addressing energy efficiency and environmental and commercial concerns in these computing settings and have been the primary objectives of optimization [1], [2], [3], [4], [5].

State-of-the-art solutions for bi-objective optimization of the execution of an application for performance and energy either optimize only the executing platform by varying the platform configuration or optimize only the application by varying the application configuration but fixing a platform configuration, but not both.

The prominent solution methods in the first category optimize only the execution platform by varying platform configurations that employ different energy governors or Dynamic Voltage and Frequency Scaling states (DVFS states) of processors. We define a *DVFS state* of a processor to be a set of (V, f) pairs where each pair contains the operating voltage and frequency of an independently powered processor component in the processor. The mainstream approach to changing a DVFS state safely and reliably is to change the frequencies since adjusting the voltages can potentially destabilize the system. Furthermore, only the frequencies of DVFS states of a processor are typically exposed to the application programmers. Setting

© 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/

a frequency adjusts the associated voltage automatically since the hardware maintains a static mapping between operational voltages and frequencies internally.

Energy governors are Unix kernel models that set the CPU frequencies dynamically depending on the system load. Each governor comprises a set of rules to determine the CPU cores' operating frequencies dynamically. Users with root privileges can select the governors in their application programs. The various types of CPU governors are described in the supplemental material (Section 4).

This work focuses on processor configurations that can be set in application programs. Therefore, we do not consider processor configurations that can be set in BIOS (for example, L1/L2/L3 cache settings, Hyperthreading, Virtualization, and QPI link frequency) needing a platform reboot to switch configurations. Hence, energy governors and DVFS states are the main decision variables covered in this category.

The platform optimization methods above can be further divided into three groups. The first group comprises methods that optimize for performance under a power cap constraint (or energy budget) or optimize for energy under an execution time constraint [6], [7], [8]. They determine a partial Pareto front by applying the power cap or an execution time constraint and select the best configuration that satisfies a user-specific criterion. The second group solves bi-objective optimization for performance and energy with no time or energy constraints [2], [9]. They build the full Pareto front.

The third group contains methods that optimize the execution of an application for a fixed application configuration on a platform by employing DVFS states during the slack phases in the application [10], [11], [12], [13], [14].

The solution methods in the second category optimize only the application by determining the optimal application configuration for performance and energy with the platform configuration fixed. The leading methods in this category optimize the application for performance and energy using workload distribution as a decision variable and a fixed platform configuration [3], [4], [5]. The platform configuration is either an energy governor or a DVFS state.

Therefore, no research addresses holistically the problem of optimizing an application execution for performance and energy through optimal combination of platform and application configurations.

In this work, we propose a holistic model-based solution to minimize the execution time and energy consumption of parallel applications executing on a heterogeneous hybrid parallel computers comprising p heterogeneous processors through the optimal combination of platform and application configurations.

Our solution is based on two theoretical models. The first is a mathematical model of execution of a parallel application on heterogeneous hybrid parallel computers. The second is a model of a processor configuration, which is a DVFS state represented by a set of (voltage, frequency) pairs for independently powered processor components.

We formulate the continuous and discrete versions of the optimization problem employing the first mathematical model of execution of a parallel application on a heterogeneous hybrid parallel computers. The problem has two decision variables: a vector of p processor configurations (representing the platform configuration) and a vector of p workload sizes (workload distribution representing the application configuration). Each processor is modeled by a pair of functions of two variables, processor configuration and workload size, returning the time and dynamic energy consumption correspondingly. The dynamic energy consumption during an application execution is the total energy consumption minus the static energy consumption of the platform during the application execution. The related terminology on bi-objective optimization for performance and energy is further detailed in the supplemental (Section 3).

We then propose exact algorithms that solve the optimization problems to determine the optimal combination of platform and application configurations. The algorithms return globally optimal Pareto fronts.

Our proposed algorithms require as input accurate time and energy functional models of the application for each processor of the platform to determine the global Pareto-optimal front. These models include the execution time and dynamic energy for each possible workload size allocated to the processor and each possible processor's configuration, making the total number of data points in these models huge. Therefore, fast and accurate construction of these models is crucial to the practical efficiency of our proposed algorithms.

In this work, we investigate the problem of efficiently constructing the accurate time and energy models for a processor consisting of one and two independently powered components and whose configuration is represented by a set of (voltage, frequency) pairs for the independently powered components (called a DVFS state of the processor). This problem has two challenges. The first is the efficient and accurate measurement of execution time and energy to obtain an experimental data point. The second is to find a method that would use a minimal number of experimentally obtained points to accurately calculate the complete set of points in the time and energy models.

Since all processors have sufficiently precise processor clocks, accurate execution time measurement (even with tightly coupled components) is not an issue. However, accurate energy measurement presents a real challenge. There are three mainstream approaches to constructing dynamic energy functions: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models.

The first approach, considered the *ground-truth,* is the most accurate but costly (in terms of execution time) [15]. For example, experimentally obtaining a data point (statistical average of dynamic energy using the Student's t-test) for a 2D fast Fourier transform application on the IceLake multicore CPU processor (Table I in the supplemental) takes $\sim$6 and $\sim$10 minutes for the DVFS states corresponding to the highest and lowest processor core frequencies, respectively. We present our statistical methodology in the supplemental (Section 6) to obtain a data point.

While on-chip power sensors, such as RAPL [16] for Intel CPUs [16] or NVML for Nvidia GPUs [15], are poorly

documented and much less accurate, the second approach is nearly as costly as the first one because it also involves statistical averaging of dynamic energy using the Student's t-test.

The third approach is still in its infancy. While it demonstrates some promising results for Intel CPUs [17], no accurate performance event-based models exist for the Nvidia A40 GPUs.

In summary, the current state of the art in constructing performance and energy models of processors as functions of their DVFS states is to build them experimentally [18]. Therefore, the only existing option to build the complete set of points in the functional models will be to experimentally obtain them for all combinations of workload sizes and DVFS states, which is very costly and infeasible at runtime since the number of DVFS states can be significant.

To understand the cost involved, experimentally building the complete set of discrete time and energy models with 30 discrete workload sizes and 27 DVFS states for a 2D fast Fourier transform application on the IceLake multicore CPU processor takes $\sim$108 hours using the ground-truth method. The time for the Nvidia A40 processors is $\sim$360 hours if we consider all the 103 DVFS states (involving only the graphics frequencies). Furthermore, experimentally building the complete set of linear time and energy models, which involves obtaining only two experimental points to build a linear function of workload size for each DVFS state, takes $\sim$7 and $\sim$24 hours for the CPU and GPU processors, respectively. The exorbitant nature of this cost is representative of similar workloads executing on these processors.

Therefore, an innovative solution is needed to solve the challenging problem of efficiently constructing the complete set of functional models for a processor. We propose a solution comprising two advances. First, we introduce and validate a circuit-level Dennard scaling power model and use it to derive scaling models of the execution time and dynamic energy as functions of operating voltages and frequencies of the independently powered processor components. Second, we propose a highly efficient, accurate and portable scaling method that employs these scaling models to build the complete time and energy models for all DVFS states of the processor using experimentally obtained time and energy profiles for only one DVFS state (if the processor consists of one independently powered component) or two DVFS states (if the processor consists of two independently powered components).

We describe and validate the scaling models and method for three state-of-the-art processors, Intel Skylake (Intel Xeon Gold 6152) and Intel Icelake (Intel Platinum 8362) multicore CPU processors and an Nvidia A40 GPU processor, showcasing their real-world relevance and applicability. Our scaling method reduces the time needed to build the complete discrete and linear functional models for the CPU processors to 4 hours and 16 minutes, resulting in 27-fold improvement. The GPU processors' times are 7 hours and 28 minutes, and the improvement is 51-fold.

To summarize, our solution's main steps to minimizing execution time and energy consumption of parallel applications on heterogeneous hybrid parallel computers through the optimal combination of workload distribution and DVFS state are:

- Our proposed scaling models and method are used to efficiently build the complete functional models (pairs of execution time and dynamic energy functions for all the DVFS states) of the p heterogeneous processors,
- Based on the functional nature of the processor models, our proposed algorithms are invoked to solve either the continuous or the discrete optimization problem. The solution is a Pareto-optimal front given by a set of pairs of vectors comprising the optimal platform configuration (vector of DVFS states) and workload distribution (vector of workload sizes) in decision variable space, and
- Finally, the parallel application is executed using a Pareto-optimal solution filtered by a user-specified criterion.

Detailed description of the workflow of the entire optimization process can be found in the supplemental (Section XI).

We validate the efficiency and accuracy of our solution through rigorous experimental studies using two hybrid parallel applications, 2D fast Fourier transform and mixed precision matrix multiplication, on a heterogeneous platform comprising an Intel Icelake multicore CPU and two Nvidia A40 GPUs.

The main contributions of this work are as follows:

- Formulation of the discrete and continuous versions of the bi-objective optimization problem minimizing the execution time and energy consumption of parallel applications executing on a heterogeneous hybrid parallel computers comprising p heterogeneous processors through the optimal combination of platform and application configurations.
- A novel holistic model-based solution solving the optimization problems based on two theoretical models, a mathematical model of execution of a parallel application on heterogeneous hybrid parallel computers and a model of a processor configuration, a DVFS state represented by a set of (voltage, frequency) pairs for independently powered processor components.
- An exact algorithm solving the continuous bi-objective optimization problem for processors modelled by linear increasing execution time and dynamic energy functions of workload size for each platform configuration. An exact algorithm solving the discrete bi-objective optimization problem for processors modelled by discrete execution time and dynamic energy functions of arbitrary shape.
- Efficient algorithms that determine globally optimal Pareto fronts given sets of discrete and piecewise linear Pareto fronts.
- A circuit-level Dennard scaling power model and using it to derive scaling models of the execution time and dynamic energy as functions of operating voltages and frequencies of processors consisting of one and two independently powered components.
- A highly efficient, accurate and portable scaling method employing the proposed scaling models to build the complete time and energy functions of a processor for all its DVFS states using experimentally obtained time and energy profiles for only one (if the processor consists of one

independently powered component) and two (if it consists of two independently powered components) DVFS states.
- Experimentally validating our scaling models and method for an Intel Icelake multicore CPU and Nvidia A40 GPU processors.
- Experimental study demonstrating the efficiency and accuracy of our solution in optimizing two parallel hybrid applications, 2D-FFT and DGEMM, on an Intel Icelake multicore CPU and two Nvidia A40 GPUs.

We organize the paper as follows. Section II contains the model of application execution on heterogeneous hybrid parallel computers. Section III contains the continuous optimization problem formulation and an algorithm solving the problem for a specific case of continuous execution time and dynamic energy functions. Section IV details the method proposed for fast construction of execution time and energy functions using a scaling method employing circuit-level Dennard scaling power and execution time models. Experimental results are provided in Section V. Section VI overviews the related work. Finally, we conclude the paper in Section VII.

II. Model of a Parallel Application Execution on Heterogeneous Hybrid Parallel Computers

In this work, we consider the mathematical model of execution of a data-parallel application comprising p software components (kernels) executing in parallel on a heterogeneous hybrid parallel computers containing p heterogeneous processors. Each processor is modelled by a pair of functions of two variables, processor configuration and workload size, returning the execution time and dynamic energy consumption correspondingly.

The model represents the execution of an application workload of size n using the p heterogeneous processors modelled by $\mathbb{M} = \{(T_1, E_1), \cdots, (T_p, E_p)\}$. Each processor i has c_i configurations with identifiers numbered from 1 to c_i. The set $\mathbb{C} = \{c_1, \cdots, c_p\}$ contains the number of configurations of the processors.

We represent a configuration of the platform consisting of these p processors by $\sigma \in C_1 \times \cdots \times C_p$, where $C_i = \{1, \cdots, c_i\}$. Therefore, σ is a vector of p scalars, $\{\sigma_1, \cdots, \sigma_p\}$.

$\sigma_i \in \{1, \cdots, c_i\}$ is the configuration index of processor i giving the execution time and dynamic energy functions, $t_{i\sigma_i}(w)$ and $e_{i\sigma_i}(w)$, in the sets T_i and E_i ($i \in \{1, \cdots, p\}$).

The complete set of functional models (pair of execution time and dynamic energy functions) of processor $i \in \{1, \cdots, p\}$ is represented by two sets, T_i and E_i, containing c_i execution time and dynamic energy functions of workload size, $T_i = \{t_{i1}(w), \ldots, t_{ic_i}(w)\}$ and $E_i = \{e_{i1}(w), \ldots, e_{ic_i}(w)\}$.

The function, $e_{ij}(w), i \in \{1, \cdots, p\}, j \in \{1, \cdots, c_i\}$, gives the dynamic energy consumption of processor i to execute the workload size w employing the processor configuration, j, and $t_{ij}(w)$ gives the execution time.

III. Continuous Optimization Problem Formulation and an Exact Algorithm

This section formulates HEPOPTGC, the continuous version of the bi-objective optimization problem minimizing the execution time and energy consumption of parallel applications executing on a heterogeneous hybrid parallel computers comprising p heterogeneous processors through the optimal combination of platform and application configurations.

The problem has two decision variables: a vector of p processor configurations (platform configuration) and a vector of p workload sizes (workload distribution representing the application configuration). The workload size solved by a processor is assumed to be a multiple of a basic computation unit that does not differ during the application execution. For example, a basic computation unit in blocked matrix multiplication is a matrix update, $c = c + a \times b$, where a, b, and c are $u \times u$ matrix blocks of fixed size u. Therefore, the workload sizes are given in multiples of $2 \times u^3$, which is the number of arithmetic operations in the basic computation unit.

HEPOPTGC considers the execution of an application workload of size $n \in \mathbb{R}_{\geq 0}$ on a heterogeneous hybrid parallel computers with base/idle power consumption, P_s, and comprising p heterogeneous processors. The set, $\mathbb{M}$, contains the processor models, which are continuous execution time and dynamic energy functions of workload size. Specifically, the sets T_i and E_i contain c_i continuous execution time and dynamic energy functions of processor i where the function, $e_{ij}(w) : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$, $j \in \{1, \cdots, c_i\}$, gives the dynamic energy consumption of processor i to execute the workload size $w \in \mathbb{R}_{\geq 0}$ employing the processor configuration, j, and $t_{ij}(w) : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ gives the execution time.

HEPOPTGC is then formulated as follows:

HEPOPTGC$(n, p, \mathbb{C}, \mathbb{M}, P_s)$:

$$G_T(\sigma, \mathbb{W}) = \max_{i=1}^{p} t_{i\sigma_i}(w_{i\sigma_i})$$

$$G_E(\sigma, \mathbb{W}) = \sum_{i=1}^{p}(P_s \times t_{i\sigma_i}(w_{i\sigma_i}) + \sum_{i=1}^{p} e_{i\sigma_i}(w_{i\sigma_i}))$$

$$\underset{(\sigma, \mathbb{W})}{\textbf{minimize}} \quad \{G_T(\sigma, \mathbb{W}), G_E(\sigma, \mathbb{W})\}$$

$$\textbf{subject to:} \quad \sum_{i=1}^{p} w_{i\sigma_i} = n, w_{i\sigma_i} \in \mathbb{R}_{\geq 0}$$
$$i \in \{1, \cdots, p\}, \sigma_i \in \{1, \cdots, c_i\}$$

The two objective functions are $G_T(\sigma, \mathbb{W})$ and $G_E(\sigma, \mathbb{W})$. The objective function $G_T(\sigma, \mathbb{W})$ gives the parallel execution time of the workload employing the platform and application configurations, $\{(\sigma_1, w_{i\sigma_1}), \cdots, (\sigma_p, w_{i\sigma_p})\}$ where $\{\sigma_1, \ldots, \sigma_p\}$ represents the platform configuration and $\{w_{1\sigma_1}, \cdots, w_{p\sigma_p}\}$ is the workload distribution signifying the application configuration. The objective function $G_E(\sigma, \mathbb{W})$ gives the total energy consumption during the execution of the workload. $G_T \times G_E : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$ denotes the objective space of this problem.

HEPOPTGC returns a set of globally Pareto-optimal solutions for execution time and total energy. The set contains elements with four components, $\{G_T(\sigma_{opt}, \mathbb{W}_{opt}), G_E(\sigma_{opt}, \mathbb{W}_{opt}), \sigma_{opt}, \mathbb{W}_{opt}\}$, where the first two components

are two scalars giving the optimal execution time and total energy and following two are vectors representing the Pareto-optimal value of the decision variable. The projection of the set onto the objective space gives the Pareto front. The Pareto-optimal value of the decision variable, $\{\sigma_{opt}, \mathbb{W}_{opt}\}$, signifies a set of pairs, $\{(\sigma_{opt_1}, w_{1\sigma_{opt_1}}), \cdots, (\sigma_{opt_p}, w_{p\sigma_{opt_p}})\}$, where the pair $(\sigma_{opt_i}, w_{i\sigma_{opt_i}})$ gives the optimal processor and application configurations for processor i. If the input parameter P_s is 0, HEPOPTGC returns a set of globally Pareto-optimal solutions for execution time and dynamic energy.

A. Algorithm Solving the Continuous Optimization Problem

We propose a solution that solves a specific case of HEPOPTGC where the execution time and dynamic energy functions of workload size are all linear and increasing. Our solution comprises two sub-algorithms. The first sub-algorithm, *HEPOPTAGC*, determines the globally optimal and locally optimal Pareto fronts. The second sub-algorithm, *HEPOPTADV*, returns the optimal platform and application configurations given a point on the Pareto front.

The inputs to HEPOPTAGC are the workload size, n; the number of heterogeneous processors, p; the set $\mathbb{C} = \{c_1, \cdots, c_p\}$ where c_i is the number of configurations of the processor i; the set of functional models of the processors, $\mathbb{M}$; the static power consumption of the heterogeneous hybrid server platform comprising the p processors, P_s.

HEPOPTAGC returns two Pareto fronts, $\mathbb{P}_g$ and $\mathbb{P}_c$. $\mathbb{P}_g$ is the set of piecewise linear segments representing the globally optimal Pareto front, which can be discontinuous locally with breaks. $\mathbb{P}_l$ is the set of piecewise linear segments containing the globally optimal Pareto front $\mathbb{P}_g$ with breaks filled by locally Pareto-optimal solutions. Each linear segment corresponds to one optimal platform configuration. Therefore, HEPOPTAGC returns three pieces of information for each linear segment in a Pareto front, the two endpoints of the linear segment and the optimal platform configuration. The objective function values for execution time and total energy represent each endpoint. Since the globally optimal Pareto front is discontinuous, it can have isolated points. HEPOPTAGC returns the two objective function values and the optimal platform configuration for an isolated point.

Algorithm 1 illustrates the pseudocode of HEPOPTAGC.

HEPOPTAGC loops over the set, $\mathbb{G}$ (Line 2). The set $\mathbb{G}$ of size $c_1 \times c_2 \times \cdots c_p$ comprises all platform configurations. Each platform configuration is represented by the vector σ of size p.

HEPOPTAGC then invokes the algorithm *LBOPA-TE* [19] using the following inputs: the workload size, n; the number of heterogeneous processors, p; the sets, $\mathbb{T}_\sigma = \{t_{1\sigma_1}(w), \cdots, t_{p\sigma_p}(w)\}$ and $\mathbb{E}_\sigma = \{e_{1\sigma_1}(w), \cdots, e_{p\sigma_p}(w)\}$, containing the p execution time and dynamic energy functions of workload size; the static power consumption of the hybrid server comprising the p processors, P_s. The output of LBOPA-TE is a piecewise linear Pareto front comprising a set of segments whose size is less than or equal to $p-1$. The piecewise Pareto front returned by *LBOPA-TE* is added to the set, $\mathbb{O}$ (Line 5).

Algorithm 1: Algorithm HEPOPTAGC solving the optimization problem HEPOPTGC for workload n using p heterogeneous processors represented by models, $\mathbb{M}$, and $\mathbb{C}$ containing the number of processor configurations. P_s is the static power consumption of the platform.

Input: n, p, $\mathbb{C}$, $\mathbb{M}$, P_s
Output: $\mathbb{P}_g, \mathbb{P}_l$

1 $\mathbb{O} \leftarrow \{\}$
2 **for** $\sigma \in \mathbb{G}$ **do**
3 $\quad \mathbb{T}_\sigma = \{t_{1\sigma_1}, \cdots, t_{p\sigma_p}\}$
4 $\quad \mathbb{E}_\sigma = \{e_{1\sigma_1}, \cdots, e_{p\sigma_p}\}$
5 $\quad \mathbb{O} \leftarrow \mathbb{O} \cup$ **LBOPA-TE**$(n, p, \mathbb{T}_\sigma, \mathbb{E}_\sigma, P_s)$
6 **end**
7 **return MergeParetoFrontsC**$(\mathbb{O})$

Algorithm 2: Algorithm HEPOPTADV returns the optimal platform and application configurations associated with a point (t_{opt}, e_{opt}) in a given Pareto front $\mathbb{P}$.

Input: n, p, $\mathbb{M}$, t_{opt}, e_{opt}, $\mathbb{P}$
Output: $\sigma_{opt}, \mathbb{W}_{opt}$

1 $\{\sigma_{opt}, t_{1\sigma_{opt_1}}, \cdots, t_{p\sigma_{opt_p}}\} \leftarrow$ **GetSigmaOpt**$(p, \mathbb{M}, t_{opt}, e_{opt}, \mathbb{P})$
2 $\mathbb{W}_{opt} \leftarrow$ **Partition**$(n, p, t_{1\sigma_{opt_1}}, \cdots, t_{p\sigma_{opt_p}}, t_{opt})$
3 **return** $\{\sigma_{opt}, \mathbb{W}_{opt}\}$

Finally, the routine *MergeParetoFrontsC* uses the Pareto fronts in $\mathbb{O}$ to determine the globally optimal and locally optimal Pareto fronts, $\mathbb{P}_g$ and $\mathbb{P}_l$ (Line 7). $\mathbb{P}_g$ is a globally optimal Pareto front, which can be discontinuous. $\mathbb{P}_l$ is a Pareto front containing the globally optimal Pareto front with any breaks in it filled by locally Pareto-optimal solutions.

The user employs the sub-algorithm HEPOPTADV to determine the optimal platform and application configurations associated with a point, (t_{opt}, e_{opt}), in either $\mathbb{P}_g$ or $\mathbb{P}_l$. The arguments to HEPOPTADV are the workload size, n; the number of heterogeneous processors, p; the functional models of the processors, $\mathbb{M}$; the point (t_{opt}, e_{opt}), and the Pareto front, $\mathbb{P}$.

Algorithm 2 illustrates the pseudocode of HEPOPTADV. HEPOPTADV invokes the sub-routine *GetSigmaOpt*, which finds the linear segment containing the point (t_{opt}, e_{opt}) in the Pareto front, $\mathbb{P}$, and extracts the optimal platform configuration, $\sigma_{opt} = \{\sigma_{opt_1}, \cdots, \sigma_{opt_p}\}$, associated with it. Then, it invokes the algorithm *Partition* [19] to find the optimal workload distribution. The arguments to *Partition* are the workload size, n; the number of heterogeneous processors, p; the set $\{t_{1\sigma_{opt_1}}(w), \cdots, t_{p\sigma_{opt_p}}(w)\}$ containing the p execution times of workload size, and t_{opt}. Finally, HEPOPTADV returns the

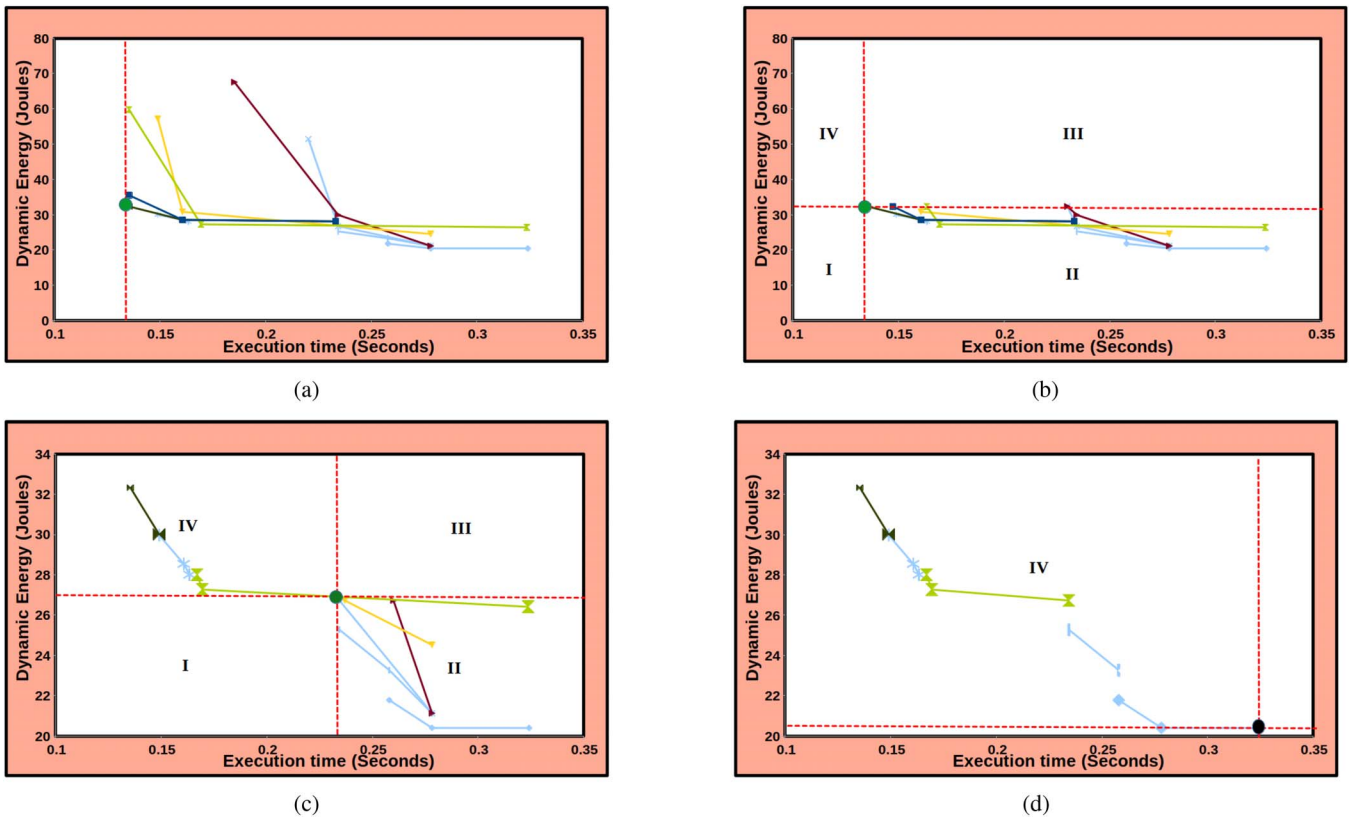


Fig. 1. Illustration of the continuous algorithm to determine the globally optimal Pareto front given a set of piecewise linear Pareto fronts. The algorithm commences by finding the point with the least execution time represented by a green sphere. It draws a vertical line through this point, moving continuously to the right. At each instant, a horizontal line is drawn intersecting the vertical line at a Pareto-optimal point. The two lines allow the objective space to be pruned. Quadrant IV will contain the partially built globally optimal Pareto front at the end of each instant. (b) and (c) Explored and unexplored objective space at the end of two different instants in the execution of the algorithm. (d) Final globally optimal Pareto front.

optimal value of the decision variable (platform and application application vectors) associated with the point (t_{opt}, e_{opt}).

B. Algorithm for Merging Piecewise Linear Pareto Fronts

We now describe our continuous algorithm *MergeParetoFrontsC* to determine the globally optimal Pareto front given k piecewise linear Pareto fronts. *MergeParetoFrontsC* outputs two Pareto fronts. One is a globally optimal Pareto front, which can be discontinuous and contains breaks. The other is a locally optimal Pareto front containing the globally optimal Pareto front with breaks filled by locally Pareto-optimal solutions.

We illustrate the algorithm using Fig. 1 for the case of three heterogeneous processors solving a workload. We consider a small set of nine platform configurations for clarity in our exposition. Therefore, the input to the algorithm is a set of nine locally optimal Pareto fronts. Each Pareto front corresponding to a platform configuration is piecewise linear and comprises two linear segments.

The algorithm commences from the left by finding the point with the least execution time. If there is more than one such point, it selects one with the least dynamic energy. Fig. 1(a) shows this point as a green sphere. The algorithm then draws a vertical line passing through this point. It moves this vertical line to the right continuously.

At each instant, it finds the Pareto-optimal point on the vertical line, which is the lowest point on the line. It draws a horizontal line intersecting the vertical line at the Pareto-optimal point. The two lines divide the objective space into four disjoint quadrants: I, II, III, and IV. By the definition of Pareto optimality, quadrant I will be always empty. Also, by the definition of Pareto optimality, all points in quadrant III will be dominated by the current Pareto-optimal point and therefore must be removed, which is illustrated in Fig. 1(b). Quadrant IV will contain the partially built globally optimal Pareto front. Fig. 1(b) shows the snapshot of the objective space at the end of the instant.

The algorithm proceeds in this fashion, pruning the objective space at each instant. Fig. 1(b) and (c) show the explored and unexplored objective space at the end of two different instants in the algorithm's execution. The algorithm terminates in Fig. 1(d), where there are no points to explore in quadrant II. Quadrant IV contains the final globally optimal Pareto front. It is discontinuous with breaks. The algorithm fills the breaks with locally Pareto optimal solutions. Due to space limitations, we do not present how this filling is done in this paper.

We present in detail the formulation of the discrete optimization problem (HEPOPTGD) and an exact algorithm (HEPOPTAGD) solving HEPOPTGD in the supplemental (Section 8).

IV. Efficient Construction of Models of Parallel Applications for Processor Configurations Represented by DVFS State

Our proposed algorithms, HEPOPTAGC and HEPOPTAGD, require each processor's complete set of functional models to determine the global Pareto-optimal front. The set comprises pairs of execution time and dynamic energy functions of workload size for all processor configurations. Therefore, fast construction of this complete set is crucial to the practical efficiency of our proposed algorithms.

Here, we investigate the problem of efficiently constructing the complete set of models for a processor whose configuration is a DVFS state represented by a set of (voltage, frequency) pairs for independently powered processor components. Therefore, the problem involves efficiently building the pairs of execution time and dynamic energy functions for all the DVFS states of a processor. Since the proposed optimization algorithms rely on supplied execution-time and dynamic-energy functions, their sensitivity to modeling inaccuracies is discussed in the Supplemental Material.

We define a *DVFS state* of a processor to be a set, $\{(V_i, f_i)\}_{i=1}^{c}$, in the $V \times f : \mathbb{R}_{\geq 0} \times \mathbb{Z}_{\geq 0}$ space, where each pair (V_i, f_i) represents the operating voltage and frequency configuration for an independently powered processor component, represented by its index i. For example, the DVFS state of a multicore Intel CPU includes (V, f) pairs for each core, L3 cache, the main memory. For Nvidia GPUs, the DVFS state includes (V, f) pairs for each Streaming Multiprocessor (SM), the main memory, and the video encoder/decoder clock.

Our theoretical notion of *DVFS state* differs from its practical application instances. For example, only the frequencies of a CPU core in the DVFS states of a multicore processor are typically exposed to the application programmers. In addition, the latest generation Nvidia GPU processors allow setting only the frequencies of the graphics processor (SMs) and main memory in their DVFS states. Furthermore, the number of frequencies supported by the main memory is constrained to a few (only two in the case of the A40 GPU).

In this work, we consider only multicore CPU DVFS states where the frequencies of all the processor cores are the same, and for the Nvidia GPU processor, only DVFS states comprising the highest memory frequency and the graphics frequency (which is the same for all the SMs).

A. Scaling Solution for Processor Consisting of One Independently-Powered Processor Component

The fundamental building blocks of our scaling solution are workload-specific circuit-level Dennard scaling power model and workload-specific execution time model. First, we describe our circuit-level Dennard scaling power model.

The original Dennard scaling power model for a CMOS transistor is

$$P_{cmos} = C_{dyn} \times V^2 \times f,$$

where C_{dyn} is called the dynamic effective capacitance, V is the supply voltage, and f is the clock frequency. C_{dyn} is equal to the product of two factors, α and C_L, signifying a transistor's switching activity and load capacitance. The energy delivered to a CMOS circuit is used for charging and discharging the load capacitance C_L. In essence, C_L represents the CMOS gate-level activity during the execution of an application workload.

We postulate that an equivalent of C_{dyn} exists, which signifies the cumulative activity of CMOS gates during the execution of an application workload. In addition, we posit that the C_{dyn} equivalent is workload-specific and will be the same for different runs of the same workload without any non-deterministic interference from the OS. Our rationale behind this presumption is that the cumulative activity of CMOS gates should not change in this situation. Therefore, the C_{dyn} equivalent should deterministically capture the overall activity of CMOS gates for a workload. Furthermore, we posit that the C_{dyn} equivalent will not depend on the DVFS state and be the same constant for all DVFS states for a given workload since changing the supply voltage affects only the clock frequency and not C_{dyn} [20].

Based on the above, we propose the following circuit-level Dennard scaling power model, an extension of the original Dennard scaling model for the dynamic power consumption of a CMOS transistor:

$$P_{avg}(w, V, f) = C(w) \times V^2 \times f, \tag{1}$$

where $P_{avg}(w, V, f)$ is the average dynamic power consumption of the processor during the execution of the application workload size, w, V and f are the operating voltage and frequency of the processor component. $C(w)$ is the workload-specific dynamic effective capacitance and signifies the cumulative activity of CMOS gates during the execution of an application workload.

We perform extensive experiments with several different application workloads to validate our hypothesis. Section IX in the supplemental presents our results for workloads belonging to two different applications, 2D fast Fourier transform and matrix multiplication. The prediction accuracy of our proposed models is 100%. The experiments validate that $C(w)$ deterministically captures the overall activity of CMOS gates for a given workload w. We can clearly see from the experiments that $C(w)$ does only depend on w being the same constant for all DVFS states for a given workload w.

It is important to note that setting the frequency f in a DVFS state adjusts V automatically in the resulting (V, f) pair so that the product $V^2 \times f$ remains constant throughout the workload execution. Research works [21], [22], [23], [24], [25] report a linear correlation between the supply voltage and clock frequency of CMOS processors, which would allow us to remove variable V from model (1). However, the above research finding is not supported by our own exhaustive experiments on modern multicore CPUs, which found no functional correlation at all.

Next, we propose a workload-specific execution time model

$$t(w, V, f) = K(w) \times \frac{1}{f}, \tag{2}$$

where $K(w)$ is a workload-specific constant and f is the processor frequency. We postulate and validate that the parameter, $K(w)$, only depends on w and has the same value for all DVFS

states for a given workload w. Intuitively, $K(w)$ represents the number of clock cycles during the execution of the workload w, and $\frac{1}{f}$ is the duration of one clock cycle. Section IX of the supplemental material validates the proposed time model, showing 100% prediction accuracy for the two applications, and discusses the reproducibility conditions supporting the assumption that workload-specific parameters remain constant across DVFS states. Reproducibility is promoted by using the same workload distributions, fixed thread-to-core mapping, maximum fan speed to prevent thermal throttling, and controls that make execution as deterministic as possible. In particular, hybrid applications with frequent CPU–GPU synchronization or repeated CPU–GPU data movement during kernel iterations require workload-specific profiling and validation, because such communication patterns may change the execution-time and dynamic-energy profiles across DVFS states.

Now we can derive the dynamic energy model from the power model (1) and the time model (2) as follows:

$$\begin{aligned} E_d(w,V,f) &= P_{avg}(w,V,f) \times t(w,V,f) \\ &= C(w) \times V^2 \times f \times K(w) \times \frac{1}{f} \\ &= C(w) \times K(w) \times V^2 = C_1(w) \times V^2 \end{aligned} \tag{3}$$

The models (2) and (3) reveal an interesting relationship between the time and the dynamic energy consumption of the execution of an application by a processor and the DVFS state of the processor. Namely, the energy consumption only depends on operating voltage and does not depend on frequency, while the execution time only depends on frequency and does not depend on voltage. Thus, different components of the DVFS state govern different conflicting properties (performance or energy) of the application execution.

1) Execution Time and Dynamic Energy Scaling Models: Finally, from models (2) and (3), we can derive scaling models, relating the time and dynamic energy of the execution of a given workload w for a reference DVFS state (V_h, f_h) and those for an arbitrary DVFS state (V, F).

As

$$t(w,V_h,f_h) = K(w) \times \frac{1}{f_h}, \quad t(w,f) = K(w) \times \frac{1}{f},$$

then

$$t(w,V,f) = t(w,V_h,f_h) \times \frac{f_h}{f} \tag{4}$$

Similarly, as

$$E_d(w,V_h,f_h) = C_1(w) \times V_h^2, \quad E_d(w,V,f) = C_1(w) \times V^2,$$

then

$$E_d(w,V,f) = E_d(w,V_h,f_h) \times \left(\frac{V}{V_h}\right)^2 \tag{5}$$

2) Scaling Method to Determine the Complete Set of Time and Energy Profiles of a Processor: Now we are equipped to propose an efficient, accurate and portable scaling method, which builds the complete set of time and energy profiles for the processor consisting of one independently powered component using only one experimentally obtained time and energy profile. The method is based on the scaling models (4) - (5).

Let $\mathbb{VF}$ be the complete set of DVFS states for the processor, $\mathbb{VF} = \{(V_l, f_l), \cdots, (V_h, f_h)\}$, where f_l and f_h are the lowest and highest processor frequencies. Then the steps of the scaling method are as follows:

1) For the DVFS state (V_h, f_h), experimentally build the execution time and dynamic energy profiles, $t(w, V_h, f_h)$ and $E_d(w, V_h, f_h)$. Generally speaking, we can use any other state from $\mathbb{VF}$ at this step, but we choose to use (V_h, f_h) as it minimizes the execution time of the experiments running them at the highest frequency.
2) For each DVFS state $(V, f) \in \mathbb{VF} \setminus (V_h, f_h)$, calculate the execution time profile $t(w, V, f)$ using formula (4) and the dynamic energy profile $E_d(w, V, f)$ using formula (5).

B. Application of the Scaling Solution to Multicore CPU Processors

The proposed models and scaling method for a processor with one independently powered processor component can be directly applied to multithreaded workloads executing on modern multicore CPU processors with identical cores given the following constraints are satisfied.

First of all, the configuration of the workload should be fixed and be as deterministic as possible to guarantee the same operation of the hardware during different runs of the workload. In particular, the number of threads and their mapping to the cores should be fixed and do not change neither during the execution of the workload nor between different executions. If this constraint is not satisfied, the workload-specific constants in our models will not be constants anymore.

Second, the dynamic energy consumption of the independently powered uncore components, including the main memory, should be negligible in comparison with the dynamic energy consumption of the cores. This constraint is typically satisfied for mainstream multicore CPUs, including the Intel processors used in our experiments. This allows us to accurately model the CPU dynamic energy consumption by only considering the CPU cores.

Third, the dynamic power consumption for the reference DVFS state (V_h, f_h) must be high enough to be accurately and reliably measured by the ground-truth method.

Finally, we only consider DVFS states of a multicore CPU where all cores of the CPU operate at the same frequency. This allows us to represent an individual DVFS state by a single (V, f) pair, where f represents the frequency of all the cores in the processor. While limiting the space of processor configurations, this constraint is typically followed by the mainstream DVFS governors.

We experimentally validate our scaling models and method for an Intel multicore CPU processor. Section IX in the supplemental contains the results. The estimation accuracy of the

scaling method is close to 100% for the DGEMM workloads, in the range of 94-98% for the 2D-FFT workloads.

C. Scaling Solution for Processor With Two Independently-Powered Processor Components, Only One of Which is Scaled

While we can directly apply the scaling solution developed for a processor with one indepenedently powered component to mainstream multicore CPUs, it will not work for mainstream GPUs. The Nvidia A40 GPU processor (Table I in the supplemental) used in our experiments comprises a graphics component with 144 Streaming Multiprocessors (SMs) and a main memory (GDDR6 SRAM) component. Both components are independently powered with different supply voltages (V_g and V_{mem}). V_{mem} is 1.35 V and does not vary. The main memory supports two (V_{mem}, f_{mem}) combinations, where the frequencies are 7251 MHz and 405 MHz.

We observe that the dynamic power consumption of the main memory component for the (V_{mem}, f_{mem}) pair with f_{mem} equal to 405 MHz is very low, to the extent that it can not be measured reliably. Therefore, in the case of f_{mem}= 405 MHz, we can realistically ignore the main memory dynamic energy consumption and apply the scaling models and the scaling method developed for a processor with one independently powered component as we did for a CPU processor. However, when f_{mem} = 7251 MHz, the main memory dynamic energy consumption is significant and cannot be ignored. Therefore, to address this case, we need new average dynamic power and dynamic energy models that would account for two independently powered components of the processor. As the second component, the main GPU memory in our case, is always powered with the same voltage and frequency, we do not need V_{mem} and f_{mem} as variables in these new models, which therefore take the following form:

$$\begin{aligned} P_{avg}(w, V_g, f_g) &= I(w) + C_g(w) \times V_g^2 \times f_g \\ t(w, V_g, f_g) &= K_g(w) \times \frac{1}{f_g} \\ E_d(w, V_g, f_g) &= P_{avg}(w, V_g, f_g) \times t(w, f_g) \\ &= I(w) \times K_g(w) \times \frac{1}{f_g} \\ &\quad + C_g(w) \times K_g(w) \times V_g^2 \\ &= I_1(w) \times \frac{1}{f_g} + I_2(w) \times V_g^2 \end{aligned}$$

In the power model, the intercept $I(w)$ is a workload-specific constant representing the dynamic energy consumption of the second independently powered processor component. As its voltage and frequency are fixed, they do not appear as variables in the models, and the DVFS state of the processor is thus represented by the voltage, V_g, and frequency, f_g, of the first component only.

We perform experiments with several application workloads executing on GPUs to validate these models. Section IX.C in the supplemental presents the results. The plots clearly demonstrate that the execution time is linearly proportional to the reciprocal of the CUDA core frequency. These models reveal that the processor with two independently powered components is qualitatively different from the processor consisting of one independently powered component, even in the simple case when the voltage and frequency of the second component are the same for all DVFS states of the processor. Namely, the dynamic energy consumption of the two-component processor will depend not on voltage only as for the single-component processor but on both voltage and frequency of its first component, making it more complex for performance/energy management.

In the supplemental (Section IX), we derive the scaling models relating the time and dynamic energy of the execution of a given workload w for two reference DVFS states (V_{g1}, f_{g1}) and (V_{g2}, f_{g2}), and those for an arbitrary DVFS state (V_g, f_g). Unlike the scaling method for a one-component processor, the method for a two-component processor needs not one but two experimentally obtained energy profiles to build the complete set of time and energy profiles. It stll uses one time profile though. We employ the scaling models and method to multithreaded workloads executing on a Nvidia A40 GPU processor.

V. Experimental Results

We demonstrate the efficiency and accuracy of our solution in optimizing two heterogeneous hybrid applications, double-precision 2D fast Fourier transform and mixed precision matrix multiplication, on a heterogeneous platform comprising an Intel Icelake multicore CPU and two identical Nvidia A40 GPUs. The Icelake multicore CPU (Intel Platinum 8362) has a single socket comprising 32 physical cores (and two logical cores per physical core). Its main memory is 64 GB DDR4-3200, and its thermal design power (TDP) is 265 W. The Nvidia A40 GPU comprises 10,752 Ampere cores, a total board memory of 48 GB GDDR6 (with ECC) and a TDP of 300 W.

We use the Likwid tool [26] to collect voltage readings from the logical CPU cores on our heterogeneous platforms. Likwid is a well-established performance profiling and monitoring tool that provides access to low-level hardware events, including cache misses, branch mispredictions, and floating-point operations. The voltage readings are obtained through the performance event labeled *VOLTAGE_CORE*, which is part of a performance monitoring group known as *VTG0*.

A hybrid parallel application comprises several software components (kernels) executing in parallel. A one-to-one mapping exists between the software components and computing devices of the hybrid platform on which the application is executed. Executing an accelerator component involves a dedicated CPU core, running the hosting thread, and the accelerator itself, performing the accelerator code. The execution of the accelerator component includes data transfer between the CPU and accelerator memory, computations by the accelerator code, and data transfer between the accelerator memory and CPU. Executing a CPU component involves only the CPU cores performing the multithreaded CPU code. This design is implemented using OpenH [27] where the main thread of the program creates a group of hosting Pthreads responsible for leading the execution of the CPU and accelerator software components in

parallel. The CPU hosting Pthread leads the execution of a multithreaded CPU software component, while an accelerator hosting Pthread leads the execution of an accelerator component. Finally, OpenH provides API that allows programmers to discover the configuration of the executing hybrid server and bind the hosting Pthreads to the CPU cores to get the best performance.

There are three software components in each hybrid application employed in this work, CPU, A40 GPU_1, and A40 GPU_2. CPU comprises 62 (out of a total of 64) logical CPU cores. A40 GPU_1 symbolizes the first Nvidia A40 GPU and a host CPU core connected to this GPU via a dedicated PCI-E link. A40 GPU_2 denotes the second Nvidia A40 GPU and a host CPU core connected to this GPU via a dedicated PCI-E link.

We experiment with 27 DVFS states for the CPU processor where a DVFS state comprises a (V_{core}, f_{core}) pair for 64 processor cores. Therefore, all the cores are set to the same processor frequency f_{core} in each DVFS state. The range of f is {3.6Ghz, 3.5Ghz, . . ., 1.0GHz}. Furthermore, the (V, f) pair for the main memory is fixed and only the (V_{core}, f_{core}) pair for the processor cores is varied.

For both HFFT and HGEMM applications, we employ 7 DVFS states for a A40 GPU where a DVFS state comprises a (V_{gra}, f_{gra}) pair for graphics and a (V_{mem}, f_{mem}) pair for memory. For the HFFT application, we keep f_{mem} constant at 1740 MHz and vary f_{gra} in the range {1740 MHz, 1650 MHz,. . ., 1005 MHz}. For the HGEMM application, we keep f_{mem} constant at 1740 MHz and vary f_{gra} in the range {1530 MHz, 1470 MHz, . . ., 1005 MHz}. Furthermore, we also fix the (V, f) pair for the video encoder/decoder clock. Therefore, there are 1323 (27 * 7 * 7) DVFS state combinations in total to consider in our heterogeneous platform since the two A40 GPUs are identical.

The statistical methodology to measure the execution time and energy is presented in the supplemental (Section VI). In brief, a sample mean for a response variable (execution time and energy) is obtained from multiple experimental runs. The sample mean is calculated by executing the application repeatedly until it lies in the 95% confidence interval and a precision of 0.05 (5%) is achieved. For this purpose, Student's t-test is used, assuming that the individual observations are independent and their population follows the normal distribution. We verify the validity of these assumptions using Pearson's chi-squared test.

A. Experimental Results for HFFT

The hybrid parallel 2D fast Fourier Transform application (HFFT) computes the 2D-FFT of a signal matrix S of size $N \times N$. Fig. 10(a) in the supplemental illustrates the application. The application comprises eight stages. The first stage consists of data transfers of P_2 from the host CPU to A40 GPU_1 and P_3 from the host CPU to A40 GPU_2. The second stage involves local computations in the software components. The computations in the software component involving the multicore CPU are performed using the Intel MKL FFT library routine. The computations in the software components involving the A40 GPUs are performed using the CUFFT library routine. The third stage involves data transfer of the result matrices P_2 and P_3 from A40 GPU_2 and A40 GPU_1 to the host CPU. The fourth stage involves transpose of the matrix S that is performed on the CPU side. The four stages are repeated again.

The complete set of functional models (pair of execution time and dynamic energy functions) for the processors are built using the processors' scaling methods. Each profile is a function of workload size representing the number of 1D-FFTs (x) of size N in the 2D-FFT of a signal matrix, $x \times N$ where N is 30720 and x varies from 1024 to 30720 in steps of 1024. Therefore, the workload size is $5 \times x \times N \times \log_2(N)$ and proportional to x.

For the CPU processor, the execution time and dynamic energy profiles corresponding to the processor core frequency of 3.6 GHz are built experimentally. The functional models for the other frequencies ({3.5 GHz, . . ., 1.0 GHz}) are estimated using the scaling method. For the A40 GPU processor, the execution time and dynamic energy profiles corresponding to the graphics frequencies of 1740 MHz and 1530 MHz are built experimentally. The functional models for the other frequencies are estimated using the scaling method. The matrix transposition performed on the CPU side has insignificant execution time and dynamic energy consumption and, hence, is ignored in the CPU processor's functional modelling.

1) Application of HEPOPTAGD: The complete execution time and dynamic energy profiles of the GPU processor are not linear functions of workload size. They are given in Section X in the supplemental. Therefore, we employ the discrete optimization algorithm (HEPOPTAGD) to determine the discrete globally optimal Pareto front.

HEPOPTAGD determines a locally optimal Pareto front for each DVFS state combination [Fig. 2(a)]. Note a Pareto front is a set of points. In the figure, we join the points by a line to highlight the slope and distinguish it from the other fronts. The total number of locally optimal Pareto fronts is 1323, and the total number of points in the fronts is 4329. The total number of points in the globally optimal Pareto front [Fig. 2(b)] output by HEPOPTAGD is 22.

2) Application of HEPOPTAGC: To illustrate the application of HEPOPTAGC to solve the continuous optimization problem for HFFT application for performance and total energy, we consider partial dynamic energy and execution time profiles that are linear. We approximate the linear profiles by linear functions of workload size. Section X in the supplemental shows the dynamic energy and execution time functions of the two heterogeneous hybrid applications that are input to the continuous algorithm. We solve a problem that computes 15360 1D FFTs of size 30720. Fig. 2(d) shows the globally optimal Pareto front.

HEPOPTAGC determines a locally optimal Pareto front for each DVFS state combination. The Pareto front is piecewise linear. For three heterogeneous processors, the front comprises two linear segments. Therefore, if the DVFS state combination contains different graphics frequencies for the two A40 GPU processors, the A40 GPUs would be heterogeneous, and the Pareto

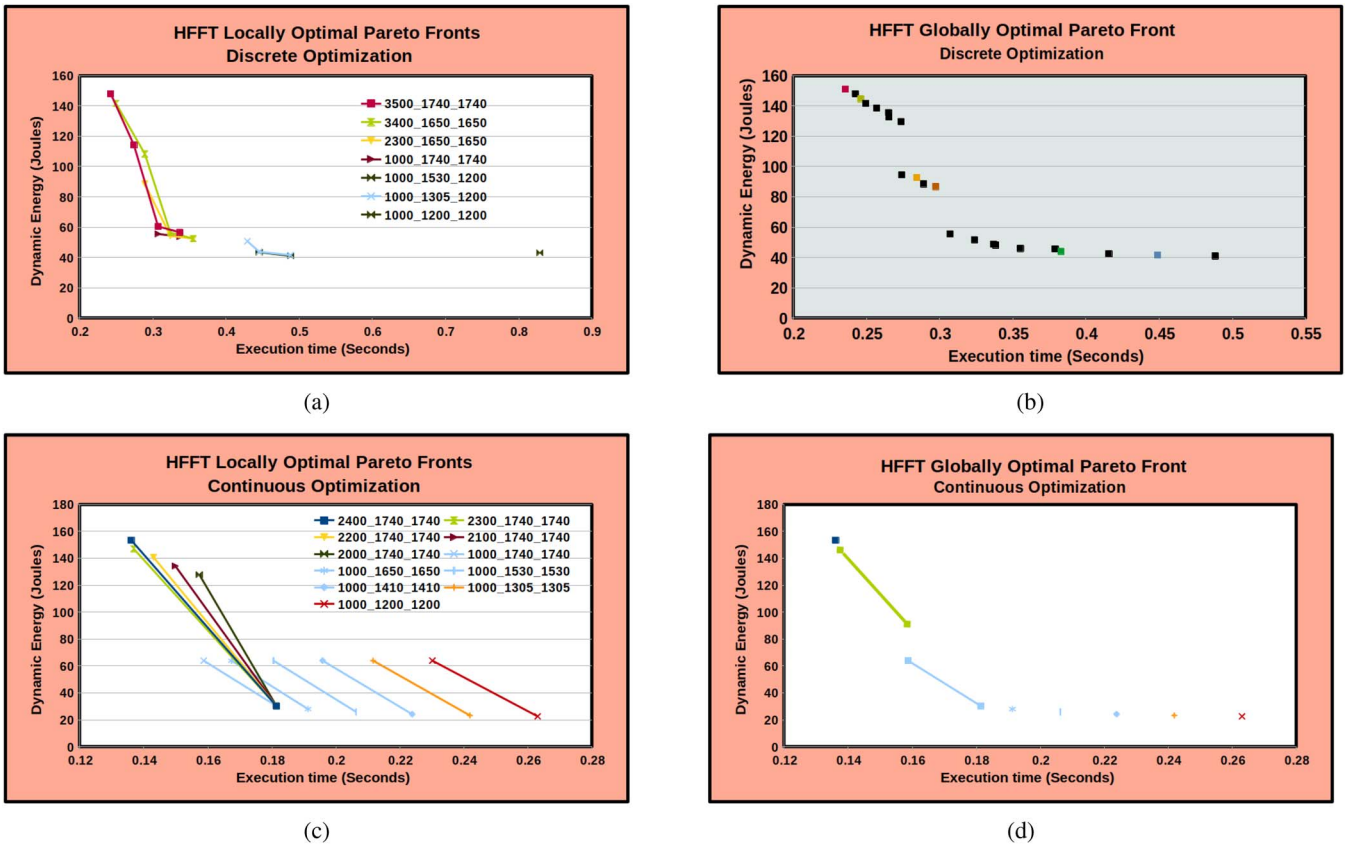


Fig. 2. (a) and (b) Application of HEPOPTAGD to solve the **discrete optimization problem** for the HFFT application. (a) Locally optimal Pareto fronts for optimal platform combinations (frequencies separated by '-') obtained during the execution of HEPOPTGD. We join the points in a Pareto front by a line to highlight the slope and distinguish it from the other fronts. (b) The globally optimal Pareto front output by HEPOPTGD. The signal matrix size solved is **30720 x 30720**. (c) and (d) Application of HEPOPTAGC to solve the **continuous optimization problem** for the HFFT application: (c) locally optimal Pareto fronts for optimal platform combinations obtained during the execution of HEPOPTGC; (d) the globally optimal Pareto front output by HEPOPTGC. The problem solved is 15360 1D FFTs of size 30720.

front will contain two linear segments. However, the Pareto front comprises only one linear segment if the DVFS state combination contains the same graphics frequencies for the two A40 GPU processors, rendering them identical. Fig. 2(c) shows the locally optimal Pareto fronts, some of whose points are globally optimal and end up on the globally optimal Pareto front.

HEPOPTAGC invokes our proposed continuous algorithm (Section III-B) to merge the piecewise linear locally optimal Pareto fronts into the globally optimal Pareto front. Fig. 2(d) shows the globally optimal Pareto front.

B. Experimental Results for HGEMM

The hybrid parallel mixed precision matrix multiplication application (HGEMM) computes the matrix product ($C+=A\times B$) of two dense square matrices A and B of size $N\times N$. Fig. 10(b) in the supplemental illustrates the application. The application comprises three main stages. The first stage consists of data transfers of A_2, B, and C_2 from the host CPU to A40 GPU_1 and A_3, B, and C_3 from the host CPU to A40 GPU_2. The second stage involves local computations in the software components. The computations in the CPU software component are performed using the Intel MKL DGEMM (double-precision floating point) library routine. The computations in the software components involving the A40 GPUs are accomplished using the CUBLAS SGEMM (single-precision floating point) library routine.

The original plan was to use vendor-optimized DGEMM kernels on all devices. However, we found out that CUBLAS DGEMM performs very slowly on A40 GPUs (up to 55 times slower than CUBLAS SGEMM) and consumes negligible dynamic power, which is so low that cannot be reliably measured using the power meter. This makes the hybrid GEMM employing CUBLAS DGEMM an impractical degenerative case, not representative of performance and power characteristics of realistic hybrid applications. Therefore, we decided to use CUBLAS SGEMM instead, resulting in a mixed-precision GEMM.

Therefore, the matrix data (A_i, B and C_i, $i=\{1,2\}$) is copied from a double-precision floating-point buffer to a single-precision floating-point buffer on the CPU side, which is transferred to the GPU for the computations. The third stage involves data transfer of the result matrices C_2 and C_3 from A40 GPU_2 and A40 GPU_1 to the host CPU. The single-precision floating-point output from the GPU is copied to the double-precision

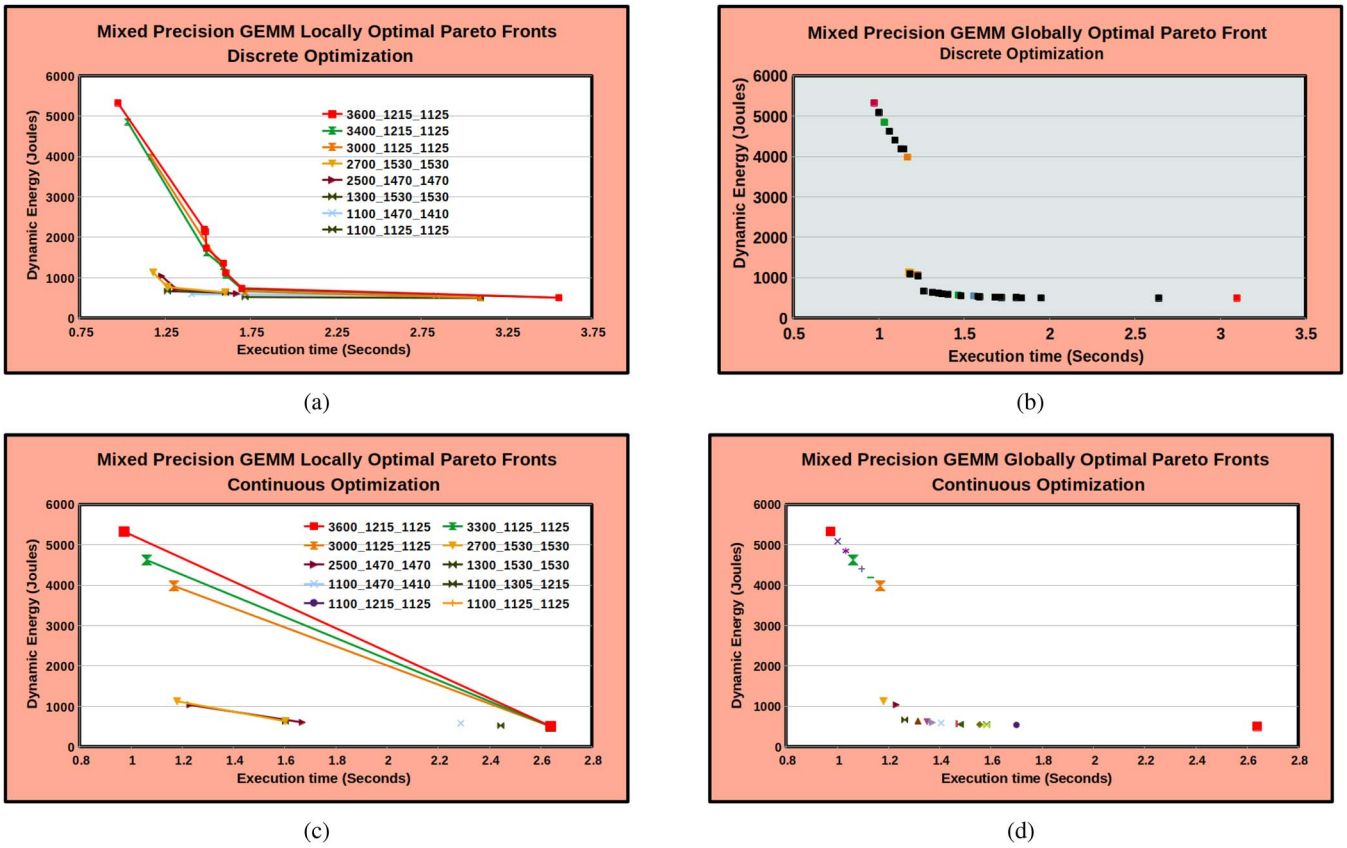


Fig. 3. (a) and (b) Application of HEPOPTAGD to solve the **discrete optimization problem** for the HGEMM application. (c) and (d) Application of HEPOPTAGC to solve the **continuous optimization problem** for the HGEMM application. The matrix size ($N \times N$) solved is **32768 x 32768.**

floating-point buffer on the CPU side. Note that the execution time and dynamic energy functional models include the time and energy of these data copies.

The processors' complete set of functional models are built using the processors' scaling methods. Each profile is a function of workload size representing the matrix product of $x \times N$ and $N \times N$ matrices where N is 32768 and x varies from 1024 to 32768 in steps of 1024. Therefore, the workload size is proportional to $x \times N^2$ and hence x.

For the CPU processor, the execution time and dynamic energy profiles corresponding to the processor core frequency of 3.6 GHz are built experimentally. The functional models for the other frequencies ($\{3.5$ GHz, . . ., 1.0 GHz$\}$) are estimated using the scaling method. For the A40 GPU processor, the execution time and dynamic energy profiles corresponding to the graphics frequencies of 1530 MHz and 1470 MHz are built experimentally. The functional models for the other frequencies are estimated using the scaling method.

HEPOPTAGD takes input the complete execution time and dynamic energy profiles of the CPU and GPU processors (provided in Section X in the supplemental). The CPU profiles are not linear functions of workload size whereas the GPU profiles are. It determines the globally optimal Pareto front shown in Fig. 3(b) that contains 30 points. Furthermore, a 20% performance degradation (increase of execution time by 0.19 seconds) results in dynamic energy savings of 34% (decrease in energy consumption by 1344 J). The energy-optimal solution gives dynamic energy reduction of 9.7-fold (decrease in energy consumption by 4828 J) while resulting in 2.2-fold performance degradation.

To illustrate the application of HEPOPTAGC, we input linear dynamic energy and execution time functions. The functions are linear approximations of partial profiles (Section X in the supplemental). Fig. 3(d) shows the globally optimal Pareto front. Note each Pareto front is one linear segment. However, some Pareto fronts are represented by single points since the endpoints of the linear segment do not have a statistically significant difference in dynamic energy.

C. Brief Discussion of Experimental Results

Here we summarize the most interesting observations:

- While the nominal peak performance of A40 GPU (only considering CUDA cores) is 10x higher than that of CPU, the effective relative speed of GPU and CPU FFT and GEMM kernels is much more moderate. For example, it will be equal to 1.3x for FFT and 3.3x for GEMM kernels processing matrices of dimension 20000 at the highest studied frequencies (Figs. 11 and 13 in the supplemental). One of the reasons is that in contrast to CPU kernels, execution of GPU kernels includes costly data transfers between memories of CPU and GPU.
- All globally Pareto-optimal solutions for both applications involve in computations all three computing devices of our experimental server.

- The shapes of the global Pareto fronts offer a wide range of very interesting optimization opportunities. For example, for HGEMM (Fig. 3), at the expense of a minor increase in execution time from 1.167 s to 1.18 s (1% increase) the user can decrease the energy consumption from 3979 J to 1127 J (3.5x decrease). Alternatively, at the expense of a 12% increase in energy consumption from 495 J to 553 J, its execution time can be decreased more than twice, from 3.1 s to 1.48 s.
- For performance optimal HFFT (the left endpoint of the Pareto front) all devices operate at their highest frequency levels. GPUs are given equal workloads, which is bigger than the workload executed by CPU. In contrast, for performance optimal HGEMM only CPU operates at the highest frequency while GPUs operate at one of the lowest frequency levels. Also, GPUs are given different workloads, each of which is less than CPU's workload.
- For both applications, as we move along the Pareto front from the left endpoint to the right one (representing energy-optimal solution), CPU's frequency is monotonically decreasing from the highest level to the lowest one. There is no such monotony for GPUs, although they also operate at the lowest frequency for the energy-optimal solution. CPU's workload is also monotonically decreasing, while there is no particular pattern in distribution of the remaining workload between the GPUs.

VI. Related Work

Application-level Bi-objective Optimization on Multicore CPUs for Performance and Energy: We first review research works proposing solution methods solving optimization problems that determine the optimal application configuration for performance and energy with the platform configuration set to one energy governor throughout the application execution.

Reddy and Lastovetsky [4] solve the discrete bi-objective optimization problem of data-parallel applications on homogeneous clusters of modern multicore CPUs for performance and energy. Khaleghzadeh et al. [5] solve the discrete bi-objective optimization problems of data-parallel applications on heterogeneous processors for performance and energy. Khaleghzadeh et al. [19] study the continuous bi-objective optimization problem of data-parallel applications on heterogeneous HPC platforms for performance and energy where the performance functions are continuous and strictly increasing and energy functions are linear increasing. The decision variable in all three works is workload distribution.

The research works below propose methods that optimize the execution of an application for a fixed application configuration on a platform by employing DVFS during the slack phases in the application.

Kappiah et al. [10] move the nodes that arrive early and wait at synchronization points to lower DVFS states to reduce energy consumption in MPI applications. Lim et al. [13] set the CPU processors to lower DVFS states in the MPI runtime system during MPI communications to reduce energy consumption while incurring a modest performance penalty. Rountree et al. [14] also employ lower DVFS states to reduce the energy consumption of MPI applications during slack phases.

Application-level Bi-objective Optimization on GPUs for Performance and Energy: Ge et al. [28] study the effect of DVFS states on the performance and energy consumption of scientific workloads executing on Nvidia K20c GPU. Lee et al. [29] employ DVFS to improve the performance of applications on Nvidia GPUs under an energy constraint/budget. Tang et al. [30] investigate the impact of DVFS on the performance and energy consumption of deep learning workloads on Nvidia GPUs. Note that the application configuration is kept constant in the research works above, while the platform configuration represented by a DVFS state is varied.

System-level Bi-objective Optimization for Performance and Energy: Research works here deal with optimization problems that determine the optimal platform configuration represented by a DVFS state during an application execution where the application configuration is fixed.

Freeh et al. [31] analyze the performance-energy trade-offs of serial and parallel applications on a cluster of DVFS-capable AMD nodes. The decision variable is a DVFS state. Rountree et al. [14] propose a runtime system that employs DVFS during predicted slack periods to achieve significant energy savings while incurring negligible performance degradation. Lee et al. [29] propose energy-conscious scheduling heuristics that employ DVFS for bi-objective optimization of parallel applications on HPC platforms. Sharma and Moulik [32] propose HEAT, a heterogeneous multicore real-time scheduler that employs task clustering, temperature-aware scheduling, and DVFS-based energy-aware scheduling to improve energy and thermal management for periodic tasks.

Fard et al. [1] present a four-objective case study comprising performance, economic cost, energy consumption, and reliability for optimization of scientific workflows in heterogeneous computing environments. The decision variable is the task assignment or mapping. Kessaci et al. [9] present a multi-objective genetic algorithm that minimizes energy consumption and CO2 emissions and maximizes the generated profit of a cloud computing infrastructure. The decision variable is the arrival rate of applications. Kolodziej et al. [2] propose multi-objective genetic algorithms for bi-objective optimization on green grid clusters and clouds for performance and energy. The decision variable is a DVFS state.

Energy Predictive Models Employing Operating Voltage and Frequency: Research work [33] models the power consumption of a multicore processor based on CPU, memory, and network utilization. The CPU component power consumption model employs the Dennard scaling power model using the cores' average operating voltage and frequency at maximum utilization. Gamell et al. [34] propose a platform-level energy predictive model where the CPU energy component is also based on the Dennard scaling power model using the CPU cores' average operating voltage and frequency. The effective capacitance in both works is a platform-level constant. However, we show that the CPU utilization is unsuitable for energy modelling on modern multicore processors since the dynamic power consumption exhibits a non-functional

relationship against utilization [35]. Furthermore, the effective capacitance used in our models is a workload-specific constant. We demonstrate in our experiments that a platform-level constant for effective capacitance does not exist.

Finally, we are the first research work that proposes an energy predictive model for GPUs based on the Dennard scaling power model employing graphics' operating voltage and frequency.

Overall, prior studies mainly optimize one side of the problem, either the application configuration or the platform configuration. In contrast, this work optimizes workload distribution and processor configuration vectors, yielding globally Pareto-optimal application-platform configurations.

VII. Conclusion

In this work, we proposed to jointly optimize the operating voltage and frequency of the heterogeneous hybrid parallel computer and the workload distribution of the application across its processors to minimize the execution time and energy consumption. We mathematically formulated the corresponding bi-objective optimization problems and proposed their exact solutions.

The proposed solutions require the construction of the time and energy profiles of the execution of the application's workload on each processor of the computer for each operating voltage/frequency pair. The state of the art suggests to experimentally build these profiles for all such pairs. The second major contribution of this work is a methodology that depending on the type of the processor (CPU or GPU) only requires experiments for one or two reference voltage/frequency pair and calculates profiles for remaining pairs using the proposed analytical scaling models. These scaling models were derived from postulated and validated circuit-level Dennard scaling power model and scaling time model.

Finally, we have demonstrated the real-world efficiency and accuracy of our solutions.

References

[1] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," in *Proc. CCGRID*, 2012, pp. 300–309.

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

[3] A. Lastovetsky and R. Reddy, "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.

[4] R. Reddy and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy," *IEEE Trans. Comput.*, vol. 64, no. 2, pp. 160–177, Feb. 2018.

[5] 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.

[6] 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. 2025.

[7] N. Gholkar, F. Mueller, and B. Rountree, "Power tuning HPC jobs on power-constrained systems," in *Proc. PACT*. New York, NY, USA: ACM, 2016, pp. 179–191.

[8] B. Rountree et al., "Bounding energy consumption in large-scale MPI programs," in *Proc. SC*, Nov. 2007, pp. 1–9.

[9] 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.

[10] N. Kappiah, V. Freeh, and D. Lowenthal, "Just in time dynamic voltage scaling: exploiting inter-node slack to save energy in MPI programs," in *Proc. SC*, 2005, p. 33.

[11] G. Chen, K. Malkowski, M. Kandemir, and P. Raghavan, "Reducing power with performance constraints for parallel sparse applications," in *Proc. IPDPS*, 2005, p. 8.

[12] L. Yan, L. Zhong, and N. K. Jha, "User-perceived latency driven voltage scaling for interactive applications," in *Proc. 42nd Annu. Des. Automat. Conf.*, New York, NY, USA: ACM, 2005, pp. 624–627.

[13] M. Y. Lim, V. W. Freeh, and D. K, Lowenthal, "Adaptive, transparent frequency and voltage scaling of communication phases in MPI programs," in *Proc. SC*, 2006, p. 14.

[14] B. Rountree et al., "Adagio: Making DVS practical for complex HPC applications," in *Proc. ICS*. New York, NY, USA: ACM, 2009, pp. 460–469.

[15] M. Fahad, A. Shahid, R. R, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, pp. 1–14, 2019.

[16] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar. 2012.

[17] A. Shahid, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "Improving the accuracy of energy predictive models for multicore CPUs by combining utilization and performance events model variables," *J. Parallel Distrib. Comput.*, vol. 151, pp. 38–51, May 2021.

[18] S. Mazzola, T. Benz, B. Forsberg, and L. Benini, "A data-driven approach to lightweight DVFS-Aware counter-based power modeling for heterogeneous platforms," in *Embedded Computer Systems: Architectures, Modeling, and Simulation.* Berlin, Germany: Springer, 2022.

[19] H. Khaleghzadeh, R. Reddy 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 Computation: Pract. Experience*, vol. 35, no. 20, 2023, Art. no. e7285.

[20] T. D. Burd and R. W. Brodersen, "Energy efficient CMOS microprocessor design," in *Proc. HICSS.* Piscataway, NJ, USA: IEEE Press, 1995, vol. 1, pp. 288–297.

[21] J. Leng, A. Buyuktosunoglu, R. Bertran, P. Bose, and V. J. Reddi, "Safe limits on voltage reduction efficiency in GPUs: A direct measurement approach," in *Proc. 48th Int. Symp. Microarchit.*, 2015, pp. 294–307.

[22] M. Gupta, L. Bhargava, and I. Sreedevi, "Dynamic voltage frequency scaling in multi-core systems using adaptive regression model," in *Proc. 4th Int. Conf. I-SMAC*. Piscataway, NJ, USA: IEEE Press, 2020, pp. 1201–1206.

[23] S. Liu and A. Karanth, "Dynamic voltage and frequency scaling to improve energy-efficiency of hardware accelerators," in *Proc. HiPC*. Piscataway, NJ, USA: IEEE Press, 2021, pp. 232–241.

[24] J. Mao, X. Peng, T. Cao, T. Bhattacharya, and X. Qin, "A frequency-aware management strategy for virtual machines in DVFS-enabled clouds," *Sustain. Comput.: Inform. Syst.*, vol. 33, 2022, Art. no. 100643.

[25] P. Bratek, L. Szustak, R. Wyrzykowski, and T. Olas, "Reducing energy consumption using heterogeneous voltage frequency scaling of data-parallel applications for multicore systems," *J. Parallel Distrib. Comput.*, vol. 175, pp. 121–133, May 2023.

[26] J. Treibig, G. Hager, and G. Wellein, "LIKWID: A lightweight performance-oriented tool suite for x86 multicore environments," in *Proc. 39th Int. Conf. Parallel Process. Workshops*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 207–216.

[27] S. Farrelly, R. R. Manumachu, and A. Lastovetsky, "OpenH: A novel programming model and API for developing portable parallel programs on heterogeneous hybrid servers," *IEEE Access*, vol. 12, pp. 23666–23694, 2024.

[28] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a K20 GPU," in *Proc. ICPP*, 2013, pp. 826–833.

[29] J. Lee, V. Sathisha, M. Schulte, K. Compton, and N. S. Kim, "Improving throughput of power-constrained GPUs using dynamic voltage/frequency and core scaling," in *Proc. PACT*. Piscataway, NJ, USA: IEEE Press, 2011, pp. 111–120.
[30] Z. Tang, Y. Wang, Q. Wang, and X. Chu, "The impact of GPU DVFS on the energy and performance of deep learning: an empirical study," in *Proc. 10th ACM Int. Conf. Future Energy Syst.*, 2019, pp. 315–325.
[31] V. W. Freeh et al., "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.
[32] Y. Sharma and S. Moulik, "HEAT: A heterogeneous multicore real-time scheduler with efficient energy and temperature management," *ACM SIGAPP Appl. Comput. Rev.*, vol. 22, no. 2, pp. 34–43, 2022.
[33] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, and G. Giuliani, "A methodology to predict the power consumption of servers in data centres," in *Proc. 2nd Int. Conf. Energy-Efficient Comput. Netw.* New York, NY, USA: ACM, 2011.
[34] M. Gamell et al., "Exploring power behaviors and trade-offs of in-situ data analytics," in *Proc. SC*. New York, NY, USA: ACM, 2013.
[35] R. R. Manumachu and A. Lastovetsky, "On energy nonproportionality of CPUs and GPUs," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, 2022, pp. 34–44.



Hesam Nejati Sharifaldin (Member, IEEE) received the master's degree in software engineering from the Islamic Azad University of Mashhad, Iran, in 2017. He is currently working toward the Ph.D. degree with the School of Computer Science, University College Dublin. His research interests include parallel computing, high-performance computing, and energy-efficient computing.



Sourodip Ghoshdastidar received the B.E. degree from Jadavpur University, India, in 2015, and the M.Tech. degree from the Indian Institute of Science, Bangalore, in 2023. He is currently working toward the Ph.D. degree with the School of Computer Science, University College Dublin. His research interests include high performance computing, energy-efficient computing, and molecular dynamics simulations.



Ravi Reddy Manumachu (Member, IEEE) received the B.Tech. degree from I.I.T. Madras in 1997, and the Ph.D. degree from the School of Computer Science, University College Dublin (UCD) in 2005. He is currently an Assistant Professor with the School of Computer Science, UCD. His research interests include high performance heterogeneous computing and energy-efficient computing.



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