Bi-Objective Optimization of Data-Parallel Applications on Heterogeneous HPC Platforms for Performance and Energy Through Workload Distribution

Hamidreza Khaleghzadeh, Muhammad Fahad, Arsalan Shahid, Ravi Reddy Manumachu, and Alexey Lastovetsky, *Member, IEEE*

Abstract—Performance and energy are the two most important objectives for optimization on modern parallel platforms. In this article, we show that moving from single-objective optimization for performance or energy to their bi-objective optimization on heterogeneous processors results in a tremendous increase in the number of optimal solutions (workload distributions) even for the simple case of linear performance and energy profiles. We then study full performance and energy profiles of two real-life data-parallel applications and find that they exhibit shapes that are non-linear and complex enough to prevent good approximation of them as analytical functions for input to exact algorithms or optimization software for determining the Pareto front. We, therefore, propose a solution method solving the bi-objective optimization problem on heterogeneous processors. The method's novel component is an efficient and exact global optimization algorithm that takes as an input performance and energy profiles as arbitrary discrete functions of workload size, which accurately and realistically take into account resource contention and NUMA inherent in modern parallel platforms, and returns the Pareto-optimal solutions (generally speaking, load imbalanced). To construct the input discrete energy functions, the method employs a methodology that accurately models the energy consumption by a hybrid data-parallel application executing on a heterogeneous HPC platform containing different computing devices using system-level power measurements provided by power meters. We experimentally analyse the proposed solution method using three data-parallel applications, matrix multiplication, 2D fast Fourier transform (2D-FFT), and gene sequencing, on two connected heterogeneous servers consisting of multicore CPUs, GPUs, and Intel Xeon Phi. We show that it determines a superior Pareto front containing the best load balanced solutions and all the load imbalanced solutions that are ignored by *load balancing* methods.

Index Terms—Heterogeneous platforms, data-parallel applications, workload partitioning, performance optimization, energy optimization, bi-objective optimization, workload distribution, multicore CPU, GPU, Intel Xeon Phi

1 Introduction

Performance and energy are the two most important objectives for optimization on modern parallel platforms such as supercomputers, heterogeneous HPC clusters, and cloud computing infrastructures ([1], [2], [3], [4]).

State-of-the-art solutions for the bi-objective optimization problem for performance and energy on heterogeneous HPC platforms can be broadly classified into *system-level* and *application-level* categories. System-level solution methods aim to optimize the performance and energy of the environment where the applications are executed. The methods employ application-agnostic models and hardware parameters as decision variables. The dominant decision variable in this category is Dynamic Voltage and Frequency Scaling (DVFS). The majority of the works in this category can be further grouped as follows: a). Methods optimizing for performance under a power cap constraint (or energy budget) or optimizing for energy under an execution time constraint [5], [6], [7]. They determine a partial Pareto front by applying the power cap or an execution time constraint and then selecting the best configuration to fulfill a user-specific criterion. b). Methods solving unconstrained bi-objective optimization for performance and energy (with no time or energy constraints) [2], [3], [8]. They build the full Pareto front.

The application-level solution methods proposed in [9], [10], [11], [12], [13], [14] use application-level parameters as decision variables and application-level models for predicting the performance and energy consumption of applications to solve the bi-objective optimization problem. The application-level parameters include the number of threads, number of processors, loop tile size, and workload distribution. The methods in [9], [10], [11] do not consider workload distribution as a decision variable. The methods proposed in [13], [14] demonstrate by executing real-life data-parallel applications on modern multicore CPUs that the functional relationships between performance and workload size and

• *The authors are with the School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland. E-mail: {hamidreza.khaleghzadeh, arsalan.shahid, ravi.manumachu, alexey.lastovetsky}@ucd.ie, muhammad.fahad@ucdconnect.ie.*

Manuscript received 19 Mar. 2019; revised 21 Aug. 2020; accepted 24 Sept. 2020. Date of publication 28 Sept. 2020; date of current version 8 Oct. 2020. (Corresponding author: Ravi Reddy Manumachu.) Recommended for acceptance by R. Tolosana. Digital Object Identifier no. 10.1109/TPDS.2020.3027338

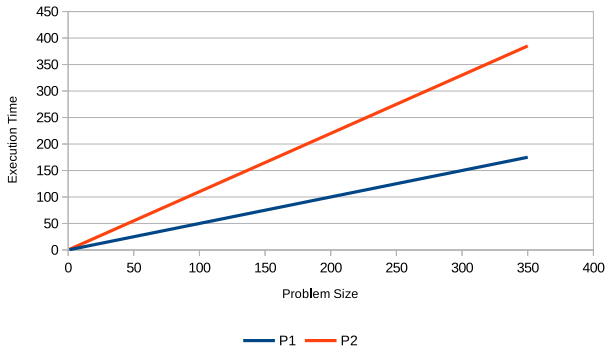

Fig. 1. Linear execution time functions of the processors P_1 and P_2.

between energy and workload size have complex (non-linear) properties and show that workload distribution has become an important decision variable that can no longer be ignored. The methods target homogeneous HPC platforms. Research work [12] considers the effect of heterogeneous workload distribution on bi-objective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters. The performance is represented by a linear function of problem size and the energy is predicted using historical data tables.

In this work, we study the bi-objective optimization problem for data-parallel applications for performance and energy on *heterogeneous* HPC systems. The problem aims to optimize the parallel execution of a given workload of n by a set of p heterogeneous processors. It has one decision variable, the *workload distribution*. The motivation for the study comes from our observation of the effect of heterogeneity on the solution space as we move from single-objective optimization for performance or energy to bi-objective optimization for performance and energy for the simple case where the execution time and energy functions are linear.

Consider two processors P_1 and P_2, whose linear execution time and energy functions are shown in the Figs. 1 and 2. The functions are real-life profiles of a data-parallel matrix multiplication application executed using a single core of a multicore CPU. For a given input workload size n, an algorithm, which we designed for the case of linear time and energy profiles and which is detailed in the supplemental, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2020.3027338, determines the Pareto front (distributions (x_1, x_2) of workload n where $x_1 + x_2 = n$). The solution for single-objective optimization for performance (minimizing the execution time of

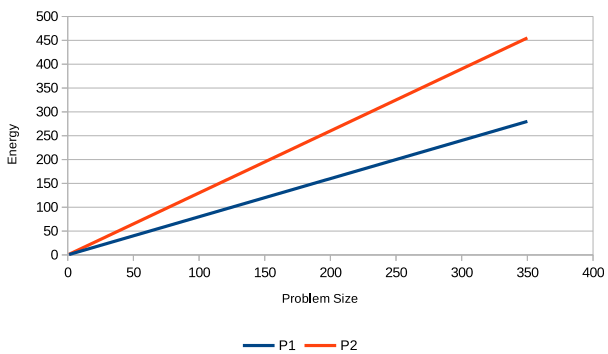

Fig. 2. Linear energy functions of the processors P_1 and P_2.

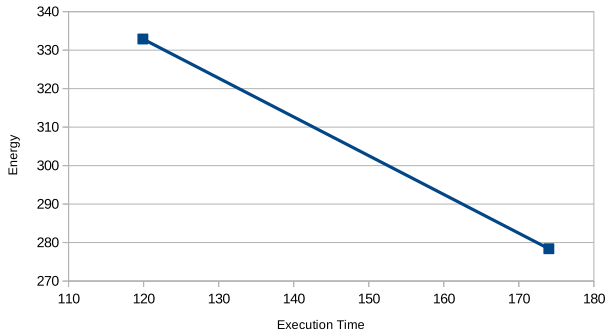

Fig. 3. Pareto-optimal solutions for a workload size 348. The front is linear. The end points are the solutions for single-objective optimization for performance and for energy.

computations during the parallel execution of the workload) is the load balanced solution where all the processors involved in the parallel execution of a given workload have equal execution times. The solution for single-objective optimization for energy allocates the entire workload to the most energy efficient processor, P_1.

We prove (in the supplemental, available online) that for two processors characterized by linear execution time and energy functions, the Pareto front is given by a linear segment connecting two endpoints, which is illustrated in Fig. 3. The first endpoint represents the performance optimal solution, which minimizes the execution time and distributes the workload in proportion to the speeds of the processors. The second endpoint represents the energy-optimal solution and allocates the total workload to the most energy-efficient processor. Apart from the performance optimal solution, all other Pareto-optimal solutions are load imbalanced.

We thus discover that moving from single-objective optimization for performance or energy to the bi-objective optimization for performance and energy on heterogeneous processors results in a drastic increase in the number of optimal solutions for the simple case of linear performance and energy profiles, with practically all the solutions load imbalanced.

Motivated by this finding, we study the performance and energy profiles of two data-parallel applications executed on two connected heterogeneous multi-accelerator NUMA nodes, HCLServer01, and HCLServer02. The applications employ two standard and highly optimized scientific kernels, matrix multiplication (DGEMM) that is highly compute-intensive and 2D fast Fourier transform (2D-FFT), which is memory intensive. We observe that the shapes of the performance and energy functions are non-linear and complex. Therefore, they are challenging to approximate as analytical functions that can be used as inputs to exact mathematical algorithms or optimization software for determining the Pareto front.

We, therefore, propose a solution method solving the bi-objective optimisation problem for performance and energy. It comprises two principal components. The first component is an efficient and exact global optimization algorithm, **H**eterogeneous **E**nergy **P**erformance **OPT**imization (HEPOPTA). The algorithm takes as inputs, the workload size, n; the number of available heterogeneous processors, p; p discrete performance functions (one for each processor); p discrete dynamic

energy functions (one for each processor); and the base/idle power of the platform. Briefly, the total energy consumption is the sum of dynamic and static energy consumptions. The static energy consumption is calculated by multiplying the idle power of the platform (without application execution) with the execution time of the application. Dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumed by the platform during the application execution. The performance and dynamic energy of a processor are represented by discrete functions of workload size. HEPOPTA returns the Pareto-optimal solutions for performance and energy. Each solution in the set is a distribution of workload n between the p heterogeneous processors, which, generally speaking, is not load balanced. In contrast, the traditional approaches to optimization for performance and energy do not consider non-balanced solutions as optimal. We prove the computational complexity of HEPOPTA to be $O(m^3 \times p^3 \times \log_2(m \times p))$, where m is the cardinality of the discrete execution time and dynamic energy functions. The second component is a methodology used to construct the dynamic energy profiles that are input to the algorithm [15]. The methodology is based solely on system-level power measurements using power meters. It accurately models the energy consumption by a hybrid scientific data-parallel application executing on a heterogeneous HPC platform containing different computing devices such as CPU, GPU, and Xeon Phi. The algorithm, HEPOPTA, employs as a building block, **H**eterogeneous **D**ynamic energy **P**erformance **OPT**imization (HDePOPTA), which solves the bi-objective optimisation problem for performance and dynamic energy. The inputs to HDePOPTA are the same as those for HEPOPTA and the base power of the platform is set to 0. The computational complexity of HDePOPTA is the same as HEPOPTA.

We analyse HEPOPTA experimentally using three data-parallel applications, matrix multiplication, 2D-FFT, and gene sequencing using the Smith-Waterman algorithm. The average and maximum reductions in execution time and dynamic energy consumption against load balanced solution are (26%, 102%) and (130%, 257%) for matrix multiplication, (7%, 44%) and (44%, 105%) for 2D-FFT, and (2.5%, 13.5%) and (64%, 507%) for gene sequencing. The average and the maximum number of Pareto-optimal solutions for the three applications are (55, 96), (11, 33), and (6,22). We demonstrate that minimisation of the dynamic energy consumption may not necessarily minimise the total energy consumption. The average and the maximum differences in total energy consumption between the dynamic-energy optimal and total-energy optimal solutions are (11%, 37%) for matrix multiplication, (29%, 106%) for 2D-FFT, and (9.6%, 53%) for gene sequencing.

The Pareto-optimal solutions determined by HEPOPTA contain the best load balanced solutions (mostly one solution in very few cases), whereas the rest of the solutions are load imbalanced. We distinguish between two types of load imbalanced solutions, *strong* and *weak*. We will define the two types in terms of both the workload distribution and the ratio of execution times of load balanced and load imbalanced solutions (called the load imbalance ratio (LIR)). A strong load imbalanced solution is one where one or more processors can be assigned zero workloads, the same as for the load balanced solution, but the rest of the processors, different workloads. A weak load imbalanced solution represents the case where all the processors are assigned workloads that are different from the workloads in the load balanced solution. The LIR for a strong load imbalanced solution is higher than that for a weak load imbalanced solution. A very high percentage of solutions determined by HEPOPTA are strong load imbalanced where PHI_1 gets zero workload. Therefore, HEPOPTA determines a superior Pareto front containing all the strong load imbalanced solutions ignored by *load balancing* approaches (Figs. 6 and 7).

The main original contributions of this work are:

- We discover that moving from the single-objective optimization for performance or energy to the bi-objective optimization for performance and energy on heterogeneous processors results in a drastic increase in the number of optimal solutions in the case of linear performance and energy profiles, with practically all the solutions load imbalanced. We prove that for two processors with linear execution time and energy functions, the Pareto front is linear and contains an infinite number of solutions out of which one solution is load balanced while the rest are load imbalanced.
- We propose a model-based data partitioning algorithm, HEPOPTA solving the bi-objective optimization problem for execution time and energy for data-parallel applications on heterogeneous HPC platforms. The algorithm takes input discrete execution time and dynamic energy functions with any arbitrary shape and returns the Pareto front of load imbalanced solutions and best load balanced solutions.
- We experimentally study the applicability of HEPOPTA to the optimization of real-life state-of-the-art data-parallel applications on two connected hybrid heterogeneous multi-accelerator servers consisting of multicore CPUs, GPUs, and Intel Xeon Phi. We demonstrate that the algorithm's solutions significantly improve the performance and reduce the energy consumption compared with the load balanced configuration of the applications.

The rest of the paper is organized as follows. Section 2 presents the challenges posed by performance and energy profiles of real-life scientific kernels for solving the bi-objective optimization problem. Related work is discussed in Section 3. Section 4 contains the formulation of the bi-objective optimization problem for performance and energy. Section 5 presents our algorithm, HEPOPTA, solving the bi-objective optimization problem. In Section 6, the methodology to construct the input discrete energy functions is described. We present the experimental results for HEPOPTA in Section 7. Finally, we conclude the paper in Section 8.

2 Motivation: Performance and Energy Profiles of Real-Life Scientific Kernels

We discover that moving from single-objective optimization for performance or energy to the bi-objective optimization for performance and energy on heterogeneous processors results in a drastic increase in the number of optimal solutions for the

TABLE 1
HCLServer1: Specifications of the Intel Haswell Multicore CPU, Nvidia K40c, and Intel Xeon Phi 3120P

Intel Haswell E5-2670V3	
No. of cores per socket	12
Socket(s)	2
CPU MHz	1200.402
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30720 KB
Total main memory	64 GB DDR4
Memory bandwidth	68 GB/sec
NVIDIA K40c	
No. of processor cores	2880
Total board memory	12 GB GDDR5
L2 cache size	1536 KB
Memory bandwidth	288 GB/sec
Intel Xeon Phi 3120P	
No. of processor cores	57
Total main memory	6 GB GDDR5
Memory bandwidth	240 GB/sec

TABLE 2
HCLServer2: Specifications of the Intel Skylake Multicore CPU and Nvidia P100 PCIe

Intel Xeon Gold 6152	
Socket(s)	1
Cores per socket	22
L1d cache, L1i cache	32 KB, 32 KB
L2 cache, L3 cache	256 KB, 30976 KB
Main memory	96 GB
NVIDIA P100 PCIe	
No. of processor cores	3584
Total board memory	12 GB CoWoS HBM2
Memory bandwidth	549 GB/sec

simple case of linear performance and energy profiles, with practically all the solutions load imbalanced.

Motivated by this finding, we study the performance and energy profiles of two data-parallel applications executed on two connected heterogeneous multi-accelerator NUMA nodes, HCLServer01, and HCLServer02. We observe that the shapes of the speed and energy functions are non-linear and complex. Therefore, they are challenging to approximate as analytical functions that can be used as inputs to exact mathematical algorithms or optimization software for determining the Pareto front. We employ HEPOPTA that takes input discrete execution time and dynamic energy functions with arbitrary shape to determine the Pareto front. Using this algorithm, we now present an analysis of the Pareto fronts for the two applications.

The first node, HCLServer01, consists of an Intel Haswell multicore CPU involving 24 physical cores with 64 GB main memory, which hosts two accelerators, one Nvidia K40c GPU and one Intel Xeon Phi 3120P (specifications in Table 1). HCLServer02 contains an Intel Skylake multicore CPU consisting of 22 cores and 96 GB main memory. The multicore CPU is integrated with one Nvidia P100 GPU (specifications in Table 2). Each accelerator connects to a dedicated host core via a separate PCI-E link.

A data-parallel application executing on this heterogeneous hybrid platform, consists of several kernels (generally speaking, multithreaded), running in parallel on different computing devices of the platform. The proposed algorithm for solving the bi-objective optimisation problem for performance and energy requires individual performance and energy profiles of all the kernels. Due to tight integration and severe resource contention in heterogeneous hybrid platforms, the load of one computational kernel in a given hybrid application may significantly impact others' performance to the extent of preventing the ability to model the performance and energy consumption of each kernel in hybrid applications individually [16]. To address this issue, we restrict our study in this work to configurations of hybrid applications, where individual kernels are coupled loosely enough to allow us to build their performance and energy profiles with the accuracy sufficient for successful application of the proposed algorithms. To achieve this objective, we only consider configurations where no more than one CPU kernel or accelerator kernel runs on the corresponding device. To apply our optimization algorithms, each group of cores executing an individual kernel of the application is modelled as an abstract processor [16], so that the executing platform is represented as a set of heterogeneous abstract processors. We make sure that the sharing of system resources is maximized within groups of computational cores representing the abstract processors and minimized between the groups. This way, the contention and mutual dependence between abstract processors are minimized.

We thus model HCLServer01 by three abstract processors, CPU_1, GPU_1, and PHI_1. CPU_1 represents 22 (out of total 24) CPU cores. GPU_1 involves the Nvidia K40c GPU and a host CPU core connected to this GPU via a dedicated PCI-E link. PHI_1 is made up of one Intel Xeon Phi 3120P and its host CPU core connected via a dedicated PCI-E link. In the same manner, HCLServer02 is modelled by two abstract processors, CPU_2 and GPU_2. Since there should be a one-to-one mapping between the abstract processors and computational kernels, any hybrid application executing on the servers in parallel should consist of five kernels, one kernel per computational device. Because the abstract processors contain CPU cores that share some resources such as main memory and QPI, they cannot be considered entirely independent. Therefore, the performance of these loosely-coupled abstract processors must be measured simultaneously, thereby taking into account the influence of resource contention [16].

The execution time of any computational kernel can be measured accurately using high precision processor clocks and used to model the performance of a parallel application and build its speed functions. There is, however, no such effective equivalent for measuring energy consumption. There are two dominant approaches to determine energy consumption: a). Hardware-based, such as using on-chip power sensors or system-level physical measurements using external power meters, and b). Software-based, such as energy predictive models using performance monitoring counters (PMCs). While energy predictive models provide the decomposition of energy consumption at the component level, they exhibit poor prediction accuracy and demonstrate high implementation complexity ([17], [18], [19], [20]). We present an overview of the issues with measurements using

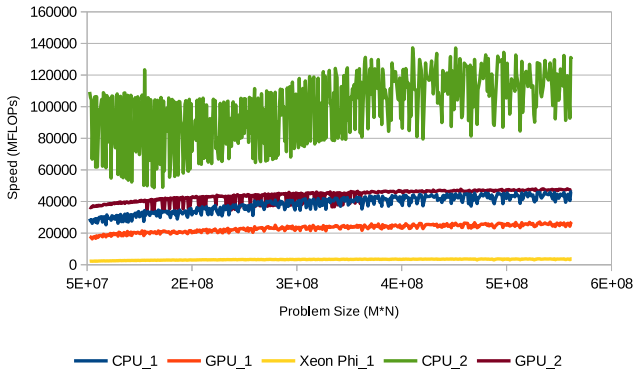

Fig. 4. Speed functions of heterogeneous 2D-FFT application executing on HCLServer01 and HCLServer02. The application computes the 2D-DFT of a matrix of size $M \times N$, where M ranges from 1024 to 10000 and N is 51200.

on-chip power sensors and energy predictive models in the experimental results section.

System-level physical measurements using power meters are accurate, but they do not provide a fine-grained decomposition of the energy consumption during the application run in a hybrid platform. Fahad *et al.* [15] propose a methodology to determine this decomposition, which employs only system-level power measurements using power meters. The methodology allows us to build discrete dynamic energy functions of abstract processors with sufficient accuracy for applying the proposed optimization algorithms in our use cases.

In our first use case, we study the performance and dynamic energy profiles of a 2D fast Fourier transform (2D-FFT) application on HCLServer01 and HCLServer02. The application computes 2D-DFT of a complex signal matrix of size $m \times n$. It employs Intel MKL FFT routines for CPUs and Xeon Phis, and CUFFT routines for Nvidia GPUs. All computations are in-card. Workloads range from 1024×51200 to 10000×51200 with the step size of 16 for m. The experimental data set does not include problem sizes that cannot be factored into primes less than or equal to 127. For these problem sizes, CUFFT for GPU gives failures. The speed of execution of a 2D-DFT of size $m \times n$ is calculated as $(2.5 \times m \times n \times \log_2(m \times n))/t$ where t is the execution time.

Figs. 4 and 5 show the speed and dynamic energy functions of the abstract processors. For each data point in these

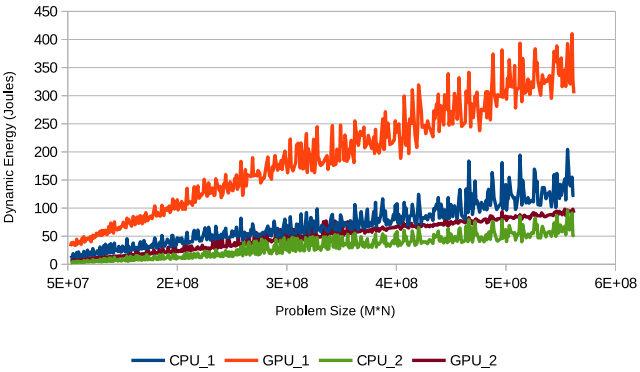

Fig. 5. Dynamic energy functions of heterogeneous 2D-FFT application executing on HCLServer01 and HCLServer02. The dynamic energy profile for Phi_1 is ignored since it consumes 10 times more energy and dwarfs the other profiles. The application calculates the 2D-DFT of a matrix size $M \times N$, where M ranges from 1024 to 10000 and N is 51200.

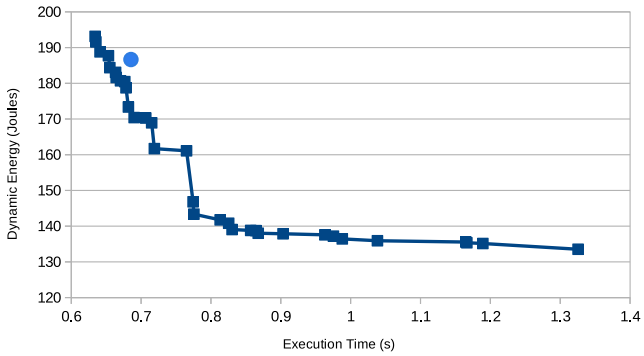

Fig. 6. Pareto-optimal solutions for 2D-FFT for a given workload size, $w = 19248 \times 51200$. Blue circle is the load balanced solution.

functions, the experiments are repeated until sample means of all the five kernels running on the abstract processors fall in the confidence interval of 95 percent. Our experimental methodology is detailed in the supplemental, available online. Fig. 6 shows the Pareto front containing 24 solutions for the input workload size $w = 19248 \times 51200$. The workload distribution maximizing the performance has the execution time of 0.63 seconds and dynamic energy consumption of 189 joules. The workload distribution with the minimal dynamic energy consumption of 131 joules has an execution time of 1.33 seconds. Optimizing for dynamic energy consumption alone degrades performance by 111 percent, and optimizing for performance alone increases dynamic energy consumption by 44 percent. The blue circle in the figure is the load balanced solution, which is close to the Pareto front.

In our second use case, we experiment with a matrix multiplication application, DGEMM. The application computes $C = \alpha \times A \times B + \beta \times C$, where A, B, and C are matrices of size $m \times n$, $n \times n$, and $m \times n$, and α and β are floating-point constants. The application uses Intel MKL DGEMM for CPUs, ZZGEMMOOC out-of-card package [21] for Nvidia GPUs, and XeonPhiOOC out-of-card package [21] for Intel Xeon Phis. ZZGEMMOOC and XeonPhiOOC packages reuse CUBLAS and MKL BLAS for in-card DGEMM calls. The out-of-card packages allow the GPUs and Xeon Phis to execute computations of arbitrary size. The Intel MKL and CUDA versions used on HCLServer01 are 2017.0.2 and 7.5, and on HCLServer02 are 2017.0.2 and 9.2.148. Workload sizes range from 64×10112 to 28800×10112 with a step size of 64 for the first dimension m. The speed of execution of a given problem size $m \times n$ is calculated as $(2 \times m \times n^2)/t$ where t is the execution time. The speed and energy profiles are provided in the supplemental, available online.

Fig. 7 shows the Pareto front containing 68 solutions for the given workload size $w = 17152 \times 10112$. The solutions are the workload distributions employing one or more of the five abstract processors. The workload distribution with the maximum performance has an execution time of 1.08 seconds, and dynamic energy consumption of 604 joules. The workload distribution with the minimum dynamic energy consumption of 167 joules has an execution time of 1.63 seconds. Optimizing for dynamic energy consumption degrades performance by 51 percent, whereas optimizing for execution time increases dynamic energy consumption by 260 percent. Thus, we observe a good number of trade-

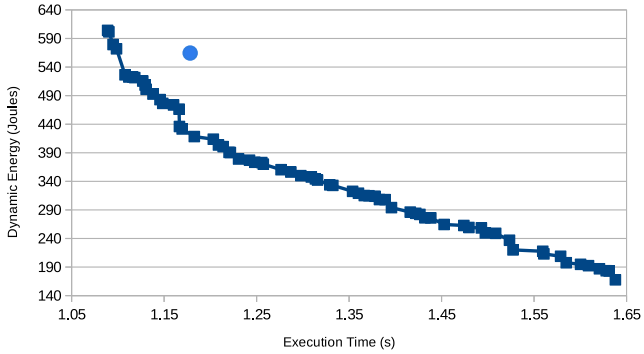

Fig. 7. Pareto-optimal solutions for heterogeneous DGEMM application for a given workload size $w = 17152 \times 10112$. Blue circle represents the load balanced solution. The discrete speed and the energy functions are provided in the supplemental, available online.

off solutions for performance and dynamic energy when workload distribution is used as the decision variable.

To summarize, we can conclude that due to the effect of heterogeneity, there is a drastic increase in the number of optimal solutions (workload distributions) as we move from single-objective optimization for performance or energy to bi-objective optimization for performance and energy on heterogeneous processors. We show using an exact workload partitioning algorithm designed by us that for the simple case of linear performance and energy profiles for two heterogeneous processors, the Pareto front is linear with an infinite number of solutions where practically all the solutions are load imbalanced.

Motivated by this finding, we study the performance and energy profiles of two standard and highly optimized scientific kernels, matrix multiplication, and 2D fast Fourier transform. We observe that the shapes of the profiles are non-linear and non-smooth. To the best of our knowledge, there is no bi-objective optimization method (or workload partitioning algorithm) that takes the profiles as an input without making any assumptions about their shapes to determine the Pareto front. To bridge this gap, we propose a bi-objective optimization algorithm for performance and total energy. The algorithm is general because it accepts discrete performance and energy profiles that can be linear or non-linear.

3 Related Work

Realistic and accurate performance and energy models of computations are essential building blocks for data partitioning algorithms solving the bi-objective optimization problem for performance and energy. We first cover them in our literature survey. We follow this with notable methods solving bi-objective optimization problem on HPC platforms.

3.1 Performance Models of Computation

Performance models of computations can be classified into analytical and non-analytical categories.

Analytical models use techniques such as linear regression, analysing patterns of computation and memory accesses, and static code analysis to estimate performance for CPUs and accelerators [22], [23]. In the non-analytical category, the most simple model is a constant performance model (CPM) where different notions such as normalized cycle time, normalized processor speed, average execution time, and task computation time. characterize the speed of an application [24], [25]. In CPMs, no dependence is assumed between the performance of a processor and the workload size.

CPMs are too simplistic to accurately model the performance of data-parallel applications executing on modern heterogeneous platforms. The most advanced load balancing algorithms employ functional performance models (FPMs) that are application-specific and that represent the speed of a processor by a continuous function of problem size [26], [27]. The FPMs capture realistically and accurately the real-life behaviour of applications executing on nodes consisting of uniprocessors (single-core CPUs).

The complex nodal architecture of modern HPC systems, consisting of tightly integrated processors with inherent severe resource contention and NUMA, pose serious challenges to load balancing algorithms based on the FPMs. These inherent traits result in significant variations (drops) in the performance profiles of parallel applications executing on these platforms, thereby violating the assumptions on the shapes of the performance profiles considered by the FPM-based load balancing algorithms. In [13], [28], [29], [30], novel model-based data partitioning algorithms are proposed that employ load imbalancing parallel computing method to address the new challenges.

3.2 Energy Modelling Techniques

There are two dominant approaches to provide an accurate measurement of energy consumption during an application execution: a). Physical measurements using external power meters or on-chip power sensors, and b). Energy predictive models. While the first approach is known to be accurate, it can only provide the measurement at a computer level and cannot, therefore, provide a fine-grained component-level decomposition of the energy consumption of an application, which is required by data partitioning algorithms optimizing the application for energy.

State-of-the-art energy predictive models predominantly use Performance Monitoring Counts (PMCs) to predict energy consumption during application execution. A typical approach is to model the energy consumption of a hardware component (such as CPU, DRAM, fans, and disks (HDD)) using linear regression of the performance events occurring in the component during application execution.

3.2.1 *Energy Predictive Models for CPUs*

Component level energy predictive models based on high positively correlated performance events such as integer operations, floating-point operations, and cache misses include [31], [32], [33]. They construct models for different hardware components such as CPU, disk, and network, based on their utilizations. Basmadjian *et al.* [34] construct a power model of a server using the summation of power models of its components: the processor (CPU), memory (RAM), fans, and disk (HDD). Lastovetsky *et al.* [13] propose a model representing the energy consumption of a multicore CPU by a non-linear function of workload size.

3.2.2 *Energy Predictive Models for Accelerators*

Hong *et al.* [35] present an energy model for an Nvidia GPU based on a PMC-based power prediction approach similar

to [36]. Nagasaka *et al.* [37] propose a PMC-based statistical power consumption modelling technique for GPUs that run CUDA applications. Song *et al.* [38] present power and energy prediction models based on machine learning algorithms such as backpropagation in artificial neural networks (ANNs). Shao *et al.* [39] develop an instruction-level energy consumption model for a Xeon Phi processor.

3.2.3 Critiques of PMC-Based Modelling

Although PMC based energy predictive software models have become popular in the scientific community, several research works highlight the poor prediction accuracy and limitations of these models. McCullough *et al.* [17] present a study on the accuracy of predictive power models for new multicore architectures and show that PMC models based on linear regression give prediction errors as high as 150 percent. O'Brien *et al.* [18] survey predictive power and energy models focusing on the highly heterogeneous and hierarchical node architecture in modern HPC computing platforms. They also present an experimental study with linear PMC based energy models where they give an average prediction error equal to 60 percent. Economou *et al.* [32] highlight the fundamental limitation of PMC-based models, which is the restricted access to read PMCs (generally four at a single run of an application). Shahid *et al.* [19] propose a selection criterion called the *additivity* for choosing a subset of PMCs to improve the accuracy of linear energy predictive models. They show that many PMCs in modern multicore CPU platforms fail the *additivity test* and hence are not reliable parameters.

3.3 Notable Works Involving Performance and Energy as Objectives

Research works [1], [2], [40], [41] propose methods for multi-objective optimization involving performance and energy as objectives. A parallel method to solve a bi-objective optimization problem for performance and energy consumption in cloud computing infrastructures is presented in [40]. It deploys a genetic algorithm to find bi-objective solutions. The parameters, input to the algorithm, are the task computation cost (w) and the communication costs between two tasks. The supply voltage (V) of the processor is the only decision variable. The consumed energy is modelled as a polynomial function of $V^2 \times w$. Fard *et al.* [1] consider four objectives, which are execution time, economic cost, energy, and reliability. Beloglazov *et al.* [41] consider twin objectives of energy efficiency and Quality of Service (QoS) for provisioning data center resources. Kessaci *et al.* [2] present a multi-objective genetic algorithm that minimizes the energy consumption, CO2 emissions, and maximizes the generated profit of a cloud computing infrastructure.

Research works [42], [43], [44] propose methods optimizing Energy-Delay Product (EDP) at the hardware level. The EDP objective function is constructed using analytical expressions for performance and energy. EDP is also used to solve software optimization problems in [45], [46], [47]. However, in [48], it is demonstrated that EDP-based techniques are not suitable for bi-objective optimization for performance and energy at the software level. These approaches are not shown to be scalable.

Research works [10], [49], [50], [51] are analytical studies of bi-objective optimization for performance and energy. Choi *et al.* [49] extend the energy roofline model by adding an extra parameter, power cap, to their execution time model. Drozdowski *et al.* [50] use iso-energy map, which are points of equal energy consumption in a multi-dimensional space of system and application parameters, to study performance-energy trade-offs. Marszałkowski *et al.* [51] analyze the impact of memory hierarchies on time-energy trade-off in parallel computations, which are represented as divisible loads.

The works reviewed in this section do not consider workload distribution as a decision variable.

4 Formulation of Heterogeneous Performance-Energy Optimization Problem (HEPOPT)

Consider a workload size n executed using p heterogeneous processors, whose execution time and dynamic energy functions are represented by $T = \{t_0(x), \ldots, t_{p-1}(x)\}$ and $E = \{e_0(x), \ldots, e_{p-1}(x)\}$ where $e_i(x)$ ($t_i(x)$), $i \in \{0, 1, \ldots, p-1\}$, is a discrete dynamic energy (execution time) function with cardinality m for processor P_i, and P_S is the base power of the platform. The function $e_i(x)$ represents the amount of dynamic energy consumed by P_i to execute the problem size x, and $t_i(x)$ is the execution time of the problem size on this processor. Without loss of generality, we assume $x \in \{1, 2, \ldots, m\}$.

The bi-objective optimization problem to find a workload distribution minimizing the execution time and the total energy consumption of computations during the parallel execution of workload n using the p processors is formulated as follows:

$$
\begin{aligned}
&HEPOPT(n, p, m, T, E, P_S): \\
&\quad \min_X \quad \left\{ \max_{i=0}^{p-1} t_i(x_i), P_S \times \max_{i=0}^{p-1} t_i(x_i) + \sum_{i=0}^{p-1} e_i(x_i) \right\} \\
&\quad \text{Subject to:} \sum_{i=0}^{p-1} x_i = n, 0 \le x_i \le m, i \in [0, p-1] \\
&\text{where} \quad p, n, m \in \mathbb{Z}_{>0}, x_i \in \mathbb{Z}_{\ge 0}, t_i(x), e_i(x), P_S \in \mathbb{R}_{\ge 0}.
\end{aligned}
\tag{1}
$$

For each given workload distribution, $X = \{x_0, \ldots, x_{p-1}\}$, HEPOPT calculates the parallel execution time, which is the time taken by the longest running processor to execute its workload, and the total energy consumption. HEPOPT returns a set of Pareto-optimal solutions, which are the workload distributions. One or more processors in an optimal solution can be allocated a workload of size zero.

5 HEPOPTA: Algorithm Finding Pareto-Optimal Solutions for Execution Time and Energy

The algorithm, HEPOPTA, solves HEPOPT. It employs **H**eterogeneous **D**ynamic energy **P**erformance **OPT**imization (HDePOPTA) as a building block, solving the bi-objective optimization problem for performance and *dynamic* energy. HDePOPTA solves the particular case of HEPOPT, where

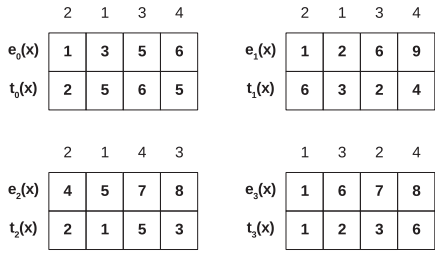

Fig. 8. Sample dynamic energy and execution time functions sorted in non-decreasing order of dynamic energies.

P_S is considered to be zero and returns Pareto-optimal solutions for performance and dynamic energy.

We describe HDePOPTA using a simple example. Suppose there are four heterogeneous processors ($p = 4$) executing a given workload size $n = 4$. The other inputs are four discrete dynamic energy functions, $E = \{e_0(x), \ldots, e_3(x)\}$, as well as four discrete time functions, $T = \{t_0(x), \ldots, t_3(x)\}$, shown in Fig. 8. The functions are sorted by dynamic energy in non-decreasing order. The functions represent the execution time and dynamic energy profiles of a real-life data-parallel application.

A naive algorithm explores the full solution tree and finds all possible workload distributions to construct the Pareto front for execution time and dynamic energy. Fig. 9 shows the tree, which is constructed by such an algorithm. Due to lack of space, only a partial tree is shown.

The tree consist of 4 levels $\{L_0, L_1, L_2, L_3\}$ where all problem sizes given to processor P_i are examined in level L_i. Each node in L_i, $i \in \{0, 1, 2, 3\}$, is labelled by a positive value representing the workload size that is distributed between processors $\{P_i, \ldots, P_3\}$. Each edge connecting a node at level L_i to its ancestor is labelled by a triple (w, e, t) where w is the problem size assigned to P_i, along with its consumed dynamic energy ($e_i(w)$) and its execution time ($t_i(w)$).

The exploration process begins from the root to find all distributions for the workload size four between four processors $\{P_0, P_1, P_2, P_3\}$. Five problem sizes, including all data points in the function $e_0(x)$ and a zero problem size, are assigned to the processor P_0 one after another. Although there is no ordering assumption, we examine the problem sizes in this example in non-decreasing order of their dynamic energy consumption. Assigning the problem sizes $\{0, 2, 1, 3, 4\}$ to P_0 expands the root into 5 children at L_1 representing the remaining workload to be distributed between processors $\{P_1, P_2, P_3\}$. For instance, the edge (2,1,2), highlighted in blue in Fig. 9, indicates that a problem size 2 with a dynamic energy consumption of 1 and an execution time of 2 is given to P_0, and its child is labelled by 2 which equals the remaining size distributed at the level L_1.

In the same manner, each node in levels $\{L_1, L_2, L_3\}$ is expanded towards the leaves. Any leaf node, labelled by 0, illustrates a solution that its dynamic energy consumption is the summation of dynamic energy consumptions, and its execution time is the maximum of the execution times labelling the edges in the path from the root to the leaf. For example, the blue path $\{(2, 1, 2), (2, 1, 6)\}$ in the tree highlights a solution distributing the workload 4 on two processors P_0 and P_1. The dynamic energy consumption of this workload distribution is 2 $(= 1 + 1)$, and the corresponding execution time is 6 $(= \max\{2, 6\})$. It is obvious that the other two processors $\{P_2, P_3\}$ are assigned a zero problem size.

Due to lack of space, we have not shown the branches that do not provide any solution. In a non-solution branch, the summation of problem sizes labelling the edges from the root to its leaf is greater than 4.

In this example, each internal node in the solution tree has either 5 children (or $m + 1$ in the general case) or just one child in which case the child is always a leaf. There are two types of leaves: *solution* leaves, labelled by 0 along with its dynamic energy consumption and execution time beneath it, and *no-solution* leaves, eliminated from, and therefore, not shown in the tree. Each internal node at level L_i, labelled by a positive number w, becomes a root of a solution tree for distribution of the workload w between processors $\{P_i, \ldots, P_3\}$ and is therefore constructed recursively.

Once a solution is found, the algorithm updates the Pareto front. In the end, the Pareto front includes three members, $\{(\langle 2, 6\rangle, \{2, 2, 0, 0\}), (\langle 4, 3\rangle, \{2, 1, 0, 1\}), (\langle 5, 2\rangle, \{2, 0, 2, 0\})\}$, where each element, like $(\langle eng, eTime\rangle, \{x_0, \ldots, x_3\})$, in the set determines the dynamic energy consumption (eng) and the execution time ($eTime$) of the workload distribution $\{x_0, \ldots, x_3\}$.

The naive algorithm has exponential complexity. We propose HDePOPTA, which is an efficient recursive algorithm

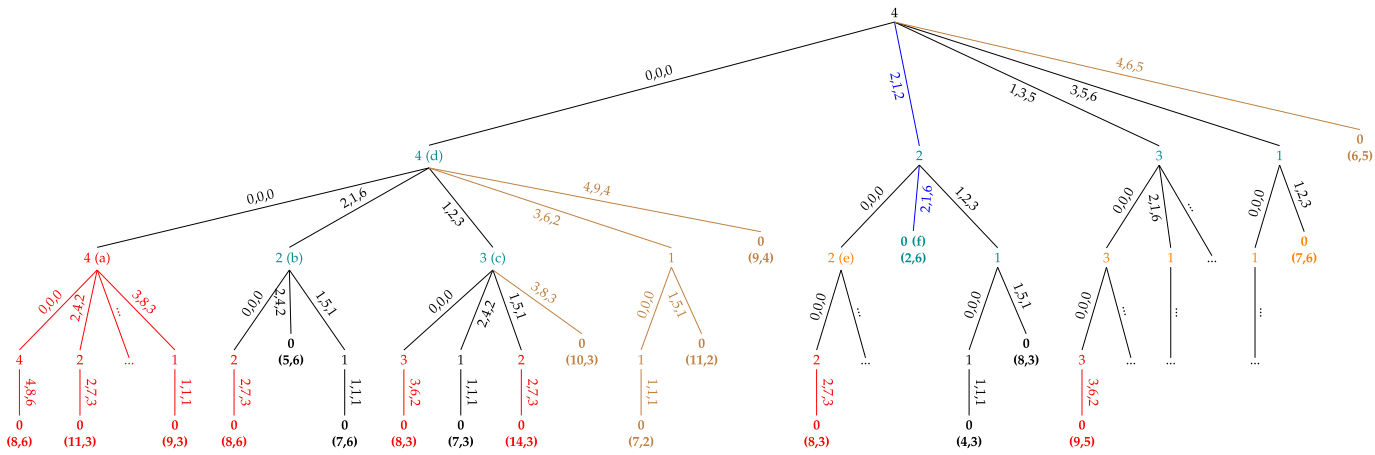

Fig. 9. For a workload $n = 4$ on four processors, the solution tree explored by the naive algorithm to find all the workload distributions and its Pareto-optimal solutions.

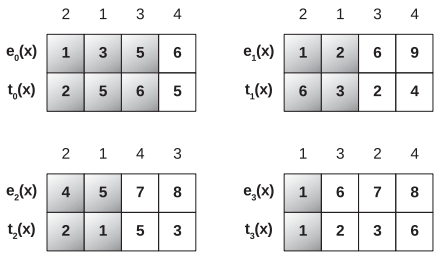

Fig. 10. Removing data points (not in bold cells) from the profiles by applying the energy threshold ε.

to determine the Pareto-optimal solutions for data-parallel applications executing on heterogeneous processors. It has polynomial computational complexity. The algorithm reduces the search space by utilizing three optimizations to avoid exploring whole subtrees in the solution tree.

We will now explain how HDePOPTA efficiently solves the example above. It scans dynamic energy functions, starting with $e_0(x)$, from left to right in non-decreasing order of dynamic energy consumption. The first optimization concerns the upper bound for dynamic energy consumption, which we call it *energy threshold* represented by ε. It is the dynamic energy consumption of the workload distribution which minimizes the execution time of the workload 4 on the processors. We determine this optimal distribution by using the algorithm HPOPTA [29]. We then initialize the variable ε to the dynamic energy consumption of this distribution. Applying the energy threshold enables HDePOPTA to reduce the search space by ignoring all data points with consumed dynamic energies greater than ε. In the example, the optimal workload distribution, returned by HPOPTA, is $X_{t_{opt}} = \{2, 0, 2, 0\}$ with an execution time (t_{opt}) of 2. Therefore, ε in this example is set to 5, which is the dynamic energy consumption for this distribution ($\sum_{i=0}^{p-1} e_i(x_{t_{opt}}[i]) = 5$). HDePOPTA, as shown in Fig. 10, ignores all data points whose dynamic energy consumptions are greater than 5. We highlight in brown all nodes and branches eliminated from the solution tree by deploying the energy threshold in Fig. 9. There may exist more than one workload distribution minimizing the execution time but with different dynamic energy consumptions. The best solution is the distribution, which minimizes ε. Nevertheless, using a non-optimal ε does not restrain HDePOPTA from obtaining the Pareto front.

To reduce the search space further, HDePOPTA assigns each level of the tree a size threshold $\sigma_i, i \in \{0, \ldots, p-1\}$. It represents the maximum workload which can be executed in parallel on processors $\{P_i, \ldots, P_{p-1}\}$ so that the dynamic energy consumption of each processor in $\{P_i, \ldots, P_{p-1}\}$ is not greater than ε. In this example, the size threshold vector σ contains four elements, $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\} = \{8, 5, 3, 1\}$. Before expanding each node, HDePOPTA compares its workload with its corresponding size threshold. If the workload exceeds the size threshold, the node is not expanded since it results in a solution with a dynamic energy consumption greater than ε.

After calculating the energy threshold ε, and the size threshold vector σ, HDePOPTA explores the solution tree from its root in the left-to-right and depth-first order. It, first, allocates zero problem sizes to P_0 and P_1 (Fig. 9). The remaining workload at the level L_2 is 4, which is labelled by 4(a) in the tree. Since the workload 4 is greater than the corresponding size threshold, σ_2, the node is not expanded further and is cut. This optimization is called operation *Cut*. We highlight in red all sub-trees eliminated from the search space using the operation *Cut*.

Returning to the tree exploration, HDePOPTA examines the next node 2(b) at the level L_2. The expansion of this node results in two solutions partitioning workload 2 on processors P_2 and P_3. HDePOPTA updates the Pareto front for this node and saves the solution in memory called $PMem$.

HDePOPTA memorizes solutions for each node in levels $\{L_1, \ldots, L_{p-2}\}$. The information stored for a node with a workload of w at a given level $L_i, i \in \{1, \ldots, p-2\}$, is a quintuple $< eng, time, part, P\#, key >$ where eng is the dynamic energy consumption of the solution, $time$ is its parallel execution time on processors $\{P_i, \ldots, P_{p-1}\}$, $part$ is the problem size given to P_i, $P\#$ is the number of active processors in the solution and finally, key, is set to the dynamic energy consumption of a saved Pareto-optimal solution for workload $w - c$ at level L_{i+1}. We call this Pareto-optimal solution at level L_{i+1} a *partial solution* for the workload w. This partial solution may not exist for some nodes, which in this case is represented by $\emptyset$. Since dynamic energies are unique in a Pareto front, we use key as a pointer to partial solutions. For each solution leaf in levels $\{L_1, \ldots, L_{p-2}\}$, like $0(f)$ in Fig. 9, HDePOPTA memorizes a solution $\{< 0, 0, 0, 0, \emptyset >\}$.

Thus, the information saved for the node 2(b) is a Pareto front including two members, $\{< 4, 2, 2, 1, \emptyset >, < 6, 1, 1, 2, \emptyset >\}$. We call this key operation, *SavePareto*. Green nodes in the solution tree highlight ones whose Pareto fronts are saved. After 2(b), the node 3(c) is examined. The solution saved for this node is $\{< 5, 2, 2, 2, \emptyset >\}$.

HDePOPTA then backtracks to the node 4(d) on L_1 and builds its Pareto front by merging Pareto fronts saved for its children, 2(b) and 3(c). Consider the edge (2,1,6) connecting the node 4(d) to 2(b). Merging this edge with the Pareto front, which has been already saved for 2(b), $\{< 4, 2, 2, 1, \emptyset >, < 6, 1, 1, 2, \emptyset >\}$, results in one Pareto-optimal solution for the node 4(d), which is saved as the quintuple $< 5, 6, 2, 2, \mathbf{4} >$. In this solution, the last element, 4, which is highlighted in bold, points to its partial solution in the node 2(b) at L_2, which is $\{< 4, 2, 2, 1, \emptyset >\}$. Merging the edge (1,2,3) with the Pareto front for 3(c), $\{< 5, 2, 2, 2, \emptyset >\}$, results in a new solution $\{< 7, 3, 1, 3, 5 >\}$. Therefore, the Pareto front for the node 4(d) is $\{< 7, 3, 1, 3, 5 >, < 5, 6, 2, 2, 4 >\}$, which is saved in the memory.

After building and saving the Pareto front of the node 4 (d), HDePOPTA visits the node 2(e) at the level L_2. This node has already been explored, and therefore, its Pareto front is retrieved from $PMem$. We call this key operation, *ReadParetoMem*. The nodes whose solutions are retrieved from the memory are highlighted in orange.

After visiting the other remaining nodes, HDePOPTA backtracks to the root and builds the Pareto-optimal solutions for the workload 4 executing on processors $\{P_0, \ldots, P_3\}$ using the Pareto fronts saved for its children. Then it terminates.

HDePOPTA thus deploys three key operations, which are a). *Cut*, b). *SavePareto*, and c). *ReadParetoMem*, to efficiently explore solution trees and build Pareto-optimal solutions optimizing for execution time and dynamic energy.

The formal description of HDePOPTA, its correctness, and complexity proofs are presented in the supplemental, available online.

We prove (in the supplemental, available online) that the solution found by HEPOPTA is a subset of the Pareto front for execution time and dynamic energy determined by HDePOPTA. Therefore, HEPOPTA calls HDePOPTA to find the Pareto front for execution time and energy. Its pseudocode and correctness proof are described in the supplemental, available online.

6 Hybrid Heterogeneous Server Energy Modeling

We present an overview of the methodology to solve the problem of modelling the dynamic energy consumption during application execution on a hybrid server composed of heterogeneous computing elements [15]. The methodology employs system-level power measurements using power meters.

To motivate the case for modelling, let us consider the optimization problem for minimizing the dynamic energy consumption during the parallel execution of a workload. To obtain the optimal workload distribution, a naïve approach explores all possible workload distributions. For each workload distribution, it determines the total dynamic energy consumption during the parallel execution of the workload from the system-level power measurements. It returns the workload distribution with the minimum total dynamic energy consumption. This approach, however, has exponential complexity.

Therefore, to reduce this complexity, we need energy models of the heterogeneous computing elements that are inputs to HDePOPTA to determine the workload distribution minimizing the dynamic energy consumption during the parallel execution of the workload.

The methodology comprises two main steps. The first step is identifying or grouping the computing elements satisfying properties that allow measurement of their energy consumptions to sufficient accuracy. We call these groups as *abstract processors*. The second step is constructing the dynamic energy models of the abstract processors, where the principal goal, apart from minimizing the time taken for model construction, is to maximize the accuracy of measurements.

6.1 Grouping of Computing Elements

We group individual computing elements executing an application together in such a way that we can accurately measure the energy consumption of the group. We call these groups, *abstract processors*. We consider two properties essential to composing the groups: (a). *Completeness:* An abstract processor must contain only those computing elements which execute the given application kernel, and (b). *Loose coupling:* Abstract processors do not interfere with each other during the application. That is, the dynamic energy consumption of one abstract processor is not affected by the activities of other abstract processor.

Based on this grouping into abstract processors, we hypothesize that the total dynamic energy consumption will equal the sum of energies consumed by all the abstract processors during an application execution. So, if E_T is the total dynamic energy consumption of the system incorporating p abstract processors $\{AP_1, \ldots, AP_p\}$, then $E_T = \sum_{i=1}^{p} E_T(AP_i)$ where $E_T(AP_i)$ is the dynamic energy consumption of the abstract processor AP_i. We call this our *additive* hypothesis.

6.2 Energy Models of Abstract Processors

The second main step of the methodology builds the dynamic energy models of the p abstract processors. We represent the dynamic energy model of an abstract processor by a discrete function composed of a set of points of cardinality m.

The total number of experiments to build the dynamic energy models is $(2^p - 1) \times m$. Consider, for example, three abstract processors $\{A, B, C\}$. The experiments can be classified into the following categories: $\{A, B, C, \{AB, C\}, \{A, BC\}, \{AC, B\}, ABC\}$. The category $\{AB, C\}$ represents parallel execution of application kernels on A and B followed by application kernel execution on C. For each workload size x, the total dynamic energy consumption is obtained from the system-level power measurements for this combined execution of kernels. The categories $\{AB, C\}$ and $\{BA, C\}$ are considered indistinguishable. There are m experiments in each category. The goal is to construct the dynamic energy models of the three abstract processors $\{A, B, C\}$ from the experimental points to sufficient accuracy.

We reduce the number of experiments to $p \times m$ by employing our additive hypothesis.

7 Experimental Results

We experimentally analyse the practical performance of HEPOPTA using three data-parallel applications, matrix multiplication, 2D-FFT, and gene sequencing using the Smith-Waterman algorithm, on the platform consisting of two connected heterogeneous multi-accelerator NUMA nodes, HCLServer01, and HCLServer02. We start with a summary of state-of-the-art energy measurement methods before analysing the additive approach for determining dynamic energy functions using the data-parallel applications, matrix multiplication, and 2D-FFT.

7.1 State-of-the-Art Energy Measurement Methods

Accurate measurement of energy consumption during an application execution is key to energy minimization at the application level and is a fundamental building block of the additive approach for constructing the dynamic energy functions. There are three popular approaches for energy measurement: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models. We consider the first approach to be the ground truth. The additive method employs this approach.

The energy measurement approach based on on-chip power sensors is now available in mainstream processors such as Intel and AMD Multicore CPUs, Nvidia GPUs, and Intel Xeon Phis. There are vendor specific libraries to acquire the power data from these sensors. For example, Running Average Power Limit (RAPL) [52] can be used to monitor power and control frequency (and voltage) of Intel CPUs, and Nvidia NVIDIA Management Library (NVML)

[53] and Intel System Management Controller chip (SMC) [54] provide the power consumption by Nvidia GPUs and Intel Xeon Phi respectively. While NVML manual specifies the accuracy of GPU on-chip sensors (±5%) [53], the accuracy of the other sensors are not known.

Fahad *et al.* [20] present the first comprehensive comparative study comparing the accuracy of state-of-the-art on-chip power sensors and energy predictive models against system-level physical measurements using external power meters. They find that the average error of the dynamic energy profiles obtained using on-chip power sensors can be as high as 73 percent, and the maximum reaches 300 percent for the two scientific applications, matrix multiplication, and 2D fast Fourier transform. They also show that the shape of a dynamic energy profile determined using on-chip sensors differs significantly from the shape obtained with the ground truth. Therefore, the energy measurements using on-chip sensors do not capture the holistic picture of the dynamic energy consumption during application execution.

Consider the dynamic energy profile of CUBLAS matrix multiplication on Nvidia Tesla K40c GPU (HCLServer01) for workload sizes ranging from 12,032 x 21,504 to 21,504 x 21,504 using a constant step size of 256. The dynamic energy consumption by DGEMM is measured with RAPL and NVML [53]. We term them collectively as *on-chip sensors*. The energy measurements using sensors are compared against HCLWattsUp API, which provides the system-level energy measurements using power meters. Fig. 11 presents the dynamic energy profiles of DGEMM using HCLWattsUp and on-chip sensors. The energy readings from the sensors exhibit a linear profile, whereas HCLWattsUp does not. The maximum and average errors of profiles given by the sensors are 35.32 and 10.62 percent, respectively.

The third approach based on software energy predictive models emerged as a popular alternative to determine the energy consumption of an application. While the models allow determination of fine-grained decomposition of energy consumption during the execution of an application at low cost compared to the other approaches, there are research works highlighting their poor accuracy [17], [18], [19], [32]. Fahad *et al.* [20] study the accuracy of platform-level and application-level energy predictive models based on linear regression. The models employ PMCs as predictor variables that have high positive correlation with dynamic energy consumption. They show that the average error between the energy predictive models and the ground truth ranges from 14 to 32 percent, and the maximum reaches 100 percent.

To summarize, a crucial requirement for the high accuracy of our solution method is the construction of accurate component-level energy functions of workload size. Power measurement APIs (PowerAPI [55]) are proven to be accurate for system-level optimization where dynamic voltage and frequency scaling (DVFS) is the key decision variable. However, based on our research, we observed that the APIs are not accurate enough to determine component-level dynamic energy profiles that can be used for energy optimization of data-parallel applications executing on multiple independent devices on a computer. In other words, while the power APIs correlate with real measurements if we vary frequency/voltage (and fix the workload), they do not correlate with real measurements if we vary the workload, as shown in the Fig. 11. Therefore, we cannot use them in methods aiming to find energy-optimal workload distributions.

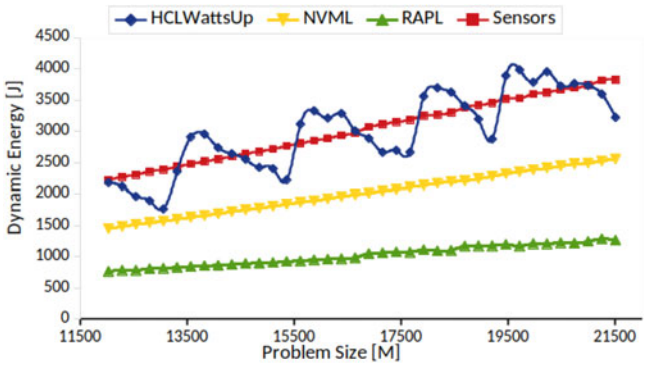

Fig. 11. Dynamic energy profiles of DGEMM on Nvidia Tesla K40c GPU. Sensors represent the summation of RAPL and NVML measurements.

7.2 Construction of Discrete Time and Dynamic Energy Functions

Based on our additive approach, we group the processing units of the platform into five abstract processors following the properties explained in Section 6.1. We name the abstract processors on HCLServer01 as CPU_1, GPU_1, Phi_1, and on HCLServer02, as CPU_2 and GPU_2.

The execution time and the dynamic energy functions of the abstract processors are experimentally built separately using an automated build procedure using five parallel processes where one process is mapped to one abstract processor. To ensure the reliability of our experimental results, we follow a detailed statistical methodology explained in Section 2 of the supplemental, available online. Briefly, to obtain a data point for each function, the software uses Student's t-test and executes the application repeatedly until the sample mean of the measurement (execution time\dynamic energy\total energy) lies in the user-defined confidence interval and a user-defined precision is achieved. The confidence interval and the precision are set to 95 and 10 percent for our experiments.

An automated tool, HCLWATTSUP [56], developed by us, is used to determine the dynamic energy and total energy consumptions of a given application kernel based on system-level power measurements using external power meters. HCLWATTSUP has no extra overhead and, therefore, does not influence the energy consumption of the application kernel. The HCLWATTSUP interface is explained in the supplemental, available online.

Several precautions are taken in computing energy measurements to eliminate the potential disturbance due to components such as Solid State Drives (SSD) and fans. They are detailed in the supplemental, available online.

The execution times of all the abstract processors executing the same workload are measured simultaneously, thereby taking into account the influence of resource contention. The execution time for accelerators includes the time taken to transfer data between the host and devices.

Dynamic energy functions for each abstract processor are constructed using the methodology explained in Section 6.2. To verify if the additive hypothesis is valid, we build four profiles for HCLServer01 (one parallel and one for each of the three abstract processors), and three profiles for HCLServer02 (one parallel and one for each of the two abstract processors).

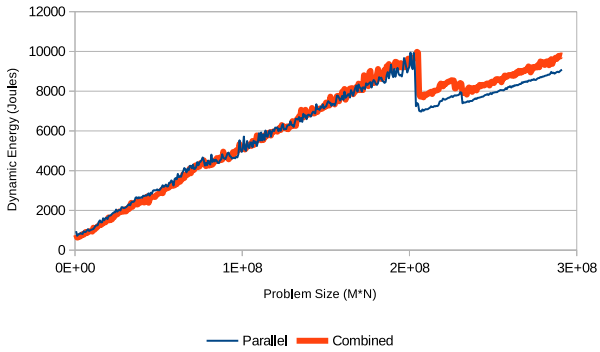

Fig. 12. Parallel and combined dynamic energy profiles for matrix multiplication application. Each data point shows the amount of dynamic energy consumed for the matrix multiplication execution of a problem size $M \times N$, where M ranges from 64 to 28800 and N is 10112.

Figs. 12 and 13 show the parallel and combined dynamic energy profiles of matrix multiplication and FFT. Here, combined refers to the sum of dynamic energy consumptions of all abstract processors when running the given workload sequentially. Table 3 shows the statistics for percentage difference of parallel to combined.

We find an average difference of 5.9 and 8.3 percent between parallel and combined dynamic energy profiles on both HCLServer01 and HCLServer02 for matrix multiplication and 2D-FFT. Despite the percentage error, both parallel and combined profiles follow the same pattern for both applications.

The parallel profiles always consume more energy than the combined profiles due to two reasons: a). Resource contention and NUMA, when all abstract processors execute the given workload in parallel. This can be seen from the relatively higher error rate for HCLServer01 compared to HCLServer02 since HCLServer01 contains three abstract processors whereas HCLServer02 has two abstract processors. b). The high precision setting of 10 percent for our experiments. This setting means that HCLWATTSUP keeps executing the given application workload until the sample mean lies in the precision interval of 10 percent.

The error rate between parallel and combined dynamic energy consumption can be reduced significantly if the precision is set to 2.5 percent. However, it will drastically increase the execution time to determine the sample mean for the given experimental data point since we need to build seven profiles: four on HCLServer01 and three on HCLServer02. We will study in our future work how to leverage the additive

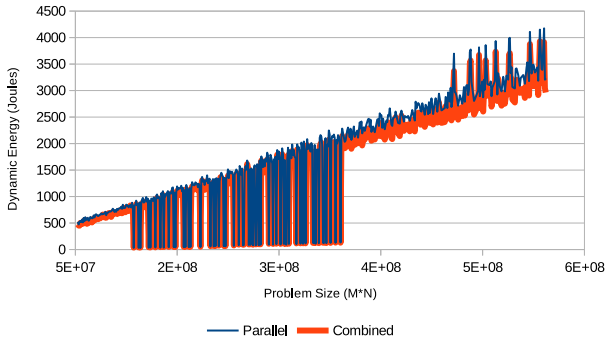

Fig. 13. Parallel and combined dynamic energy profiles for 2D-FFT application. The application calculates the 2D-DFT of a matrix of size $M \times N$, where M ranges from 1024 to 10000 and N is 51200.

TABLE 3
Percentage Difference of Dynamic Energy Consumption of Parallel to Combined

Platform	Application	Min	Max	Average
HCLServer01	DGEMM	0.026%	29.2%	6.38%
HCLServer02	DGEMM	0.001%	29.03%	3.8%
BOTH	DGEMM	0.04%	26.1%	5.9%
HCLServer01	2D-FFT	1.8%	18.4%	9.1%
HCLServer02	2D-FFT	0.02%	28.8%	12.4%
BOTH	2D-FFT	0.16%	24.7%	8.3%

component energy modelling without incurring a significant time penalty.

For each data point in the discrete performance and energy functions in the figures, all the abstract processors solve the same workload size. HEPOPTA takes the discrete functions as input and determines the optimal workload distribution. The workload distribution typically will contain different workload sizes assigned to the abstract processors, but they are still members of the input discrete sets. For a given workload, the time and energy of parallel execution of the workload is calculated as the maximum of the execution times of the assigned workload sizes and summation of their respective energies. We can call them the predicted time and energy. We confirm through exhaustive experimentation that the real time and energy of parallel execution of the workload do not differ significantly from the predicted time and energy. In Section 3 of the supplemental, available online, we experimentally show that the execution times of problem sizes executed in parallel do not differ significantly from those present in the discrete time functions.

7.3 Consideration of Cost of Communications

In this section, we discuss how our proposed solution method considers the energy consumptions of the intra-node and inter-node communications during the execution of a data-parallel application.

The execution times and the dynamic energy consumptions of the intra-node communications are included in the performance and dynamic energy models of an abstract processor. Consider for example, the abstract processor GPU_1 in HCLServer01. For each data point in the performance and dynamic energy models of GPU_1, the execution time during the execution of a workload size by GPU_1 is the total execution time that includes the transfer of data from the host core to the GPU, kernel invocation on the accelerator, and copying results back from the accelerator to the host core. Similarly, the dynamic energy consumption during the execution of a workload size by GPU_1 is the total dynamic energy consumption that includes the transfer of data from the host core to the GPU, kernel invocation on the accelerator, and copying results back from the accelerator to the host core. Therefore, the performance and dynamic energy models of an abstract processor involving accelerators integrate the data movements and take into consideration the execution times and energy consumptions of communications between a host core and accelerators inside a node.

HCLServer01 and HCLServer02 are connected by a Gigabit Ethernet network switch that consumes a constant power of 5.6 W irrespective of the amount of data transferred

between the two servers. The energy consumption by the network due to the inter-node communications is included in the static energy consumption of the platform. Since the contribution to the dynamic energy consumption due to the network during the execution of an application is zero, the dynamic energy models of the abstract processor are not affected. The total energy models will include an additional component, $5.6 \times t$, where t is the execution time of the data-parallel application. This component forms a negligible portion of the total static energy consumption of the platform.

A holistic approach to solve the bi-objective optimization problem for performance and energy must consider the performance models and energy models of both computations and communications. Rico-Gallego, Lastovetskty, and Díaz-Martín [57] proposed a model-based approach where a functional performance model for computations and a contention-aware communication model are combined for performance optimization of data-parallel applications on heterogeneous clusters of nodes containing CPUs and GPU accelerators. We plan to extend this approach to take into account the energy of computations and communications in our future work.

7.4 Analysis of HDePOPTA

The experimental data set for matrix multiplication is $\{64 \times 10112, 128 \times 10112, 196 \times 10112, \ldots, 57600 \times 10112\}$, and for 2D-FFT is $\{1024 \times 51200, 1040 \times 51200, 1056 \times 51200, \ldots, 20000 \times 51200\}$. We determine the minimum, average and maximum cardinality of Pareto fronts determined by HDePOPTA. These values for the matrix multiplication application are (1, 55, 96), and for the 2D-FFT application, (1, 11, 33).

We study improvements in performance and reductions in dynamic energy consumption of optimal solutions determined by HDePOPTA in comparison with load balanced solution. A load balanced solution (or workload distribution) is one with the minimum difference between the execution times of processors. The number of active processors in a load balanced solution may be less than the total number of available processors. The percentage performance improvement is obtained using $(t_{balance} - t_{opt})/t_{opt} * 100$, where $t_{balance}$ represents the execution time of the load balanced solution, and t_{opt} is the optimal execution time. The percentage dynamic energy saving is calculated as $(e_{balance} - e_{opt})/e_{opt} * 100$, where $e_{balance}$ represent the dynamic energy consumption of load balanced solution, and e_{opt} is optimal dynamic energy consumption. For matrix multiplication, the average and maximum performance improvements are 26 and 102 percent. The average and maximum energy savings are 130 and 257 percent. For 2D-FFT, the average and maximum performance improvements are 7 and 44 percent. The average and maximum dynamic energy savings are found to be 44 and 105 percent.

We obtain to what extent performance can be improved when the dynamic energy consumption is increased by up to 5 percent over the optimal and to what extent dynamic energy can be reduced with 5 percent degradation in performance over the optimal. The percentage performance improvement is obtained using $(t_{e_{opt}} - t_{e_{opt} \times 1.05})/t_{e_{opt} \times 1.05} * 100$, where $t_{e_{opt}}$ and $t_{e_{opt} \times 1.05}$ are the execution time of the energy-optimal endpoint and execution time associated with 5 percent increase in energy consumption over the optimal. The percentage dynamic energy saving is obtained using $(e_{t_{opt}} - e_{t_{opt} \times 1.05})/e_{t_{opt} \times 1.05} * 100$, where $e_{t_{opt}}$ and $e_{t_{opt} \times 1.05}$ are the dynamic energy consumption of the performance-optimal endpoint in the Pareto front and the dynamic energy consumption associated with 5 percent degradation in performance over the optimal.

The average and maximum performance improvements for the matrix multiplication application are 5 and 50 percent. These values for the 2D-FFT application are 19 and 109 percent. The average and maximum savings of dynamic energy consumption for our matrix multiplication application are 18 and 116 percent, and for the 2D-FFT are 6 and 63 percent.

7.5 Analysis of HEPOPTA

We use the same experimental data sets as those employed for analysis of HDePOPTA. First, the minimum, average, and maximum cardinality of Pareto fronts for execution time and total energy are determined. These values for the matrix multiplication application are $(1, 15, 35)$, and for the 2D-FFT application are $(1, 2, 8)$. The cardinalities are less than the corresponding values for the Pareto fronts for execution time and dynamic energy since the Pareto front for execution time and total energy is a subset of Pareto front for execution time and dynamic energy. Pareto-optimal solutions for execution time and total energy for matrix multiplication and FFT are shown in Figs. 14 and 15. In these figures, the blue circles above the Pareto fronts represent the execution time and total energy consumption of load balanced solutions. The Pareto front in Fig. 15 is virtually a single point signifying that optimizing for performance results in optimizing for total energy.

To study the trade-off between execution time and total energy consumption, we calculate how much performance can be gained in case the total energy consumption is increased by up to 5 percent over the optimal and to what extent dynamic energy can be reduced with 5 percent degradation in performance over the optimal. The average and maximum performance improvements for the matrix multiplication application are 8 and 17 percent. These values for the 2D-FFT application are 0.7 and 9 percent. The average and maximum savings of total energy consumption for the matrix multiplication application are 4 and 13 percent, and for the 2D-FFT are 0.4 and 6 percent. The performance improvements and total energy savings for the 2D-FFT application are negligible (considering the accuracy of measurement in our experiments). We provide a discussion for this below.

To demonstrate that dynamic energy optimization does not always result in minimizing total energy, we calculate the percentage total energy saving over HDePOPTA solutions for the data set mentioned above. Zero total energy saving means that HEPOPTA and HDePOPTA determine the same workload distribution. The minimum, average, and maximum total energy savings for the matrix multiplication application are 0, 11, and 37 percent. These values for the 2D-FFT application are 0, 29, and 106 percent.

7.6 Performance, Energy, and Scalability of HEPOPTA

The sequential algorithm HEPOPTA is executed on a single core of a multicore CPU in our experimental testbed. For any workload size in the experimental data sets for matrix

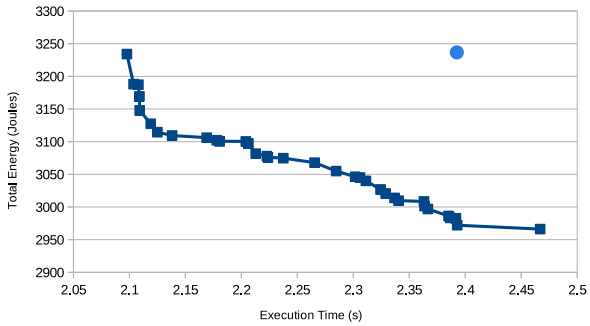

Fig. 14. Pareto-optimal solutions for execution time and total energy for a workload size 41728×10112 determined by HEPOPTA for the matrix multiplication application. The blue circle is the load balanced solution.

multiplication and 2D-FFT applications, the algorithms have execution times less than one second. The amount of energy consumed by the algorithms is also negligible compared to the energy consumption of the applications.

To describe the scalability of the proposed method on large clusters, consider a cluster of h identical nodes where each node consists of c heterogeneous processors. Since all the nodes are identical, only c execution time functions and c energy functions are constructed on one node. HEPOPTA uses h copies of each of the c execution time functions and c energy functions, that is, $h \times c$ execution time and $h \times c$ energy functions, to determine the optimal workload distributions. The time complexity of HEPOPTA will be $O(m^3 \times p^3 \times \log_2(m \times p))$ where $p = h \times c$ and m represents the cardinality of the discrete functions representing the execution time and dynamic energy. The practical time complexity is observed to be negligible compared to the execution time of the application. Therefore, the cost of our method is dominated by the construction of the execution time and energy functions for a single node.

To derive the time complexity of HEPOPTA, we have only considered the memorization technique. There are two other important optimisations, energy and size thresholds, that have been ignored because their occurrence is a function of the shape of the input discrete functions and the input workload size. These optimizations play an important role leading to very less practical complexity. HEPOPTA falls into the class of branch-and-cut algorithms. We can think of two approaches to reduce the time complexity: (a). Parallel branch-and-cut techniques, and (b). Hierarchical algorithmic approaches, which can potentially reduce the

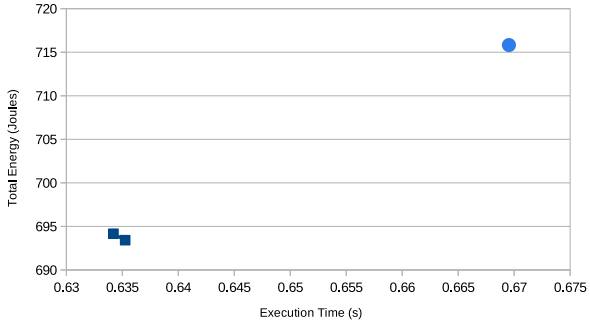

Fig. 15. Pareto-optimal solutions for execution time and total energy for a workload size 19248×51200 determined by HEPOPTA for the 2D-FFT application. The blue circle is the load balanced solution.

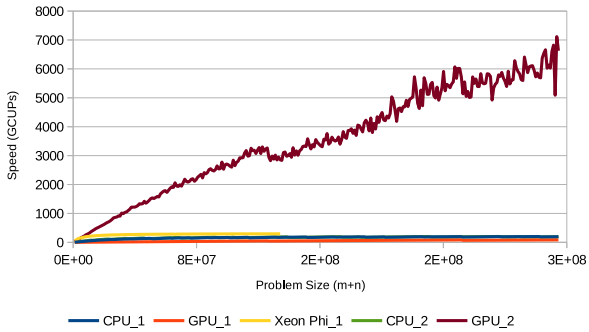

Fig. 16. Speed functions of heterogeneous SW application executing on HCLServer01 and HCLServer02.

complexity to $O((m^2 \times h + m^3 \times c^3) \times \log_2(m \times p))$. This is the subject of our future work.

7.7 Gene Sequencing Using Smith-Waterman Algorithm

We use a gene sequencing application executing the Smith-Waterman algorithm (SW) ([58], [59]) to analyse our proposed algorithms. The application deals with the alignment of DNA or protein sequences. It employs the SW algorithm, which uses a dynamic programming (DP) approach to determine the optimal local alignment score of two sequences, a query sequence of length m and a database sequence of length n. The time and space complexities of the SW DP algorithm are $O(m \times n)$ and $O(m)$, where $m < n$, assuming the use of refined linear-space methods. The speed of execution of the application for a given workload size, $(m + n)$, is calculated as $(m \times n)/t$ where t is the execution time. The speed is usually measured in GCUPS, which stands for Billions of Cell Updated per Second. A detailed description of the application is presented in the supplemental, available online.

Three abstract processors model the application, CPU_1, GPU_1 and PHI_1, on HCLServer01, and two abstract processors, CPU_2 and GPU_2, on HCLServer02. The application employs optimized SW routines provided by SWIPE for Multicore CPUs [60], CUDASW++3.0 for Nvidia GPU accelerators [61], and SWAPHI for Intel Xeon Phi accelerators [62]. All the computations are in-card.

Figs. 16 and 17 show speed and dynamic energy functions of the application executing on HCLServer01 and HCLServer02. Workloads range from 64×16384 to 8192×16384 for Phi_1 abstract processor and to 19200×16384 for the other abstract processors with the step size 64 for

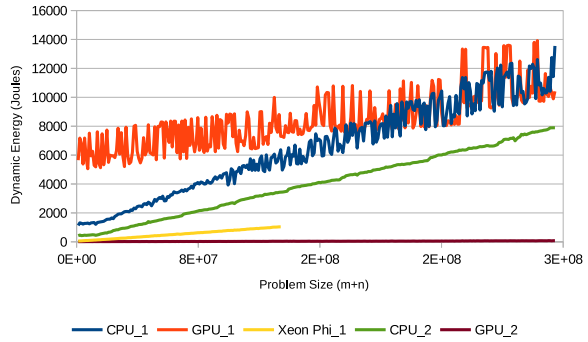

Fig. 17. Dynamic energy functions of heterogeneous SW application executing on HCLServer01 and HCLServer02.

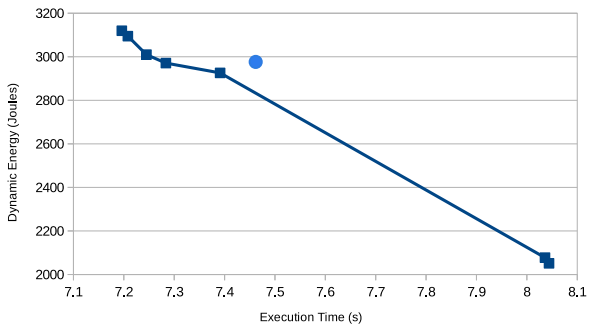

Fig. 18. Pareto-optimal solutions for execution time and dynamic energy for a workload size 29312×16384 determined by HDePOPTA for the SW application. The blue circle is the load balanced solution.

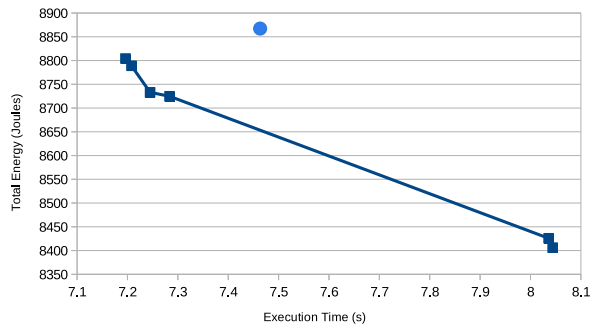

Fig. 19. Pareto-optimal solutions for execution time and total energy for a workload size 29312×16384 determined by HEPOPTA for the SW application. The blue circle is the load balanced solution.

m. The figures contain real execution times and energy consumptions.

The experimental data set in this case study is $\{64 \times 16384, 128 \times 16384, \ldots, 40000 \times 16384\}$. The minimum, average and maximum cardinality of Pareto fronts determined by HDePOPTA are $(1, 6, 22)$. These values for the sets obtained by HEPOPTA are $(1, 2, 6)$. Pareto-optimal solutions for execution time and dynamic energy and execution time and total energy for a workload size 29312×16384 are shown in Figs. 18 and 19.

For this application, the average and maximum performance improvements over load balanced solutions are 2.5 and 13.5 percent, respectively. The average and maximum dynamic energy saving over load balanced solutions are 64 and 507 percent. The average and maximum performance improvements when the dynamic energy consumption is increased by 5 percent over the optimal are 0.5 and 40 percent. The average and maximum dynamic energy savings with 5 percent degradation in performance over the optimal are 48 and 528 percent.

The average and maximum performance improvements when the total energy consumption is increased by 5 percent over the optimal are 1.5 and 18 percent. The average and maximum savings in total energy with 5 percent degradation in performance over the optimal are 2.2 and 19 percent. We also calculate the total energy saving over HDePOPTA solutions (solutions minimising dynamic energy consumption) to experimentally show that dynamic energy optimization does not always result in minimizing total energy. The minimum, average, and maximum total energy savings over optimal solutions for dynamic energy are 0, 10, and 53 percent.

7.8 Analysis of Pareto Fronts

In this section, we will analyse the interplay between performance and energy (plotted on x and y axes, respectively) in the Pareto fronts and provide guidelines to decide which objective to optimize. Since the state-of-the-art optimization methods are based on load balancing, we will focus on guidelines that employ the load balancing solution as the starting point of optimization [14]. The endpoints of a Pareto front represent the optimal solutions for single-objective optimization for performance and energy. A steep slope close to time-optimal endpoint means that allowing a small degradation in performance can result in significant energy savings. Similarly, a steep slope close to the energy-optimal endpoint means that large performance improvement can be achieved with a small increase in energy consumption. A few other interesting guidelines follow:

Figs. 6 and 7 show that the load balanced solution lies inside the intervals bounded by the projections of the endpoints on the x-axis and y-axis. In that case, minimizing for one objective will result in worsening the other objective. Fig. 14 shows that the total energy of the load balanced solution lies outside the interval bounded by the projections of the endpoints on the y-axis. In this case, maximizing performance will also result in a reduction in energy. Suppose the execution time of the load balanced solution lies outside the interval bounded by the projections of the endpoints on the x-axis. In that case, minimizing for energy will also result in performance improvement. If the load balanced solution lies outside the intervals bounded by the projections of the endpoints on the x-axis and y-axis, then minimizing for one objective will result in improvement for the other objective.

Suppose the execution time of the load balanced solution is much closer to projection of the energy-optimal endpoint on the x-axis than the time-optimal endpoint. In that case, a major reduction in energy will only result in minor performance improvement and maximizing for performance will lead to large increase in energy. Similarly, suppose the energy of the load balanced solution is much closer to projection of the time-optimal endpoint on the y-axis than the energy-optimal endpoint (Figs. 18 and 19). In that case, a major reduction in execution time will only result in minor reduction in energy and minimizing for energy will lead to significant performance degradation.

7.9 Discussion

Our solution method provides good tradeoffs for performance and total energy for the matrix multiplication and gene sequencing applications. However, it is not the case for the 2D-FFT application (Fig. 15), where the performance optimal solution is also the optimal solution for total energy. This behaviour is due to the large total static energy consumption during the application execution. Our platforms, HCLServer01, and HCLServer02, are already six and four years old and therefore consume high static energy, and its share in the total energy consumption is especially high for heavily memory-bound applications such as 2D-FFT. The tradeoffs for performance and dynamic energy, however, are good for this application (Fig. 6). Therefore, we believe that as the hardware platforms become more energy-efficient,

thereby consuming lower static energy, the dynamic energy consumption will become more prominent, and our solution method will find good tradeoffs for performance and total energy even for heavily memory-bound applications.

8 Conclusion

Performance and energy are the two most important objectives for optimization on modern parallel platforms such as supercomputers, heterogeneous HPC clusters, and cloud computing infrastructures. We discovered in this work that moving from single-objective optimization for performance or energy to their bi-objective optimization on heterogeneous processors results in a drastic increase in the number of optimal solutions (workload distributions) even in the simple case of linear performance and energy profiles. Motivated by this finding, we studied the full performance and energy profiles of two data-parallel applications executed on two connected heterogeneous hybrid nodes and found them to be non-linear and complex. Therefore, the profiles are challenging to approximate as analytical functions that can be used as inputs to exact mathematical algorithms or optimization software for determining the Pareto front.

We then proposed an efficient global optimization algorithm solving the bi-objective optimization problem on heterogeneous HPC platforms for performance and energy. The problem aims to optimize the parallel execution of a given workload of n by a set of p heterogeneous processors. It has one decision variable, the *workload distribution*. The algorithm takes discrete speed and dynamic energy functions with arbitrary shape and returns the Pareto-optimal solutions (generally speaking, load imbalanced). The input dynamic energy functions are constructed using a methodology which is purely based on system-level power measurements using power meters and which accurately models the energy consumption of a hybrid scientific application executing on a heterogeneous HPC platform incorporating different computing devices.

We experimentally analysed our solution method using three data-parallel applications, matrix multiplication, 2D fast Fourier transform, and gene sequencing using the Smith-Waterman algorithm on two connected heterogeneous servers consisting of multicore CPUs, GPUs, and Intel Xeon Phi. We demonstrated that the solutions provided by our method significantly improve the performance and reduce the energy consumption in comparison with the load balanced configuration of the applications. We have shown that our method determines a superior Pareto front containing all *strong* load imbalanced solutions that are ignored by *load balancing* approaches and best load balanced solutions.

Acknowledgment

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant 14/IA/2474.

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. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2012, pp. 300–309.

[2] 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.

[3] J. J. Durillo, V. Nae, and R. Prodan, "Multi-objective energy-efficient workflow scheduling using list-based heuristics," *Future Gener. Comput. Syst.*, vol. 36, pp. 221–236, 2014.

[4] F. D. Rossi, M. G. Xavier, C. A. 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, 2017.

[5] 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, 2015.

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

[7] 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. Supercomput.*, 2007, pp. 1–9.

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

[9] 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. Soc. Comput.*, 2010, pp. 139–146.

[10] J. Demmel, A. Gearhart, B. Lipshitz, and O. Schwartz, "Perfect strong scaling using no additional energy," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process.*, 2013, pp. 649–660.

[11] 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, 2014.

[12] 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.*, 2017, pp. 533–542.

[13] 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.

[14] 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.

[15] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements," *IEEE Access*, vol. 8, pp. 93793–93829, 2020.

[16] Z. Zhong, V. Rychkov, and A. Lastovetsky, "Data partitioning on multicore and multi-GPU platforms using functional performance models," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2506–2518, Sep. 2015.

[17] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2011, Art. no. 12.

[18] K. O'Brien, I. Pietri, R. Reddy, A. Lastovetsky, and R. Sakellariou, "A survey of power and energy predictive models in HPC systems and applications," *ACM Comput. Surv.*, vol. 50, no. 3, 2017, Art. no. 37.

[19] A. Shahid, M. Fahad, R. Reddy, and A. Lastovetsky, "Additivity: A selection criterion for performance events for reliable energy predictive modeling," *Supercomput. Front. Innov.*, vol. 4, no. 4, pp. 50–65, 2017.

[20] 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, 2019, Art. no. 2204.

[21] H. Khaleghzadeh, Z. Zhong, R. Reddy, and A. Lastovetsky, "Out-of-core implementation for accelerator kernels on heterogeneous clouds," *J. Supercomput.*, vol. 74, no. 2, pp. 551–568, 2018.

[22] K.-H. Kim, K. Kim, and Q.-H. Park, "Performance analysis and optimization of three-dimensional FDTD on GPU using roofline model," *Comput. Phys. Commun.*, vol. 182, no. 6, pp. 1201–1207, 2011.

[23] J. Shen, A. L. Varbanescu, Y. Lu, P. Zou, and H. Sips, "Workload partitioning for accelerating applications on heterogeneous platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2766–2780, Sep. 2016.
[24] A. Kalinov and A. Lastovetsky, "Heterogeneous distribution of computations solving linear algebra problems on networks of heterogeneous computers," *J. Parallel Distrib. Comput.*, vol. 61, no. 4, pp. 520–535, 2001.
[25] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert, "Matrix multiplication on heterogeneous platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1033–1051, Oct. 2001.
[26] A. Lastovetsky and R. Reddy, "Data partitioning for multiprocessors with memory heterogeneity and memory constraints," *Sci. Program.*, vol. 13, no. 2, pp. 93–112, 2005.
[27] L. A. Lastovetsky and R. Reddy, "Data partitioning with a functional performance model of heterogeneous processors," *Int. J. High Perform. Comput. Appl.*, vol. 21, no. 1, pp. 76–90, 2007.
[28] A. Lastovetsky, L. Szustak, and R. Wyrzykowski, "Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 787–797, Mar. 2017.
[29] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, "A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous HPC platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 10, pp. 2176–2190, Oct. 2018.
[30] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, "A hierarchical data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous multi-accelerator NUMA nodes," *IEEE Access*, vol. 8, pp. 7861–7876, 2019.
[31] T. Heath, B. Diniz, B. Horizonte, E. V. Carrera, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proc. 10th ACM SIGPLAN Symp. Princ. Practice Parallel Program.*, 2005, pp. 186–195.
[32] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Proc. Workshop Model. Benchmarking Simul.*, 2006, pp. 70–77.
[33] W. L. Bircher and L. K. John, "Complete system power estimation using processor performance events," *IEEE Trans. Comput.*, vol. 61, no. 4, pp. 563–577, Apr. 2012.
[34] 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.*, 2011, pp. 1–10.
[35] S. Hong and H. Kim, "An integrated GPU power and performance model," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 280–289, 2010.
[36] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proc. 36th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2003, Art. no. 93.
[37] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," in *Proc. Int. Green Comput. Conf. Workshops*, 2010, pp. 115–122.
[38] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent GPU architectures," in *Proc. 27th IEEE Int. Parallel Distrib. Process. Symp.*, 2013, pp. 673–686.
[39] Y. S. Shao and D. Brooks, "Energy characterization and instruction-level energy model of Intel's Xeon Phi processor," in *Proc. Int. Symp. Low Power Electron. Des.*, 2013, pp. 389–394.
[40] M. Mezmaz *et al.*, "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, 2011.
[41] 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, 2012.
[42] A. J. Martin, "Towards an energy complexity of computation," *Inf. Process. Lett.*, vol. 77, no. 2–4, pp. 181–187, 2001.
[43] A. J. Martin, M. Nyström, and P. I. Pénzes, "Et2: A metric for time and energy efficiency of computation," in *Power Aware Computing*. Berlin, Germany: Springer, 2002, pp. 293–315.
[44] V. Srinivasan *et al.*, "Optimizing pipelines for power and performance," in *Proc. 35th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2002, pp. 333–344.
[45] 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.
[46] B. D. Bingham and M. R. Greenstreet, "Computation with energy-time trade-offs: Models, algorithms and lower-bounds," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2008, pp. 143–152.
[47] J. H. Laros III *et al.*, "Energy delay product," in *Energy-Efficient High Performance Computing*. Berlin, Germany: Springer, 2013, pp. 51–55.
[48] S. I. Roberts, "Energy-aware performance engineering in high performance computing," PhD dissertation, Dept. Comput. Sci., Univ. Warwick, Coventry, U.K., 2017.
[49] J. Choi, M. Dukhan, X. Liu, and R. Vuduc, "Algorithmic time, energy, and power on candidate HPC compute building blocks," in *Proc. IEEE 28th Int. Parallel Distrib. Process. Symp.*, 2014, pp. 447–457.
[50] M. Drozdowski, J. M. Marszalkowski, and J. Marszalkowski, "Energy trade-offs analysis using equal-energy maps," *Future Gener. Comput. Syst.*, vol. 36, pp. 311–321, 2014.
[51] J. M. Marszałkowski, M. Drozdowski, and J. Marszałkowski, "Time and energy performance of parallel systems with hierarchical memory," *J. Grid Comput.*, vol. 14, no. 1, pp. 153–170, 2016.
[52] 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./Apr. 2012.
[53] Nvidia, "Nvidia management library: NVML reference manual," Oct. 2018. [Online]. Available: https://docs.nvidia.com/pdf/NVML_API_Reference_Guide.pdf
[54] R. Rahman, "Intel Xeon Phi coprocessor architecture and tools: The guide for application developers," 1st ed. USA: Apress, 2013.
[55] A. Bourdon, A. Noureddine, R. Rouvoy, and L. Seinturier, "PowerAPI: A software library to monitor the energy consumed at the process-level," *ERCIM News*, vol. 92, pp. 43–44, Jan. 2013. [Online]. Available: https://hal.inria.fr/hal-00772454
[56] M. Fahad and R. R. Manumachu, "HCLWattsUp: Energy API using system-level physical power measurements provided by power meters," Heterogeneous Comput. Lab., Univ. College Dublin, 2020. [Online]. Available: https://csgitlab.ucd.ie/manumachu/hclwattsup
[57] J.-A. Rico-Gallego, A. L. Lastovetsky, and J.-C. Díaz-Martín, "Model-based estimation of the communication cost of hybrid data-parallel applications on heterogeneous clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 11, pp. 3215–3228, Nov. 2017.
[58] T. Smith and M. Waterman, "Identification of common molecular subsequences," *J. Mol. Biol.*, vol. 147, no. 1, pp. 195–197, 1981.
[59] O. Gotoh, "An improved algorithm for matching biological sequences," *J. Mol. Biol.*, vol. 162, no. 3, pp. 705–708, 1982.
[60] T. Rognes, "Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation," *BMC Bioinf.*, vol. 12, no. 1, 2011, Art. no. 1.
[61] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: Accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions," *BMC Bioinf.*, vol. 14, no. 1, 2013, Art. no. 1.
[62] Y. Liu and B. Schmidt, "SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors," in *Proc. IEEE 25th Int. Conf. Appl.-Specific Syst. Archit. Processors*, 2014, pp. 184–185.

Hamidreza Khaleghzadeh received the BSc and MSc degrees in computer engineering (software), in 2007 and 2011, respectively, and the PhD degree from the School of Computer Science, University College Dublin, Dublin, Ireland, in 2019. His research interests include performance and energy consumption optimization in massively heterogeneous systems, high-performance heterogeneous systems, energy efficiency, and parallel/distributed computing.

Muhammad Fahad received the BS degree from the International Islamic University Islamabad, Islamabad, Pakistan, in 2008, and the MS degree from the KTH - Royal Institute of Technology, Stockholm, Sweden, in 2012. He is a PhD researcher with Heterogeneous Computing Lab (HCL), University College Dublin, Ireland. His main research interests include high performance heterogeneous computing, energy efficient computing, parallel/distributed, and peer-to-peer computing.

Arsalan Shahid received the BS degree in electrical engineering from HITEC University, Taxila, Pakistan, in 2016, and the PhD degree from the University College Dublin, Dublin, Ireland, in 2020. His research interests include energy-aware high-performance heterogeneous computing and intelligent cloud computing.

Ravi Reddy Manumachu received the BTech degree from I.I.T. Madras, Chennai, India, in 1997, and the PhD degree from the School of Computer Science, University College Dublin, Dublin, Ireland, in 2005. His main research interests include high performance heterogeneous computing, distributed computing, sparse matrix computations, and energy-efficient computing.

Alexey Lastovetsky (Member, IEEE) received the PhD degree from the Moscow Aviation Institute, Moscow, Russia, in 1986, and the doctor of science degree from the Russian Academy of Sciences, Moscow, Russia, in 1997. His main research interests include algorithms, models, and programming tools for high performance heterogeneous computing. He has published more than a 100 technical papers in refereed journals, edited books, and international conferences. He authored *the Monographs Parallel Computing on Heterogeneous Networks* (Wiley, 2003) and *High performance heterogeneous computing* (Wiley, 2009).