Received: 18 October 2019 | Revised: 3 June 2020 | Accepted: 4 June 2020

DOI: 10.1002/cpe.5928

SPECIAL ISSUE PAPER

WILEY

A novel data partitioning algorithm for dynamic energy optimization on heterogeneous high-performance computing platforms

Hamidreza Khaleghzadeh | **Muhammad Fahad** | **Ravi Reddy Manumachu** | **Alexey Lastovetsky**

School of Computer Science, University College Dublin, Belfield, Ireland

Correspondence
Hamidreza Khaleghzadeh, School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland.
Email: hamidreza.khaleghzadeh@ucd.ie

Funding information
Science Foundation Ireland, Grant/Award Number: 14/IA/2474

Summary

Energy is one of the most important objectives for optimization on modern heterogeneous high-performance computing (HPC) platforms. The tight integration of multicore CPUs with accelerators such as graphical processing units (GPUs) and Xeon Phi coprocessors in these platforms presents several challenges to the optimization of multithreaded data-parallel applications for energy. In this work, the problem of optimization of data-parallel applications on heterogeneous HPC platforms for dynamic energy through *workload distribution* is formulated. We propose a workload partitioning algorithm to solve this problem. It employs load-imbalancing technique to determine the workload distribution minimizing the dynamic energy consumption of the parallel execution of an application. The inputs to the algorithm are discrete dynamic energy profiles of individual computing devices. The profiles are practically constructed using an approach that accurately models the energy consumption by execution of a hybrid scientific data-parallel application on a heterogeneous platform containing different computing devices such as CPU, GPU, and Xeon Phi. The proposed algorithm is experimentally analyzed using two multithreaded data-parallel applications, matrix multiplication and 2D fast Fourier transform. The load-imbalanced solutions provided by the algorithm achieve significant dynamic energy reductions for the two applications (in average by 130% and 44%, respectively) compared with the load-balanced solutions.

KEYWORDS

energy of computation, energy optimization, GPU, heterogeneous platforms, high-performance computing, multicore CPU, Xeon Phi

1 | INTRODUCTION

Energy consumption is one of the main challenges hindering high-performance computing (HPC) community from breaking the exascale barrier.[1] Current HPC systems are already consuming megawatts of energy. For example, the world's most powerful supercomputer as of 2018, Summit, consumes around 10 MW of power (including the cooling power),[2] and US Department of Energy aims to deploy an exascale supercomputer, capable of performing exaflops (10^{18}) in a power envelope of 20 to 30 MW.[3] Because of such high power consumption, energy optimization has become a first-class constraint at both hardware and software levels.

Energy optimization in HPC is studied chiefly in the context of biobjective optimization for performance and energy. State-of-the-art solution methods for the biobjective optimization problem can be broadly classified into *system-level* and *application-level* categories. System-level solution methods aim to optimize 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. Majority of the works in this category optimize for performance with energy budget as a constraint. Application-level solutions proposed in References 4-7 use application-level parameters as decision variables and application-level models for predicting the performance and energy consumption of applications. The application-level parameters include the number of threads, number of processors, loop tile size, workload distribution, and so on. Chakrabarti et al[5] consider the effect of heterogeneous workload distribution on biobjective optimization of data analytics applications by simulating heterogeneity on homogeneous clusters. The performance is represented by a linear function of problem size and the total energy is predicted using historical data tables. Research works[6,7] demonstrate by executing real-life data-parallel applications on modern multicore CPUs that the functional relationships between performance and workload distribution and between energy and workload distribution have complex (nonlinear) properties. They target homogeneous HPC platforms.

Modern heterogeneous HPC platforms feature tight integration of multicore CPUs with accelerators such as graphics processing units (GPUs) and Xeon Phi coprocessors to provide cutting-edge computational power and increased energy efficiency. This has resulted in inherent complexities such as severe resource contention for shared on-chip resources [last level cache (LLC), interconnect] and nonuniform memory access (NUMA). One visible manifestation of these complexities is a complex functional relationship between energy consumption and workload size of applications executing on these platforms where the shape of energy profiles may be highly nonlinear and nonsmooth with drastic variations. This provides an opportunity for application-level energy optimization through workload distribution as a decision variable.

Consider the dynamic energy profiles of multithreaded matrix-matrix multiplication (DGEMM) and 2D fast Fourier transform (2D-FFT) applications executed on two connected heterogeneous multiaccelerator NUMA nodes, HCLServer1 (Table 1) and HCLServer2 (Table 2). The multicore CPU in HCLServer1 is integrated with one Nvidia K40c GPU and one Intel Xeon Phi 3120P. The multicore CPU in HCLServer2 is integrated with one Nvidia P100 GPU. DGEMM computes the matrix product, $C = \alpha \times A \times B + \beta \times C$, where A, B, and C are, respectively, dense matrices of size $m \times n$, $n \times n$, and $m \times n$ and α and β are constant floating-point numbers. 2D-FFT computes the Fourier transform of a complex matrix of size $m \times n$. The structure of the applications can be found in Reference 8 and the supplemental. The Intel MKL and Nvidia CUDA versions used on HCLServer1 are, respectively, 2017.0.2 and 7.5. The CUDA version 9.2.148 is installed on HCLServer2.

A data-parallel application executing on this heterogeneous platform, consists of a number of kernels (generally speaking, multithreaded), running in parallel on different computing devices of the platform. The proposed algorithm for solving the optimization problem for dynamic energy requires individual dynamic 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 the performance of others to the extent of preventing the ability to model the energy consumption of each kernel in hybrid applications individually. To address this issue, we restrict our study in this work to such configurations of hybrid applications, where individual kernels are coupled loosely enough to allow us to build their individual energy profiles with the accuracy sufficient for successful application of the proposed algorithm. To achieve this objective, we only consider configurations where no more than one CPU kernel or accelerator kernel is running on the corresponding device. In order to apply our optimization algorithm, each group of cores executing an individual kernel of the application is modeled as an abstract processor[9] so that the executing platform is represented as a set of abstract processors. Each abstract processor solely constitutes the processing elements and resources which are involved

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

Intel Haswell E5-2670V3		Nvidia K40c		Intel Xeon Phi 3120P	
Socket(s), Cores per socket	2, 12	No. of processor cores	2880	No. of processor cores	57
Main memory	64 GB	Total board memory	12 GB	Total main memory	6 GB
Idle Power (W)	60	Idle Power (W)	68	Idle Power (W)	91
TDP (W)	120	TDP (W)	245	TDP (W)	300

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

Intel Xeon Gold 6152		Nvidia P100 PCIe	
Socket(s), Cores per socket	1, 22	No. of processor cores	3584
Main memory	96 GB	Total board memory	12 GB
Idle Power (W)	60	Idle Power (W)	30
TDP (W)	140	TDP (W)	250

in the execution of a given application kernel on it. 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 loosely coupled abstract processors are minimized.

HCLServer1 is modeled by three abstract processors, CPU_1, GPU_1, and PHI_1, as shown in Figure 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, HCLServer2 is modeled 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. Each server is equipped with a Watts Up Pro power meter to provide system-level physical power measurements.

The dynamic energy profiles for the two applications are shown in Figure 2. Each profile presents the dynamic energy consumption of a given processor vs workload size executed on the processor. The dynamic energy consumptions are determined using the *HCLWattsUp* API,[10] which gathers the system-level physical power measurements from the power meter. Each data point in the profiles is obtained using Student's *t*-test. The application is run repeatedly until the sample mean of the measurement (execution time\dynamic energy) lies in the confidence interval (95%) and a user-defined precision (10%) is achieved.

Consider the execution of DGEMM for the workload size 31360×10112 employing all the five abstract processors, {CPU_1, CPU_2, GPU_1, GPU_2, PHI_1}. The solution determined by load-balanced algorithm is {2560, 2688, 5376, 20736, 0} and its dynamic energy consumption is 1003 J. The optimal workload distribution is {0, 1216, 1344, 28800, 0} resulting in dynamic energy consumption of 475 J and thereby providing 52.6% reduction in energy. Consider the execution of 2D-FFT for the workload size 10960×51200 (2D signal) employing all the five abstract processors. The solution (workload distribution) determined by load-balanced algorithm is {1232, 7040, 1024, 1664, 0} and its dynamic energy consumption is 94 J. The load-balancing algorithm employs horizontal decomposition of the rows of the 2D signal. The optimal workload distribution is {0, 9072, 0,

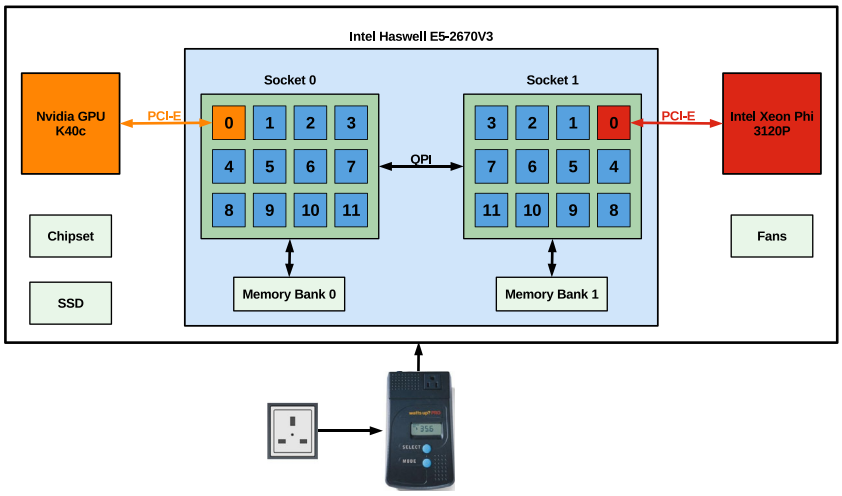
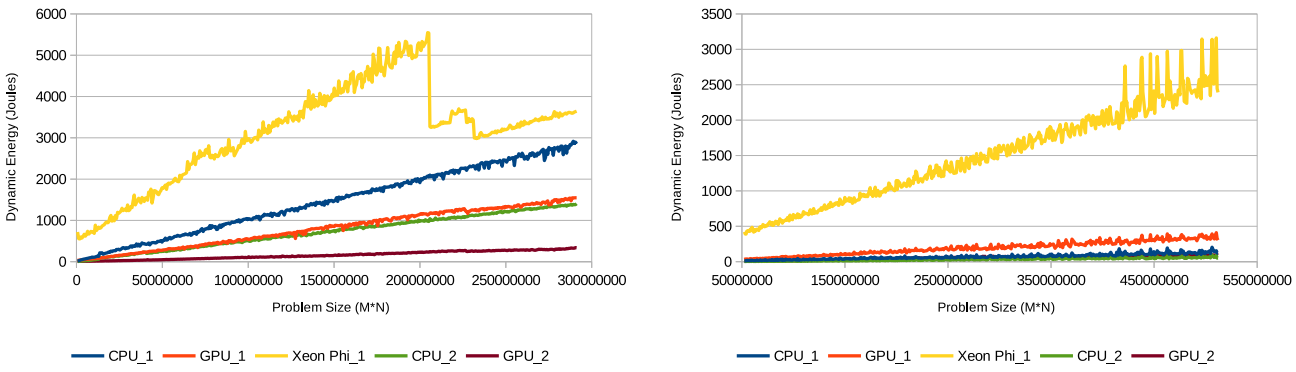


FIGURE 1 Block diagram of HCLServer1 containing three abstract processors



(A) DGEMM **(B)** 2D-FFT

FIGURE 2 Dynamic energy functions of the abstract processors on HCLServer1 and HCLServer2. A, DGEMM and B, 2D-FFT. 2D-FFT, 2D fast Fourier transform

1888, 0} resulting in dynamic energy consumption of 54 J and thereby providing 42.5% reduction in energy. These results motivate our work on a novel data partitioning algorithm for minimization of dynamic energy.

In this work, we propose a novel workload partitioning algorithm, heterogeneous energy optimization algorithm (HEOPTA) that determines optimal workload distribution minimizing the dynamic energy consumption of data-parallel applications executing on heterogeneous platforms for the most general shapes of dynamic energy profiles of the participating processors. HEOPTA is a workload partitioning algorithm that takes as input dynamic energy profiles of computations during the execution of a data-parallel application.

A significant challenge to the application of *HEOPTA* is the *energy modeling* of heterogeneous parallel applications running on hybrid platforms. Consider a parallel application running on a CPU, a GPU and an Intel Xeon Phi. HEOPTA requires the dynamic energy profile of each computational kernel separately. To model the performance of a parallel application and build its speed functions, the execution time of any computational kernel can be measured accurately using high precision processor clocks. There is, however, no such effective equivalent for measuring the energy consumption.

There are three mainstream approaches to providing the energy consumption of an application: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors, and (c) Energy predictive models. The first approach using power meters is known to be accurate at system-level[11] but it does not provide a fine-grained decomposition of the energy consumption during the application run in a hybrid platform. Now, an overview of the deficiencies with measurements using on-chip power sensors and energy predictive models is presented.

The energy measurement approach based on on-chip power sensors is now pervasively available in mainstream processors such as Intel and AMD Multicore CPUs, Intel Xeon Phis, and Nvidia GPUs. There are vendor-specific libraries to obtain the power data from these sensors. For example, running average power limit (RAPL)[12] can be used to monitor power and control frequency (and voltage) of Intel CPUs. Intel system management controller chip[13] and Nvidia NVIDIA management library (NVML)[14] provide the power consumption by Intel Xeon Phi and Nvidia GPUs, respectively. The accuracy of GPU on-chip sensors is reported in the NVML manual (±5%).[14] However, in general, there is not much documentation available on the accuracy of these vendor-specific libraries. Furthermore, there are many other issues related to power data obtained using on-chip sensors. For the GPU and Xeon Phi on-chip sensors, there is no information about how a power reading is determined that would allow one to determine its accuracy. However, for the CPU on-chip sensors, RAPL uses voltage regulators (VR IMON) for CPU and DRAM. VR IMON is an analog circuit within voltage regulator (VR), which keeps track of an estimate of the current.[15] However, there are two main issues with these measurements. First, how this estimate is determined. Second, the accuracy of the estimates is not reported in the vendor manual.

The third approach based on software energy predictive models emerged as a popular alternative to estimate energy consumption by an application. While the models provide the fine-grain estimation of energy consumption during the execution of an application at relatively low cost compared with the other approaches, there are research works highlighting their poor accuracy.[16-19]

We propose a practical methodology to determine this decomposition, which employs only system-level energy measurements using power meters. The methodology allows us to build discrete dynamic energy functions of abstract processors with sufficient accuracy for the application of HEOPTA.

The accuracy of our energy modeling methodology and the performance of HEOPTA are experimentally analyzed using two data-parallel applications, DGEMM and 2D-FFT, on a cluster of two heterogeneous nodes. We show that the load-imbalanced solutions provided by the algorithm achieve significant dynamic energy reductions (on the average 130% and 44%) compared with the load-balanced solutions.

An effective approach to save energy in green computing clusters and big data centers is to switch idle nodes off or to put them in the sleep mode.[20-24] Both strategies reduce the base energy of idle nodes resulting in less total energy consumption of the platform. However, applying a hybrid approach where each node consumes energy optimally during its computations and is switched off at other times can reduce the total energy consumption of the whole platform more effectively. Therefore, apart from minimizing dynamic energy consumption, minimizing total energy consumption is also an important objective. In this article, we experimentally examine if dynamic energy minimization using HEOPTA leads to minimizing total energy consumption.

The main original contributions of this work are:

- We present the first study on dynamic energy optimization of data-parallel applications on heterogeneous processors through optimal workload distribution.
- We propose a novel data partitioning algorithm that determines optimal workload distribution minimizing the dynamic energy consumption of applications executing on heterogeneous platforms for the most general shapes of dynamic energy profiles of the participating processors. The algorithm returns, generally speaking, nonbalanced solutions.
- We propose a methodology to determine the decomposition of dynamic energy consumption using system-level power measurements for heterogeneous platforms with sufficient accuracy, and experimental validation of the methodology on two modern heterogeneous hybrid servers.
- We experimentally demonstrate that performance optimization does not lead to dynamic energy optimization and also minimizing dynamic energy does not result in minimizing total energy on modern heterogeneous platforms.

The article is organized as follows. Section 3 presents related work. Section 4 presents the formulation of the heterogeneous dynamic energy optimization problem. Section 5 describes the data partitioning algorithm solving the problem. In Section 6, the device-level approach for dynamic energy modeling is illustrated. Section 7 presents the experimental results. Finally, Section 9 concludes the article.

2 | TERMINOLOGY

There are two types of energy consumptions, static energy, and dynamic energy. 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. The dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumed by the platform during the application execution. That is, if P_S is the static power consumption of the platform, E_T is the total energy consumption of the platform during the execution of an application, which takes T_E seconds, then the dynamic energy E_D is equal to $E_D = E_T - (P_S \times T_E)$.

3 | RELATED WORK

There are three popular approaches to measure energy in computing platforms: (a) Energy predictive (analytical) models, (b) measurements using on-chip power sensors, and (c) system-level physical measurements using external power meters.

In this section, we present an overview of analytical and nonanalytical methods for power/energy modeling and energy optimization proposed for multicore CPUs and accelerators. We then explain the rationale in using system-level power measurements using power meters, which is considered to be the ground truth.

3.1 | Analytical methods

Basmadjian et al[25] 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).

Rotem et al[12] present a software-based power model RAPL in Intel Sandybridge. RAPL estimates the energy consumption of core and uncore components based on some performance monitoring counters (PMCs) which are not disclosed. Lively et al[26] present power-predictive models for hybrid applications (MPI/OpenMP) based on PMCs. They apply Spearman's rank correlation and PCA to select appropriate performance counters.

Rofouei et al[27] estimate energy consumption on CPU-GPU systems using the multiplication of execution time with average power consumption of a device. Therefore, $E_{\text{CPU}} = t_{\text{CPU}} \times P_{\text{avg-CPU}}$, and $E_{\text{GPU}} = t_{\text{GPU}} \times (P_{\text{avg-GPU}} + P_{\text{idle-CPU}}) + E_{\text{transfer}}$, where t_{CPU} and t_{GPU}, respectively, represent CPU and GPU usages, $P_{\text{avg-CPU}}$ and $P_{\text{avg-GPU}}$ are the average power consumed by CPU and GPU, and E_{transfer} is the amount of energy consumed for data transfer between CPU and GPU.

Multicore power, area, and timing (McPAT)[28] is a framework modeling power, area, and timing for multicore and manycore processors. It considers most of the fundamental components in multicore processors such as cores, interconnects, shared caches, and memory controllers. Lim et al[29] highlight that power consumption of a GPU can be represented as the summation of the power consumptions of all modeled components. They use McPAT to estimate the power consumption of GPU components and adjust their model.

A component-level power consumption model is proposed in Reference 30. It analytically models the power consumption of 12 components in GPUs, from ALU to memory units. The power consumption is then estimated as the sum of power consumptions of all these components along with their access rates.

Nagasaka et al[31] propose PMC-based statistical power consumption modelling technique for GPUs that run CUDA applications based on linear regression and utilizes 13 PMCs

A power consumption model for GPUs is presented in Reference 32. It is based on linear regression tree and random forest methods. The model collects 22 runtime characteristics including 18 types of operations and four architecture parameters.

Song et al[33] present power and energy prediction models for GPUs which is based on machine learning algorithms such as back-propagation in artificial neural networks.

Kestor et al[34] present a system monitor interface between the OS and the user runtime that accounts for each core's power consumption. The proposed model considers the number of integer instructions, stalled cycles, LLC misses and the number of floating-point operations.

Choi et al[35] proposed an arch-line model which is an energy-based analogue of the time-based roofline model presented in Reference 36. The model visualizes energy and power consumptions based on algorithm-related parameters, including arithmetic and memory operations and

computation intensity of algorithm; as well as the machine characteristics, such as the time and energy costs per operation or per word of communication.

Shao and Brooks[37] develop an instruction-level energy consumption model for a Xeon Phi processor.

Jarus et al[38] present system-wide energy consumption models for servers, which is based on the analysis of performance counters. They use decision trees for finding an appropriate model for a given application.

Al-Khatib and Abdi[39] presents an operand-value-based model to estimate the dynamic energy consumption of FPGAs.

Shahid et al[40] propose a novel criterion called *additivity* to determine a subset of PMCs that can potentially be considered for reliable energy predictive modelling for an Intel Haswell-E5-2670 CPU. This criterion is based on the experimental observation that the energy consumption of a serial execution of two applications is the sum of energy consumptions observed for the individual execution of each application. They show that lots of PMCs, used in energy predictive models, are not additive, and therefore, bring into question the reliability and reported prediction accuracy of these models.

Chakrabarti et al[5] propose a data partitioning scheme addressing the energy consumption optimization on heterogeneous clusters. They estimate energy consumption using PVWATTS simulator. The proposed approach uses a linear programming formulation to solve the optimization problem.

3.2 | Nonanalytical methods

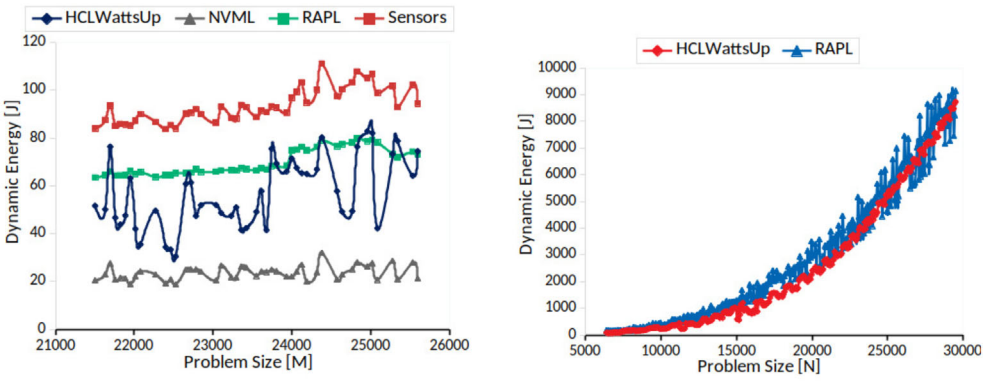
McCullough et al[41] demonstrate that linear-based power modelling approaches show high prediction error in modern computing platforms because of inherent complexities such as multiple cores, hidden device states, and large dynamic power components. They show power prediction errors can reach as high as 150% and propose direct measurement as an alternative to analytical-based techniques to deal with the inherent complexities arise by modern architectures.

O'Brien et al[18] study proposed models for power and energy prediction on the highly heterogeneous and hierarchical node architectures in modern HPC platforms. They come up with the idea that the inherent complexities, such as resource contention on LLC, NUMA, and dynamic power management, make analytical-based approaches less-accurate enough to model performance and energy on modern HPC systems. Finally, they conclude that direct measurement is the only accurate way to model the energy consumption of HPC platforms.

Lastovetsky and Manumachu[6,42] propose a model-based energy optimization algorithm on tightly integrated multicore CPUs. Due to inherent complexities (contentions on shared resources and NUMA), they highlighted that the shapes of energy profiles get so complicated that cannot be modeled using linear techniques. They studied the real-life profiles of single and multithread applications and concluded as the number of threads increases, the fluctuations in the energy profiles also increase. Due to the inherent complexities, they used direct energy measurement, rather than analytical modelling techniques, to build real-life energy profiles of parallel applications.

Using on-chip power sensors is another popular approach to measure energy. Fahad et al[43] present a comprehensive comparative study to illustrate the shortcomings of this approach using two case studies. For the first case study, they use the dynamic energy profile of 2D (CUDA) FFT on Nvidia Tesla P100 GPU for workload sizes ranging from 21504×25600 to 25600×25600 using a constant step size of 64. The dynamic energy consumption by 2D FFT is measured with RAPL and NVML.[14] They term them collectively as *sensors*. The energy measurements using sensors are compared against HCLWattsUp API, which provides the system-level energy measurements using power meters. Figure 3A presents the dynamic energy profiles of 2D FFT using HCLWattsUp and the on-chip sensors. The maximum and average errors of profiles given by the sensors and HCLWattsUp are 176% and 73%, respectively.

For the second case study, consider the dynamic energy profile of 2D (MKL) FFT on Intel Xeon Gold 6152 (HCLServer2) for workload sizes ranging from 6400×6400 to 29504×29504 using a constant step size of 64. The dynamic energy consumption by 2D FFT is measured with RAPL.



(A) CUDA FFT (B) MKL FFT

FIGURE 3 Dynamic energy functions of FFT on (i) Nvidia Tesla P100 and (ii) Intel Xeon Gold 6152

The energy measurements using RAPL are compared against HCLWattsUp API. Figure 3B presents the dynamic energy profiles of 2D FFT using HCLWattsUp and RAPL. The maximum and average errors of profiles given by the sensors and HCLWattsUp are 205% and 36%, respectively.

To summarize, in the single-core processors' era, analytical approaches were able to precisely estimate the energy consumption of applications using a few architectural and program parameters. However, the tight integration of multicore CPUs with many-core accelerators incurs new complexities, such as contentions on shared resources and NUMA. These complexities make the state-of-the-art energy measurements based on on-chip sensors and energy predictive models suffer from poor accuracy. Therefore, an alternative is to use system-level energy measurements of applications provided by power meters to model their energy consumptions.

In addition, apart from a few variation-aware algorithms which consider workload distribution as a decision variable for energy optimization on *homogeneous* HPC platforms,[6,42] all proposed methods assume a linear relationship between workload size and energy consumption. Nevertheless, regarding some aforementioned efforts,[6,18,41,42] one can conclude that profiles on modern HPC platforms are highly nonlinear that makes the relationship between workload size and energy consumption so complex, nonlinear and even non-convex. Therefore, workload distribution has now become an important decision variable that cannot be ignored in solving the energy optimization problem on modern heterogeneous platforms.

4 | FORMULATION OF HETEROGENEOUS DYNAMIC ENERGY OPTIMIZATION PROBLEM

Consider a workload size n executing on p abstract processors (multithreaded kernels). The dynamic energy consumption of each abstract processor p_i is modeled by a discrete dynamic energy function, $e_i(x)$, and contains m data points. $e_i(x)$ represents the dynamic energy consumption during the execution of the workload size x. The set of dynamic energy functions of the p abstract processors is given by $E = \{e_0(x), \ldots, e_{p-1}(x)\}$.

The heterogeneous dynamic energy optimization problem can be formulated as follows:

Heterogeneous dynamic energy optimization problem, HEOPT($n, p, m, E, X_{opt}, e_{opt}$): The problem is to find a distribution, $X_{opt} = \{x_0, \ldots, x_{p-1}\}$, for the workload n on p abstract processors that minimizes total dynamic energy consumption during parallel execution of n. The parameters (n, p, m, E) are the inputs to the problem. The outputs are X_{opt}, which is the optimal solution (workload distribution), and e_{opt}, which represents the dynamic energy consumption of the optimal solution. The formulation below is an integer nonlinear programming problem.

$$e_{opt} = \min_{X} \sum_{i=0}^{p-1} e_i(x_i) \quad \text{Subject to} \quad \sum_{i=0}^{p-1} x_i = n,$$
$$\text{where} \quad p, m, n \in \mathbb{Z}_{>0} \quad \text{and} \quad x_i \in \mathbb{Z}_{\geq 0} \quad \text{and} \quad e_i(x) \in \mathbb{R}_{>0} \tag{1}$$

The objective function in Equation (1) is a function of workload distribution X, $X = \{x_0, \ldots, x_{p-1}\}$, for a given workload n executing on the p processors. The function returns the amount of dynamic energy which is consumed by running each given distribution X on processors $\{P_0, \ldots, P_{p-1}\}$. The total dynamic energy consumption of X is calculated as the summation of all dynamic energies consumed by the p processors $\{P_0, \ldots, P_{p-1}\}$ which run X in parallel. The distribution with minimum dynamic energy consumption is returned as the optimal distribution. It should be noted that the number of active processors (processors with nonzero workload sizes) in the optimal solution determined by HEOPTA (X_{opt}) might be less than p.

5 | HEOPTA: ALGORITHM SOLVING HEOPT PROBLEM

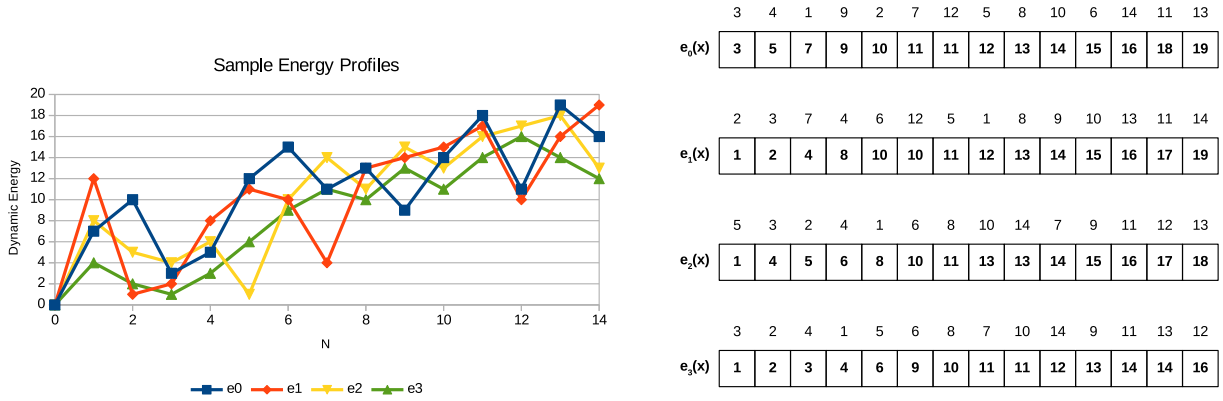
In this section, we will introduce HEOPTA, a branch-and-bound algorithm solving HEOPT. It utilizes two bounding criteria, *energy threshold* and *size threshold*, to find the optimal workload distribution in a polynomial complexity of $O(m^3 \times p^3)$.

Consider a workload $n = 12$ executed using four heterogeneous processors ($p = 4$). Figure 4A shows the discrete dynamic energy functions that are input to EOPTA, $E = \{e_0(x), \ldots, e_3(x)\}$, with a cardinality of 14 ($m = 14$). Figure 4B shows the discrete dynamic energy functions which are stored as arrays in nondecreasing order of energy consumption.

HEOPTA, as a branch-and-bound algorithm, gradually forms a rooted tree and explores branches of this tree, in depth-first order, to find optimal workload distributions. Figure 5 shows the tree explored by HEOPTA which contains all the combinations for $n = 12$ and $p = 4$. Due to the lack of space, the tree is shown partially.

HEOPTA starts tree exploration from the root at the level L_0 of the tree. The root node is labeled by 12 which represents the whole workload to be distributed between four processors $\{P_0, P_1, P_2, P_3\}$. Then, 15 ($= m + 1$) problem sizes, including a zero problem size along with all problem sizes in the dynamic energy function $e_0(x)$, are assigned to the processor P_0 one at a time in depth-first order. The problem sizes are assigned in nondecreasing order of their energy consumption. The value, which labels an internal node at level L_1 (root's children), determines the remaining workload to be distributed between processors $\{P_1, P_2, P_3\}$.

After assigning a problem size to P_0 in the level L_0, the algorithm proceeds to solve HEOPT in L_1. Therefore, each child of the root in L_0 is a root of a subtree in the next level L_1, which is a solution tree to solve HEOPT for the remaining workload between three processors $\{P_1, P_2, P_3\}$. Each



(A) Sample dynamic energy functions **(B)** Functions stored in arrays

FIGURE 4 A, Dynamic energy functions of a sample application executing on four heterogeneous processors. B, The same functions stored in arrays

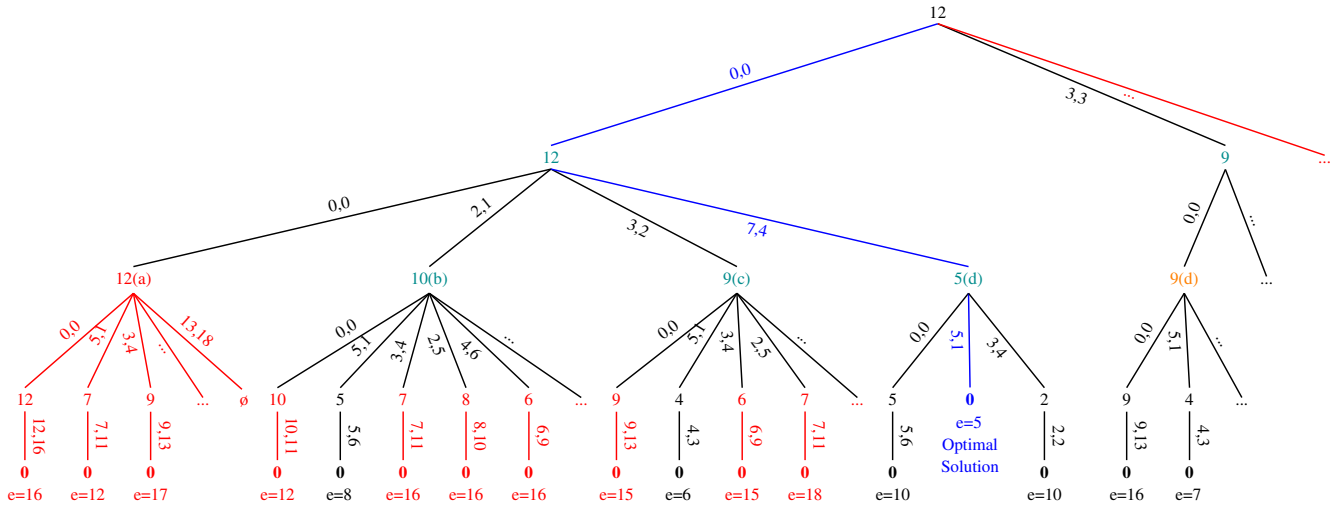


FIGURE 5 Applying naive approach to examine all combinations and select a workload distribution with the minimum dynamic energy consumption

edge, which connects the root and its child, is labeled by the problem size assigned to P_0 and its energy consumption. Similarity, HEOPTA explores branches in depth-first order until it reaches a leaf. Generally, in this tree, any leaf node labeled by 0 illustrates one of the possible solutions, where its dynamic energy consumption, is calculated as the summation of the consumed energies labeling the edges in the path connecting the root and the solution leaf. *No-solution* leaves are labeled by ⌀.

Before expanding each branch, the algorithm checks the branch against two upper estimated bounds, *energy threshold* and *size threshold*, and it is discarded if cannot produce a better distribution. The *energy threshold*, represented by ε, is initialized to the dynamic energy consumption of load-equal distribution, allocating each processor the same workload of size $\frac{n}{p}$ (assuming n is divisible by p). HEOPTA will not examine data points in the energy functions with the dynamic energy consumption greater than or equal to the energy threshold. In the example, ε will be initialized by $10\ (\sum_{i=0}^{3} e_i(\frac{12}{4}) = (3+2+4+1) = 10)$. Therefore, data points with dynamic energy consumption less than 10 will only be considered, forming the reduced search space.

The *size threshold* assigns each level of the tree a threshold, $\sigma_i, i \in \{0, \ldots, p-1\}$, which represents the maximum workload that can be executed in parallel on processors $\{P_i, \ldots, P_{p-1}\}$ so that the dynamic energy consumption by every processor $\{P_i, \ldots, P_{p-1}\}$ is less than ε. In this example, the maximum workloads with the dynamic energy consumptions less than $\varepsilon = 10$ in the dynamic energy functions for processors P_0, P_1, P_2, and P_3 are 9, 7, 5, and 6, respectively. The size threshold vector, σ contains four elements, $\sigma = \{\sigma_0, \sigma_1, \sigma_2, \sigma_3\}$, where the size threshold for L_3 (σ_3) is equal to 6, σ_2 is 11 ($= \sigma_3 + 5$), σ_1 is set to 18 ($= \sigma_2 + 7$), and finally σ_0 would be 27 ($= \sigma_1 + 9$). Once ε changes, the size threshold array σ is also updated using the new ε.

All subtrees, eliminated from the search space by applying the two bounding criteria, are highlighted in red in Figure 5. This key optimization operation is called *Cut*. Apart from cutting useless branches in the tree, HEOPTA saves the solutions which are found during the tree exploration. As an example, consider the solution $\{(0, 0), (2, 1), (5, 1), (5, 6)\}$ with an energy consumption of 8. Each pair like (a, b) in the solution represents an allocated

workload size where a is the workload size and b determines the energy consumption for executing a. First, HEOPTA updates the energy threshold to 8 ($\varepsilon = 8$). The vector of size thresholds, σ, is then updated to {21, 17, 0, 5}. The solution is also memorized, which includes saving the information pertaining to all the levels of the tree except for the first and the last and the levels whose consumed energies go beyond ε. Thus, the information that is saved is level-specific. For L_1, the memorized information includes the problem size assigned to P_1, which is 2 and the total dynamic energy consumption of the solution for processors $\{P_1, P_2, P_3\}$, which is 8, The same is done for L_2. The memorized information includes the problem size assigned to P_2, which is 5 along with the total dynamic energy consumption of the solution for processors $\{P_2, P_3\}$, which is equal to 7. Green nodes in the tree highlight ones whose solutions are saved. This key operation is called *Save*. This allows HEOPTA to read the memory before exploring a node and retrieves its solution (if it has already been saved). This key operation is called READMEMORY. The solution of the orange node in the tree is retrieved from the memory.

In this example, the distribution {(0, 0), (7, 4), (5, 1), (0, 0)}, highlighted in blue, with the consumed dynamic energy of 5, represents the optimal solution.

In summary, HEOPTA uses three key operations, *Cut*, *Save*, and READMEMORY, to find the optimal solutions. In supplemental available online, we elucidate using an example how these key operations reduce the search space of solutions. The correctness and complexity proofs of HEOPTA are presented in the supplemental available online.

5.1 | Formal description of HEOPTA

Algorithm 1 shows the pseudocode of HEOPTA. It takes as inputs: the problem size, n, the number of heterogeneous processors, p, and an array of p discrete dynamic energy functions, $E = \{E_0, E_1, \ldots, E_{p-1}\}$. E_i represents the dynamic energy function of processor P_i and consists of m pairs (x_{ij}, e_{ij}), $j \in [0, m)$ where x_{ij} is the jth problem size in the function, and e_{ij} represents the amount of dynamic energy consumed by P_i to run x_{ij}. HEOPTA returns two outputs: the optimal workload distribution, X_{opt}, and its optimal dynamic energy consumption, e_{opt}. It should be noted that the number of processors selected by the algorithm (processors with nonzero workloads) in the optimal workload distribution may be less than p.

Algorithm 1. Algorithm Finding Optimal Workload Distribution of Size n for Minimizing Dynamic Energy Consumption

1: **function** HEOPTA($n, p, E, X_{opt}, e_{opt}$)
INPUT:
Problem size, $n \in \mathbb{Z}_{>0}$
Number of processors, $p \in \mathbb{Z}_{>0}$
Dynamic energy functions, $E = \{E_0, ..., E_{p-1}\}$,
$E_i = \{(x_{ij}, e_{ij}) \mid i \in [0, p), j \in [0, m), x_{ij} \in \mathbb{Z}_{>0}, e_{ij} \in \mathbb{R}_{>0}\}$.
OUTPUT:
Optimal workload distribution, $X_{opt} = \{x_{opt}[0], ..., x_{opt}[p-1]\}$,
$x_{opt}[i] \in \{\bigcup_{j=0}^{m-1} x_{ij} \cup \{0\}\}, i \in [0, p)$.
Total dynamic energy consumption, $e_{opt} \in \mathbb{R}_{>0}$

2: $E \leftarrow E \cup Sort_{\uparrow}(E)$
3: $x_{opt}[i] \leftarrow \frac{n}{p}, \forall i \in [0, p-1]$
4: $x_{opt}[i] \leftarrow x_{opt}[i] + 1, \forall i \in [0, n\%p)$
5: $\varepsilon \leftarrow \sum_{i=0}^{p-1}$ GETENG($E_i, x_{opt}[i]$)
6: $\sigma \leftarrow$ SIZETHRESHOLDCALC(p, E, ε)
7: $Mem[i][j] \leftarrow \emptyset, \forall i \in [1, \cdots, p-2], j \in [0, \cdots, n]$
8: HEOPTA_KERNEL($n, p, 0, E, \varepsilon, \sigma, 0, X_{cur}, Mem, X_{opt}$)
9: $e_{opt} \leftarrow \varepsilon$
10: **return** (X_{opt}, e_{opt})
11: **end function**

The algorithm first sorts each profile in nondecreasing order of dynamic energy consumption (Line 2). After that, the array X_{opt} and the energy threshold ε are initialized to the load-equal distribution and its corresponding dynamic energy consumption, respectively (Lines 3-5). The vector of size thresholds, σ, is then determined using the function SIZETHRESHOLDCALC (Line 6).

In line 7, the data structure for saving solutions, matrix *Mem*, which consists of $(p-2) \times (n+1)$ elements, is initialized. It will save the solutions found for processors $\{P_1, \ldots, P_{p-2}\}$. Next, HEOPTA invokes the recursive routine HEOPTA_KERNEL to find the optimal workload distribution.

Function GETENGE_i, x (called in Line 5) returns the dynamic energy consumption of a problem size x running on P_i (The value is extracted from E_i). It returns 0 when x equals 0. The pseudocodes of all functions, used in Algorithms 1 and 2, and the structure of *Mem* can be found in the supplemental which is available online.

5.1.1 | Recursive algorithm *HEOPTA_Kernel*

HEOPTA_KERNEL (Algorithm 2) is a recursive function, deploying the key three operations, *Cut*, *Save* and READMEMORY to solve HEOPT problem efficiently. The variable c indicates the level of a node which is being processed in a solution tree. It is initialized to 0 in the first invocation of HEOPTA_KERNEL, the next recursive invocation deals with L_1 (ie, $c = 1$), and so on. The vector $X_{\text{opt}} = \{x_{\text{opt}}[0], \ldots, x_{\text{opt}}[p-1]\}$ holds the best solution found so far where its dynamic energy consumption is in ε. The array X_{cur} is used to hold problem sizes currently assigned to processors $P_i (i \in [0, p-1])$.

Algorithm 2. Algorithm of Recursive Kernel Invoked by Algorithm 1

1: **function** HEOPTA_KERNEL($n, p, c, E, \varepsilon, \sigma, X_{cur}, Mem, X_{opt}$)
2: **if** CUT(n, σ_c) **then**
3: **return**
4: **end if**
5: **if** $c = p - 1$ **then**
6: **if** GETENG(E_c, n) $< \varepsilon$ **then**
7: $x_{cur}[p-1] \leftarrow n$
8: PROCESSSOLUTION($p, E, \varepsilon, \sigma, X_{cur}, Mem, -1, X_{opt}$)
9: **end if**
10: **return**
11: **end if**
12: **if** $c > 0 \wedge c \leq p - 2$ **then**
13: $status \leftarrow$ READMEMORY($n, p, c, \varepsilon, E, X_{cur}, Mem, idx$)
14: **if** $status = NOT\_SOLUTION$ **then**
15: **return**
16: **else if** $status = SOLUTION$ **then**
17: PROCESSSOLUTION($p, E, \varepsilon, \sigma, X_{cur}, Mem, c, X_{opt}$)
18: **return**
19: **end if**
20: **end if**
21: $idx \leftarrow -1, x_{c\ idx} \leftarrow 0$
22: **while** GETENG($E_c, x_{c\ idx}$) $< \varepsilon$ **do**
23: $x_{cur}[c] \leftarrow x_{c\ idx}$
24: **if** $x_{c\ idx} = n$ **then**
25: $x_{cur}[i] \leftarrow 0, \forall i \in [c+1, \cdots, p-1]$
26: PROCESSSOLUTION($p, E, \varepsilon, \sigma, X_{cur}, Mem, -1, X_{opt}$)
27: **end if**
28: **if** $n > x_{c\ idx}$ **then**
29: HEOPTA_KERNEL($n - x_{c\ idx}, p, c+1, E, \varepsilon, \sigma, X_{cur}, Mem, X_{opt}$)
30: **end if**
31: **if** $idx + 1 = m$ **then**
32: **break**
33: **end if**
34: $idx \leftarrow idx + 1$
35: **end while**
36: MAKEFINAL($Mem[c][n]$)
37: **end function**

The function CUT(n, σ_c), applying the key operation *Cut*, compares the workload n with the corresponding size threshold σ_c to decide whether to expand the node or cut the subtree in level c (Lines 2-4).

Lines 5 to 11 process the solutions found in the last level L_{p-1}. Generally, once a solution is found, the routine PROCESSSOLUTION is invoked to perform the following operations:

- If X_{cur} is better than the current best solution, X_{opt}, the energy threshold ε will be reduced to the dynamic energy consumption of X_{cur}, and X_{opt} will be updated to X_{cur}.
- Should ε decrease, the size threshold vector σ is correspondingly updated.
- The operation *Save* is invoked to save X_{cur} in the memory.

Prior to expanding a node with a label of n at a given level c, the function READMEMORY is called to retrieve the solution for n on processors $\{P_c, \ldots, P_{p-1}\}$, provided it exists (Lines 12-20).

The optimal and intermediate solutions are stored in *Mem*. A memory cell which contains the optimal distribution is labeled *Finalized*. The variable *status* determines the type of the retrieved solution. If there is no solution stored in a finalized cell or the total amount of dynamic energy consumption for the retrieved solution is greater than or equal to ε (given by the status, *NOT_SOLUTION*), we return from HEOPTA_KERNEL. If the stored solution in the *Mem* is the optimal one (given by the status, *SOLUTION*), the retrieved solution is used and the process returns from HEOPTA_KERNEL. If none of the above cases occur, it means that the node has not already been examined, and the routine starts expanding the current node by scanning the dynamic energy profile E_c from left to right.

The variable *idx*, ranging from -1 to $m-1$, determines indexes of data points in the sorted dynamic energy functions. Line 21 initializes *idx* to -1 and x_{cidx} to zero. Generally, x_{cidx} determines the *idx*th problem size in profile E_c, in case *idx* is not -1.

The *while* loop (Lines 22-35) scans the dynamic energy profile E_c from left to right examining data points with dynamic energy consumption less than the energy threshold ε. The array $X_{\text{cur}} = \{x_{\text{cur}}[0], \ldots, x_{\text{cur}}[p-1]\}$ where $x_{\text{cur}}[i] \in \{\cup_{j=0}^{m-1} x_{ij} \cup \{0\}\}$, $i \in [0, p)$, is used to store problem sizes currently assigned to processors P_i. In each iteration, the data point *idx* is extracted from E_c, and its workload, which is x_{cidx}, is stored in array X_{cur} (Line 23). If this workload (x_{cidx}) is equal to n, we found a solution. In this case, the solution is processed using PROCESSSOLUTION, otherwise, if x_{cidx} is less than n, HEOPTA_KERNEL is reinvoked to solve HEOPT for the remaining workload $n-x_{cidx}$ at the next level L_{c+1} (Lines 24-30). If x_{cidx} greater than n, this data point is declined and the next one is processed.

Lines 31 to 34 check if the algorithm reaches the end of the function E_c. If this is the case, the *while* loop (Line 22-35) terminates, and the corresponding memory cell is finalized (Line 36). Otherwise, *idx* is incremented moving to the next data point in the dynamic energy function E_c.

6 | DEVICE-LEVEL DYNAMIC ENERGY DECOMPOSITION IN HETEROGENEOUS HYBRID PLATFORMS

We describe our practical approach here to construct the discrete dynamic energy profiles of the abstract processors in a hybrid heterogeneous server. The method is based purely on system-level power measurements. The approach comprises of two main steps. The first step is the identification or grouping of 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 the construction of 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

Individual computing elements executing an application are grouped together in such a way that we can accurately measure the energy consumption of the group. These groups are called *abstract processors*. We consider two properties essential to composing the groups:

- *Completeness:* An abstract processor must contain only those computing elements which execute the given application kernel.
- *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 processors.

Based on this grouping into abstract processors, we hypothesize that the total dynamic energy consumption during an application execution will equal the sum of energies consumed by all the abstract processors. 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.

Apart from computing elements, there are other resources such as network interface controller (NIC), solid state drive, fans, chipsets, and so on, which are almost shared between all abstract processors and consume energy during application execution. To eliminate their potential contribution in the dynamic energy consumption of a given abstract processor, the following precautions for energy measurements are taken into consideration:

- Since consuming a significant and variable amount of energy during application execution, fans are set at full speed before running the application to eliminate their contribution. Thus, they run consistently at the same speed and consume the same amount of energy which is then considered part of the static energy of the platform. This way, the dynamic energy consumption of a given abstract processor is not affected by fans.
- Disk utilization is monitored during the application run to ensure that the disk activity is negligible during the execution of our applications. It is ensured that the problem size used in the execution of an application does not exceed the main memory, where swapping (paging) does not occur. That is, problem sizes are bounded by main memory size.
- NIC is also monitored to make sure that the network is not used by the application. It should be mentioned that communications are out of the scope of this work.

6.2 | Energy models of abstract processors

We describe here the second main step of our approach, which is to build 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 available 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 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, the total dynamic energy consumption is obtained from the system-level measurement 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. The number of experiments reduces to $p \times m$ by employing our additive hypothesis.

7 | EXPERIMENTAL RESULTS

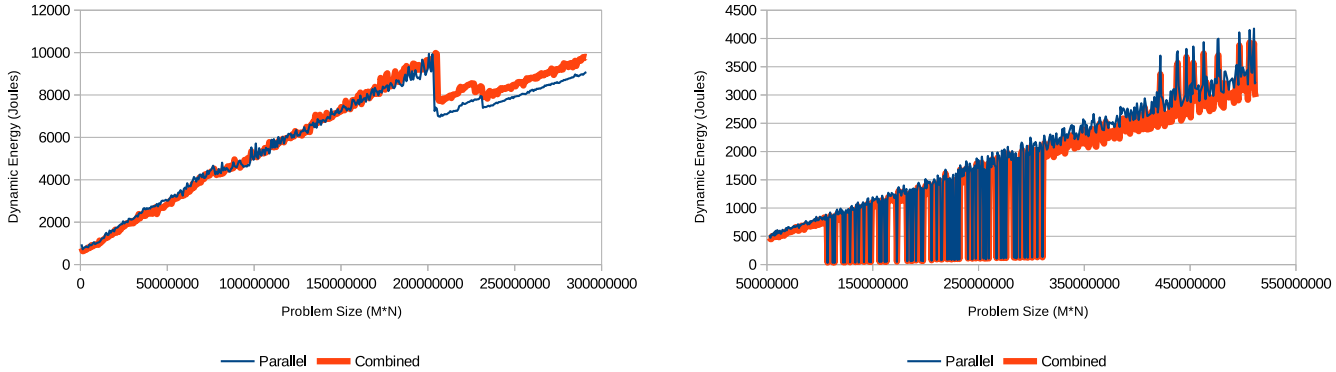
We employ two connected heterogeneous multiaccelerator NUMA nodes, HCLServer1 (Table 1) and HCLServer2 (Table 2). HCLServer1 is modeled by three abstract processors, CPU_1, GPU_1 and PHI_1, as described earlier. HCLServer2 is modeled by two abstract processors, CPU_2 and GPU_2.

Two popular data-parallel applications, matrix-matrix multiplication (DGEMM) and 2D-FFT are used for our experimental analysis. A hybrid data-parallel application executing on the servers in parallel consists of five kernels, one kernel per computational device. Figure 2 shows discrete dynamic energy functions for the five abstract processors for DGEMM and 2D-FFT. For the DGEMM application, the workload sizes range from 64×10112 to 28800×10112 with a step size of 64 for the first dimension m. For the 2D-FFT application, the workload sizes range from 1024×51200 to 10000×51200 with a step size of 16 for the first dimension m.

We use the member function *gettimeofday()* of the Linux library *sys/time.h* to measure the execution time of each kernel in our applications separately. For measuring dynamic energy consumption, each node is facilitated with one WattsUp Pro power meter which sits between the wall A/C outlets and the input power sockets of the node. Each power meter captures the energy consumption of one node. *HCLWattsUp* API[10] is used to gather the readings from the meter to determine the dynamic energy consumption during the execution of an application using two macros *HCL_WATTSUP_START* and *HCL_WATTSUP_STOP*. The first macro starts gathering power readings from the power meter, whereas the *HCL_WATTSUP_STOP* stops gathering and return the total energy as a sum of these power readings. This approach is explained in detail in the supplemental available online. HCLWattsUp has no extra overhead and therefore does not influence the energy consumption of the application execution. Fans are significant contributors to energy consumption. To rule out the contribution of fans in dynamic energy consumption, we set the fans at full speed before executing an application so that the energy consumption due to fans is included only in the static energy consumption of the platform. For each data point in the functions, the experiments are repeated until sample means of all the five kernels executing on the abstract processors fall in the confidence interval of 95%, and precision of 0.1 (10%) is achieved.

7.1 | Analyzing the accuracy of the additive approach

In this section, the accuracy of the additive approach is experimentally validated. We determine the dynamic energy consumption of application kernels executing in parallel on their corresponding abstract processors, and term it as *parallel* dynamic energy profile for illustration purposes. We



(A) DGEMM **(B)** 2D-FFT

FIGURE 6 Parallel and Combined dynamic energy functions for, A, DGEMM and B, 2D-FFT applications on HCLServer1 and HCLServer2. 2D-FFT, 2D fast Fourier transform

TABLE 3 Percentage difference of dynamic energy consumption of parallel to combined for the DGEMM and 2D-FFT applications

Platform	Min	Max	Average
(a) DGEMM			
HCLServer01	0.026%	29.2%	6.38%
HCLServer02	0.012%	29.03%	3.8%
BOTH	0.004%	26.1%	5.9%
Platform	Min	Max	Average
(b) DGEMM			
HCLServer01	1.8%	18.4%	9.1%
HCLServer02	0.02%	28.8%	12.4%
BOTH	0.16%	24.7%	8.3%

Abbreviation: 2D-FFT, 2D fast Fourier transform.

determine the dynamic energy consumption of the same application kernels executing serially by keeping all other experiment settings the same. We compute the sum value of the dynamic energy consumptions by these individual application kernels. This sum is termed the *combined* dynamic energy profile for illustration purposes. The additive hypothesis holds if the percentage error between combined and parallel dynamic energy profiles is within the user-specified accuracy.

Four profiles for HCLServer1 (one parallel and one for each of the three abstract processors), and three profiles for HCLServer2 (one parallel and one for each of the two abstract processors) are built. Figure 6A, B show the dynamic energy functions for parallel and combined executions of DGEMM and 2D-FFT applications on all the abstract processors.

Table 3A, B summarize the percentage difference of parallel to combined for the two applications. The percentage error between combined and parallel dynamic energy profiles are measured as $Error(\%) = |(E(x)_{\text{combined}} - E(x)_{\text{parallel}})|/E(x)_{\text{parallel}} \times 100$, where $E(x)_{\text{parallel}}$ and $E(x)_{\text{combined}}$ represent the energy consumption by parallel and combined profiles for workload size x. We find the average percentage error of DGEMM combined energy profile with parallel as 6.38% on HCLServer1, and 3.8% on HCLServer2. The average error of 2D-FFT on HCLServer1 is 9.1% and 12.4% on HCLServer2.

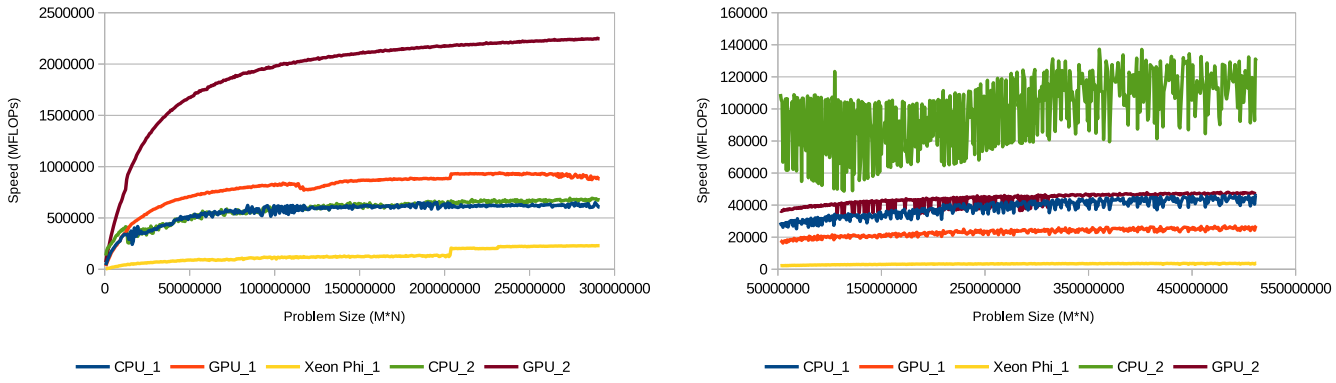
7.2 | Analyzing HEOPTA

HEOPTA is analyzed using two sets of experiments. For the first set, the dynamic energy consumption of solutions determined by HEOPTA is compared against the dynamic energy of load-balanced solutions. Load-balanced solutions are workload distributions with equal execution times for each abstract processor. The number of active processors in a solution (those assigned nonzero workload size) may be less than the total number of available processors. The dynamic energy saving against load-balancing algorithm is obtained as follows: $\text{Energy\_Saving}_{\text{balance}}(\%) = \frac{e_{\text{balance}} - e_{\text{heopta}}}{e_{\text{heopta}}} \times 100$, where e_{balance} and e_{heopta} are the dynamic energy consumptions of solutions determined by load-balancing and HEOPTA algorithms.

For the second set, we examine the interplay between dynamic energy optimization and performance optimization using the workload distribution determined by heterogeneous performance optimization algorithm (HPOPTA). HPOPTA[44] is a model-based data partitioning algorithm that minimizes the parallel execution time for the most general shapes of performance profiles for data-parallel applications executing on heterogeneous clusters of hybrid nodes. In fact, HPOPTA finds workload distributions minimizing the execution time of parallel applications. Optimal solutions found by HPOPTA may not be balanced in terms of execution time. The inputs to HPOPTA are discrete execution time (or performance) functions. The energy saving of HEOPTA against HPOPTA is obtained as follows: $\text{Energy\_Saving}_{\text{hpopta}}(\%) = \frac{e_{hpopta} - e_{\text{heopta}}}{e_{\text{heopta}}} \times 100$, where e_{hpopta} represents the dynamic energy consumption of the solution determined by HPOPTA. The goal of the second set of experiments is to find out if optimizing a data-parallel application for performance also optimizes the application for dynamic energy. If it is not the case, then we have a biobjective optimization problem for dynamic energy and performance to solve.

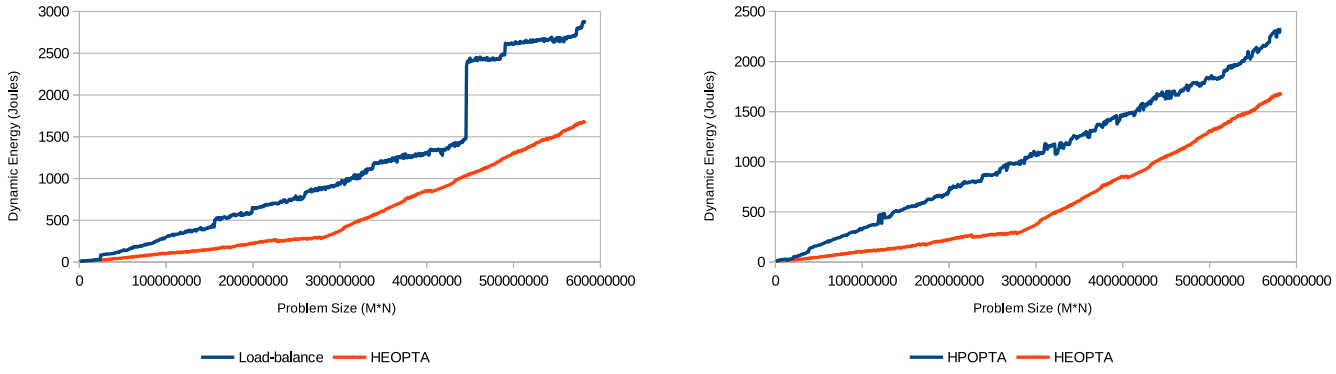
The experimental dataset for DGEMM contains the workload sizes, $\{64 \times 10112, 128 \times 10112, \ldots, 57600 \times 10112\}$. Figures 2A and 7A show energy and performance functions of DGEMM for the five abstract processors. We need the speed functions to obtain e_{balance} and e_{hpopta}. The minimum, average, and maximum reductions in the dynamic energy consumption of HEOPTA against load-balancing algorithm, $\textit{Energy\_Saving}_{\text{balance}}$, are 0%, 130%, and 257%. Zero percentage improvement represents the same workload distribution is determined by HEOPTA and load-balancing algorithm. These values for $\text{Energy\_Saving}_{\text{hpopta}}$ are 0%, 145%, and 314%. Figure 8 compares HEOPTA against the dynamic energy consumption of solutions determined by load-balancing and HPOPTA. According to these results, optimization for performance increases dynamic energy consumption by an average of 145%.

The experimental dataset for 2D-FFT includes workload sizes, $\{1024 \times 51200, 1040 \times 51200, \ldots, 20000 \times 51200\}$. Figures 2B and 7B show energy and performance functions of DGEMM for the five abstract processors. The minimum, average, and maximum dynamic energy reductions of HEOPTA against load-balancing algorithm, $\text{Energy\_Saving}_{\text{balance}}$, are 0%, 44%, and 105%. The minimum, average, and maximum of $\text{Energy\_Saving}_{\text{HPOPTA}}$ are 0%, 32%, and 77%. Figure 9 compares HEOPTA against the dynamic energy consumption of solutions determined by load-balancing and HPOPTA. Therefore, optimization for performance increases dynamic energy consumption by an average of 32%.



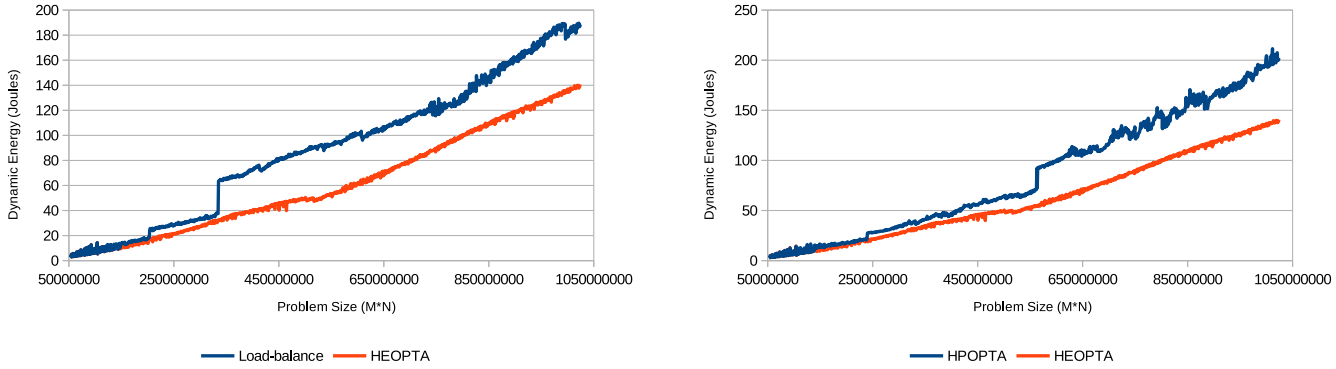
(A) DGEMM **(B)** 2D-FFT

FIGURE 7 Performance functions for the five abstract processors on HCLServer1 and HCLServer2 for, A, DGEMM and B, 2D-FFT. 2D-FFT, 2D fast Fourier transform



(A) HEOPTA vs load-balancing **(B)** HEOPTA vs HPOPTA

FIGURE 8 Dynamic energy consumption of DGEMM executed using HEOPTA in comparison with A, load-balanced solutions and B, HPOPTA. HPOPTA, heterogeneous performance optimization algorithm



(A)HEOPTA vs load-balancing (B) HEOPTA vs HPOPTA

FIGURE 9 Dynamic energy consumption of the 2D-FFT application executed using HEOPTA in comparison with A, load-balanced solutions and B, HPOPTA. 2D-FFT, 2D fast Fourier transform; HPOPTA, heterogeneous performance optimization algorithm

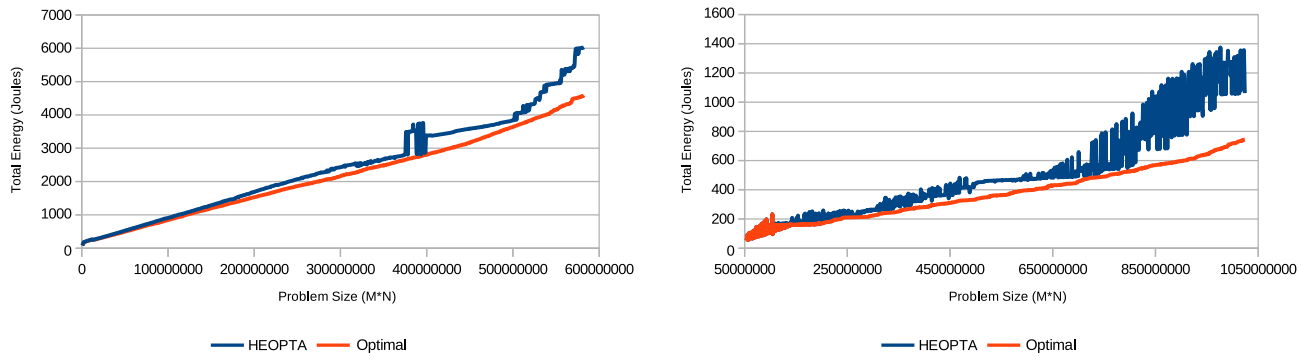
HEOPTA is executed on a single core of a multicore CPU in our experimental testbed. For any workload size in the experimental datasets for matrix multiplication and 2D-FFT applications, the algorithm has execution times less than 1 second. The amount of energy consumed by the algorithm is negligible compared with the energy consumption of the applications.

7.3 | Dynamic energy optimization vs total energy optimization

In this section, we analyse results of HEOPTA in terms of total energy minimization. Our experiments are conducted with DGEMM and 2D-FFT applications and compare the total energy consumption of solutions returned by HEOPTA over minimum total energy consumption for any problem size in the same experimental datasets employed earlier in Section 7. Total energy consumption of solutions returned by HEOPTA is calculated as explained in Section 2. Since there is no algorithm to find the optimal workload distribution minimizing total energy consumption, we exhaustively explore all possible distributions in the solution space to obtain optimal total energy. It should be mentioned that this algorithm has exponential time complexity.

The percentage of total energy saving over HEOPTA solutions is calculated for the aforementioned datasets. Total energy saving is calculated as follows: Total Energy Saving $= \frac{te_{\text{HEOPTA}} - te_{\text{opt}}}{te_{\text{opt}}} \times 100$, where te_{HEOPTA} is total energy consumption of the solution with optimal dynamic energy consumption and te_{opt} is the optimal total energy consumption obtained via exhaustive exploration of the solution space. The minimum, average and maximum total energy savings for the DGEMM application are 0, 11% and 37%, respectively. These values for the 2D-FFT application are, respectively, 0, 29% and 106%.

Figure 10A,B show the optimal total energy consumption over the total energy consumption of the distribution with minimum dynamic energy consumption for DGEMM and 2D-FFT, respectively.



(A) DGEMM (B) 2D-FFT

FIGURE 10 Total energy profiles of A, DGEMM and B, 2D-FFT applications for two different workload distributions HEOPTA and optimal solutions executing on HCLServer1 and HCLServer2. 2D-FFT, 2D fast Fourier transform; HPOPTA, heterogeneous performance optimization algorithm

8 | DISCUSSION

We experimentally validate our practical approach to construct the discrete dynamic energy profiles of the abstract processors in our heterogeneous servers. Since the servers are composed of CPUs and accelerators that are loosely coupled, we find that the discrete dynamic energy profiles are accurate enough to be used as input to our workload partitioning algorithm, HEOPTA.

We show that HEOPTA provides considerable improvements in average and maximum dynamic energy consumptions in comparison with the state-of-the-art load-balancing algorithm, which is based on smooth functional performance models of abstract processors and load-imbalancing algorithm (HPOPTA), which takes as input nonsmooth and nonlinear performance profiles and optimizes the data-parallel application for performance. Therefore, we conclude that performance optimization does not always lead to minimization of dynamic energy consumption.

We also experimentally demonstrate that minimization of dynamic energy does not always result in minimization of total energy consumption during the execution of a data-parallel application.

The limitations of our method are that it does not take into account the energy consumptions of fans as well as the energy consumptions of the network due to internode communications. We will extend our method to address the limitations. We will also explore the applicability of our additive modelling approach for platforms where the constraint of loosely coupling for the compute devices is relaxed to some extent.

9 | CONCLUSION

Modern heterogeneous HPC platforms feature tight integration of multicore CPUs with accelerators which resulted in inherent complexities such as severe resource contention for shared on-chip resources and NUMA. These complexities result in a complex functional relationship between energy consumption and workload size of applications executing on these platforms thereby providing an opportunity for application-level energy optimization through workload distribution as a decision variable.

We proposed HEOPTA, a novel data partitioning algorithm that determines optimal workload distributions minimizing the dynamic energy consumption of data-parallel applications running on heterogeneous HPC platforms. The optimality of solutions found by HEOPTA is analyzed using two popular data-parallel applications, matrix-matrix multiplication and 2D-FFT. It was showed that the load-imbalanced solutions provided by the algorithm achieve significant dynamic energy reductions compared with the load-balanced solutions. We also demonstrated that optimizing for performance increases dynamic energy consumption by an average of 145% and 32% for both the applications. As future work, we will further study the impact of dynamic energy optimization on performance and vice versa and will propose a biobjective optimization algorithm for performance and dynamic energy to make a trade-off between these two objectives.

We also examined the impact of dynamic energy optimization on total energy consumption. It is experimentally demonstrated that solutions minimizing dynamic energy consumption do not always optimize total energy consumption. In our future work, it is intended to propose an algorithm finding optimal workload distribution minimizing total energy consumption with polynomial time complexity.

The software implementation for HEOPTA is available at Reference 45.

ACKNOWLEDGMENTS

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

ORCID

Hamidreza Khaleghzadeh https://orcid.org/0000-0003-4070-7468
Ravi Reddy Manumachu https://orcid.org/0000-0001-9181-3290
Alexey Lastovetsky https://orcid.org/0000-0001-9460-3897

REFERENCES

1. Hsu J. Three paths to exascale supercomputing. *IEEE Spectrum*. 2016;53(1):14-15.
2. Top500. Top500; 2018. https://www.top500.org/lists/2018/11/.
3. DOE. Preliminary conceptual design for an exascale computing initiative; 2014. https://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20141121/Exascale_Preliminary_Plan_V11_sb03c.pdf.
4. Lang J, Rünger G. An execution time and energy model for an energy-aware execution of a conjugate gradient method with CPU/GPU collaboration. *J Parall Distrib Comput*. 2014;74(9):2884-2897.
5. Chakrabarti A, Parthasarathy S, Stewart C. A pareto framework for data analytics on heterogeneous systems: implications for green energy usage and performance. Paper presented at: Proceedings of the 46th International Conference on Parallel Processing (ICPP); 2017:533-542; IEEE.
6. Lastovetsky A, Reddy R. New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters. *IEEE Trans Parall Distrib Syst*. 2017;28(4):1119-1133.
7. Manumachu RR, Lastovetsky A. Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy. *IEEE Trans Comput*. 2018;67(2):160-177.

8. Khaleghzadeh H, Zhong Z, Reddy R, Lastovetsky A. Out-of-core implementation for accelerator kernels on heterogeneous clouds. *J Supercomput*. 2018;74(2):551-568.
9. Zhong Z, Rychkov V, Lastovetsky A. Data partitioning on multicore and multi-GPU platforms using functional performance models. *IEEE Trans Comput*. 2015;64(9):2506-2518.
10. HCL HCLWattsUp: API for power and energy measurements using WattsUp Pro Meter; 2016. https://csgitlab.ucd.ie/ucd-hcl/hclwattsup.
11. Konstantakos V, Chatzigeorgiou A, Nikolaidis S, Laopoulos T. Energy consumption estimation in embedded systems. *IEEE Trans Instrument Measur*. 2008;57(4):797-804.
12. Rotem E, Naveh A, Ananthakrishnan A, Weissmann E, Rajwan D. Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*. 2012;32(2):20-27.
13. Intel Corporation, *Intel® Xeon Phi™ Coprocessor System Software Developers Guide*. Intel Corporation; 2014. https://software.intel.com/sites/default/files/managed/09/07/xeon-phi-coprocessor-system-software-developers-guide.pdf.
14. Nvidia Nvidia management library: NVML reference manual; 2018.
15. Gough C, Steiner I, Saunders W. *Energy Efficient Servers Blueprints for Data Center Optimization*. Springer Nature; 2015.
16. Economou D, Rivoire S, Kozyrakis C, Ranganathan P. Full-system power analysis and modeling for server environments. Paper presented at: Proceedings of Workshop on Modeling, Benchmarking, and Simulation; 2006:70-77.
17. McCullough JC, Agarwal Y, Chandrashekar J, Kuppuswamy S, Snoeren AC, Gupta RK. Evaluating the effectiveness of model-based power characterization. Paper presented at: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference. USENIX Association; 2011:12.
18. O'Brien K, Pietri I, Reddy R, Lastovetsky A, Sakellariou R. A survey of power and energy predictive models in HPC systems and applications. *ACM Comput Surv*. 2017;50(3):37.
19. Shahid A, Fahad M, Reddy R, Lastovetsky A. Additivity: a selection criterion for performance events for reliable energy predictive modeling. *Supercomput Front Innov Int J*. 2017;4(4):50-65.
20. Liu Y, Zhu H, Lu K, Wang X. Self-adaptive management of the sleep depths of idle nodes in large scale systems to balance between energy consumption and response times. Paper presented at: Proceedings of the 2012 IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom); 2012:633-639; IEEE.
21. Benoit A, Lefèvre L, Orgerie AC, Rais I. Reducing the energy consumption of large-scale computing systems through combined shutdown policies with multiple constraints. *Int J High Perf Comput Appl*. 2018;32(1):176-188.
22. Rossi FD, Xavier MG, De Rose CA, Calheiros RN, Buyya R. E-eco: Performance-aware energy-efficient cloud data center orchestration. *J Netw Comput Appl*. 2017;78:83-96.
23. Chen K, Lenhardt J, Schiffmann W. Improving energy efficiency of web servers by using a load distribution algorithm and shutting down idle nodes. Paper presented at: Proceedings of the 2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid); 2015:745-748; IEEE.
24. Rajamani K, Lefurgy C. On evaluating request-distribution schemes for saving energy in server clusters. Paper presented at: Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2003; 2003:111-122; IEEE.
25. Basmadjian R, Ali N, Niedermeier F, Meer DH, Giuliani G. A methodology to predict the power consumption of servers in data centres. Paper presented at: Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking; 2011; ACM.
26. Lively C, Wu X, Taylor V, et al. Power-aware predictive models of hybrid (MPI/OpenMP) scientific applications on multicore systems. *Comput Sci-Res Dev*. 2012;27(4):245-253.
27. Rofouei M, Stathopoulos T, Ryffel S, Kaiser W, Sarrafzadeh M. Energy-aware high performance computing with graphic processing units. Paper presented at: Proceedings of the Workshop on Power Aware Computing and System; 2008.
28. Li S, Ahn JH, Strong RD, Brockman JB, Tullsen DM, Jouppi NP. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. Paper presented at: Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture; 2009:469-480; ACM.
29. Lim J, Lakshminarayana NB, Kim H, Song W, Yalamanchili S, Sung W. Power modeling for GPU architectures using McPAT. *ACM Trans Des Automat Electron Syst (TODAES)*. 2014;19(3):26.
30. Hong S, Kim H. An integrated GPU power and performance model. Paper presented at: Proceedings of the 38 of ACM SIGARCH Computer Architecture News; 2010:280-289; ACM.
31. Nagasaka H, Maruyama N, Nukada A, Endo T, Matsuoka S. Statistical power modeling of GPU kernels using performance counters. Paper presented at: Proceedings of the 2010 International IEEE Green Computing Conference; 2010:115-122.
32. Chen J, Li B, Zhang Y, Peng L, Peir JK. Statistical GPU power analysis using tree-based methods. Paper presented at: Proceedings of the 2011 International Green Computing Conference and Workshops (IGCC); 2011:1-6; IEEE.
33. Song S, Su C, Rountree B, Cameron KW. A simplified and accurate model of power-performance efficiency on emergent GPU architectures. Paper presented at: Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS). IEEE Computer Society; 2013:673-686.
34. Kestor G, Gioiosa R, Kerbyson DJ, Hoisie A. Enabling accurate power profiling of HPC applications on exascale systems. Paper presented at: Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers; 2013:4; ACM.
35. Choi JW, Bedard D, Fowler R, Vuduc R. A roofline model of energy. Paper presented at: Proceedings of the 2013 IEEE 27th International Symposium on Parallel & Distributed Processing (IPDPS); 2013:661-672; IEEE.
36. Williams S, Waterman A, Patterson D. Roofline: an insightful visual performance model for multicore architectures. *Commun ACM*. 2009;52(4):65-76.
37. Shao YS, Brooks D. Energy characterization and instruction-level energy model of Intel's Xeon Phi processor. Paper presented at: Proceedings of the 2013 International Symposium on Low Power Electronics and Design; 2013; IEEE Press.
38. Jarus M, Oleksiak A, Piontek T, Węglarz J. Runtime power usage estimation of HPC servers for various classes of real-life applications. *Future Generat Comput Syst*. 2014;36:299-310.
39. Al-Khatib Z, Abdi S. Operand-value-based modeling of dynamic energy consumption of soft processors in FPGA. Paper presented at: Proceedings of the International Symposium on Applied Reconfigurable Computing; 2015:65-76; Springer.
40. Shahid A, Fahad M, Reddy R, Lastovetsky A. Additivity: a selection criterion for performance events for reliable energy predictive modeling. *Supercomput Front Innovat*. 2017;4(4):50-65.

41. McCullough JC, Agarwal Y, Chandrashekar J, Kuppuswamy S, Snoeren AC, Gupta RK. Evaluating the effectiveness of model-based power characterization. Paper presented at: Proceedings of the 20 of USENIX Annual Technical Conference; 2011.
42. Manumachu RR, Lastovetsky A. Parallel data partitioning algorithms for optimization of data-parallel applications on modern extreme-scale multicore platforms for performance and energy. *IEEE Access*. 2018;6:69075-69106.
43. Fahad M, Shahid A, Manumachu RR, Lastovetsky A. A comparative study of methods for measurement of energy of computing. *Energies*. 2019;12(11):2204.
44. Khaleghzadeh H, Manumachu RR, Lastovetsky A. A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous HPC platforms. *IEEE Trans Parall Distrib Syst*. 2018;29(10):2176-2190.
45. Khaleghzadeh H, Reddy R, Lastovetsky A. HEOPTA: heterogeneous model-based data partitioning algorithm for optimization of data-parallel applications for dynamic energy; 2019. https://csgitlab.ucd.ie/HKhaleghzadeh/heopt.

How to cite this article: Khaleghzadeh H, Fahad M, Reddy Manumachu R, Lastovetsky A. A novel data partitioning algorithm for dynamic energy optimization on heterogeneous high-performance computing platforms. *Concurrency Computat Pract Exper*. 2020;e5928. https://doi.org/10.1002/cpe.5928