Journal of Parallel and Distributed Computing 175 (2023) 121–133

Contents lists available at ScienceDirect

Journal of Parallel and Distributed Computing

journal homepage: www.elsevier.com/locate/jpdc

Reducing energy consumption using heterogeneous voltage frequency scaling of data-parallel applications for multicore systems

Pawel Bratek [*], Lukasz Szustak, Roman Wyrzykowski, Tomasz Olas

Department of Computer Science, Czestochowa University of Technology, Dabrowskiego 69, Czestochowa, 42-201, Poland

ARTICLE INFO

Article history:
Received 13 May 2022
Received in revised form 3 January 2023
Accepted 10 January 2023
Available online 18 January 2023

Keywords:
Data-parallel applications
Energy saving
Heterogeneous voltage frequency scaling
Multicore
ccNUMA

ABSTRACT

This paper investigates the exploitation of heterogeneous DVFS (dynamic voltage frequency scaling) control for improving the energy efficiency of data-parallel applications on ccNUMA shared-memory systems. We propose to adjust the clock frequency individually for the appropriately selected groups of cores, taking into account the diversified costs of parallel computation. This paper aims to evaluate the proposed approach using two different data-parallel applications: solving the 3D diffusion problem, and MPDATA fluid dynamics application. As a result, we observe the energy-savings gains of up to 20 percentage points over the traditional homogeneous frequency scaling approach on the server with two 18-core Intel Xeon Gold 6240. Additionally, we confirm the effectiveness of our strategy using two 64-core AMD EPYC 7773X. This paper also introduces two pruning algorithms that help select the optimal heterogeneous DVFS setups taking into account the energy or performance profile of studied applications. Finally, the cost and efficiency of developed algorithms are verified and compared experimentally against the brute-force search.

1. Introduction

Energy efficiency has become an essential factor in the high-performance computing (HPC) race towards Exascale [22]. Publishing the Green500 rank that includes the power efficiency metric expressed in the number of FLOPS per Watt together with the TOP500 list confirms the importance of this aspect in the architecture of modern HPC systems. The scientific community is attempting to address this challenge in different ways on both hardware and software levels [13].

Dynamic voltage and frequency scaling (DVFS) is a commonly accepted technique provided at the hardware level [25]. It is known [16] as an efficient way to save energy for memory-bound codes. Using DVFS permits lowering the voltage/frequency at the cost of potentially longer runtime [6].

Different granularity levels of DVFS are possible for shared-memory multicore processors [20]: (i) when the frequency is changed *per chip*, (ii) *cluster-level* DVFS assuming multiple on-chip voltage regulators handling a set of DVFS domains, and (iii) *per-core* DVFS with an individual regulator for every core. The per-core frequency management is provided in more recent Intel CPUs (with the Haswell architecture and later) by fixing the minimum and maximum frequencies for a given core [37]. While the first level corresponds to homogeneous voltage frequency scaling over the processor, the remaining two levels refer to heterogeneous scaling. Compared with the per-chip approach, controlling the voltage at the core- or cluster-level can allow [3] to yield the most performance out of the system.

The popular parallel decomposition strategy is data parallelism [29], by which an application's data domain is divided into as many data partitions as threads performing the computation. In the data-parallel model, tasks are assigned to threads, and every task performs similar types of operations on different data. Typically, data-parallel programs contain a loop body executed on different parts of the input data.

Data-parallel codes are gaining in importance, are becoming more diverse, and require increased hardware performance while keeping energy consumption to a minimum. In our previous papers, we studied the usability of the DVFS technique to counterbalance energy savings with acceptable performance losses for such data-parallel algorithms and applications as 3D MPDATA (Multidimensional Positive Definite Advection Transport Algorithm) for solving computational fluid dynamics (CFD) problems [26,35], and conjugate gradient solver [27].

A further example of utilizing DVFS for data-parallel computation is investigated in work [8], which studies the relationship between task scheduling and energy constraints for stencil comput-

[*] Corresponding author.
E-mail addresses: pawel.bratek@pcz.pl (P. Bratek), lszustak@icis.pcz.pl (L. Szustak).

https://doi.org/10.1016/j.jpdc.2023.01.005
0743-7315/© 2023 Elsevier Inc. All rights reserved.

ing, a class of memory-bound applications that are widely found in scientific computing. This work and our previous papers utilize homogeneous DVFS for multicore processors, alternatively supported by concurrency throttling. This approach seems to be a realistic option for homogeneous multicore, and regular data-parallel applications that feature a consistent behavior when all cores or threads perform a similar type of work [3].

Commonly, the usage of heterogeneous DVFS in homogeneous multicore processors is justified [3] for irregular or unstructured applications where all cores can perform different kinds of work at a given time. On the contrary, this paper investigates the advantages of the heterogeneous DVFS control to improve the energy-performance behavior of regular data-parallel applications on homogeneous multicore CPU platforms with shared memory, including ccNUMA (cache-coherent non-uniform memory access) systems. The justification for these advantages lies in the evolving relationship between the sizes of applications (corresponding to sizes of data sets) and the growing variety of the number of cores. The consequence is the thread divergence within threads of an application causing deterioration of energy efficiency.

The heterogeneous voltage frequency scaling for data-parallel applications running on multicore systems was for the first time proposed in our previous work [5]. In that work, we introduced an energy-based pruning algorithm for selecting heterogeneous DVFS configurations and confirmed its effectiveness by testing it using the 3D diffusion problem as a use case.

This paper expands the research by considering another use case of regular data-parallel application – the 3D MPDATA code from the CFD area. Both applications represent a broad group of iterative stencil-based codes. In this work, we also propose a new algorithm that avoids measuring energy consumption – it is enough to measure the execution time only. As a result, both proposed algorithms for selecting heterogeneous DVFS configurations are validated using two different data-parallel codes. In addition, apart from considering 18-core Intel Xeon Gold 6240 CPUs (Cascade Lake architecture), we confirm the efficiency of our approach for the latest 64-core AMD EPYC 7773X CPUs (Milan-X architecture).

The material of this paper is organized as follows. Section 2 provides an overview of related work, while Section 3 presents the basics of our approach, including two use cases of applications studied in the article and a description of a target computing platform. A detailed motivation for this research and the problem statement is the topic of Section 4. A brute-force search and two pruning algorithms for selecting heterogeneous DVFS settings are proposed in Section 5 and Section 6, respectively. The experimental evaluation of the proposed algorithms is the subject of Section 7. Finally, Section 8 shows the results for AMD EPYC CPUs, while Section 9 presents conclusions.

2. Related works

The energy efficiency of HPC parallel workloads has recently become a focus of attention [18,21]. Besides the trend of energy reduction achieved through re-engineering hardware, there is another one with significant potential for energy savings that is implemented by transforming and completely rethinking the algorithms and software developed for HPC platforms [12].

The solutions from the second category focus largely on optimizing applications with carefully analyzing the trade-off between time and power/energy consumption [11,40]. These solutions use application-level models to predict performance and energy consumption. Similarly to our study, a lot of research efforts aim to advance the state-of-the-art in power-aware computing [17], focusing on different aspects of the field.

A common technique applied is DVFS which allows the hardware to lower the operational voltage and frequency and gives the possibility to optimize the energy efficiency when CPU (or GPU) cycles are typically being wasted since they are stalled on memory resources [16]. Many of the past research uses chip-level (or homogeneous) DVFS to do energy and power optimizations [10,15]. In particular, works [15,7] studied the use of hardware power capping mechanisms to reduce energy consumption in the parallel execution of applications with various workloads and memory usage.

For data-parallel applications, the usage of DFVS is studied by Reddy and Lastovetsky in work [21]. They propose a method to solve the bi-objective optimization problem of an application for performance and energy on homogeneous clusters of multicore CPUs. This method gives a set of Pareto-optimal solutions and is combined with DVFS to provide a better set of solutions. However, the proposed approach target only homogeneous DVFS policies. Such policies are also exploited in many works devoted to special cases of data-parallel applications, including stencil computations [26,8], and linear algebra kernels [42]. In particular, in work [42], the authors devise a systematic machine learning algorithm to predict the performance and energy costs of the matrix-vector product, considering the effect of DVFS.

The capability of implementing heterogeneous DFVS by processors (per core or at cluster level) brings the premise of improved energy efficiency thanks to fine-grained optimization [3]. Manufacturing-related process variation [2] and heterogeneity of cores in a processor architecture [23,41,24] are among the major motivations for using core-level voltage regulators. For example, work [23] considers the heterogeneity of cores in ARM architectures for enhancing the default Linux scheduler to take task allocation decisions for balanced performance and energy efficiency. Paper [24] explores the benefits of voltage-frequency settings for individual cores in heterogeneous mobile platforms running data-parallel applications.

Many studies aim to develop DVFS-aware schedulers based on per-core policies. For example, paper [1] studies a case when an application running on some cores coexists with its co-runners – applications running on other cores. Machine learning methods such as random forest regression are used to model the energy-performance trade-off. The application-agnostic model is investigated in work [14], where the core statistics are collected and employed to predict the next interval power consumption. As a result, it is possible to determine a suitable voltage-frequency setting for each core using an adaptive regression model. The methods used in these studies are primarily deployed at the operating system (OS) level or as a runtime system extending the OS. Therefore, they require changes to the OS or runtime system.

An important motivation for core-level DVFS is applications that execute code with different characteristics simultaneously on different cores within a processor [3]. In contrast to applications when all cores or processes within a chip execute similar type of work, some HPC applications are irregular or unstructured with dynamic behavior. In these applications, cores may do various types of work at a given time. Work [3] shows that implementing an adaptive runtime module based on collected statistics can lead to considerably higher energy efficiency of such applications.

At the same time, to our best knowledge, there is no other research where the benefits of the heterogeneous DVFS strategy are employed to improve the energy-performance behavior of regular data-parallel applications for homogeneous multicore platforms. This strategy allows us to avoid deterioration of energy efficiency due to thread divergence and load imbalance, which result from the discrepancy between the sizes of applications and the number of cores.

```
#pragma omp for
for(int i=0; i<X; i++) // i - dimension
 for(int j=0; j<Y; j++) // j - dimension
  #pragma omp simd
  for(int k=0; k<Z; k++) // k - dimension
   v(i,j,k) = u(i,j,k)
   v(i,j,k) += xS * (u(i-1,j,k) - 2*u(i,j,k) + u(i+1,j,k))
   v(i,j,k) += yS * (u(i,j-1,k) - 2*u(i,j,k) + u(i,j+1,k))
   v(i,j,k) += zS * (u(i,j,k-1) - 2*u(i,j,k) + u(i,j,k+1))
```

Listing 1: Parallelization of data-parallel application using OpenMP [5].

3. Use cases of data parallel applications and target platform

3.1. Use case of data-parallel application: 3D diffusion problem

The diffusion process is described [9] by the following partial differential equation that is the consequence of Fick's law and the law of conservation of mass:

$$\partial U/\partial t = \partial^2 U/\partial x^2 + \partial^2 U/\partial y^2 + \partial^2 U/\partial z^2 \tag{1}$$

The function $U = U(x, y, z, t)$ from Eqn. (1) describes the concentration of a given substance in point (x, y, z) at moment t. The presented equation can also be used to analyze other physical phenomena. Under certain assumptions, it can describe the process of heat transfer. In this case, the unknown function U represents temperature. Applying the finite difference method [43] to this partial differential equation produces the following numerical scheme:

$$\begin{aligned} U^{h+1}_{i,j,k} = \Delta t\big[&(U^h_{i-1,j,k} - 2U^h_{i,j,k} + U^h_{i+1,j,k})/\Delta x^2 + \\ &+ (U^h_{i,j-1,k} - 2U^h_{i,j,k} + U^h_{i,j+1,k})/\Delta y^2 + \\ &+ (U^h_{i,j,k-1} - 2U^h_{i,j,k} + U^h_{i,j,k+1})/\Delta z^2\big] + U^h_{i,j,k} \end{aligned} \tag{2}$$

In this use case, we consider the 3D diffusion problems defined over the structured rectilinear domain of sizes $X \times Y \times Z$ in $i-$, $j-$, and k-dimensions, respectively. The presented explicit method requires meeting certain conditions regarding discretization. To ensure the numerical stability of the algorithm, time steps need to be sufficiently short compared to space steps. Therefore, performed simulations are usually long and consist of many thousands of time steps.

Listing 1 shows the basic code of this data-parallel application. The computing kernel is implemented in parallel using the OpenMP standard. The parallelization scheme implements data parallelism across i-dimension, based on distributing data over threads by `#pragma omp for` directive, and then exploits vectorization along k-dimension using `#pragma vector` directive.

The adoption of vector-friendly data structures allows for the benefits of vectorization. In this code, the main memory bandwidth is the primary constraint of parallel efficiency. The performance bottleneck is especially observable for problems with large domain sizes that significantly exceed the available cache capacity.

3.2. Use case of data-parallel application: MPDATA

The MPDATA application implements a general approach to modeling a wide range of complex geophysical flows on micro-to-planetary scales [28,33], including numerical weather prediction, simulation of urban flows, turbulences, and ocean currents. MPDATA belongs to the class of methods for the numerical simulation of fluid flows that are based on the sign-preserving properties of upstream differencing. It is mainly used to solve the advection problems on moving grids for a sequence of time steps that allows us to include MPDATA in the group of forward-in-time algorithms. In this paper, we consider solving 3D problems. The MPDATA numerical scheme is described in detail in [30].

Each MPDATA step performs a collection of 17 kernels that depend on each other. The outcomes of prior kernels typically are inputs for the subsequent ones. All kernels are executed sequentially, one by one, with each kernel processed in parallel using OpenMP. Like the 3D diffusion problem, MPDATA allows us to use the 1D decomposition where the data parallelism is implemented across the outer loop of each kernel (see Listing 2), distributing data across cores by `#pragma omp for` directive. Then the vectorization is applied efficiently to the most inner loop using `#pragma omp simd` directive. As shown in our previous papers [35,31,34], such a 1D decomposition enables more efficient memory utilization for ccNUMA shared memory systems than 2D and 3D workload distributions across cores. At the same time, 2D and 3D approaches can be successfully applied across nodes of distributed memory systems.

MPDATA is typically used for long simulations that run thousands of time steps. A single step operates on five input matrices (arrays), and returns a single output array that is used in the next step. As shown in our previous works [32,36], MPDATA is a memory-bound application.

3.3. Target computing platform

We run both applications on the Intel-based ccNUMA server S2600WFT with two 18-core Intel Xeon Gold 6240 CPUs (Cascade Lake architecture) containing 72 logical cores in total. Each processor comes with 24.75MB of L3 cache. The thermal and power limitations of the test platform allow setting the minimum clock frequency to 1.0 GHz and then sampling it at every 0.1 GHz to reach the maximum Turbo Boost speed of about 2.5 GHz.

The Yokogawa WT310 power meter [35] monitors the entire platform and provides all energy measurements presented in this paper. This power meter passes the power to the server under the load and measures the total energy consumption. It guarantees that the results of conducted experiments are maximally precise and trustworthy. Moreover, we ensure that simulations for both use cases last long enough to provide the credibility and repeatability of energy measurements. We customize the number of time steps for every tested domain size to keep the execution time at least 700 seconds. It allows us to keep the relative standard deviation (RSD) of measurements at a fixed level. This factor does not exceed 1.5% for execution time and 2.5% for energy consumption measurements. Besides, we ensure the server is located in an air-conditioned server room providing stable temperature, as well as it is fully dedicated to our experiments.

4. Motivation for the research and problem statement

The starting point of our research is the applied parallelization strategy (see Listings 1 and 2) that is identical for both use cases. Using OpenMP, we split up iterations within the first loop (i-dimension) and assign them to threads within the parallel region. For this reason, the uniform workload distribution takes place only

```
/*...*/
//Kernel 4
#pragma omp for
for(int i=0; i<X; i++) // i - dimension
 for(int j=0; j<Y; j++) // j - dimension
  #pragma omp simd
  for(int k=0; k<Z; k++) // k - dimension
   x[i,j,k]=XIn[i,j,k]-(((F1[i+1,j,k]-F1[i,j,k])+(F2[i,j+1,k]
        -F2[i,j,k])+(F3[i,j,k+1]-F3[i,j,k]))/H[i,j,k]);
/*...*/
```

Listing 2: Part of 3D MPDATA code that corresponding to the 4-th computing kernel.

when the number X of iterations is equally divided by the number LC of logical cores. In practice, this situation is infrequent as real-life simulations rarely meet this condition. Consequently, the proposed parallelization scenario leads to workload imbalance and thread divergence in most cases.

We define two parameters that describe and formalize the presented observation. The first one is the load imbalancing percentage that corresponds to the ratio of the total number of threads that perform more loop iterations to the total number of logical cores in the computing platform:

$$LIP = (X \, mod \, LC)/LC \cdot 100\% \tag{3}$$

The second parameter specifies the disproportion in assigned iterations between threads. Assuming $X > LC$, the more loaded threads perform R times more iterations than the rest of the threads, where the parameter R can be expressed as:

$$R = \lceil X/LC \rceil / (\lceil X/LC \rceil - 1) \tag{4}$$

These parameters have a significant influence on workload imbalance and thread divergence. Fig. 1 presents the execution time and energy consumption for different domain sizes of the 3D diffusion problem. In all examples, $R = 2$ while LIP varies from 8% to 50%. This study assumes a limit of 2% performance losses to achieve energy savings.

In the left parts of Fig. 1, we explore the relationship between execution time and energy consumption, assuming the usage of homogeneous DFVS across cores for different frequencies. As shown in Fig.1a and Fig.1b, the optimal performance-energy trade-off (marked with red points) corresponds to the highest CPU frequency. It turns out that even a slight reduction in the frequency provides energy savings but, at the same time, exceeds the assumed limit for performance losses. The rationale behind this behavior is that all necessary data reside in the cache hierarchy, and computing units do not wait for loading data from the main memory.

We can observe totally different time and energy characteristics in Fig.1c and Fig.1d corresponding to large domain sizes. For all examined frequencies, the execution time is almost the same. Here, the frequency is not a bottleneck, and reducing it does not affect performance. The reason is that the most critical limitation is the main memory speed.

The most significant energy savings achieved with DVFS are about 30%, with negligible performance losses not exceeding 1-2%. However, our experiments expose the potential of revising the DVFS approach even in this case that commonly assumes homogeneous frequency scaling across cores. To illustrate this, let us move on to the analysis of the right parts of Figs. 1a–1d, that present the execution time distribution across threads for different domain sizes and a variety of workloads characterized by diverse values of parameter LIP. These plots illustrate the total execution time that every OpenMP thread spends on (i) computation (area marked in blue color) and (ii) synchronization. Additionally, we mark two stages responsible for the synchronization costs: (i) the arrival stage that puts threads arriving at the barrier into a waiting state (area marked in red color), and (ii) the departure stage that releases from the barrier all waiting threads (marked in gray color).

While the departure stage for small domain sizes is relatively high, it is practically unnoticeable for large domains. Its cost depends on the number of synchronization points, which in our study increases linearly with the number of time steps. As expected, the time of this stage is nearly uniform for every OpenMP thread. For a single synchronization point, this time mainly depends on the OpenMP implementation chosen and hardware limitations.

At the same time, the actual computation time (see blue regions in the right parts of Figs. 1a–1d) differs visibly across threads. The same observation is valid for the arrival stage of synchronization. This heterogeneity comes from non-uniform workload distribution over available cores/threads, characterized by LIP and R parameters.

Furthermore, we note that the inter-socket data traffic negatively influences thread divergence. From the right parts of Figs. 1a and 1b, it follows that threads pinned to cores located on the boundaries between sockets feature the highest time of actual computation even though they execute twice fewer iterations than more loaded threads. This undesirable effect becomes crucial only when threads operate on data that fit entirely in the cache hierarchy, and the inter-socket data exchange overhead is a fundamental limitation. On the contrary, the overhead is negligible or imperceptible for larger domains when the main memory speed restricts the computation time (see Figs. 1c and 1d, respectively).

We perform a similar set of experiments for the MPDATA application to validate our observations. Figs. 2a–2c present the results obtained for three different domain sizes corresponding to various load imbalance ratios defined by the LIP parameter. The examined problems concern distinct R coefficients. In the first two cases, $R = 2$, while in the last one, $R = 3/2$.

Analogously to the first application, the left parts of Figs. 2a–2c expose the execution time and energy consumption of MPDATA depending on the frequency set for all cores. In contrast to the 3D diffusion problem, it is possible to lower frequency without a rapid increase in the execution time in all three analyzed cases. Assuming a limit of 2% for performance losses, we can obtain over 12% energy savings. The reason is that MPDATA requires much more memory than the 3D diffusion problem. Consequently, now the bottleneck is not the frequency but the time of loading data from the main memory.

The analysis of the right parts of Fig. 2 shows again that the actual computation time differs across threads. We can distinguish a group of threads with a longer execution time resulting from the load imbalance specified by the LIP parameter. Compared to the 3D diffusion application, the main difference is that the inter-socket traffic has less impact on the thread divergence. It is a consequence of the fact that the size of the working set for considered domain sizes significantly exceeds the available cache capacity. Finding domains that fit in the cache hierarchy is feasible

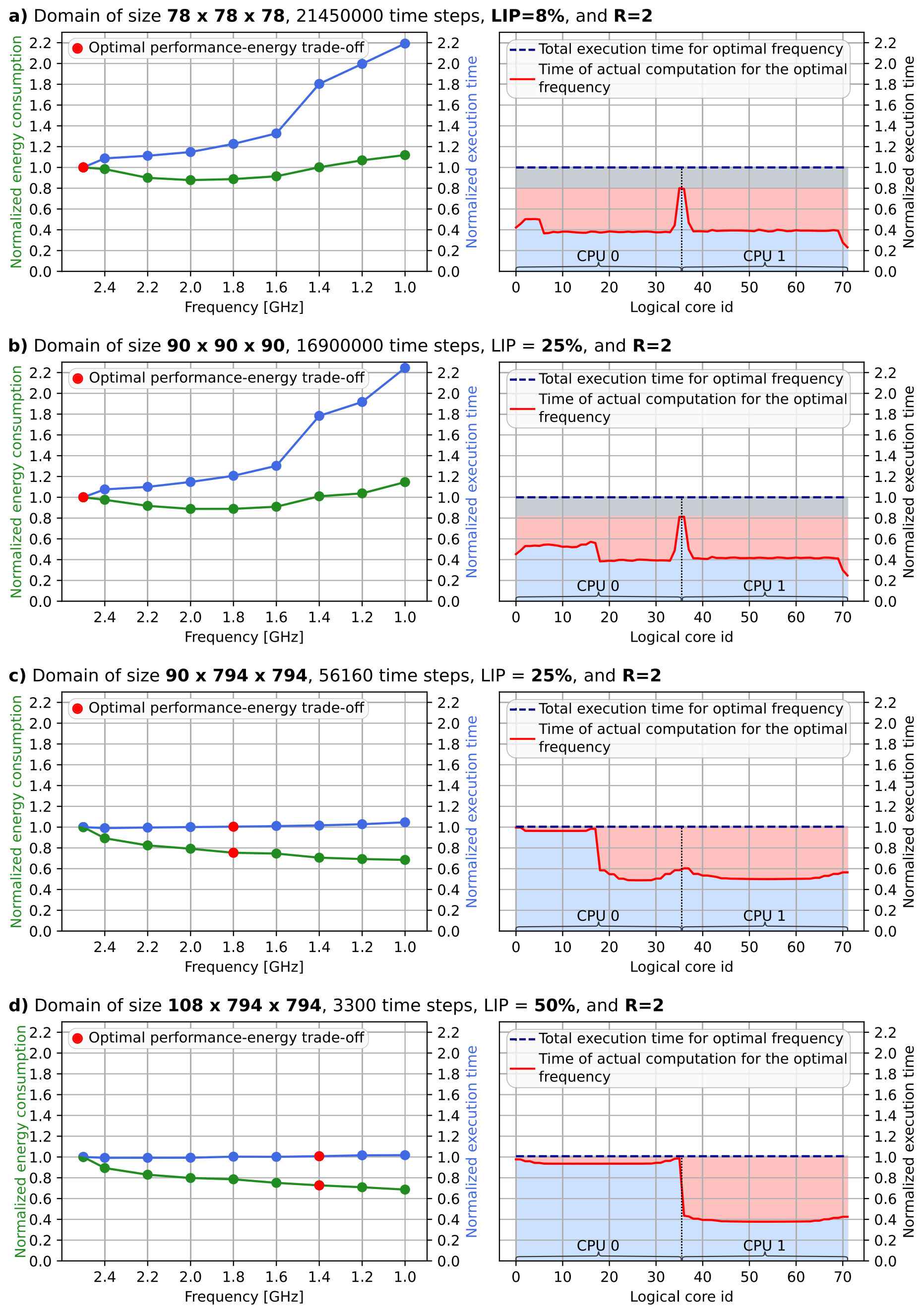

Fig. 1. Execution time and energy consumption for solving the 3D diffusion problem with various domain sizes, using homogeneous DFVS [5].

but not of practical importance since these domains correspond to unrealistically small sizes.

Summarizing our observations for the two use cases, the homogeneous variant of the DVFS method allows selecting the performance-energy tradeoff considering the total execution time that reflects the maximum computation time obtained across a pool of threads. However, threads that process fewer loop iterations waste energy waiting at points of synchronization in the arrival stage. This disadvantage is exacerbated by the steadily increasing number of cores offered by modern multicore architectures.

5. Heterogeneous DVFS: brute-force search

The considerations from Section 4 allow us to see the potential of adjusting the frequency and supply voltage to individual cores in workload imbalance problems. We perceive two heterogeneous DVFS scenarios that manage the frequency of correctly specified sets of cores. Our key idea is to customize the clock frequency individually for the appropriately selected groups of cores corresponding to the diversified time of actual computation.

In the first scenario, we focus on smaller problem sizes (Figs. 1a and 1b), where we can identify cores with the time of actual computations limited by inter-socket data traffic overheads. Here, we can indicate three groups of cores, including (i) cores located on the boundaries between sockets; cores with higher (ii) and fewer (iii) loop iterations. The numbers of cores in the second and third groups result from the LIP parameter and depend on the domain size and number of cores. The strategy assumes assigning the highest frequency to the first group and then lowering it successively for the second and last group. We disclose that scaling down the clock speed for cores located on the socket boundaries

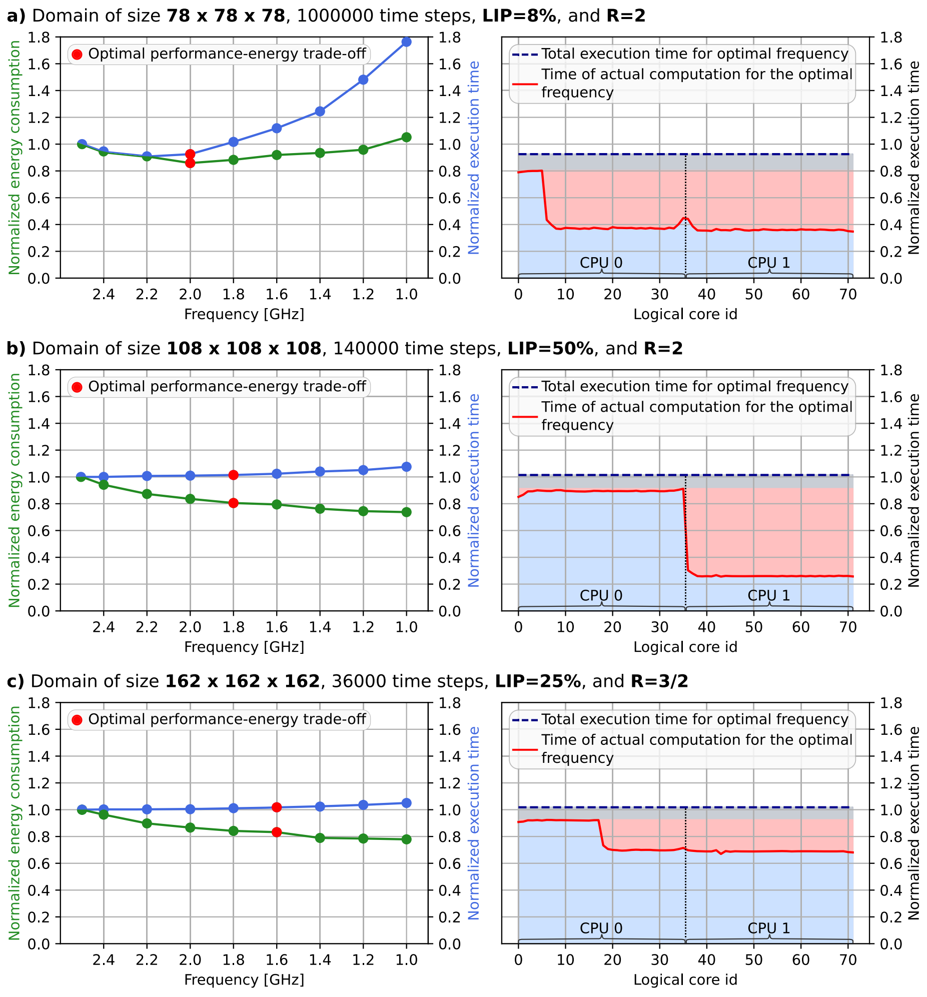

Fig. 2. Execution time and energy consumption for solving the MPDATA application with various domain sizes, using homogeneous DFVS.

is not necessary. Applying the maximum available frequency/voltage for those cores should improve the efficiency of the load and store instructions, opening the way for optimal performance and energy consumption results.

The second scenario corresponds to larger problem sizes where the main memory constraints result in performance degradation (Fig. 1c, Fig. 1d, and Fig. 2). In this scenario, we indicate only two groups of cores with higher (ii) and fewer (iii) loop iterations. Consequently, a higher priority and thus a higher voltage/frequency refers to the first group.

We perform a set of brute-force experiments for both use cases to fully explore the benefits of the heterogeneous DVFS approach. This approach involves testing all possible clock speed configurations, considering various domain sizes. More precisely, while examining a given frequency for the group with a higher priority, we test different frequency combinations for the group with a lower priority by scaling down the frequency from a fixed level to the minimum, sampling it at every 0.2 GHz. For example, when setting clock speed to 2.0 GHz for cores in the first group, we combine it with the set $\{2.0, 1.8, ..., 1.0\}$ GHz of frequencies for the second group.

The summary of performed tests is presented in Fig. 3 and Fig. 4. We select a set of 33 domain sizes that allow a comprehensive analysis of the homo- and heterogeneous DVFS strategies based on examining different ratio $R \in \{2, 3/2, 4/3\}$ and the LIP parameter varying for each ratio from 8 to 92%. The top graphs in the presented figures correspond to the homogeneous approach, showing the frequency of all cores for which the energy consumption is minimal, with performance losses not exceeding 2% compared to the results obtained for the frequency of 2.5 GHz. The same assumptions are valid for middle graphs in the figures, where we mark the optimal heterogeneous frequency setups. Finally, the bottom graphs compare both approaches regarding the gained energy profit.

For the 3D diffusion application (Fig. 3), we select three groups of clock speeds for domains of sizes not exceeding 138x138x138 (these domains fit in the cache), while two groups are enough for larger domains. The heterogeneous approach permits a maximum energy reduction of about 25% for the domain of size 150x150x150 when the profit for homogeneous scaling is about 9% only. In general, the heterogeneous approach allows us to achieve better energy profits for all performed tests. The most significant energy improvement compared to the homogeneous solution is obtained for the size of 96x96x96. In this case, we observe the advantage of about 20 percentage points over the traditional frequency scaling, which does not bring any energy improvement.

In the case of MPDATA applications (Fig. 4), all domain sizes do not fit in the cache memory. Consequently, the best solution is to designate only two groups of clock speed. The most notable contrast between both approaches occurs while $R = 2$. For the domain of size 78x78x78, the heterogeneous solution permits a 30% reduction of energy consumption compared to about 14% obtained by the traditional homogeneous method. To sum up, in 28 out of 33 analyzed cases, it is possible to determine a heterogeneous configuration that allows clear energy savings compared to the best homogeneous configuration. In 21 of 33 tested cases, the group of

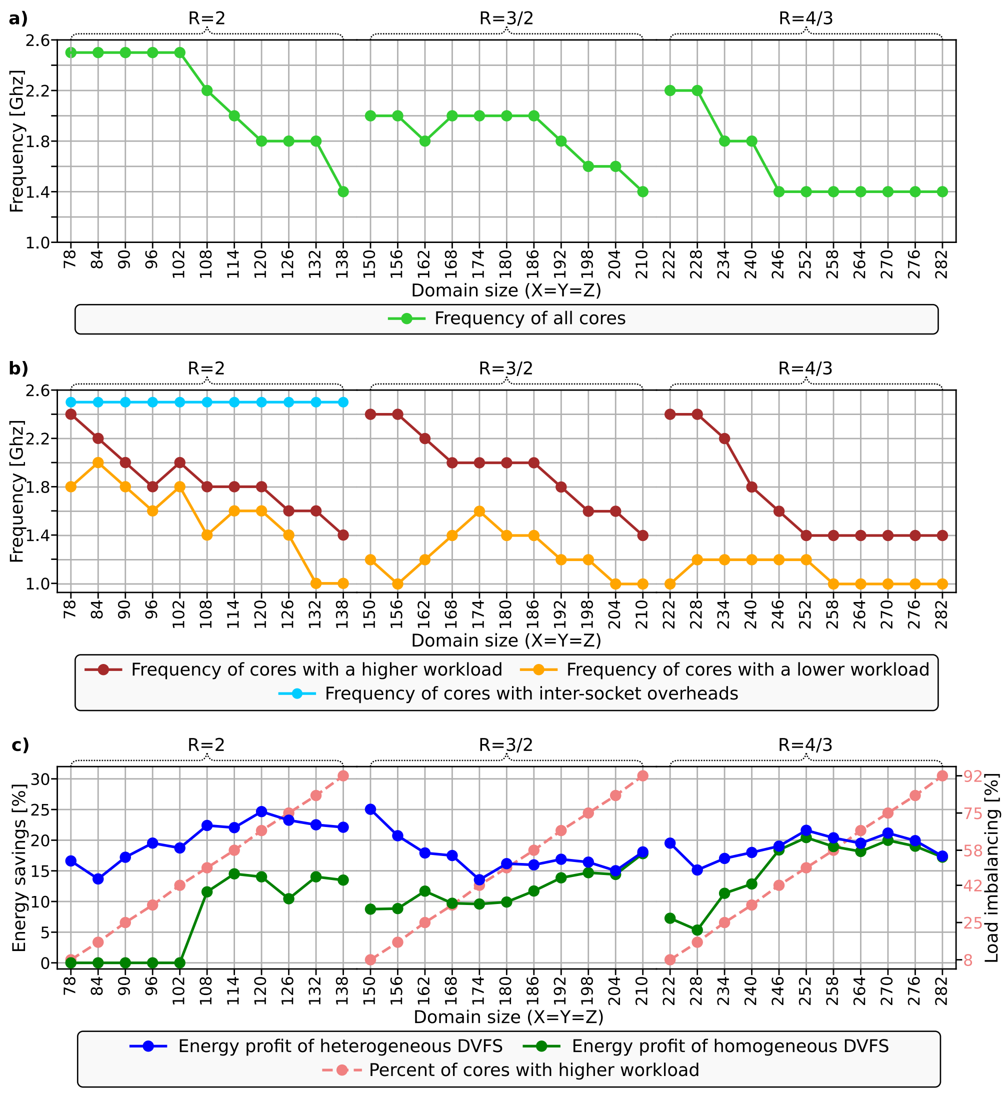

Fig. 3. Comparison of energy-savings for homogeneous and heterogeneous DVFS approaches applied to various domain sizes when solving 3D diffusion problem.

cores with a lower workload has assigned the frequency of 1.0 GHz – the minimum one available for the test platform. Thus energy savings could be even higher if further frequency lowering were possible.

6. Pruning algorithms for selecting heterogeneous DVFS configurations

The brute-force search technique allows us to discover a combination of frequencies that provides minimal energy consumption without performance degradation. However, the significantly high cost of this approach makes it an impractical way of finding optimal frequency settings. Let us assume we begin tests with a clock speed equal to f_{max}. Then we decrease frequency starting from f_{start} up to f_{stop} by every f_{step}. Let $\mathbf{F} = \{f_{max}, f_{start}, \ldots, f_{stop}\}$ be a set of tested frequencies, where the cardinality N of the set $\mathbf{F}$ is as follows:

$$N = |\mathbf{F}| = (f_{start} - f_{stop})/f_{step} + 2 \tag{5}$$

According to the brute-force technique of searching for frequencies for cores with higher and lower workloads, we combine each frequency f_i from the set $\mathbf{F}$ with a subset of $\mathbf{F}$ containing elements f_j such that $f_j \le f_i$. As a result, the number TC of tested configuration is expressed as:

$$TC = N(N+1)/2 \tag{6}$$

For example, the brute-force strategy with f_{max}, f_{start}, f_{stop}, and f_{step} equal to respectively 2.5, 2.4, 1.0, and 0.2 GHz, requires testing $TC = 45$ frequency configurations. We perform energy consumption tests for each of them, and their run time should be long enough to ensure the stability of measurements. Therefore, finding the best frequency configuration in this way is a highly time-consuming task. In this section, we outline two versions of the pruning algorithm for selecting the best heterogeneous DVFS configuration. These algorithms allow us to significantly reduce the number of tested configurations and speed up finding the best frequency configuration.

6.1. Algorithm based on energy measurement

In the first version of the pruning algorithm, decisions are made based on energy consumption measurements. We introduced this approach in our previous work [5], where we proved its effectiveness using the 3D diffusion problem as a use case. The proposed algorithm is shown in Fig. 5. It consists of two stages, the aim of which is to find the best frequencies f_{hw}, f_{lw} for cores with

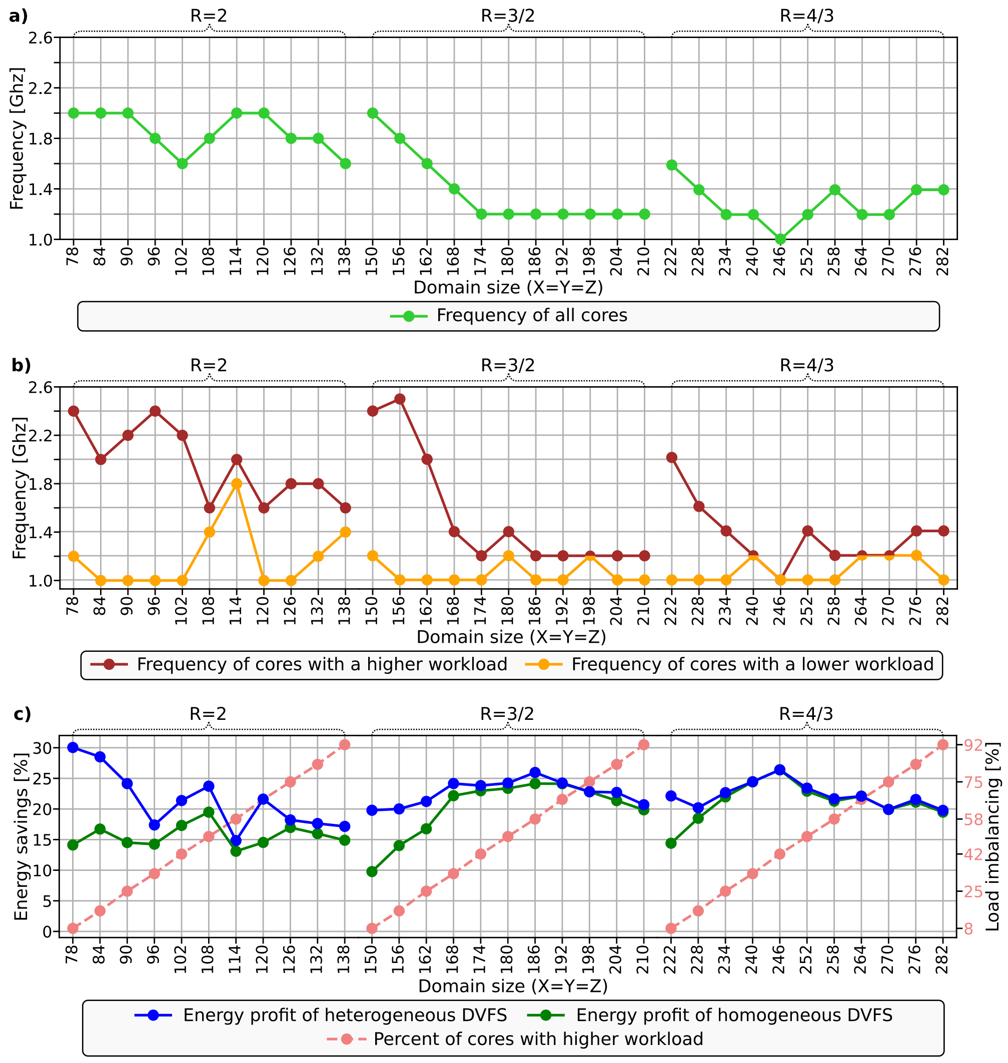

Fig. 4. Comparison of energy-savings for homogeneous and heterogeneous DVFS approaches applied to various domain sizes in MPDATA use case.

higher and lower workloads, respectively. The input parameters of the algorithm include L_a, which is the acceptable performance loss. For example, $L_a = 1.02$ allows the frequency to be reduced until a 2% increase in the execution time is achieved. We use the routine $Set\,(f_1, f_2)$ to examine frequency configurations - it sets the clock speeds f_1 and f_2 for groups of cores with higher and lower workloads, respectively. The routine $Set\_b(f)$ sets the frequency f for cores located on the boundaries between sockets, while the routine $Measure\,(T, E)$ is responsible for measuring the current configuration's execution time T and energy consumption E.

To provide the accuracy of results, we repeat all measurements m times and calculate their median value ($m = 5$ as default). The algorithm checks if all data reside in the cache memory. When this condition is met, cores located on the boundaries between sockets have the highest time of actual computation (Figs. 1a and 1b). In this case, the best strategy is to let these cores work with the maximum frequency to deal with data exchange overheads.

6.2. Algorithm based on execution time measurement

Energy consumption measurements are required to implement the algorithm outlined in the previous section (see Fig. 5). Monitoring energy consumption is widely available using hardware- and software-based techniques [35]. In this work, we employ the first technique using the Yokogawa WT310 digital power meter that monitors the entire HPC platform (see Sec. 3.3). An alternative solution can be Intel's Running Average Power Limit (RAPL) [19] interface, which represents the software technique. In contrast to digital power meters, the RAPL provides energy readings for different parts of the HPC platform, including CPU packages and DRAM domains.

However, during our study, we observe that both techniques can bring challenging tasks of accurate and reliable energy/power measurements. We also notice that RAPL tools are not available on some clusters equipped with AMD-based CPUs, making energy measurements unavailable. Furthermore, the standard tools that provide measurements from RAPL (e.g., PAPI [38] or LIKWID [39]) sometimes do not support the latest architectures correctly.

To increase the usability of the proposed heterogeneous DVFS strategy, we develop a solution that does not require energy consumption measurements. Instead, the algorithm makes decisions based on the execution time at a given frequency. Such an approach widens the scope of our strategy since time measurements do not require additional effort and are trivial even for nonadvanced users.

Fig. 6 presents the proposed algorithm. Its idea is similar to the algorithm based on energy measurements. The input parameters and routines for setting the given frequency for the groups of cores remain unchanged. The update affects the *Measure* routine, with only one parameter related to the execution time. The

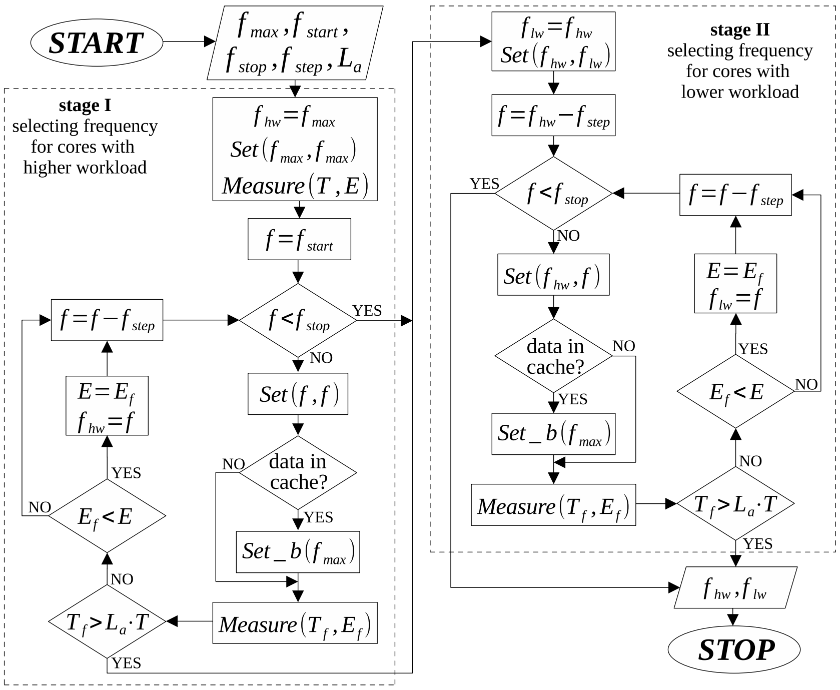

Fig. 5. Block diagram of the pruning algorithm based on energy measurements for selecting heterogeneous DVFS configurations [5].

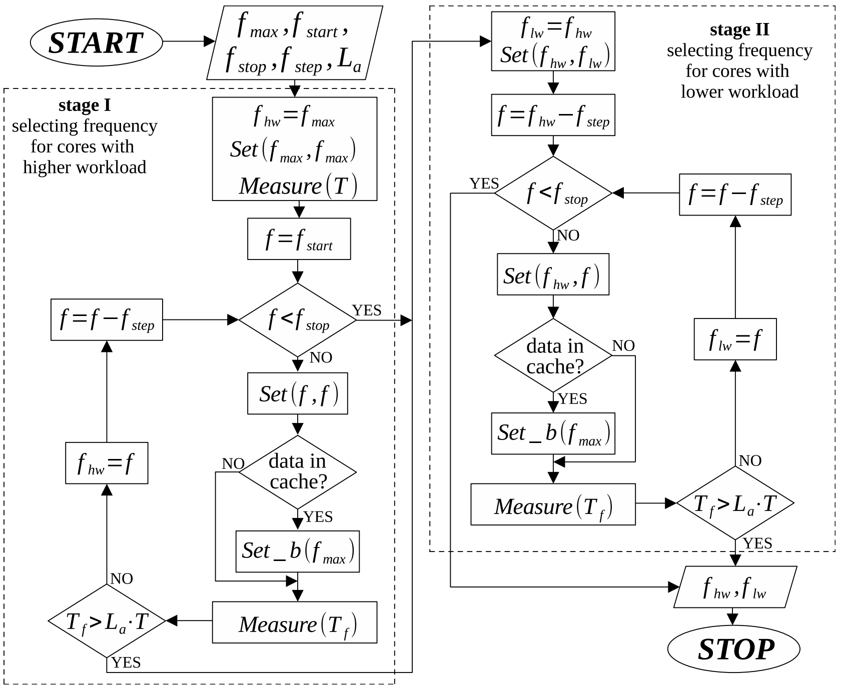

Fig. 6. Block diagram of the pruning algorithm based on execution time measurements for selecting heterogeneous DVFS configurations.

key difference is the refusal to analyze energy consumption for the subsequent launches of an application. At each stage, the algorithm selects the lowest frequency that does not result in exceeding the fixed execution time limit.

7. Evaluation of the proposed approach

7.1. Experimental validation

We perform a set of tests to explore the efficiency of the proposed algorithms for both use cases. The analysis concerns the three search strategies: the brute-force (BF) search approach as a baseline and two proposed algorithms based on measurements of energy (AE) and time (AT). We evaluate each solution based on the number of tested configurations required to return a final answer by a given strategy and the percentage of energy reduction achieved. Additionally, we assess each algorithm's efficiency as the ratio (in %) of the energy reduction achieved by a pruning algorithm and brute-force search. The comparison concerns nine domain sizes corresponding to various R and LIP parameters.

Table 1 presents the results obtained for the first use case. The expected consequence of testing all possible configurations is that the BF strategy always offers the best results regarding energy re-

Table 1
Comparison of the brute-force search (BF) and proposed algorithms (AE, AT) for the 3D diffusion application with different domain sizes $(X = Y = Z)$.

R		2			3/2			4/3		
LIP [%]		8	25	50	8	25	50	8	25	50
Domain size		78	90	108	150	162	180	222	234	252
BF	f_{hw} [GHz]	2.4	2.0	1.8	2.4	2.2	2.0	2.4	2.2	1.4
	f_{lw} [GHz]	1.8	1.8	1.4	1.2	1.2	1.4	1.0	1.2	1.2
	f_{bs} [GHz]	2.5	2.5	2.5	–	–	–	–	–	–
	TC	45	45	45	45	45	45	45	45	45
	Energy reduction [%]	16.6	17.1	22.9	25.0	17.9	16.2	19.5	17.0	21.6
AE	f_{hw} [GHz]	2.4	2.0	1.6	2.0	2.2	2.0	2.2	1.8	1.4
	f_{lw} [GHz]	1.8	1.8	1.2	1.0	1.2	1.4	1.2	1.4	1.2
	f_{bs} [GHz]	2.5	2.5	2.5	–	–	–	–	–	–
	TC	8	7	11	10	10	9	10	9	10
	Energy reduction [%]	16.6	17.1	22.4	24.0	17.9	16.2	18.3	16.0	21.6
	Efficiency [%]	100	100	97.9	95.8	100	100	93.9	93.8	100
AT	f_{hw} [GHz]	2.4	2.0	1.4	2.0	2.2	2.0	2.2	1.8	1.4
	f_{lw} [GHz]	1.6	1.8	1.2	1.0	1.2	1.4	1.2	1.4	1.0
	f_{bs} [GHz]	2.5	2.5	2.5	–	–	–	–	–	–
	TC	8	7	10	10	10	9	10	9	10
	Energy reduction [%]	15.89	17.1	20.7	24.0	17.9	16.2	18.3	16.0	21.0
	Efficiency [%]	95.7	100	90.4	95.8	100	100	93.9	93.8	97.1

Table 2
Comparison of the three search strategies for the MPDATA application with different domain sizes $(X = Y = Z)$.

R		2			3/2			4/3		
LIP [%]		8	25	50	8	25	50	8	25	50
Domain size		78	90	108	150	162	180	222	234	252
BF	f_{hw} [GHz]	2.4	2.2	1.6	2.4	2.0	1.4	2.0	1.4	1.4
	f_{lw} [GHz]	1.2	1.0	1.4	1.2	1.0	1.2	1.0	1.0	1.0
	TC	45	45	45	45	45	45	45	45	45
	Energy reduction [%]	30.0	24.1	23.7	19.8	21.2	24.2	22.1	22.7	23.4
AE	f_{hw} [GHz]	2.0	2.0	2.0	2.2	2.0	1.4	2.0	1.6	1.8
	f_{lw} [GHz]	1.2	1.2	1.2	1.4	1.0	1.2	1.0	1.0	1.0
	TC	10	10	10	9	10	10	10	10	10
	Energy reduction [%]	28.7	24.0	23.1	18.9	21.2	24.2	22.1	21.5	21.1
	Efficiency [%]	95.5	99.2	97.5	95.7	100	100	100	95.0	90.3
AT	f_{hw} [GHz]	2.0	2.0	2.0	2.2	2.0	1.4	2.0	1.6	1.8
	f_{lw} [GHz]	1.2	1.0	1.2	1.4	1.0	1.0	1.0	1.0	1.0
	TC	10	10	10	9	10	10	10	10	10
	Energy reduction [%]	28.7	22.1	23.1	18.9	21.2	23.4	22.1	21.5	21.1
	Efficiency [%]	95.5	91.5	97.5	95.7	100	96.6	100	95.0	90.3

duction. However, both pruning algorithms reach at least 90% of efficiency in all tests. Furthermore, the algorithm based on energy measurements in more than half tests achieves 100% efficiency, while its time-based counterpart reaches this goal in 3 out of 9 domain sizes.

In particular, for the domain of size 78x78x78, the best solution is to set frequencies for groups of cores with higher and lower workloads as 2.4 GHz and 1.8 GHz, respectively, while the cores located on the border between the sockets have to work with the maximum available frequency of 2.5 GHz to amortize inter-socket overheads. This setting reduces energy consumption by 16.6% while preserving less than a 2% increase in the execution time compared to the result obtained for the homogeneous frequency of 2.5 GHz. For this domain, the efficiency of the AE strategy is 100%, which means that it returns the same set of frequencies as the BF approach. For comparison, the AT algorithm recommends setting the frequency of lower workload cores to 1.6 GHz, which permits reducing energy consumption by 15.9% and achieves an efficiency of 95.7%.

For the first use case, the proposed pruning algorithms significantly reduce the cost of selecting heterogeneous DVFS configurations. For instance, while the brute-force search always requires exploring 45 frequency configurations, both pruning algorithms need only 7-11 configurations.

The same conclusions stay valid for the second use case – the MPDATA application. It follows from Table 2 that in all tests, the efficiency of both pruning algorithms exceeds 90%. They require checking 9-10 configurations to find a solution compared to the fixed number of 45 configurations needed by the brute-force method.

For both pruning algorithms, the lowest efficiency is achieved for the domain of size 252x252x252. Setting frequencies at 1.8 and 1.0 GHz reduces energy consumption by 21.1% with an efficiency of 90.3% against the brute-force approach. Here the BF strategy achieves a better result by lowering the frequency of cores with a higher workload to 1.4 GHz. For large domains (see $R = 4/3$ in Table 2), the frequency for a group of cores with a lower workload is 1.0 GHz in both pruning algorithms. Therefore, we believe that obtained results could be even better if the target computing platform allows further frequency lowering.

7.2. Cost of selecting heterogeneous DVFS configuration

Proposed algorithms assume an offline approach and require running an application with some number (TC) of different DVFS

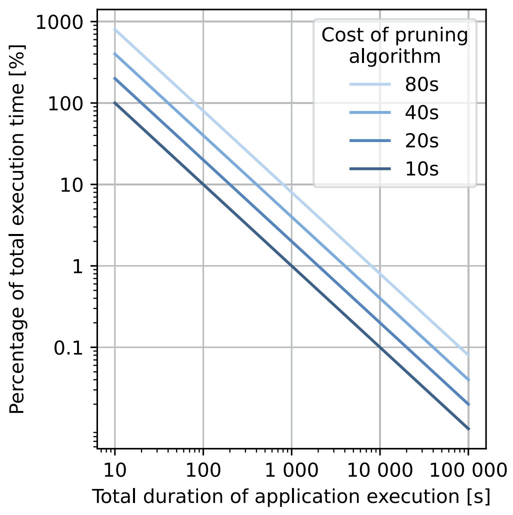

Fig. 7. The predicted cost of searching for a heterogeneous DVFS configuration using pruning algorithms as percentage of total execution time.

configurations to select the optimal one. The overhead introduced by such a strategy depends on a specific case and hence is not easy to state a-priori.

The behavior of algorithms presented in Tables 1 and 2 is inferred from the experiments where application runs last several hundred seconds. However, in practice there is no need for such long runtimes to keep the reliability of measurements, and algorithms can make proper decisions based on much shorter simulations. We note that managing the duration of considered applications is simple since we can customize the number of time steps which does not affect the workload distribution described by R and LIP parameters.

We test both use cases and notice that about 10-second simulations provide sufficient time and energy measurement accuracy for selecting the optimal configuration using proposed algorithms. Hence, assuming a relatively small number of tested DVFS configurations ($TC \leq 11$ in our experiments), we can estimate the cost of our strategy as tens of seconds. Fig. 7 shows the impact of such overhead on the overall performance depending on different target application durations. If the target run of the application is supposed to be single and short, the overhead of selecting a DVFS configuration may be too high. However, in many practical scenarios of considered use cases, calculations last for hours and hence cost of a few dozen seconds is negligible. For instance, assuming the target application is expected to run for at least 2 hours, the overhead will not exceed 1%. Moreover, the result obtained with pruning algorithms can be used multiple times. An example is the field of numerical weather prediction, where simulations with a fixed domain size and given hardware are repeated even thousands of times using MPDATA.

8. Exploration of heterogeneous DVFS approach using the 3rd generation of AMD EPYC CPUs

In this section, we explore the advantages of the heterogeneous DVFS strategy for the Asus RS720A-E11-RS12 server equipped with two top-of-the-line AMD EPYC 7773X CPUs (Milan-X). Each CPU has 64 cores grouped into 8 Core Complex Dies and provides a massive 768MB L3 cache [4]. As a result, the entire computing platform offers 256 logical cores and over 1.5 GB of the last-level cache.

Using such a large number of cores in the AMD EPYC computing platform allows us to evaluate our approach using larger domain sizes. While for the Intel Cascade Lake architecture $R = 2$ corresponds to domains with size X in the range $(72, 144)$, for AMD EPYC this value of R refers to $(256, 512)$. At the same time, AMD Milan-X CPUs offer reduced capabilities of frequency management in comparison to Intel CPUs, limiting the frequency setups only to four levels: 1.5 GHz, 1.9 GHz, 2.2 GHz, and the turbo mode that raises the clock up to 3.5 GHz.

We perform similar brute-force search experiments as presented in Section 5. For each use case, we select eight domain sizes corresponding to $R = 2$ and values of LIP varying from 6 to 88%. Fig. 8 shows a summary of performed tests for both use cases. As before, we search for frequency configurations that provide the lowest energy consumption and do not cause more than 2% performance loss compared to results obtained for the server running with turbo frequency. While the top and middle graphs illustrate selected frequency settings for homogeneous and heterogeneous approaches, the bottom charts compare them regarding energy savings. As shown in Fig. 8, the proposed strategy achieves better results than the homogeneous variant of DVFS. Both use cases follow a similar pattern and benefit the most from heterogeneous DVFS for the lowest values of the LIP parameter. As a result, the highest difference in energy savings between homogeneous and heterogeneous variants for 3D Diffusion and MPDATA applications is about 13 and 12 percentage points, respectively.

In all performed tests, the frequency selected for cores with a lower workload equals 1.5 GHz, which corresponds to the lowest available level. Hence, we expect that heterogeneous DVFS could provide even better results if AMD EPYC CPUs offered a broader range of available frequencies. At the same time, the relatively small number of frequency setups for the AMD-based platform reduces the searching space of the optimal heterogeneous DVFS configuration compared to the Intel-based platform. Even in the Brute-Force Search approach, the total number of explored frequency setups does not exceed 10, and the proposed pruning algorithms decrease it twice.

9. Conclusions

This paper explores the advantages of heterogeneous DVFS control and performance heterogeneity in enhancing the energy-performance behavior of data-parallel applications on homogeneous multicore CPU systems. Using two different use cases, we prove the significant energy improvement of heterogeneous voltage/frequency scaling compared to the homogeneous technique. The proposed approach provides the advantage of up to 20 percentage points over the homogeneous frequency scaling during experiments on the ccNUMA server with two 18-core Intel Xeon 6240 CPUs.

To enable the application of heterogeneous DVFS, we develop two pruning algorithms and compare their cost and efficiency against the brute-force search experimentally. For both use cases, the efficiency of the pruning algorithms exceeds 90% and often reaches 100%, which indicates that both techniques return similar energy savings. At the same time, in this way, we significantly reduce the cost of selecting a heterogeneous DVFS configuration. While the brute-force search always requires exploring 45 frequency configurations, the pruning algorithms achieve the goal by examining only 7-11 configurations.

Moreover, we prove the effectiveness of our approach using the state-of-the-art 64-core AMD EPYC processors. Despite limited ability to manage the clock frequency, we achieve up to 13 percentage points advantage compared to the homogeneous variant of DVFS for both use cases.

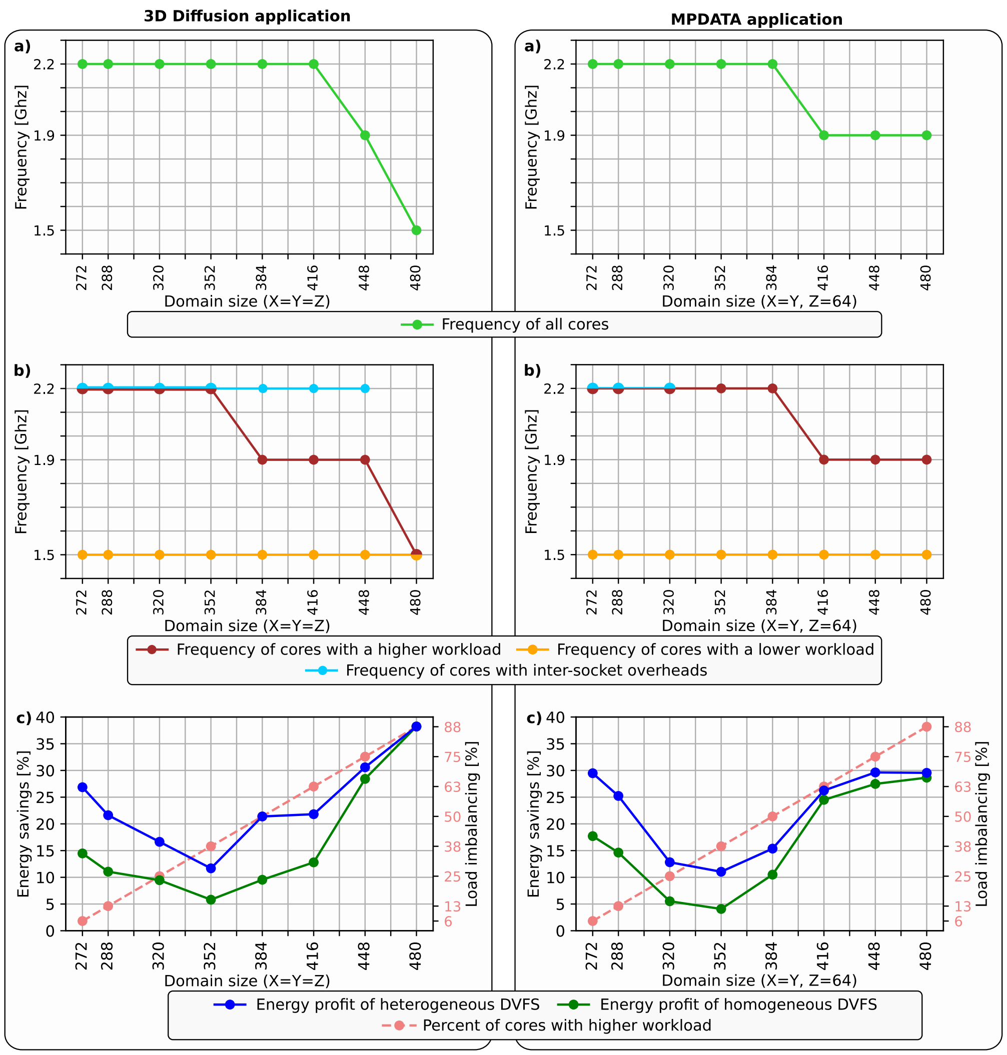

Fig. 8. Comparison of energy-savings for homogeneous and heterogeneous DVFS approaches using two 64-core AMD EPYC 7773X CPUs.

CRediT authorship contribution statement

We the undersigned declare that this manuscript is original, has not been published before and is not currently being considered for publication elsewhere. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us. We understand that the Corresponding Author is the sole contact for the Editorial process. He/she is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the National Science Centre (Poland) under grant no. UMO-2017/26/D/ST6/00687, and by the project no. 020/RID/2018/19 financed within the program of the Polish Ministry of Science and Higher Education “Regional Initiative of Excellence” (years 2019 – 2023, the amount of financing 12 000 000 PLN).

References

[1] S. Abera, M. Balakrishnan, A. Kumar, Performance-energy trade-off in CMPs with per-core DVFS, in: Int. Conf. Architecture of Computing Systems (ARCS 2018), 2018, pp. 225–238.
[2] B. Acun, P. Miller, L. Kale, Variation among processors under turbo boost in HPC systems, in: ICS'16: Proc. 2016 Int. Conf. on Supercomputing, 2016, pp. 1–12.
[3] B. Acun, K. Chandrasekar, L. Kale, Fine-grained energy efficiency using per-core DVFS with an adaptive runtime systems, in: 2019 Tenth Int. Green and Sustainable Computing Conf. (IGSC), vol. 1, 2019, pp. 1–8.
[4] AMD EPYC 7773X specification, https://www.amd.com/pl/products/cpu/amd-epyc-7773x.
[5] P. Bratek, L. Szustak, R. Wyrzykowski, T. Olas, T. Chmiel, Heterogeneous voltage frequency scaling of data-parallel applications for energy saving on homogeneous multicore platforms, in: Proc. Euro-Par 2021: Parallel Processing Workshops, in: Lect. Notes Comp. Sci., vol. 13098, 2022, pp. 1–12.
[6] E. Calore, A. Gabbana, S. Schifano, R. Tripiccione, Software and DVFS tuning for performance and energy-efficiency on Intel KNL processors, J. Low Power Electron. Appl. 8 (2) (2018).

[7] D. Cesarini, A. Bartolini, L. Benini, Benefits in relaxing the power capping constraint, in: ANDARE'17: Proc. 1st Workshop on AutotuniNg and aDaptivity ApPRoaches for Energy Efficient HPC Systems, ACM, 2017, pp. 1–6.
[8] M. Ciznicki, K. Kurowski, Resource management strategies with energy profiles for stencil computing, in: HiStencil 2015: 2nd Int. Workshop on High-Performance Stencil Computating, 2015, pp. 943–950.
[9] J. Crank, Mathematics of Diffusion, second edition, Clarendon Press, 1975.
[10] Q. Deng, D. Meisner, A. Bhattacharjee, T. Wenisch, R. Bianchini, CoScale: coordinating CPU and memory system DVFS in server systems, in: MICRO-45: Proc. 2012 45th Annual IEEE/ACM Int. Symp. on Microarchitecture, IEEE Xplore, 2012, pp. 143–154.
[11] K. Eder, J.P. Gallagher, Energy-aware software engineering, in: G. Fagas, L. Gammaitoni, J.P. Gallagher, D.J. Paul (Eds.), ICT - Energy Concepts for Energy Efficiency and Sustainability, IntechOpen, Rijeka, 2017.
[12] P. Ezzatti, E.S. Quintana-Ortí, A. Remón, J. Saak, Power-aware computing, Concurr. Comput., Pract. Exp. 31 (6) (2019).
[13] M. Fahad, A. Shahid, R. Manumachu, A. Lastovetsky, Energy predictive models of computing: theory, practical implications and experimental analysis on multicore processors, IEEE Access 9 (2021) 63149–63172.
[14] M. Gupta, L. Bhargava, I. Sreedevi, Dynamic voltage frequency scaling in multicore systems using adaptive regression model, in: Proc. 4th Int. Conf. IoT in Social, Mobile, Analytics and Cloud (I-SMAC), 2020, pp. 1201–1206.
[15] A. Haidar, H. Jagode, P. Vaccaro, A. Yarkhan, S. Tomov, J. Dongarra, Investigating power capping toward energy-efficient scientific applications, Concurr. Comput., Pract. Exp. (2018), https://doi.org/10.1002/cpe.4485.
[16] J. Haj-Yahya, A. Mendelson, Y. Ben-asher, Energy Efficient High Performance Processors: Recent Approaches for Designing Green High Performance Computing, Springer, 2018.
[17] H.H. Hassan, A.S. Moussa, I. Farag, Performance vs. power and energy consumption: impact of coding style and compiler, Int. J. Adv. Comput. Sci. Appl. 8 (12) (2017) 132–142.
[18] J. Jeffers, J. Reinders, A. Sodani, Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition, Morgan Kaufmann, 2016.
[19] K.N. Khan, M. Hirki, T. Niemi, J.K. Nurminen, Z. Ou, Rapl in action: experiences in using rapl for power measurements, ACM Trans. Model. Perform. Eval. Comput. Syst. 3 (2) (mar 2018), https://doi.org/10.1145/3177754.
[20] T. Kolpe, A. Zhai, S. Sapatnekar, Enabling improved power management in multicore processors through clustered DVFS, in: 2011 Design, Automation & Test in Europe, 2011, pp. 1–6.
[21] A. Lastovetsky, R. Manumachu, Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy, IEEE Trans. Comput. 67 (2) (2018) 160–177.
[22] J. Mair, Z. Huang, D. Eyers, Y. Chen, Quantifying the energy efficiency challenges of achieving exascale computing, in: 15th IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing, 2015, pp. 943–950.
[23] A. Papadimitriou, A. Chatzidimitriou, D. Gizopoulos, Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore CPUs, in: 2019 IEEE Int. Symp. High Performance Computer Architecture (HPCA), 2019, pp. 133–146.
[24] A. Prakash, S. Wang, A. Irimiea, T. Mitra, Energy-efficient execution of data-parallel applications on heterogeneous mobile platforms, in: 2015 33rd IEEE Int. Conf. on Computer Design (ICCD), IEEE, 2015.
[25] T. Rauber, G. Rünger, M. Stachowski, Performance and energy metrics for multi-threaded applications on DVFS processors, Sustain. Comput., Inf. Syst. 17 (2018) 55–68.
[26] K. Rojek, A. Ilic, R. Wyrzykowski, L. Sousa, Energy-aware mechanism for stencil-based MPDATA algorithm with constraints, Concurr. Comput., Pract. Exp. 29 (8) (2017).
[27] K. Rojek, E. Quintana-Orti, R. Wyrzykowski, Modeling power consumption of 3D MPDATA and the CG method on ARM and Intel multicore architectures, Concurr. Comput., Pract. Exp. 73 (2017) 4373–4389.
[28] B. Rosa, L. Szustak, A. Wyszogrodzki, K. Rojek, D. Wojcik, R. Wyrzykowski, Adaptation of multidimensional positive definite advection transport algorithm to modern high-performance computing platforms, Int. J. Model. Optim. 5 (3) (2015) 171–176.
[29] K. Sankaralingam, S. Keckler, W. Mark, D. Burger, Universal mechanisms for data-parallel applications, in: Proc. 36th Int. Symp. Microarchitecture (MICRO-36), 2003.
[30] P. Smolarkiewicz, Multidimensional positive definite advection transport algorithm: an overview, Int. J. Numer. Methods Fluids 50 (10) (2006) 1123–1144.
[31] L. Szustak, Strategy for data-flow synchronizations in stencil parallel computations on multi-/manycore systems, J. Supercomput. 74 (4) (2018) 1534–1546.
[32] L. Szustak, P. Bratek, Performance portable parallel programming of heterogeneous stencils across shared-memory platforms with modern Intel processors, Int. J. High Perform. Comput. Appl. 33 (3) (2019) 507–526.
[33] L. Szustak, K. Halbiniak, R. Wyrzykowski, O. Jakl, Unleashing the performance of ccNUMA multiprocessor architectures in heterogeneous stencil computations, J. Supercomput. 75 (2019) 7765–7777.
[34] L. Szustak, R. Wyrzykowski, K.K. Halbiniak, P. Bratek, Toward heterogeneous MPI+MPI programming: comparison of OpenMP and MPI shared memory models, in: Proc. Euro-Par 2019: Parallel Processing Workshops, in: Lect. Notes Comp. Sci., vol. 11997, 2020, pp. 270–281.
[35] L. Szustak, R. Wyrzykowski, T. Olas, V. Mele, Correlation of performance optimizations and energy consumption for stencil-based application on Intel Xeon scalable processors, IEEE Trans. Parallel Distrib. Syst. 31 (11) (2020) 2582–2593.
[36] L. Szustak, et al., Architectural adaptation and performance-energy optimization for CFD application on AMD EPYC Rome, IEEE Trans. Parallel Distrib. Syst. 32 (12) (2021) 2852–2866.
[37] Technology Guide, Intel speed select technology - core power, https://networkbuilders.intel.com/solutionslibrary/intel-speed-select-technology-core-power-intel-sst-cp-overview-technology-guide, April 2022.
[38] D. Terpstra, H. Jagode, H. You, J. Dongarra, Collecting performance data with PAPI-C, in: Tools for High Performance Computing, 2010, pp. 157–173.
[39] J. Treibig, G. Hager, G. Wellein, Likwid: a lightweight performance-oriented tool suite for x86 multicore environments, in: Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures, San Diego, CA, 2010.
[40] Y. Wang, D. Nörtershäuser, S. Le Masson, J.-M. Menaud, An empirical study of power characterization approaches for servers, in: ENERGY 2019 - 9th Int. Conf. on Smart Grids, Green Communications and IT Energy-Aware Technologies, Jun 2019, Athens, Greece, 2019, pp. 1–5.
[41] J. Winter, C. Albonesi, D.H. Shoemaker, Scalable thread scheduling and global power management for heterogeneous many-core architecture, in: Proc. PACT'10, 2010, pp. 29–40.
[42] R. Wyrzykowski, K. Rojek, L. Szustak, The impact of voltage-frequency scaling for the matrix-vector product on the IBM POWER8, in: Euro-Par 2016: 22nd European Conference on Parallel Processing, in: Lect. Notes Comp. Sci., vol. 9833, Springer, 2016, pp. 103–116.
[43] D. Zill, W. Wright, Differential Equations with Boundary-Value Problems, eight edition, Brooks/Cole Cengage Learning, 2012.

Pawel Bratek received the MSc degree in mathematics and BEng degree in computer science from the Czestochowa University of Technology in 2018 and 2019. Currently, he is PhD student in computer science. His main research interests include adapting parallel computing to modern HPC architectures and developing efficient strategies for handling data access queries in machine learning applications.

Lukasz Szustak received a D.Sc. degree in Computer Science in 2019 and a Ph.D. granted by the Czestochowa University of Technology in 2012. His main research interests include parallel computing and mapping algorithms onto parallel architectures. His current work is focused on the development of methods for performance portability, scheduling, and load balancing, including the adaptation of stencil-based computations to modern HPC architectures.

Roman Wyrzykowski received the M.Sc. and Ph.D. degrees from the Kiev Polytechnic Institute in Computer Science, in 1982 and 1986, respectively. Since 1982, he is employed at Czestochowa University of Technology, Poland. His fields of expertise are: parallel and distributed computing, mapping algorithms onto cluster and cloud systems. Since 1994, he has chaired the program committee of the PPAM series of international conferences on parallel processing.

Tomasz Olas received a M.Sc. degree in Computer Science in 1999 from the Czestochowa University of Technology, Poland. In 2004 he received his Ph.D. degree in Computer Science for the dissertation on mapping FEM computations onto parallel and distributed systems. His main research interests include parallel and distributed computing, mapping algorithms onto parallel architectures, cluster and cloud technologies.