Heterogeneous Voltage Frequency Scaling of Data-Parallel Applications for Energy Saving on Homogeneous Multicore Platforms

Pawel Bratek(✉), Lukasz Szustak, Roman Wyrzykowski, Tomasz Olas, and Tomasz Chmiel

Department of Computer Science, Czestochowa University of Technology, Dabrowskiego 69, 42-201 Czestochowa, Poland
pawel.bratek@pcz.pl, {lszustak,roman,olas,tchmiel}@icis.pcz.pl

Abstract. In this paper, for the first time, we explore and establish the combined benefits of heterogeneous DVFS (dynamic voltage frequency scaling) control in improving the energy-performance behavior of data-parallel applications on shared-memory multicore systems. We propose to customize the clock frequency individually for the appropriately selected groups of cores corresponding to the diversified time of actual computation. In consequence, the advantage of up to 20% points over the homogeneous frequency scaling is achieved on the ccNUMA server with two 18-core Intel Xeon Gold 6240 containing 72 logical cores in total. The cost and efficiency of the proposed pruning algorithm for selecting heterogeneous DVFS configurations against the brute-force search are verified and compared experimentally.

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

1 Introduction

Energy efficiency becomes one of the main challenges in the race of high-performance computing (HPC) to Exascale [11]. The scientific community is attempting to address this challenge in different ways on both hardware and software levels. State-of-the-art solution methods in this area can be generally divided [6] into system-level and application-level categories.

A widely accepted technique from the first category is dynamic voltage and frequency scaling (DVFS) [13]. It is known [8] as an efficient method to save energy for memory-bound applications when CPU cycles are being wasted as they are stalled on the main memory [8]. Using DVFS allows lowering the operational voltage/frequency at the cost of possibly higher execution time [3].

For multicore processors with shared memory, DVFS can be performed [9] at various level of granularity: (i) *per-chip* DVFS with changing the whole chip's

R. Chaves et al. (Eds.): Euro-Par 2021, LNCS 13098, pp. 141–153, 2022.
https://doi.org/10.1007/978-3-031-06156-1_12

frequency, (ii) *cluster-level* DVFS with multiple on-chip voltage regulators driving a set of DVFS domains, and (iii) per-core DVS with a separate regulator for each core. In particular, the per-core frequency control is available in more recent Intel processors (based on the Haswell architecture and later) by limiting the minimum and maximum frequencies for a given CPU core [18]. While the first level represents homogeneous voltage frequency scaling across the processor, the other two levels correspond to heterogeneous scaling, which can help [2] getting the most performance out of the system.

Data parallelism is the most common parallel decomposition strategy, by which an application's data domain is decomposed into as many data partitions as threads assigned to the computation. In data-parallel model, tasks are assigned to threads, and each task performs similar types of operations on different data. At an abstract programming level, data-parallel programs consist of a loop body executing on different parts of the input data [16].

Data-parallel programs are growing in importance, increasing in diversity, and demanding increased performance from hardware while preserving minimized energy consumption. In our previous works, we explore the usability of the DVFS technique as a tool for balancing energy savings with admissible performance losses for such data-parallel algorithms/applications as 3D MPDATA from computational fluid dynamics [14,17], and conjugate gradient [15].

Another example of using DVFS for data-parallel applications is presented in paper [4], which explores the relationship between task scheduling and energy constraints for stencil computation, a class of memory-bound applications that are quite common in scientific computing. This paper and our previous works apply homogeneous voltage frequency scaling for multicore CPUs, possibly combined with concurrency throttling. This solution seems to be a natural choice for homogeneous multicore and regular data-parallel applications structured with a uniform behavior when all cores or threads execute a similar type of work [2].

Typically, using heterogeneous DVFS across homogeneous multicore CPUs is justified [2] for irregular or unstructured applications when at a given time, cores might do different types of work. On the contrary, in this paper, *we explore and establish for the first time the combined benefits of heterogeneous DVFS control and performance heterogeneity in improving the energy-performance behavior of regular data-parallel applications on homogeneous multicore CPU systems with shared memory*, including ccNUMA (cache-coherent non-uniform memory access) ones. The rationale behind these benefits lies in the evolving relationship between feasible sizes of applications (determined by sizes of data sets) and increasing variety in the core number. The consequence is thread divergence within an application and load imbalancing across cores, resulting in deteriorating energy efficiency. The usage of heterogeneous DVFS allows us to mitigate the deterioration and reduce energy consumption without the performance loss.

The material of this paper is organized in the following way. Section 2 describes the basics of our approach, including a use case of application studied in the paper, as well as a more detailed motivation and the problem statement. A brute-force search and a pruning algorithm for selecting heterogeneous DVFS

configuration across homogeneous multicore platforms are proposed in Sect. 3 and Sect. 4, respectively. Section 5 provides an overview of related work, while Sect. 6 presents conclusions.

2 Basics

2.1 Use Case of Data-Parallel Application: 3D Diffusion Problem

As the bulk of physics phenomena, the diffusion process is described [5] by a partial differential equation shown below:

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

In Eq. (1), the function $U = U(x, y, z, t)$ describes concentration of a physical quantity in point (x, y, z) at moment t. In some sense, this equation is universal. For example, it can describe the process of heat transfer where the unknown function $U(x, y, z, t)$ represents temperature. The studied application is based on the finite difference method [20], which results in the following equation:

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

We focus on modeling 3D diffusion problems defined over the structured rectilinear domain of sizes $X \times Y \times Z$ in $i-$, $j-$, and $k-$dimensions, respectively. This iterative numerical algorithm is intended to run long simulations engaging even many thousands of time steps. Each step performs a single computing kernel - a 3D stencil code. The application code features a constant computation intensity for all steps, so the simulation time is proportional to their number.

In the basic code of the application (Listing 2.1), the computing kernel is implemented in parallel using the OpenMP standard. The parallelization strategy exploits data parallelism across $i-$dimension, based on distributing data across available resources by `#pragma omp for` directive, and then incorporates vectorization along $k-$dimension using `#pragma vector` directive.

The studied application is characterized by vector-friendly data structures that enable taking advantage of the vectorization process. However, the data traffic requirements constraint the parallel efficiency of the code by the main memory bandwidth. The performance bottleneck is mainly noticeable for rather large problems with domain sizes that significantly exceed the cache capacity.

In this work, all experiments are performed 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 is equipped with 24.75 MB of L3 cache. The thermal and power limitations of the test platform permit 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.

All energy measurements are provided by the Yokogawa WT310 power meter [17], monitoring the entire platform. This power meter passes the power to the

Listing 2.1. Parallelization of data-parallel application using OpenMP

```
#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))
```

server under the load and measures the total energy consumption in real-time [6, 17]. It allows us to obtain maximally accurate and reliable energy measurements. Moreover, to make sure the experimental results for energy are trustworthy, we customize the number of time steps for every tested domain size to keep the execution time at the level of at least 700 s. As a result, the relative standard deviation (RSD) for all benchmarks does not exceed 1.5% and 2.5% for execution time and energy consumption measurements, respectively. 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.

2.2 Motivation for the Research and Problem Statement

The proposed parallelization (see Listing 2.1) splits up iterations within the first loop (i-dimension) and distributes them over the available threads corresponding to the OpenMP parallel region. As a result, the uniform workload distribution occurs only when the number X of iterations is equally divided among the number LC of logical cores. Otherwise, the parallelization scenario leads to workload imbalance and thread divergence. In this case, the load imbalancing percentage (LIP) corresponds to the ratio of total number of threads that perform more loop iterations to the total number of logical cores:

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

Assuming $X > LC$, the more loaded threads perform R times more iterations than the rest of threads, where the parameter R can be expressed as:

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

Figure 1 characterizes the execution time and energy consumption when solving the 3D diffusion problem with various domain sizes. In all examples, $R = 2$. This study assumes a limit of 2% performance losses to achieve energy savings.

The left parts of Fig. 1 present the execution time and energy consumption for various domain sizes, 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 since reducing the frequency leads to energy savings but breaks the

assumed limit for performance losses. The reason is that all required data reside in the cache hierarchy, and computing units are not waiting for loading data from the main memory.

In contrast, in Figs. 1c and 1d, we observe that reducing the clock frequency does not affect performance for large domain sizes. Now the execution time is practically constant despite decreasing the frequency from the highest to the lowest one. In this case, the performance depends primarily on the main memory speed, and frequency scaling does not affect the execution time. Consequently, the DVFS method allows reducing the total energy consumption up to 30% with negligible performance losses not exceeding 1–2% for all performed tests.

However, our experiments reveal a need for revisiting the DVFS technique even in this case that commonly assumes homogeneous frequency scaling across cores. To explain this, let us move on to the analysis of the right parts of Fig. 1a–d, which demonstrate the execution time distribution across threads for different domain sizes and a variety of workloads with varying values of LIP parameter, where $LIP \in \{8, 25, 50\}\%$. More precisely, these plots show the total execution time that every OpenMP thread spends on (i) computation (area marked in blue color) and (ii) synchronization. In turn, the synchronization costs are split into two stages: (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).

The cost of the departure stage depends on the number of synchronization points, which in our study increases linearly with the number of time steps. As expected, the time required by this stage is practically uniform for every OpenMP thread. For a single synchronization point, this time mainly depends on the OpenMP implementation chosen and hardware limitations.

However, the cost of actual computation (see blue regions in the right parts of Fig. 1) differs across threads. The same is true for the cost of the arrival stage of synchronization. This heterogeneity results from non-uniform workload distributions over available cores/threads, described by LIP and R parameters.

Additionally, we observe a negative impact of the inter-socket data traffic on thread divergence. This undesirable effect becomes essential only when threads operate on data located in the cache hierarchy. As shown in the right parts of Figs. 1a and 1b, the threads pinned to cores located on the boundaries between sockets feature a higher time of actual computation even if they execute a fewer number of iterations. This time is mainly limited by the inter-socket data exchange overhead. On the contrary, the overhead is negligible or simply imperceptible for larger domains when the time of actual computation is constrained by the main memory speed (see Figs. 1c and 1d, respectively).

Summarizing our observations, the homogeneous variant of DVFS method allows selecting the performance-energy trade-off considering the total execution time that reflects the maximum computation time obtained across a pool of threads. Consequently, threads that process fewer loop iterations waste energy waiting at points of synchronization in the arrival stage. This disadvantage inten-

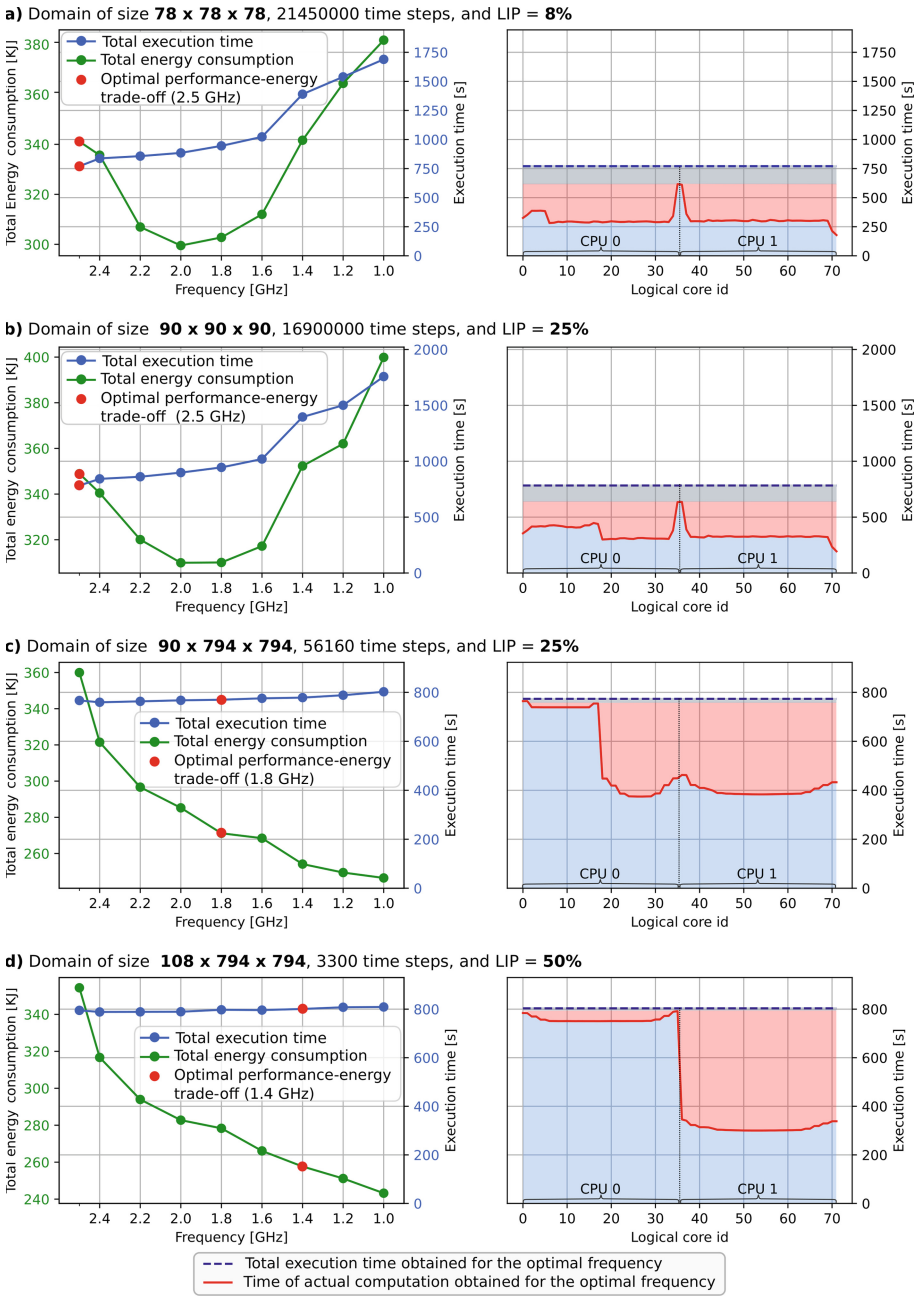

Fig. 1. Execution time and energy consumption for solving the 3D diffusion problem with various domain sizes, using homogeneous DFVS. (Color figure online)

sifies with a constantly increasing number of cores offered by modern shared-memory computing systems.

3 Heterogeneous DVFS: Brute-force Search

The analysis from Sect. 2 motivates us to adjust the frequency and supply voltage to individual 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. We distinguish two heterogeneous DVFS scenarios that prioritize the appropriately selected groups of cores.

The first scenario corresponds to smaller problem sizes (Figs. 1a and 1b), when we can identify cores with the time of actual computation limited by inter-socket data traffic overheads. In this case, we distinguish three groups of cores, including: (i) cores located on the boundaries between sockets; cores with a higher (ii) and a fewer (iii) number of loop iterations. The amounts of cores in the last two groups reflect the LIP parameter and depend on the domain size and number of cores. Based on the time of actual computation for every group, a higher voltage/frequency should be assigned to the first group, then for the second one, and the lowest for the last group. The second scenario assumes larger problem sizes where the main memory constraints result in performance degradation (Figs. 1c and 1d). Here, only two groups are indicated, including cores with a higher (ii) and a fewer (iii) number of loop iterations. Hence, a higher priority and thus a higher voltage/frequency are given to the first group.

To fully explore the heterogeneous DVFS approach, we test almost all frequency configurations for groups, 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. We also reveal no need to scale down the clock speed for cores located on the boundaries between sockets. As expected, setting the maximum frequency/voltage for these cores guarantees to achieve the best results in terms of both performance and energy consumption.

The summary of performed tests is depicted in Figs. 2 and 3. The first one shows the best heterogeneous frequency setups determined for various domain sizes with up to 1–2% performance losses. We select three groups of clock speeds for domains of sizes not exceeding $138 \times 138 \times 138$ (these domains fit in the cache), while two groups are enough for larger domains.

Figure 3 presents the advantages of using heterogeneous voltage/frequency scaling compared to homogeneous scaling, achieved for different domain sizes. The proposed approach permits a maximum energy reduction of about 25% for the domain of size $150 \times 150 \times 150$, 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

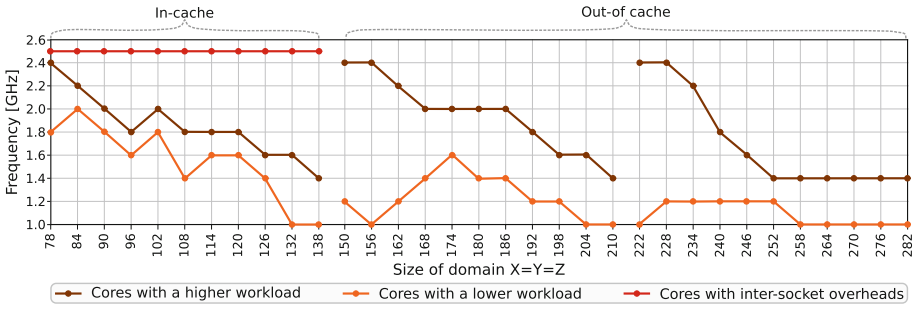

Fig. 2. The best heterogeneous frequency setups selected for various domain sizes.

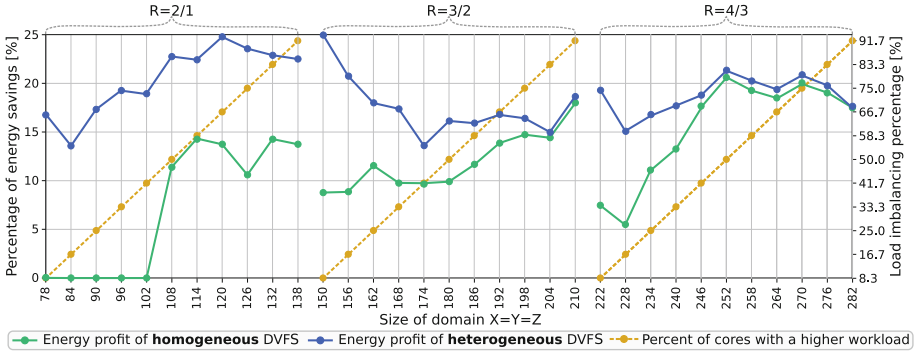

Fig. 3. Comparison of energy-savings for homogeneous and heterogeneous DVFS approaches applied to various domain sizes.

compared to the homogeneous solution is obtained for the size of $96 \times 96 \times 96$. In this case, we observe the advantage of about 20% points over the traditional frequency scaling, which does not bring any energy improvement.

4 Pruning Algorithm for Selecting Heterogeneous DVFS Configurations

The main disadvantage of the brute-force search is its high cost. 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 $\mathbf{F} = \{f_{max}, f_{start}, \ldots, f_{stop}\}$, where $|\mathbf{F}| = N$, be a set of tested frequencies. 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 $f_j \leq f_i$. As a result, the number TC of tested configuration is expressed as $TC = N(N+1)/2$, where the cardinality N of set $\mathbf{F}$ is as follows:

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

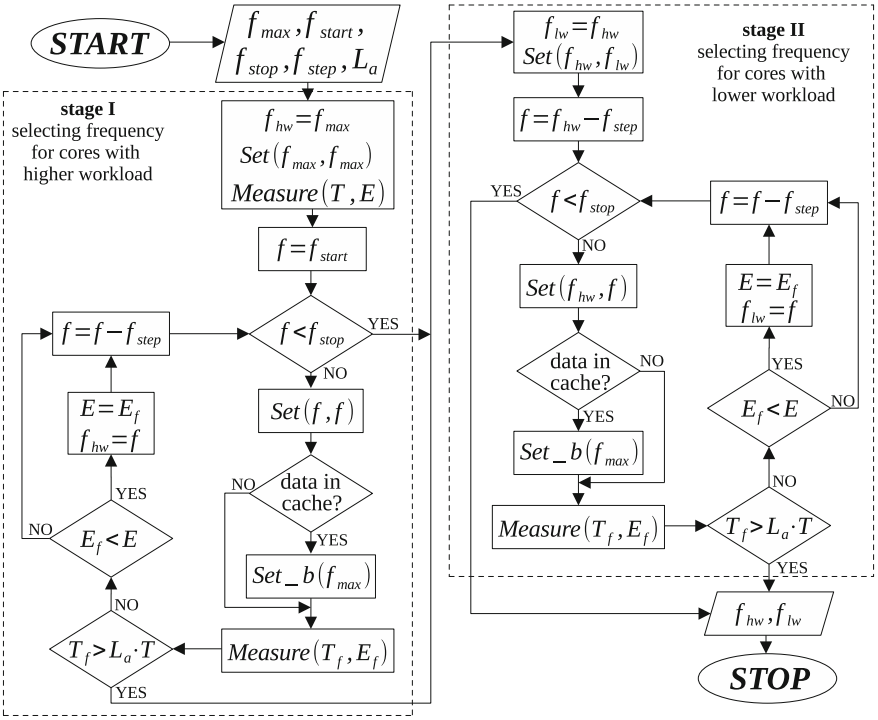

Fig. 4. Block diagram of the pruning algorithm for selecting heterogeneous DVFS configurations

For example, using 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. In each of them, we perform energy consumption tests, 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 propose a pruning algorithm for selecting the best heterogeneous DVFS configuration. This algorithm allows us to significantly reduce the value of TC and speed up finding the best frequency configuration.

The proposed algorithm is shown in Fig. 4. It consists of two stages, the aim of which is to find the best frequencies f_{hw}, f_{lw} for cores with higher and lower workloads, respectively. The input parameters of the algorithm include L_a, which is an acceptable performance loss. For example, $L_a = 1.02$ allows the frequency to be reduced until a 2% increase in run time is achieved. We use the routine $Set(f_1, f_2)$ to examine frequency configurations - it sets clock speeds f_1 and f_2 for groups of cores with higher and lower workloads, respectively. Analogously, the routine $Set\_b(f)$ sets frequency f for cores located on the boundaries between sockets. In Fig. 4, the routine $Measure(T, E)$ is responsible for measuring the execution time T and energy consumption E for the current configuration. To

Table 1. Comparison of brute-force search (B-F) and proposed algorithm (A) for domains of various size $X = Y = Z$

R	LIP [%]	Domain size	f_{hw} [GHz]		f_{lw} [GHz]		TC		Energy reduction [%]		Efficiency r_E [%]
			B-F	A	B-F	A	B-F	A	B-F	A	
2	8	78	2.4	2.4	1.8	1.8	45	8	16.62	16.62	100.00
2	25	90	2.0	2.0	1.8	1.8	45	7	17.14	17.14	100.00
2	50	108	1.8	1.6	1.4	1.2	45	11	22.88	22.40	97.90
3/2	8	150	2.4	2.0	1.2	1.0	45	10	25.04	23.98	95.77
3/2	25	162	2.2	2.2	1.2	1.2	45	10	17.92	17.92	100.00
3/2	50	180	2.0	2.0	1.4	1.4	45	9	16.17	16.17	100.00
4/3	8	222	2.4	2.2	1.0	1.2	45	10	19.52	18.33	93.90
4/3	25	234	2.2	1.8	1.2	1.4	45	9	17.02	15.97	93.83
4/3	50	252	1.4	1.4	1.2	1.2	45	10	21.61	21.61	100.00

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 (see 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.

A set of tests is performed to explore the speed and efficiency of the proposed algorithm. Table 1 presents a comparison of results obtained using the brute-force search and pruning algorithm for different values of R and LIP. The comparison is based on the number TC of configurations tested by the pruning algorithm and efficiency r_E, where r_E is the ratio (in %) of energy reduction achieved by both techniques. In all tested cases, the algorithm's efficiency exceeds 90% and in more than half of them reaches 100%. At the same time, we significantly reduce the cost of selecting a heterogeneous DVFS configuration. While the brute-force search always requires exploring 45 frequency configurations, the pruning algorithm achieves the goal by examining only 7–11 configurations.

5 Related Works

The methods of improving energy efficiency in computing can be divided [6, 10] into hardware-level or system-level. The methods from the first category aim to optimize the energy efficiency of the environment where applications are performed. The solutions from the second category focus mainly on the optimization of applications for performance and energy. These methods use application-level models for predicting the performance and energy consumption of applications. The approach proposed in this work combines methods from both categories since it is based on DVFS and the application-level model.

For data-parallel applications, the combination of methods from both categories is also considered by Reddy and Lastovetsky in work [10]. They propose a method to solve the bi-objective optimization problem of an application for performance and energy on homogeneous clusters of modern multicore CPUs. This method gives a diverse set of Pareto-optimal solutions and can be combined with the DVFS technique to provide a better set of solutions. However, the proposed approach target only homogeneous DVFS policies.

At the same time, heterogeneous voltage frequency scaling is studied in a number of works that can be included into the first category. Unlike this paper, the studied methods employ application-agnostic models [7]. They are principally deployed at the operating system (OS) level [12] or as a runtime system extending the OS [2,19]. Therefore, they require changes to the OS [6]. Typically these methods propose asymmetry-aware or heterogeneity-aware schedulers that exploit the asymmetry [1,9] or heterogeneity [12,19] between sets of cores in a multicore platform to find optimal DVFS configurations. For example, work [1] studies a case when an application running on some cores coexists together with its co-runners (applications running on other cores), while paper [12] considers the heterogeneity of cores in ARM architectures.

6 Conclusions

This paper studies the benefits of heterogeneous DVFS control and performance heterogeneity in improving the energy-performance behavior of data-parallel applications on homogeneous multicore CPU systems. Using the 3D diffusion problem as a use case, we show that using heterogeneous voltage/frequency scaling permits significant energy improvement compared to the homogeneous solution. The advantage of up to 20% points over the homogeneous frequency scaling is achieved on the ccNUMA server with two 18-core Intel Xeon 6240.

The cost and efficiency of the proposed pruning algorithm for selecting heterogeneous DVFS configurations against the brute-force search are compared experimentally. In all tests, the efficiency of the pruning algorithm exceeds 90% and in more than half of them reaches 100%, which indicates that both techniques return the same energy saving. At the same time, we significantly reduce the cost of selecting a heterogeneous DVFS configuration. While the brute-force search always requires exploring 45 frequency configurations, the pruning algorithm achieves the goal by examining only 7–11 configurations.

Acknowledgments. This research was supported by the National Science Centre (Poland) under grant no. UMO-2017/26/D/ST6/00687.

References

1. Abera, S., Balakrishnan, M., Kumar, A.: Performance-energy trade-off in CMPs with Per-Core DVFS. In: Berekovic, M., Buchty, R., Hamann, H., Koch, D., Pionteck, T. (eds.) ARCS 2018. LNCS, vol. 10793, pp. 225–238. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77610-1_17

2. Acun, B., et al.: Fine-grained energy efficiency using per-core DVFS with an adaptive runtime systems. In: 2019 Tenth International Green and Sustainable Computing Conference (IGSC), vol. 1, pp. 1–8 (2019)
3. Calore, E., Gabbana, A., Schifano, S., Tripiccione, R.: Software and DVFS tuning for performance and energy-efficiency on Intel KNL processors. J. Low Power Electron. Appl. **8**(2), 18 (2018)
4. Ciznicki, M., Kurowski, K.: Resource management strategies with energy profiles for stencil computing. In: HiStencil 2015: 2nd International Workshop on High-Performance Stencil Computating, pp. 943–950 (2015)
5. Crank, J.: Mathematics of Diffusion, 2nd edn. Clarendon Press (1975)
6. Fahad, M., Shahid, A., Manumachu, R., Lastovetsky, A.: Energy predictive models of computing: theory, practical implications and experimental analysis on multicore processors. IEEE Access **9**, 63149–63172 (2021)
7. Gupta, M., Bhargava, L., Sreedevi, I.: Dynamic voltage frequency scaling in multicore systems using adaptive regression model. In: Proceedings of 4th International Conference on IoT in Social, Mobile, Analytics and Cloud (I-SMAC), pp. 1201–1206 (2020)
8. Haj-Yahya, J., et al.: Energy Efficient High Performance Processors: Recent Approaches for Designing Green High Performance Computing. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-8554-3
9. Kolpe, T., Zhai, A., Sapatnekar, S.: Enabling improved power management in multicore processors through clustered DVFS. In: 2011 Design, Automation & Test in Europe, pp. 1–6 (2011)
10. Lastovetsky, A., Manumachu, R.: Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy. IEEE Trans. Comput. **67**(2), 160–177 (2018)
11. Mair, J., Huang, Z., Eyers, D., Chen, Y.: Quantifying the energy efficiency challenges of achieving exascale computing. In: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, pp. 943–950 (2015)
12. Papadimitriou, G., et al.: Adaptive voltage/frequency scaling and core allocation for balanced energy and performance on multicore CPUs. In: 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp. 133–146 (2019)
13. Rauber, T., Rünger, G., Stachowski, M.: Performance and energy metrics for multithreaded applications on DVFS processors. Sustain. Comput.: Inform. Syst. **17**, 55–68 (2018)
14. Rojek, K., Ilic, A., Wyrzykowski, R., Sousa, L.: Energy-aware mechanism for stencil-based MPDATA algorithm with constraints. Concurr. Computat. Pract. Exper. **29**(8), e4016 (2017)
15. Rojek, K., Quintana-Ortí, E.S., Wyrzykowski, R.: Modeling power consumption of 3D MPDATA and the CG method on ARM and Intel multicore architectures. J. Supercomput. **73**(10), 4373–4389 (2017). https://doi.org/10.1007/s11227-017-2020-z
16. Sankaralingam, K., Keckler, S., Mark, W., Burger, D.: Universal mechanisms for data-parallel applications. In: Proceedings of 36th International Symposium on Microarchitecture (MICRO-36) (2003)
17. Szustak, L., et al.: Correlation of performance optimizations and energy consumption for stencil-based application on intel xeon scalable processors. IEEE Trans. Parallel Distrib. Syst. **31**(11), 2582–2593 (2020)
18. Technology Guide: Intel Speed Select Technology - Core Power, April 2021

19. Winter, J., Albonesi D.H. Shoemaker, C.: Scalable thread scheduling and global power management for heterogeneous many-core architecture. In: Proceedings of PACT 2010, pp. 29–40 (2010)
20. Zill, D., Wright, W.: Differential Equations with Boundary-Value Problems. 8th edn. Brooks/Cole Cengage Learning (2012)