Architectural Adaptation and Performance-Energy Optimization for CFD Application on AMD EPYC Rome

Lukasz Szustak, Roman Wyrzykowski, *Senior Member, IEEE*, Lukasz Kuczynski, and Tomasz Olas

Abstract—The advantages of the second-generation AMD EPYC Rome processors can be successfully used in the race to Exascale. However, the novel architecture's complexity makes it challenging to adapt demanding scientific codes - like stencil ones - to platforms with Rome CPUs. This article tackles this challenge by exploring the adaptation of the stencil-based CFD (computational fluid dynamics) application called MPDATA to these processors' influential features. We show that the previously proposed parametric adaptation methodology can be profitably applied to extend the performance portability of the memory-bound MPDATA on the AMD EPYC architecture. The extension of the parametric adaptation on the novel architecture requires careful consideration of two relevant aspects that reflect splitting the Rome architecture into multiple dies – features of the cache hierarchy and partitioning cores into work teams. The article also investigates the correlation between the performance optimizations and energy efficiency for a ccNUMA platform powered by top-of-the-line 64-core AMD Rome 7742 CPUs, comparing the results against two servers with Intel Xeon Scalable processors of different generations. Even without appealing to prices, the achieved performance and energy efficiency results are a solid argument confirming the competitiveness of AMD Rome processors against Intel Xeon CPUs in scientific applications.

Index Terms—CFD, MPDATA, AMD EPYC Rome, shared-memory programming, performance portability, energy efficiency

1 Introduction

The choice of processors available for high performance computing (HPC) has been growing for a number of years [1]. There are no fewer than three major types of CPUs available for HPC duties, including x86, Arm, and Power architectures with more than half a dozen reliable suppliers totally, along with two –- soon to be three –- GPU architectures. Simultaneously, the overwhelming majority of HPC systems today are equipped with Intel CPUs and sometimes NVIDIA GPUs. From the year 2020, this is going to start to change.

The most significant source of diversity in the near-term is within the x86 world [1], where AMD's EPYC architecture gives Intel the most substantial competition since the times of AMD Opteron processors. In particular, the second generation of EPYC CPUs, codenamed "Rome," almost certainly will take over a significant part of Intel's share in the server market, including HPC. Rome's promising price-performance ratio was undoubtedly crucial to its selection for a number of supercomputers in the USA and Europe. An example is a new BullSequana XH2000 system that operates in the European Centre for Medium-Range Weather Forecasts (ECMWF) located in Bologna, Italy [2]. It is powered by EPYC Rome 7002 series processors.

• *The authors are with the Department of Computer Science, Czestochowa University of Technology, 42-201 Czestochowa, Poland. E-mail: {lszustak, roman, lkucz, olas}@icis.pcz.pl.*

Manuscript received 16 Nov. 2020; revised 22 Mar. 2021; accepted 23 Apr. 2021. Date of publication 7 May 2021; date of current version 4 June 2021.
(Corresponding author: Lukasz Szustak.)
Recommended for acceptance by J. Zola.
Digital Object Identifier no. 10.1109/TPDS.2021.3078153

Based on 7 nm technology, the second-generation AMD EPYC CPUs of the 7002 series bring large core counts, more efficient memory hierarchy, and the upgraded Infinity Fabric [3], all to enable high performance and better handle the massively parallel workloads. The new architecture of AMD EPYC processors offers up to 64 cores and the base clock speeds from 2.0 to 3.2 GHz. They are delivered for both a single-socket and dual-socket systems, supporting maximally 128 cores.

Emerging computing architectures such as the second-generation AMD EPYC are characterized by a large number of parameters whose diversity makes it difficult to ensure performance portability for real-life applications [4]. An application's sustained performance is affected by such features of computing architectures as number and characteristics of processors, cores, and threads; type of SIMD hardware; parameters of memory hierarchy; the relationship between scalar and SIMD frequency, and many others. Splitting AMD EPYC architecture [5] into multiple dies and NUMA (non-uniform memory access) domains adds another dimension to the space of possible solutions. Consequently, utilizing the full capabilities of AMD EPYC Rome processors in scientific and commercial environments is a significant challenge [6], [7], [8].

Our previous work [4] addressed the issue of performance portable programming of heterogeneous stencil codes for a wide range of shared-memory systems with Intel processors. The application we studied as a use case implements the Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) [9]. This CFD (computational fluid dynamics) application is one of the main parts of the EULAG geophysical model (https://www2.mmm.ucar.edu/eulag) developed for simulating thermo-fluid flows across a wide range of scales and real scenarios [10].

MPDATA executes a set of stencil kernels with heterogeneous patterns and represents a memory-bound application. The adaptation (or customization) methodology proposed in paper [4] allows us to develop the automatic transformation of the MPDATA code, achieving scalable, high performance for all tested ccNUMA platforms with Intel processors of last generations. This methodology consists of a set of parametric optimizations, including (3+1)D decomposition, islands-of-cores strategy, exploiting data parallelism and simultaneous multithreading, data flow synchronization, and vectorization.

The working hypothesis verified in this paper is the possibility of using this methodology to provide the optimization of the MPDATA code for the novel AMD Rome architecture. It is expected that the resulting code can adapt to the parameters of hardware components and their interaction with the proposed parametric optimizations. Furthermore, based on results achieved in our work [11], this study investigates the correlation between performance and energy efficiency for a ccNUMA platform powered by two top-of-the-line AMD Rome 7742 CPUs, each with 64 cores. The hardware-based technique [12] provides the accuracy and reliability of energy/power measurements with the Yokogawa WT310 power meter [13]. Finally, we expect to receive a solid argument confirming or denying the competitiveness of AMD EPYC Rome processors versus Intel Xeon CPUs in scientific applications.

The main contributions of this work are:

1) Based on a memory-bound CFD application called MPDATA, we provide a comprehensive study of two interrelated issues:
   a) systematic adaptation of a real-world scientific application to the AMD EPYC Rome architecture considering its prominent features. The authors are not aware of papers studying this issue;
   b) credible verification of the competitiveness of AMD Rome processors against Intel Xeon CPUs in scientific computing, considering both performance and energy efficiency.
2) It is demonstrated that the extension of the parametric adaptation methodology on the novel architecture requires careful consideration of two relevant aspects that reflect splitting the Rome architecture into multiple dies – features of the cache hierarchy and partitioning cores into work teams. Determining the optimal MPDATA configuration adjusted to essential features of the AMD Rome architecture instead of using a solution obtained for Intel CPUs allows improving performance up to more than 2.5 times and energy consumption up to 2.15 times.
3) It is shown that despite significant differences in the behavior of particular optimizations for various NPS (NUMA per socket) modes of configuring the Rome architecture, the combination of these optimizations leads to practically the same performance and energy consumption for all modes. The achieved performance results are verified by the Roofline-based model, which connects the MPDATA performance with the main memory and cache bandwidth.
4) Based on the performance-energy comparison of the dual-socket platform with 64-core AMD Rome 7742 CPUs versus two servers with Intel Xeon Scalable processors of different generations – Intel Xeon Platinum 8180 and 8280, we show that the Rome-based platform outperforms both Intel-based systems in the performance and energy consumption, allowing us to execute computations 1.23–1.36 times faster with up to 1.12 times fewer energy costs. Therefore, even without appealing to prices, the achieved performance and energy efficiency results are a solid argument confirming the competitiveness of AMD Rome processors against Intel Xeon CPUs in scientific applications.

This paper is organized as follows. Section 2 discusses related works, while Sections 3 and 4 outline respectively the architecture of the second generation of AMD EPYC and the parallelization methodology previously proposed for the MPDATA application. Section 5 describes details of adapting and optimizing the MPDATA parallel code for the EPYC Rome architecture, while Section 6 introduces the methodology of experimental evaluation, including energy/power measurements. Section 7 describes the results of experiments supported by the performance model presented in Section 8. The comparison to Intel CPUs is a topic of Section 9, while Section 10 concludes the paper.

2 Related Works

In July 2017, AMD launched the first generation of its EPYC processors (codenamed "Naples") that disrupted data centers and HPC installations [14]. The AMD second generation of EPYC processors changes the server ecosystem, offering over twice the first generation's peak performance in the same socket, with a similar thermal envelope. Besides upgrading CPU cores with 15 percent more performance at the same clock speeds, this innovative AMD design could double the number of cores.

Utilizing the x86 architecture, the novel AMD EPYC 7002 series allows porting existing applications practically automatically by merely compiling them using the AMD Optimizing C/C++ Compiler (AOCC), GNU compiler, PGI compiler, or even the Intel compiler [15]. An example of applying the GNU compiler is benchmarking the Weather Research and Forecasting (WRF) Model [7] with different generations of AMD EPYC processors. The Intel compiler is also used for assessing the performance of parallel application codes in materials engineering, and chemistry [6]. The focus is on the baseline cluster with 20-core Intel Skylake Gold 6148/2.4GHz processors and two clusters powered by AMD Rome CPUs – 32-core AMD EPYC 7452 and 7502. Concerning a core-to-core comparison, this assessment shows that the Rome 7452 and 7502 perform on a par with the Intel Gold 6148, but a number of applications with heavy memory bandwidth demands perform poorly on the AMD systems. Also, the achieved performance is sensitive to the effective use of the AVX vector instructions. Applications with low utilization of AVX-512 bring a weaker performance of the Skylake-based systems and better performance on the Rome-based clusters. Finally, in a node-to-node comparison, the AMD Rome systems deliver superior results compared to the Skylake Gold 6148 cluster for all applications, with an average improvement factor of 1.49.

An example of using the AOCC compiler [16] powered by AMD Optimizing CPU Libraries (AOCL) [17] is porting GROMACS molecular dynamics simulations [8]. AOCL contain a set of numerical libraries tuned specifically for the AMD EPYC processor family. Among others, this set includes the FFTW library that implements the FFT algorithm. The library compiled from the source is used in porting GROMACS. Another use case of exploring the actual cost/performance ratio of various AMD EPYC Rome processors provides work [3], which benchmarks different Rome processors against Intel Xeon Platinum 8280 on a commercial application – the Facebook's RocksDB Database. Nevertheless, at this moment, the authors are not aware of papers investigating in a more or less systematic way the issues of code adaptation and performance portability of parallel scientific codes for the AMD EPYC Rome architecture, taking into account its prominent features.

Particularly, the last statement relates to stencil-based codes – a class of memory-bound applications that are quite common among scientific applications [18]. The stencil computations have traditionally been optimized by many authors over the years, especially considering various hardware platforms such as multicore CPUs [19], short-vector SIMD architectures [20], Intel Xeon Phi [21], GPU [22] and FPGA [23] accelerators. One of the main directions of improving the efficiency of stencil computations is focused around different methods for the domain decomposition [24], including overlapping neighbor domains [25]. These methods include space and temporal blocking [26], diamond [27] and multi-dimensional [28] tiling. These methods are oriented towards exploiting the data locality and providing workloads of computing resources balanced as much as possible. These works' limitation is their focus on the homogeneous stencil computations, with a single pattern repeated for consecutive time steps. Moreover, typically the code transformations take place between successive steps. In contrast, a combination of parametric optimization techniques developed in our previous work [4] is applied within every MPDATA time step and is dedicated to a set of heterogeneous stencils with different patterns.

The closest approaches were proposed in papers [25], [29], and particularly in paper [30] devoted to generating and optimizing stencil programs automatically. Similarly to our approach, these papers consider the code transformation using the overlapped tiling technique. It enables leveraging the synchronization and enhancing the data locality at the cost of redundant computations. But again these works address only the homogeneous stencil computations with a single pattern repeated in each step.

Apart from achieving high performance, the energy efficiency of HPC parallel workloads become a focus of attention in recent times [31]. Besides the tendency of energy reduction by re-engineering the hardware, another trend is observed with great potential for energy savings realized by transforming and completely rethinking the algorithms [32] and software [33] dedicated for HPC systems. This work belongs mainly to the second trend, and partly follows our previous paper [11] that studied the impact of performance optimizations on the energy efficiency of MPDATA for the first generation of Intel Xeon Scalable processors.

TABLE 1
Specification of HPE ProLiant DL385 Gen10 Server Equipped With Two AMD 7742 CPUs (https://www.amd.com)

Base frequency [GHz]	2.25
Max. boost single-core freq. [GHz]	3.4
Sockets Cores (Logical cores) Core-Complexes (CCXs)	2 2×64 (2×128) $2 \times 8 \times 2$
Core Complex Dies (CCDs)	2×8
Type of SIMD (SIMD width)	AVX2 (256-bit)
L1 per core [KB]	64
L2 per core [KB]	512
L3 per CCX [MB]	16
L3 per CCD [MB] Total size of L3 [MB]	32 $512 = 2 \times 8 \times 32$
Main memory [GB]/DDR channels	$2 \times 8 \times 16$ / 2×8
Type of memory	DDR4-3200
Memory bandwidth [GB/s]	2×204.8
Peak performance for SIMD [Gflop/s]	2304

3 Architecture of the Second Generation of AMD EPYC Processors

This work explores the dual-socket HPE Server ProLiant DL385 Gen10 (https://www.hpe.com) consisting of two 64-core AMD 7742 CPUs with 256 GB of DDR4 memory operating at 3200 MHz. The basic software package includes the CentOS Linux operating system and AMD Optimizing C/C++ Compiler v2.1. Table 1 summarizes this platform.

The AMD EPYC 7742 CPU represents the second generation of EPYC processors [34]. The design of this CPU consists of a single central I/O hub (or I/O Die) [5] through which all CPU components communicate. The CPU uses a collection of 8-core chiplets, called Core Complex Dies (CCDs), connected to the I/O Die through dedicated high-speed Infinity Fabric links. Through this die, a given CCD can communicate with other CCDs and the main memory, as well as with external devices connected by the PCIe bus. As a result, the EPYC 7742 CPU can provide one NUMA domain for a single processor, which is equivalent to the NUMA layout offered by current Intel Xeon CPUs [35]. This mode is known as NPS1 [5].

This top-of-the-line Rome CPU contains 8 CCDs. Every CCD consists of two Core-Complexes (CCXs); each of them embraces four cores and 16 MB of L3 cache. As a result, each CCD provides 32 MB of L3 cache. A single core contains the L2 inclusive cache of 512 KB size and the L1-D cache of 32KB size. The four cores of a given CCX share 16MB of its L3 cache segment, rather than requesting access to any L3 cache from other CCXs. Thus, this design is an example of the non-uniform cache architecture (NUCA) [36]. The total capacity of the L3 cache depends on the number of CCDs, and can maximally reach 256 MB for a single 64-core CPU. Each processor also includes the 8-channel memory controller providing DDR4 memory speeds up to 3200MHz, where each memory channel supports up to 2 DIMMs.

Like other EPYC 7002 series processors, the EPYC 7742 CPU uses a NUMA architecture with separate quadrants, each with two CCDs, two memory channels, and 32 I/O lanes. The four logical quadrants allow the processor to be partitioned into different NUMA domains [5]. These domains are designated as NUMA per socket (NPS). In the NPS1 mode, all cores on the processor, all memory and PCIe devices connected to it are in one MUMA domain. It means that memory is interleaved across the eight memory channels. In the NPS2 and NPS4 modes, the processor is partitioned into two and four NUMA domains, respectively. For example, each logical quadrant of the CPU is a NUMA domain in the NPS4 mode, with memory being interleaved across the two memory channels in each quadrant. In general, NPS2 and NPS4 can provide slightly better bandwidth than NPS1. Finally, each L3 cache slice corresponding to one CCX is exposed as a NUMA domain. On the dual-processor server used in this work, 32 NUMA domains could be exposed at most. Using BIOS settings, the server could be configured as NPS1, NPS2, or NPS4, with an additional option to configure L3 cache slices as NUMA domains.

The CPU design permits simultaneous multithreading (SMT), which results in 128 logical cores for the 64-core CPU. The architecture of Rome processors offers full AVX2 support. It enables single-cycle AVX2 calculations, rather than splitting 256-bit instructions into two separate 128-bit operations, as is the case of the first generations of AMD EPYC. This design also includes two FMA (Fused Multiple-Add) units per core. As a result, each core can execute up to 16 double precision floating-point operations per cycle. There is no restriction for CPU frequency when using AVX2 instructions. The clock speed depends on temperature and voltage requirements regardless of the instructions used.

The CPUs used in the study are clocked at the base frequency of 2.25 GHz. The maximum boost for a single core is 3.4 GHz. The theoretical peak performance of the test platform is 2304 Gflop/s for SIMD. These values refer to double-precision non-FMA and the base frequency. For FMA instructions, the peak performance is twice higher.

4 Parallelization of MPDATA Application

4.1 Overview of MPDATA

The MPDATA algorithm belongs to CFD methods for numerical modeling of advection transport phenomena. MPDATA represents a general approach to modeling complex geophysical flows from micro to planetary scales [9]. It corresponds to the second-order accurate nonoscillatory iterative algorithms and is defined using a finite-difference scheme over structured rectilinear grids. As a representative of forward-in-time algorithms, MPDATA solves the advection of a non-diffusive quantity Ψ in a flow field [37]

$$\frac{\partial \Psi}{\partial t} + \mathrm{div}(V\Psi) = 0, \tag{1}$$

where V is the velocity vector. In this work, we focus on modeling 3D advection problems defined on structured rectilinear grids. This means that MPDATA is defined in a 3D domain of sizes $n \times m \times l$ according to $i-$, $j-$, and $k-$dimensions, respectively.

In general, MPDATA is intended to run long simulations that engage even many thousands of time steps. Each time step takes five 3D matrices (arrays) as an input and returns a single 3D matrix (array) reused in the next step. Each MPDATA step performs [4] a series of 17 kernels that depend on each other (the outcomes of a given kernel typically are inputs for the subsequent ones). Every MPDATA kernel represents a 3D stencil code that updates all elements of its output array, according to a particular pattern. The detailed description of MPDATA is presented in [9], [21].

4.2 Parallelization Methodology for MPDATA Code on Shared Memory Systems

In the basic version of the parallel MPDATA code [4], [21], subsequent kernels are executed sequentially, one by one, with each kernel processed in parallel using OpenMP. The data parallelism and vectorization are employed to distribute kernels across computing resources, including logical cores and vector units. Particularly, `#pragma omp for` directive across outer-most loop ($i-$dimension) is applied to split loop iterations among logical cores, and then `#pragma omp for simd` directive allows us to incorporate vectorization along inner-most loop ($k-$dimension).

Since the code is not optimized for cache reusing, the performance of the basic MPDATA parallelization is strongly limited by the main memory bandwidth. As a result, the relatively low operational intensity of each MPDATA kernel [4] is not high enough to efficiently utilize the resources of modern processors. In our works [4], [11], [21], [35], [38], a set of optimizations was developed to exploit resources of multicore ccNUMA/SMP systems more efficiently. The resulting parallelization methodology consists of the following parametric optimization steps:

- *(3+1)D decomposition of MPDATA* [21] – this step explores spatial blocking across the different kernels, employing overlapped tiling with redundant computations, while all kernels are grouped into five packages using loop fusion. Besides increasing the computational intensity, this approach reduces the main memory traffic and provides efficient utilization of L3 and L2 levels of the cache hierarchy.
- *Partitioning cores into independent work teams* [35] – this step relieves the overhead of inter- and intra-CPU data traffic within the cache hierarchy of the ccNUMA system by setting the set of groups of physical cores mapped on MPDATA work teams. As a result, two scenarios for executing MPDATA kernels are proposed: the first one performs fewer computations but requires more data traffic, while the second scenario replaces the implicit data traffic by replicating some of the calculations.
- *Data-flow strategy of synchronization* [38] – the primary purpose is to synchronize only interdependent threads following the data dependencies between the MPDATA kernels, instead of using the barrier approach that typically synchronizes all threads. This strategy reduces the cost of synchronization.
- *Vectorization of MPDATA kernel* [4] – the 7-step procedure is developed for the MPDATA code transformation that allows the compiler to automatically perform

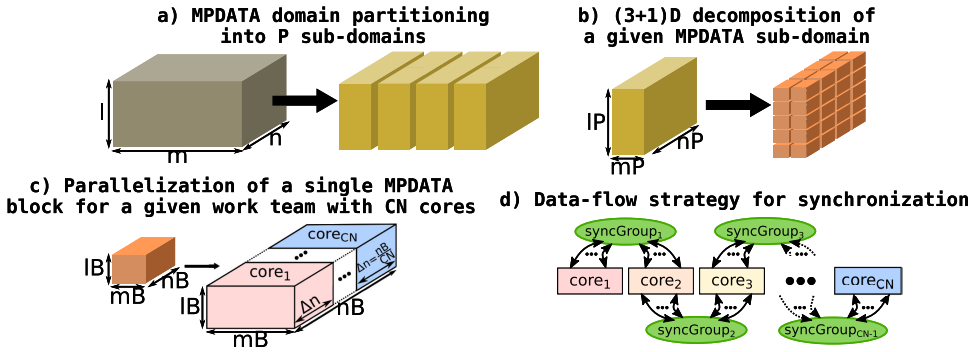

Fig. 1. MPDATA decomposition [4]: a) domain partitioning into sub-domains, b) sub-domain decomposition into blocks of size adjusted to cache capacity, c) parallel execution of kernels within a single block by a given work team, and d) synchronization.

the vectorization, and ensures the performance portability of vectorizing MPDATA computations.

Fig. 1 illustrates the proposed distribution of the MPDATA workload across computing resources. First, the MPDATA domain is split into P sub-domains that are processed in parallel by P hardware teams of cores (work teams) available in a given ccNUMA platform (Fig. 1a). Every work team processes a given sub-domain following the (3+1)D decomposition (Fig. 1b). Each sub-domain is partitioned into blocks of the size that enables efficient utilization of L3 and L2 caches. The successive blocks are processed sequentially, one by one. Each block exploits data parallelism across $i-$ and $j-$dimensions (Fig. 1c) to distribute workload across CN cores of a given work team. As a result, each MPDATA block is partitioned into a set of CN sub-blocks. Finally, the vectorization is performed along $k-$dimension for appropriate chunks of data arrays corresponding to the sub-block.

The computations performed by every core provide executing the MPDATA kernels grouped into 5 packages P1-P5. The package P1 consists of the first four kernels and requires load input data from the main memory only. The rest of the packages operate on (i) the data stored in the cache hierarchy by P1 and (ii) intermediate results of prior packages that all should be located in the cache. More precisely, the size of sub-blocks has to allow keeping data in the L2 cache during the execution of a given package, while the block size should enable keeping required data in the L3 cache for executing all packages.

To meet the performance portability challenge, we developed [4] the parametric transformation/adaptation of the MPDATA code to ccNUMA shared memory systems. As a result, the customizable MPDATA code follows various hardware architectural issues such as memory hierarchy, threading, vectorization, and their interaction with the MPDATA code. This adaptation was successfully applied to achieve sustained high performance for a wide range of Intel-based systems. Among these systems were 2-socket servers with Intel Xeon CPUs based on Skylake SP, Broadwell, and Haswell architectures. For example, for a platform built with 28-core Intel Platinum 8180 CPUs, the proposed adaptation accelerates the MPDATA application more than 10 times [11] compared to the basic version of code.

The architecture of Intel Xeon CPUs provides a single NUMA domain per every socket. Each domain contains a group of CPU cores connected to a single last-level cache (LLC) domain. There are typically two NUMA domains in a dual-socket Intel-based platform with two groups of cores and two LLC domains. As a result [4], the optimal number of MPDATA work teams is basically equal to the number of processors. At the same time, the presence of two memory controllers per CPU affects the main memory differently for different cores. Consequently, the final number of work teams is twice the number of CPUs in the server.

For modern Intel Xeon CPUs, like Intel Platinum 8180 and 8280 based respectively on Skylake SP and Cascade Lake architectures, the size of the L2 cache is relatively large (1MB per core). Simultaneously, these CPUs feature relatively small L3 caches - only 38,5MB for 28-core Intel CPUs (1.37 MB of L3 per core). Moreover, since Skylake SP and Cascade Lake processors feature the non-inclusive L3 cache [39], the effective size TCS^P of L3 cache is further limited to the aggregate size of L2 caches for all cores (TCS^P = 28MB). The reason is that instead of copying data both to the L2 and L3 caches as in the case of previous generations of CPUs, now data are loaded directly into the L2 cache of a given core. Hence, the block size optimization based on the size of the effective size (TCS^p) of the L3 cache also allows keeping all required data in the L2 cache assigned to a core [4]. In this cache, the core keeps data required to processes a sequence of kernels within a package.

Finally, details of vectorizing computation within cores are determined. For Intel 8180 and 8280 CPUs, the AVX-512 SIMD extension is applied, allowing to increase the performance on the 2-socket 8180 CPUs about 2.8 times [11].

5 Adaptation of MPDATA Parallel Code to EPYC Rome Architecture in Comparison to Intel-Based Systems

The MPDATA code's adaptation to new features of AMD Rome architecture has to ensure high performance and scalability of the resulting code implemented with OpenMP. The differences between Xeon and Rome processors radically affect the adaptation methodology for the MPDATA code, including selecting the optimal number of work teams and determining the best size of MPDATA blocks.

5.1 Adapting MPDATA Code to Core Complex Dies

In the NPS1 mode, the design of EPYC Rome CPUs provides a NUMA model comparable to Xeon processors, with a single socket and single NUMA domain. Thus similarly to dual-

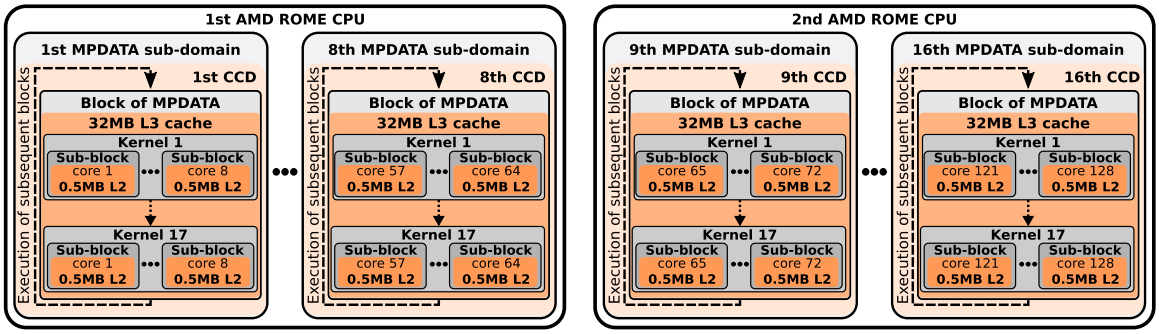

Fig. 2. Adaptation of MPDATA code for to dual-socket systems with 64-core AMD's Rome EPYC CPUs.

socket servers based on Intel Xeon CPUs, we can basically use two MPDATA work teams for two AMD Rome CPUs to alleviate the inter-CPU communication overhead. However, since Rome CPUs in general contain more NUMA domains corresponding both to the NPS2/NPS4 modes and L3 caches, we can use more MPDATA work teams to reduce the intra-CPU data traffic.

Every 64-core Rome CPU contains eight 8-core complex dies, each corresponding to an L3 cache section of 32MB, where the section contains two segments of 16MB each. Every die uses the I/O Die to connect other dies through AMD Infinity Fabric. Accessing data that reside at a single die is more efficient than moving data between different dies [3]. That is why we expect that the optimal number of MPDATA teams refers to the total number of CCDs. The test server is expected to achieve the best performance for 16 MPDATA work teams mapped on 2×8 CCDs. As a result, every MPDATA work team will exploit 8 cores with 32MB of L3 cache. The idea of adaptation of the MPDATA code to a platform with two Rome CPUs is illustrated in Fig. 2.

Remark 1. The maximum number of NUMA domains is equal to the amount of CCXs. At the same time, we select the optimal number of MPDATA teams following the amount of core complex dies. The reason is to minimize the data movements between dies performed through the interconnect. This choice is justified by performance experiments described in Section 7.

The overall performance depends on how the MPDATA domain is partitioned, and then the sub-domains are mapped on CCDs (work teams). The data layouts for all MPDATA arrays allow transfers of contiguous memory areas along the first dimension only [35]. It is the reason for avoiding 2D and 3D variants of partitioning.

The MPDATA domain of size $m \times n \times l$ is evenly decomposed into P sub-domains of size $\frac{m}{P} \times n \times l$, where P corresponds to the number of CCDs (Fig. 1a). Since data transfers take place only between borders of neighbor MPDATA sub-domains [38], the adjacent sub-domains are mapped onto CCDs that are closely connected with each other. This strategy reduces the communication paths between work teams [4] by selecting the appropriate policy for the OpenMP thread affinity interface.

5.2 Adjusting MPDATA Code to Cache Hierarchy

As shown in Fig. 1b, a single MPDATA sub-domain is partitioned into blocks following the (3+1)D decomposition. Thus, the critical point is selecting the size of MPDATA blocks to keep all necessary data in the cache hierarchy corresponding to a given work team.

In contrast to Intel processors, the second generation of EPYC CPUs provides a large L3 cache. The top-of-the-line AMD Rome 7742 CPU contains 64 cores with 256 MB of L3 (4 MB of L3 per core), while its Intel Xeon counterpart – Intel Xeon Platinum 8280 – has 28 cores with 38.5 MB of L3 (1.37 MB of L3 per core). The large capacity of the L3 cache in Rome processors supports larger MPDATA blocks compared with Intel CPUs.

However, since every block is partitioned into a set of sub-blocks processed by cores of a given work team, the 512KB volume of the L2 cache per Rome core limits the size of a sub-block to keep in the L2 cache all data required for every MPDATA package. For this reason, the final size of MPDATA blocks has to be reduced to allow residing data in L2 for a sequence of MPDATA kernels processed within each package (Fig. 2). As a result, the relatively small size of L2 (512KB) strongly bounds the size of blocks and depletes the advantage of large L3.

The MPDATA package P5 features one of the highest demands for the L2 cache. So the optimal size of sub-blocks can be successfully determined based on the constraints of this package. Table 2 includes the examples of the L2 cache requirements for all MPDATA packages, in correlation to the L3 cache demand for MPDATA blocks of various size. As follows from the last column of this table, the smallest size of $1 \times 256 \times 128$ gives the best performance. This conclusion is supported by Fig. 3, which illustrates the impact of increasing the block size on the execution time of the version **D** of the MPDATA code.

The three parameters mB, nB, and lB define the size of MPDATA blocks, with sub-blocks of size $mB \times \frac{nB}{CN} \times lB$. Considering constraints of the (3+1)D decomposition [4], the best configuration is fixed as $nB = n$, $lB = l$ assuming

TABLE 2
Requirements for Volume of L2 Cache for the MPDATA Packages versus Size of the LLC Domain Assigned to a Single Work Team, for Various Block Size ($l = 128$)

Size of block	L2 (0.5MB)					L3 (32MB)	Perf. loss
	P1	P2	P3	P4	P5		
1x256x128	0.33	0.36	0.36	0.47	0.47	16.29	–
2x256x128	0.54	0.58	0.62	0.76	0.76	20.00	1.06
4x256x128	0.98	1.02	1.12	1.34	1.34	27.41	1.15
8x256x128	1.85	1.89	2.14	2.50	2.50	42.23	1.66

The last column shows the performance loss against the smallest block size.

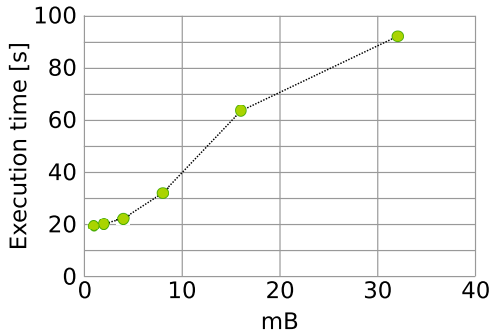

Fig. 3. Execution time for version D of MPDATA with domain of size $1024 \times 512 \times 128$ and different value mB for blocks of size $mB \times 256 \times 128$.

that the L2 and L3 cache capacity is large enough to keep all input and output data for a given size mB. Otherwise, we reduce the block size iteratively by $nB = \frac{n}{q}$, with $q = 2, 3, 4, \ldots,$, until the block size is small enough to satisfy the cache capacity restrictions. Algorithm 1 summarizes the procedure for determining the optimal sizes of the MPDATA block and sub-blocks. Here we assume that $l \in [64, 128]$, as these values are typical for numerical simulation in weather prediction applications.

Algorithm 1. Determining the Optimal Size $nB \times mB \times lB$ of MPDATA Block

$mB = 1$
$nB = n$
$lB = l$
$\Delta nB = \frac{nB}{CN}$
$q = 1$

Phase 1 - determining values of ΔnB and nB

while $P5\_in\_L2(mB, \Delta nB, lB) \leq L2S$ **and**
$allPackages\_in\_L3(mB, nB, lB) \leq L3S^T$ **do**
$q = q + 1$
$nB = \frac{n}{q}$
$\Delta nB = \frac{nB}{CN}$
end while
$nB = \frac{nP}{q-1}$
$\Delta nB = \frac{nB}{CN}$

Phase 2 - determining value of mB

while $P5\_in\_L2(mB, \Delta nB, lB) \leq L2S$ **and**
$allPackages\_in\_L3(mB, nB, lB) \leq L3S^T$ **do**
$mB = mB + 1$
end while
$mB = mB - 1$

6 Methodology of Evaluation

The four versions of the MPDATA code are considered in our benchmark: *(A)* basic, non-optimized implementation; *(B)* code with the (3+1)D decomposition of MPDATA; *(C)* version B with partitioning cores into work teams; *(D)* version C with the data-flow synchronization. Additionally, every MPDATA version is run with enabled and disabled vectorization to evaluate the influence of vector units for performance and energy. The AMD Optimizing C/C++ Compiler (AOCC) v2.3 is used with the optimization flags `-O3", -mprefer-vector-width=256"`, and `-march=-znver2"` for enabling high-level optimizations towards AMD EPYC 7002-series.

TABLE 3
Execution Time [s] for Versions **A** and **D** of MPDATA for Different Numbers of NUMA Domains Achieved for the Domain of Size $2048 \times 1024 \times 128$ and 10000 Time Steps (With Enabled Vectorization)

BIOS setting	NPS1	NPS2	NPS4	L3 as NUMA
# NUMA domains	2x1	2x2	**2x4**	2x16
$1024 \times 512 \times 128$				
Version A	2042.4	1982.9	**1903.7**	1916.1
Version D	194.4	193.3	**193.2**	194.5
$2048 \times 1024 \times 128$				
Version A	8245.9	7984.8	**7631.7**	7781.2
Version D	740.5	739.2	**737.2**	741.5

The hardware-based technique [12] is used to measure the energy and power consumed by the tested platform. We employ the Yokogawa WT310 digital power meter [13] to obtain maximally accurate and reliable energy/power measurements (100k samples per second with measurement accuracy kept at the level of 0.1 percent). The power meter passes the power to the server under the load and measures the energy and power in real-time. Yokogawa WT310 is equipped with the serial USB interface and YokoTool command-line tool [40] that allow collecting the total power and energy without any noticeable influence on measurements.

There are three types of measurements: execution time (seconds), total energy consumption (Joules), and average power (Watts). Every type of measurement is carried out independently: one run for measuring the execution time, and the other run for collecting the power and energy consumption measurements. Each type of measurement is repeated at least 15 times, and the median values are selected to obtain statistically sound results. The server is located in an air-conditioned server room, providing stable temperature. As a result, the relative standard deviation (RSD) does not exceed 0.5 percent for all measurements.

7 Results of Experiments

7.1 Effect of Using Various Numbers of NUMA Domains

This subsection contains results of investigating the impact of using various numbers of NUMA domains on the MPDATA performance. Results for two versions of MPDATA are shown in Table 3: the basic version **A** and most optimized version **D** corresponding to selecting the number of work teams equal to 16 as the total number of CCDs in the server (see Section 7.2). All the possible BIOS settings are being examined: NPS1, NPS2, NPS4, and with L3 segments exposed as NUMA domains. Following [5], we turn off the NUMA balancing option for operation system tuning to avoid undesired performance effects.

The analysis of Table 3 shows that the NPS4 mode gives the shortest execution times for both versions. For the basic code,

TABLE 4
Performance and Energy/Power Results Achieved for Different MPDATA Versions, 10000 Time Steps, and Two Sizes of MPDATA Domain

Size	Version of MPDATA	Time [s]	Speedup	Sust. perf. [Gflop/s]	% of peak perf.	Total energy [KJ]	Total energy gain	Avg. power [W]
1024×512×128	**A**	1903.76	–	70.20	3.05	1154.84	–	607
	B	1886.96	1.01	81.50	3.54	984.06	1.17	522
	C with 2-teams (C-2T)	1222.70	1.56	126.00	5.47	632.81	1.82	518
	C with 4-teams (C-4T)	583.99	3.26	264.80	11.49	308.26	3.75	528
	C with 8-teams (C-8T)	249.95	7.62	623.40	27.06	155.44	7.43	622
	C with 16-teams (C-16T)	243.19	7.83	658.40	28.58	151.49	7.62	623
	C with 32-teams (C-32T)	254.06	7.49	643.70	27.94	160.57	7.19	632
	D	193.24	9.85	828.60	35.96	125.86	9.18	651
2048×1024×128	**A**	7621.48	–	70.10	3.04	4650.48	–	610
	B	7062.59	1.08	87.10	3.78	3833.64	1.21	543
	C with 2-teams (C-2T)	4265.76	1.79	144.30	6.26	2241.11	2.08	525
	C with 4-teams (C-4T)	1723.89	4.42	358.40	15.56	977.83	4.76	567
	C with 8-teams (C-8T)	968.79	7.87	642.90	27.90	613.15	7.58	633
	C with 16-teams (C-16T)	917.88	8.30	683.70	29.67	602.96	7.71	657
	C with 32-teams (C-32T)	919.68	8.29	694.90	30.16	613.12	7.58	669
	D	734.25	10.38	854.60	37.09	493.06	9.43	672

The server is configured with NPS4 mode exposing 2x4 NUMA domains.

NPS4 allows reducing the execution time by about 7.5 percent. This advantage is practically negligible for the version **D** (less than 1 percent). The analogous conclusion can be made for other problem sizes. Thus, the NPS4 mode allows achieving the highest performance. At the same time, it is of considerable interest to study the behavior of the different versions of code for other modes, especially NPS1.

7.2 Correlation of Performance Optimizations and Energy Consumption for Different MPDATA Versions

The benchmarks outlined in this subsection allow the performance and energy/power comparison for four MPDATA code versions. The primary assumption of the tests is to reach the highest possible performance by utilizing all cores and vector units, as well as setting the maximum CPU clock frequency. Table 4 shows results obtained in the NPS4 mode for the double precision format, 10000 time steps, with two different domains: $2048 \times 1024 \times 128$ and $1024 \times 512 \times 128$. Apart from the execution time, total energy consumption, and average power, this table presents the speedup against the basic code **A**, the sustained performance (Gflop/s), and the percentage of the peak performance, as well as the total energy gain against the basic code.

We also monitor the power required by the platform during the execution of each MPDATA version. Fig. 4 shows an example of power traces for all MPDATA versions, corresponding to the domain of size $1024 \times 512 \times 128$, 1000 time steps, and enabled AVX2 (NPS4 mode). As shown in Fig. 4, the power consumed by the server is mostly kept at the same level while executing a given MPDATA version. It is the result of a constant computational intensity of MPDATA time steps. At the same time, different MPDATA versions correspond to various levels of power.

Fig. 5 illustrates performance and energy gains for various MPDATA optimizations with two problem sizes. We show the advantages of a given MPDATA version over the previous one. For example, the version **C** with four work teams (C-4T) reduces the execution time 4.10 times against the version **B** for the domain of size $2048 \times 1024 \times 128$.

The transformation of the code from the version **A** into **B** employs the (3+1)D decomposition for alleviating the memory and communication constraints by increasing the data locality and cache reusing [4]. For NPS1, this optimization allows reducing the execution time in the range from 1.92 times (for the smaller domain) to 3.01 times (for the larger one), in comparison with the version **A**. The energy consumption decreases slightly more since the version **B** not only reduces the execution time but also saves power by about 60W. This difference is because the basic version generates a considerably higher main memory traffic.

The (3+1)D decomposition moves the bulk of data traffic from the main memory to the cache, improving both cache reusing and the data locality. At the same time, this optimization results in intensive intra- and inter-cache communications within the cache hierarchy involving different NUMA domains [35]. As a result, increasing the number of NUMA domains almost annihilates the advantages of using the version **B** in the NPS4 mode since now the performance and energy gains become equal only to 1.01–1.08 and 1.17–1.21, respectively.

A remedy for this issue is partitioning cores into independent work teams. This optimization included in the version **C** is responsible for reducing the overhead of inter- and intra-CPU data traffic within the cache hierarchy of ccNUMA systems. In the first stage of testing this version, we reveal the effect of using two work teams for two Rome CPUs to

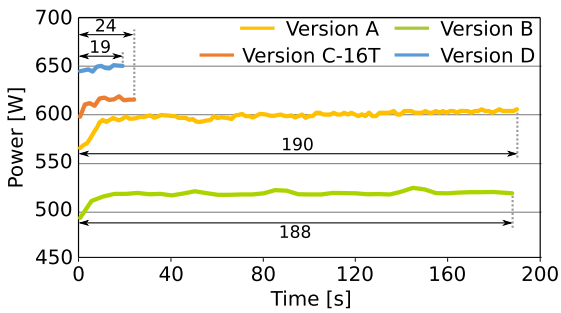

Fig. 4. Power traces for MPDATA versions with domain of size $1024 \times 512 \times 128$ in NPS4 mode (every sample is averaged over 1s interval).

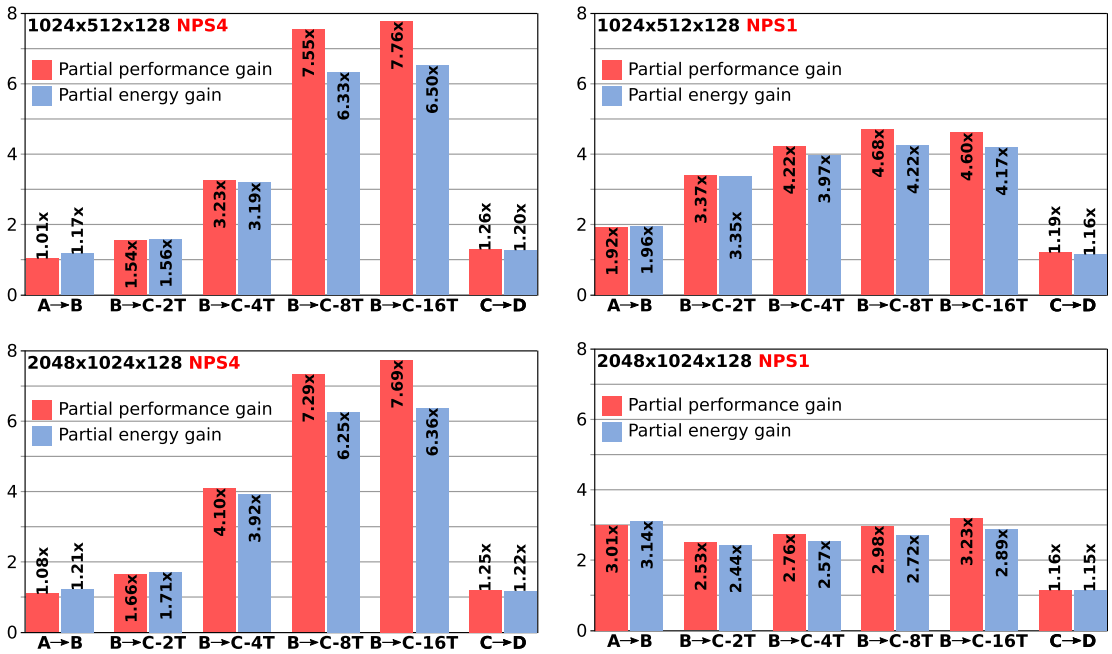

Fig. 5. Impact of optimization steps on performance and energy gains for two sizes of MPDATA domain using NPS4 and NPS1 modes.

alleviate the inter-CPU data traffic. In the NPS4 mode, for the larger MPDATA domain, the version **C** with two work teams (depicted as *C-2T*) performs computation 1.66 times faster than the pure (3+1)D decomposition (version **B**). The advantage is a bit smaller for the smaller domain (1.54 times). The energy consumption is reduced as well, with the gain at a similar level as performance.

Then we increase the number of work teams inside processors. Results presented in Table 4 show the advantage of increasing the number of work teams – up to 16 work teams since there are neither performance nor energy gains when using more work teams. Introducing work teams inside CPUs reduces the intra-CPU data traffic and allows using available cores more efficiently. In the NPS4 mode, the configuration with 16 work teams gives the highest acceleration of computations (up to 7.76 times) and reduction of the energy consumption (up to 6.50 times), as compared to the version **B**. At the same time, the power required by the server increases with the number of work teams (Table 4). More intensive utilization of cores results in higher power requirements for the server but allows us to execute the MPDATA kernels faster. Accordingly, the energy consumption is reduced with smaller profits than those for the execution time. In the NPS1 mode, the performance and energy gains obtained for the version **C** are considerably lower. In fact, using 16 work teams permits speeds up computations 4.60 times for the smaller MPDATA domain and 3.23 times for the larger domain.

The data flow synchronization implemented in the version **D** increases the performance up to 26 percent and saves up to 22 percent of energy compared to the version *C-16T* configured with the optimal number of work teams. The gains for the NPS4 mode are more significant than for NPS1.

Thus, the combination of all optimization steps radically improves performance and energy consumption. In the NPS4 mode, these steps achieve the performance gain from 9.85 to 10.38 times and energy savings from 9.18 to 9.43 times against the basic code. Another important conclusion is that despite the differences in the behavior of the optimization steps for different modes, their combination leads to practically the same performance and energy consumption results for all modes.

7.3 Impact of SIMD Processing and CPU Frequency Scaling on Performance and Energy Consumption

All MPDATA versions allow us to automatically implement the vectorization in a portable way across different architectures and various compilers. The optimization flags `-O3` and `-march=znver2` offered by the AOCC compiler support the full use of vector units for Rome CPUs. The OpenMP SIMD directives together with the special hints and keywords are used to make the vectorization more efficient. In our tests, the clock CPU frequency is equal to 2.38 GHz and 3.2 GHz for all MPDATA versions with enabled and disabled vectorization, respectively. For all versions with enabled vectorization, the average power is only slightly higher (up to 20W) compared to the scalar code.

We examine the same three performance-energy metrics as before. Table 5 presents results measured for the version **D** with different problem sizes. This version offers the highest benefits of vectorization for all sizes. The highest efficiency of vectorization is achieved for the domain of size $1024 \times 512 \times 128$ when when the performance and energy gains are equal to respectively 1.71 and 1.66 times. In contrast, enabling the AVX extension in the basic version **A** results neither in performance nor energy gains. This effect happens because the performance of the basic code is strongly limited by the main memory bandwidth [4].

Our experiments are completed by exploring CPU frequency scaling as a method to optimize the energy efficiency of MPDATA. We use the `ACPI CPUfreq` framework [41] to set three levels of clock frequency, including 1.5 GHz,

TABLE 5
Performance-Energy Comparison of Scalar (Non-AVX) and Vectorized (AVX) Codes for Version **D** of MPDATA Obtained for Double Precision Format and 10000 Time Steps, With Different Domain Sizes

Size: $1024 \times 512 \times 128$		Non-AVX	AVX	Gain
Metrics	Time [s]	329.7	193.2	1.71
	Total Energy [KJ]	209	126	1.66
	Avg. power [W]	633	651	0.97
Size: $2048 \times 1024 \times 128$		Non-AVX	AVX	Gain
Metrics	Time [s]	1222.0	739.5	1.65
	Total Energy [KJ]	794	495	1.60
	Avg. power [W]	650	669	0.97

2.0 GHz, and 3.2 GHz for the basic version of code, and 1.5 GHz, 2.0 GHz, and 2.38 GHz for the version **D**.

The version **A** features practically constant execution time for all frequencies. While frequency scaling does not affect performance, the execution of this version with the lowest clock frequency permits a maximum reduction in the energy. The average power is reduced by 175W to 432W while decreasing the clock speed from 3.2 GHz to 1.5 GHz, with negligible performance losses not exceeding 1,5 percent. As a result, the total energy consumption is reduced by 1.38 times compared to the highest clock speed. For the version **D**, the minimum energy consumption is achieved for 2.0 GHz. It allows reducing the energy on around 12 percent at the cost of increasing the execution time on about 4.5 percent.

8 Performance Model

The performance of all MPDATA versions is limited [42] by the bandwidth of the memory hierarchy which includes the main memory and the cache hierarchy with L3 as LLC. So it is of significant interest to establish a model considering the influence of the memory traffic on the performance.

We use the Roofline model [43]. It is one of the common methods of determining performance bottlenecks of compute- and memory-bound applications. The Roofline sets an upper bound on the performance of a kernel depending on the kernel's operational intensity. Particularly, the attainable performance AP of a kernel is expressed as

$$AP = min(R_{peak}, I * B_{max}), \tag{2}$$

where R_{peak} is the peak performance, B_{max} is the peak memory bandwidth, while operational intensity I denotes the number of operations per byte of the memory traffic.

Assessing the quality of performed optimizations needs to consider the data movements for the MPADTA kernels. Each MPDATA team executes a sequence of MPDATA blocks with a size suitable for the cache hierarchy. Inside every block, the threads of each MPDATA team perform 17 kernels. Executing these kernels is further grouped into 5 packages P1–P5 (see Section 4.2). The first package (P1) consists of the first four MPDATA kernels and requires to load data from the main memory only. The rest of the packages (P2–P5) operate on (i) the data loaded to the L3 cache by P1 and (ii) intermediate results of prior kernels that all should be located in L3.

For example, let us analyze the execution of a domain of size $1024 \times 512 \times 128$ with the optimal block size of $1 \times 256 \times 128$. Each of the 16 teams operates on sub-domains of size $\frac{1024}{16} \times 512 \times 128$ and performs a sequence of 128 blocks.

The first package requires to transfer five portions of input data of size $1 \times (lh + 256 + rh) \times 128$, where the parameters lh and rh correspond to the properly defined ghost regions (see [21]). Assuming the double precision format, every team that executes the package P1 should transfer about 0.00133 GB of data from the main memory. As a result, the total amount Q of data transferred by all 16 teams that perform all 128 blocks is about 2.726 GB. At the same time, computing a single output element within the package P1 requires to perform 33 operations including 8 *add*, 4 *sub*, 10 *mul*, 5 *max*, 5 *min*, and one *div* operations, which yield totally about 0.001098 Gflop for a single block. The total number W of operations is about 2.2491 Gflop for all 128 blocks and all 16 teams.

To proceed further with our model, we utilize the LIKWID Performance Tools [44] to measure the required performance parameters of the test platform. Using the benchmark called *peakflops_avx*, we fix the peak performance as 2263.34 Gflop/s for AVX SIMD and non-FMA type of instructions, which is very close to the theoretical peak of 2304 Gflop/s. Through running the *load_avx* benchmark in the NPS4 mode, the maximum DRAM bandwidth is determined as about 322 GB/s.

Remark 2. The value of 322 GB/s should be interpreted as the upper bound for the memory bandwidth. It corresponds to rather large sizes of transferred data packages. In applications dealing with smaller packages, the memory bandwidth is lower. For example, the classic STREAM measurement from LIKWID yields only 220 GB/s. This value corresponds to the *stream_avx* benchmark. It is expected to achieve about 290 GB/s for the *stream_mem_avx* benchmark, which uses non-temporal stores (this benchmark failed to run on our platform).

Following the Roofline model, the operational intensity $I = \frac{W}{Q}$ for the package P1 is $I = 0.82 \frac{flop}{byte}$. With the peak data bandwidth $B_{max} = 322$ GB/s and peak performance $R_{peak} = 2263.34$ Gflop/s, the attainable performance for the package P1 becomes $AP = 288.8$ Gflop/s.

We repeat our estimation of the operational intensity for the other packages P2–P5 that operate on the data loaded from the L3 cache. Following Eq. (2), again it is necessary to measure the data transfer bandwidth, this time assuming loading data from L3. For this purpose, we use the *load_avx* benchmark from LIKWID, which returns the sustained bandwidth in the range from 1000 to 5000 GB/s achieved for different data package sizes. To provide a fair assessment, we determine the sustained bandwidth aggregated for all threads by separating input data transfers corresponded to a single MPDATA array processed by the package P4, where the block size is selected following Section 5.2. In this way, the sought value of the bandwidth is determined as 1655 GB/s. This allows us to estimate the attainable performance AP for each kernel.

Remark 3. To simplify the validation of the model accuracy, we run the MPDATA code without the asynchronous execution of threads within packages, which is offered by the version **D**. This simplification corresponds to utilizing the barrier synchronization. Disabling the

TABLE 6
DRAM:322 L3:1655 Roofline Performance Model for Domain of 1024×512×128 and Block Size 1×256×128

Package	P1	P2	P3	P4	P5
# kernels	4	1	2	6	4
Data location	Memory	Cache hierarchy with L3 as LLC			
W $[Gflop]$	2.249	1.961	3.938	3.787	3.557
Q $[GB]$	2.726	2.713	3.802	4.869	4.831
I $[\frac{flop}{byte}]$	0.82	0.72	1.04	0.78	0.74
AP $[Gflop/s]$	265.6	1199.9	1714.1	1287.2	1218.3
SP $[Gflop/s]$	243.1	916.6	1137.4	939.3	991.9
q [%]	91.5	76.4	66.4	73.0	81.4
Exec. time T $[s]$	103.02	22.58	36.21	41.80	35.96
v [%]	43	9.4	15.1	17.4	15.01

data flow synchronization allows measuring the sustained performance SP individually for every package by dividing the amount of performed operations by the execution time T.

The summary of our study is presented in Table 6, where q denotes the ratio of the measured performance SP to the attainable performance AP. The calculated values of q reveal the model accuracy in the range from 66.4 percent (package P3) to 91.5 percent (package P1). This level of agreement between the measured and predicted performance is quite good, taking into account the runtime and architectural overheads, including thread divergence within work teams.

Thus, the performance model established in this section confirms the memory-bound nature of the MPDATA application. The performance bottleneck is noticeable for the first package, where the DRAM bandwidth strongly limits the sustained performance. The execution of this package takes around $v = 43\%$ of the total execution time, where the value of v is estimated very roughly based on the values shown in the last but one row of Table 6. This estimation does not consider, e.g., overlapping in executing the packages using the data flow synchronization. For the rest of the packages, the proposed optimizations remove the constraints imposed by the DRAM memory bandwidth. In this case, the performance is limited by the bandwidth of the cache hierarchy with L3 as the last-level cache. This bandwidth is around 5.14 times higher than the DRAM one. Consequently, the sustained performance for the packages P2-P5 is increased in the range from 3.8 times for P2 to 4.7 times for P3, compared with the package P1.

The resulting Roofline graph is shown in Fig. 6. It illustrates increasing the operational intensity I by grouping kernels into packages due to loop fusion and loop tiling. For example, while the values of I for the original kernels K1-K4 are in the range 0.14–0.31, the operational intensity for the resulting package P1 becomes equal to 0.82. For the other packages, the main reason for the improved performance becomes the efficient utilization of the cache hierarchy. In Fig. 6, it is exemplified by the displacement of the points which correspond to the kernels grouped into the packages P2–P5 from the line corresponding to the DRAM bandwidth of 322 GB/s to the line corresponding to the L3 cache bandwidth of 1655 GB/s.

9 Comparison to Intel CPUs

Our research is completed by comparing the efficiency of the platform equipped with AMD Rome CPUs and those based on Intel Xeon Scalable processors of two generations. Intel-based systems are dual-socket servers with either Skylake SP (SKL-SP) or Cascade Lake-SP (CSL-SP) Intel Xeon CPUs, each with 28 cores. The first platform includes two Intel Xeon Platinum 8180 CPUs [45] and 192GB of DDR4 memory operating at 2666 MHz. The second one contains 192GB of DDR4-2933 and is powered by one of the top-of-the-line CPUs provided by the second generation of Intel Xeon Scalable architecture – Platinum 8280 [46]. While performance results are measured for all servers, energy consumption measurements are available only for the platform with Rome and SKL-SP processors.

The servers with SKL-SP, CSL-SP, and Rome processors offer quite similar theoretical peak performance R_{peak} of respectively 2060, 2150, and 2304 Gflop/s. These values assume SIMD vectorization usage with non-FMA instructions and turbo SIMD frequency (2.3 GHz, 2.4 GHz, and 2.25 GHz for SKL-SP, CSL-SP, and 7742 processors,

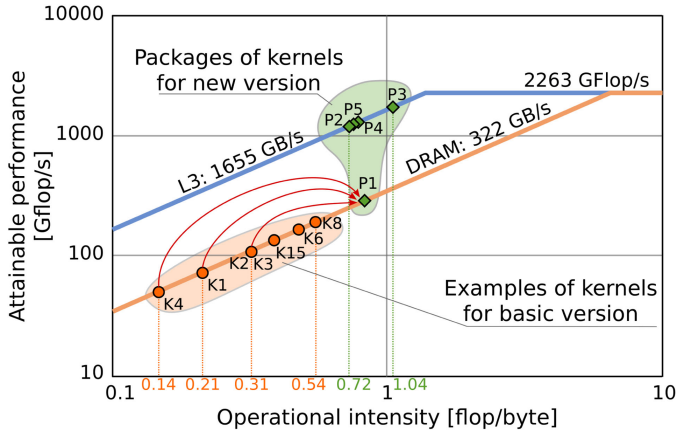

Fig. 6. The resulting Roofline graph (only some of 17 MPDATA kernels are shown).

TABLE 7
Performance and Energy Comparison Between Intel-Based and AMD-Based Platforms for MPDATA Domains of Size $1024 \times 512 \times 128$ and $2048 \times 1024 \times 128$, Double Precision Format, and 10000 Time Steps

Ver.	Computing platform	CPU freq. [GHz]	Time [s]	Sustained perf. [Gflop/s]	% of peak perf.	Total energy [KJ]	Avg. power [W]	Energy efficiency [Gflop/J]
1024×512×128								
A	2x Intel Platinum 8180	1.0	2665	51.64	2.51	947.48	355	0.15
	2x Intel Platinum 8280	1.0	2570	51.97	2.42	–	–	–
	2x AMD Rome 7742	1.5	1929	69.20	3.01	834.12	432	0.16
D	2x Intel Platinum 8180	2.3	245	630.75	30.62	140.20	572	1.10
	2x Intel Platinum 8280	2.4	240	644.39	29.97	–	–	–
	2x AMD Rome 7742	2.38	193	828.60	35.96	125.86	651	1.27
2048×1024×128								
D	2x Intel Platinum 8180	2.3	999	620.69	30.12	591.3	590	1.05
	2x Intel Platinum 8280	2.4	905	682.20	31.72	–	–	–
	2x AMD Rome 7742	2.38	734	854.60	37.09	493.06	672	1.27

respectively). To address the possible readers' surprise because of comparable values of R_{peak} for the 64-core Rome CPU and 28-core CSL-SP (or SKL-SP) processor, it should be noted that this advantage in the core count is counterbalanced by the double width of the Intel AVX-512 SIMD extension versus AMD AVX2. What is also essential, both AMD and Intel processors used by the test platforms have also quite similar thermal envelope (TDP) - 205W in the case of Intel CPUs versus 225W for Rome 7742. There are available other Cascade Lake-SP Intel Xeon CPUs [46] with a higher core count (from 32 to even 56 cores), but they have higher TDP as well - from 250W to even 400W.

All benchmarks on Intel-based servers are compiled using Intel Parallel Studio XE 2019 with the optimization flag `-O3` and properly chosen compiler arguments for AVX-512. To optimize the performance-energy trade-off for a given platform, we select the optimal clock speed individually for every MPDATA version to minimize energy consumption with negligible performance losses.

The comparison between Intel- and AMD-based systems is presented in Table 7. It shows the performance and energy/power measurements obtained for the versions **A** and **D** of MPDATA, as well as the sustained performance (Gflop/s) and the percentage of peak performance. This table also shows the energy efficiency expressed as the ratio of the total number of floating-point operations to energy consumption. All energy/power measurements are delivered by Yokogawa WT310, monitoring the entire platform.

For all tested platforms, the basic version **A** of MPDATA returns the best performance and energy results when using the lowest CPU frequency. The AMD-based server executes MPDATA 1.33–1.38 times faster than the Intel-based platforms. As a result, despite requiring 76.5W more power, the AMD-based system consumes about 1.14 less energy.

In contrast, when running the most optimized version **D**, the highest CPU frequency is set for all tested platforms to achieve the best performance. Again the AMD-based server outperforms the Intel-based ones, executing computations 1.23–1.36 times faster. Simultaneously, the energy consumption is about 1.12 times less than provided by the server with Skylake SP CPUs despite requiring 76W more.

The efficient porting of MPDATA to the tested platforms requires generating an optimal configuration for every optimization, based on parameters characterizing a given server. Table 8 shows the optimal configurations of the version **D** of MPDATA for each platform. Because of similar architectures of the Intel-based servers used in this study, the adaptation process for the MPDATA code returns the same configuration. The optimal number of MPDATA work teams in this case – 4 teams – is twice the number of available NUMA domains because of two memory controllers per CPU. As a result, every team includes 14 cores and effectively 14 MB of L3 (1 MB per core). This setup corresponds to the MPDATA block with the optimal size of 2x512x128. In this case, as shown in Table 8, all MPDATA packages consume all available L3 memory. At the same time, each package utilizes only up to half of the L2 cache capacity. In consequence, the size of L3 becomes the critical constraint in the calibration of the MPDATA code for the Intel-based servers.

For the platform with Rome CPUs, the MPDATA code requires 16 MPDATA teams to get the best performance and energy efficiency. This setup corresponds to 8 cores per

TABLE 8
Optimal Configuration of Version **D** of MPDATA for Different Platforms With Domains of Size $1024 \times 512 \times 128$ and $2048 \times 1024 \times 128$

Computing platform	Teams	Cores per team	Block size	L2 per core [MB]		L3 per team [MB]	
				Available cache	Max demanded cache (package P5)	Effective available cache	Total demanded cache (all packages)
Intel based platforms	4	14	$2 \times 512 \times 128$	1	0.44	14	14.5
AMD based platforms	16	8	$1 \times 256 \times 128$	0.5	0.47	32.0	16

each team that shares 32 MB of L3 cache (4 MB per core). As shown in Table 8, the optimal size of the MPDATA block – 1x256x128 – consumes only about half of the L3 cache capacity when performing all MPDATA packages. Simultaneously, the package with the highest L2 demand (P5) utilizes almost all L2 cache capacity. Thus, in contrast to the Intel-based platforms, the available size of L3 per team does not constrain the MPDATA adaptation.

As shown in Section 7 (Table 4), for the AMD-based server, the configuration with only 4 MPDATA teams - the best one for the systems with Intel CPUs - results in 88–140 percent longer execution time and 62–103 percent higher energy consumption compared to the optimal setup with 16 MPDATA teams. Additional degradation of up to 15 percent for the performance and 12 percent for energy consumption correspond to adjusting the block size based solely on the L3 capacity.

10 Conclusion

The second generation of AMD EPYC CPUs delivers large core counts, faster memory, and high-performance fabric. These advantages can be successfully used in the race to Exascale. However, the the novel architecture's complexity makes it challenging to adapt demanding scientific codes – like stencil ones – to platforms with AMD Rome CPUs. Moreover, the differences between Intel Xeon and AMD Rome architectures thoroughly affect optimizing parallel codes to achieve sustained high performance of real-life scientific applications.

This paper tackles this challenge and presents the architectural adaptation of a real-world CFD application to influential features of Rome processors. In the background, we place the parametric optimizations of the MPDATA application for Intel processors. Extending these optimizations allows us to develop a portable methodology for parallelizing the memory-bound MPDATA code. The goal is to optimize this code for server platforms powered by both Intel and AMD processors of the last generations.

This work explores how the proposed methodology addresses performance-energy trade-offs. In particular, we focus on the impact of the set of optimizations proposed in [4] on the MPDATA performance and energy efficiency. For this aim, we study the energy and power consumption for a dual-socket ccNUMA platform with AMD Rome 7742 CPUs (128 cores in total), using the accurate power meter. This study proves that our methodology can reduce both execution time and energy consumption of MPDATA radically – in the range of around 9.8–10.4 and 9.2-9.4 times for performance and energy consumption, respectively. It is also shown that despite significant differences in the behavior of particular optimizations for various NPS modes of configuring the Rome architecture, the combination of these optimizations leads to practically the same performance and energy consumption for all modes. The achieved performance results are verified by the Roofline-based model, which connects the MPDATA performance with the main memory and cache bandwidth.

Finally, the paper provides the performance-energy comparison of the platform with two AMD Rome 7742 CPUs against two servers with Intel Xeon Scalable processors of different generations – Intel Xeon Platinum 8180 and 8280. The dual-socket platform with Rome processors outperforms both Intel-based systems in performance and energy consumption, allowing us to execute computations 1.23–1.36 times faster with up 1.12 times fewer energy costs. Even without appealing to prices, these results are a serious reason to confirm Rome's competitiveness against Intel Xeon CPUs in scientific applications. Given the highly possible superiority of AMD Rome processors in price [14], [47], their actual level of competitiveness appears to be even higher.

Determining an optimal configuration of the MPDATA code is vital to ensure the performance portability on different architectures. The proposed adaptation methodology [4] resulted in developing the customizable MPDATA code that can follow various hardware parameters. The extension of this parametric adaptation on the novel EPYC architecture requires careful consideration of two aspects: partitioning cores into work teams and cache hierarchy features. The first one reflects splitting the Rome architecture into multiple dies, each communicating with other dies through Infinity Fabric links. The second aspect takes into account the relationship between the sizes of L3 and L2 caches.

In consequence, even for the memory-bound MPDATA application, the ratio r of sustained performance to peak performance for the analyzed AMD Rome processor ($r \approx 36 - 37\%$) is better than in the case of Intel Platinum CPUs ($r \approx 30 - 31.7\ \%$). This result is the effect of determining the optimal MPDATA configuration adjusted to the AMD Rome architecture features instead of using a solution chosen for Intel CPUs. Selecting the optimal configuration improves performance up to more than 2.5 times and energy consumption up to 2.15 times.

Acknowledgments

The authors would like to thank HPE Poland and Poznan Supercomputing and Networking Center (Poland) for granting access to HPC platforms. This work was supported in part by the National Science Center Poland under Grant UMO-2017/26/D/ST6/00687 and in part by the Polish Minister of Science and Higher Education though Regional Initiative of Excellence Project in 2019–2022 under Grant 020/RID/2018/19.

References

[1] HPC in 2020: Compute engine diversity gets real, Jan. 2020. [Online]. Available: https://www.nextplatform.com/2020/01/13/hpc-in-2020-compute-engine-diversity-gets-real/

[2] European weather center breaks tradition with upcoming supercomputer, Jan. 2020. [Online]. Available: https://www.nextplatform.com/2020/01/14/

[3] AMD infinity architecture, TIRIAS White Paper, Aug. 2019. [Online]. Available: https://www.amd.com/system/files/documents/TIRIAS-White-Paper-AMD-Infinity-Architecture.pdf

[4] L. Szustak and P. Bratek, "Performance portable parallel programming of heterogeneous stencils across shared-memory platforms with modern Intel processors," *Int. J. High Perform. Comput. Appl.*, vol. 33, no. 3, pp. 507–526, 2019.

[5] A. Kashyap, "High performance computing: Tuning guide for AMD EPYC 7002 series processors," Jan. 2020. [Online]. Available: https://developer.amd.com/wp-content/resources/56827–1-0.pdf

[6] J. Munoz, C. Kitchen, and M. Guest, "Performance analysis of AMD EPYC Rome processors," Dec. 2019. [Online]. Available: https://www.scd.stfc.ac.uk/SiteAssets/Pages/CIUK-2019-Presentations/Martyn_Guest.pdf

[7] AMD EPYC 7002 series processors weather modeling with WRF, 2020. [Online]. Available: https://www.amd.com/system/files/documents/EPYC-7002-Weather-Modeling-with-WRF.pdf
[8] AMD EPYC 7002 series processors and GROMACS molecular dynamic simulation, Aug. 2019. [Online]. Available: https://www.amd.com/system/files/documents/EPYC-7002-Gromacs-Molecular-Dynamics-Simulation.pdf
[9] P. Smolarkiewicz, "Multidimensional positive definite advection transport algorithm: An overview," *Int. J. Numer. Methods Fluids*, vol. 50, no. 10, pp. 1123–1144, 2006.
[10] P. Smolarkiewicz and P. Charbonneau, "EULAG, a computational model for multiscale flows: An MHD extension," *J. Comput. Phys.*, vol. 236, pp. 608–623, 2013.
[11] L. Szustak, R. Wyrzykowski, T. Olas, and V. Mele, "Correlation of performance optimizations and energy consumption for stencil-based application on Intel Xeon scalable processors," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2582–2593, Nov. 2020.
[12] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Energies*, vol. 12, no. 11, 2019, Art. no. 2204.
[13] WT300 series digital power meter analyzer, 2018. [Online]. Available: https://tmi.yokogawa.com
[14] AMD EPYC vs. Intel Xeon Cascadelake with Facebook's Rocksdb database, Oct. 2019. [Online]. Available: https://www.phoronix.com/scan.php?page=article&item=intel-amd-rocksdb&num=1
[15] X. Guo and O. W. Saastad, "Best practice guide - AMD EPYC," Feb. 2019. [Online]. Available: https://prace-ri.eu/wp-content/uploads/Best-Practice-Guide_AMD.pdf
[16] Clang – The C, C++ compiler. Accessed: Mar. 1, 2021. [Online]. Available: https://developer.amd.com/amd-aocc/
[17] AMD optimizing CPU libraries (AOCL)," 2020. [Online]. Available: https://developer.amd.com/amd-aocl/
[18] G. Hager and G. Wellein, *Introduction to High Performance Computing for Science and Engineers*. Boca Raton, FL, USA: CRC Press, 2011.
[19] K. Datta *et al.*, "Optimization and performance modeling of stencil computations on modern microprocessors," *SIAM Rev.*, vol. 51, no. 1, pp. 129–159, 2009.
[20] T. Henretty *et al.*"Data layout transformation for stencil computations on short-vector SIMD architectures," in *Proc. Int. Conf. Compiler Construction*, 2011, pp. 225–245.
[21] L. Szustak, K. Rojek, T. Olas, L. Kuczynski, K. Halbiniak, and P. Gepner, "Adaptation of MPDATA heterogeneous stencil computation to Intel Xeon Phi coprocessor," *Sci. Program.*, vol. 2015, 2015, Art. no. 642705.
[22] J. Holewinski, L.-N. Pouchet, and P. Sadayappan, "High-performance code generation for stencil computations on GPU architectures," in *Proc. 26th ACM Int. Conf. Supercomputing*, 2012, pp. 311–320.
[23] K. Rojek, K. Halbiniak, and L. Kuczynski, "CFD code adaptation to the FPGA architecture," *Int. J. High Perform. Comput. Appl.*, vol. 35, no. 1, pp. 33–46, 2021.
[24] A. Lastovetsky, L. Szustak, and R. Wyrzykowski, "Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 787–797, Mar. 2017.
[25] X. Zhou *et al.*"Hierarchical overlapped tiling," in *Proc. 10th Int. Symp. Code Gener. Optim.*, 2012, pp. 207–218.
[26] V. Bandishti, I. Pananilath, and U. Bondhugula, "Tiling stencil computations to maximize parallelism," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Analysis*, 2012, pp. 1–11.
[27] I. Bertolacci *et al.*, "Parameterized diamond tiling for stencil computations with chapel parallel iterators," in *Proc. 29th ACM Int. Conf. Supercomputing*, 2015, pp. 197–206.
[28] T. Malas, J. Hornich, G. Hager, H. Ltaief, C. Pflaum, and D. Keyes, "Optimization of an electromagnetics code with multicore wavefront diamond blocking and multi-dimensional intra-tile parallelization," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2016, pp. 142–151.
[29] J. Guo, G. Bikshandi, B. B. Fraguela, and D. Padua, "Writing productive stencil codes with overlapped tiling," *Concurrency Comput., Pract. Experience*, vol. 21, no. 1, pp. 25–39, 2009.
[30] B. Hagedorn, L. Stoltzfus, M. Steuwer, S. Gorlatch, and C. Dubach, "High performance stencil code generation with LIFT," in *Proc. Int. Symp. Code Gener. Optim.*, 2018, pp. 100–112.
[31] R. Manumachu and A. Lastovetsky, "Bi-objective optimization of data-parallel applications on homogeneous multicore clusters for performance and energy," *IEEE Trans. Comput.*, vol. 67, no. 2, pp. 160–177, Feb. 2018.
[32] H. Anzt, "Asynchronous and multiprecision linear solvers: Scalable and fault-tolerant numerics for energy efficient high performance computing," Ph.D. dissertation, Inst. Appl. Numer. Math., Karlsruhe Inst. Technol., 2012.
[33] K. Eder and J. P. Gallagher, "Energy-aware software engineering," in *ICT - Energy Concepts for Energy Efficiency and Sustainability*, G. Fagas, L. Gammaitoni, J. P. Gallagher, and D. J. Paul, Eds., Rijeka, Croatia: IntechOpen, pp. 1–165, 2017.
[34] AMD EPYC 7002 series processors, 2020. [Online]. Available: https://www.amd.com/en/processors/epyc-7002-series
[35] L. Szustak, K. Halbiniak, R. Wyrzykowski, and O. Jakl, "Unleashing the performance of ccNUMA multiprocessor architectures in heterogeneous stencil computations," *J. Supercomputing*, vol. 75, pp. 7765–7777, 2019.
[36] J. Hennessy and D. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed., Burlington, MA, USA: Morgan Kaufmann, 2019.
[37] B. Rosa, L. Szustak, A. Wyszogrodzki, K. Rojek, D. Wojcik, and R. Wyrzykowski, "Adaptation of multidimensional positive definite advection transport algorithm to modern high-performance computing platforms," *Int. J. Model. Optim.*, vol. 5, no. 3, pp. 171–176, 2015.
[38] L. Szustak, "Strategy for data-flow Synchronizations in stencil parallel computations on multi-/manycore systems," *J. Supercomputing*, vol. 74, no. 4, pp. 1534–1546, 2018.
[39] Intel 64 and IA-32 architectures optimization reference manual, Apr. 2018. [Online]. Available: https://software.intel.com, April 2018.
[40] Yoko tool, 2020. [Online]. Available: https://01.org/yoko-tool
[41] E. Calore, A. Gabbana, S. Schifano, and R. Tripiccione, "Software and DVFS tuning for performance and energy-efficiency on Intel KNL processors," *J. Low Power Electronics Appl.*, vol. 8, no. 2, 2018, Art. no. 18.
[42] L. Szustak, K. Halbiniak, L. Kuczynski, J. Wrobel, and A. Kulawik, "Porting and optimization of solidification application for CPU–MIC hybrid platforms," *Int. J. High Perform. Comput. Appl.*, vol. 32, no. 4, pp. 523–539, 2018.
[43] S. Williams, A. Watterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, 2009.
[44] LIKWID performance tools, Mar. 2021. [Online]. Available: https://hpc.fau.de/research/tools/likwid/
[45] Xeon Platinum 8180 – Intel, Jul. 2017. [Online]. Available: https://en.wikichip.org/wiki/intel/xeon_platinum/8280
[46] 2nd generation Intel® Xeon® scalable processors, 2019. [Online]. Available: https://ark.intel.com/content/www/us/en/ark/products/series/192283/2nd-generation-intel-xeon-scalable-processors.html
[47] Detailed specifications of the AMD EPYC Rome CPUs, 2020. [Online]. Available: https://www.microway.com/knowledge-center-articles/detailed-specifications-of-the-amd-epyc-rome-cpus/

Lukasz Szustak received the PhD degree from the Czestochowa University of Technology in 2012 and the DSc degree in computer science in 2019. His research interests include parallel computing and mapping algorithms onto parallel architectures. His current research interests include the development of methods for performance portability, scheduling, and load balancing, including the adaptation of stencil-based computations to modern HPC architectures.

Roman Wyrzykowski (Senior Member, IEEE) received the MSc and PhD degrees in computer science from the Kiev Polytechnic Institute in 1982 and 1986, respectively. Since 1982, he is with the Czestochowa University of Technology, Poland. Since 1994, he has been the chair of the program committee of the PPAM series of international conferences on parallel processing. His research interests include parallel and distributed computing, mapping algorithms onto cluster, and cloud systems.

Lukasz Kuczynski received the MSc degree in computer science from the Czestochowa University of Technology, in 2001 and the PhD degree in computer science, the dissertation on efficient data management in PC meta-clusters, in 2010. His research interests include parallel and distributed computing, mapping algorithms onto FPGAs, and other parallel architectures.

Tomasz Olas received the MSc degree in computer science from the the Czestochowa University of Technology, Poland, in 1999 and the PhD degree in computer science, the dissertation on mapping FEM computations onto parallel and distributed systems, in 2004. His main research interests include parallel and distributed computing, mapping algorithms onto parallel architectures, and cluster and cloud technologies.