# Correlation of Performance Optimizations and Energy Consumption for Stencil-Based Application on Intel Xeon Scalable Processors

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

**Abstract**—This article provides a comprehensive study of the impact of performance optimizations on the energy efficiency of a real-world CFD application called MPDATA, as well as an insightful analysis of performance-energy interaction of these optimizations with the underlying hardware that represents the first generation of Intel Xeon Scalable processors. Considering the MPDATA iterative application as a use case, we explore the fundamentals of energy and performance analysis for a memory-bound application when exposed to a set of optimization steps that increase the application performance, by improving the operational intensity of code and utilizing resources more efficiently. It is shown that for memory-bound applications, optimizing toward high performance could be a powerful strategy for improving the energy efficiency as well. In fact, for the considered performance optimizations, the energy gain is correlated with the performance gain but with varying degrees. As a result, these optimizations allow improving both performance and energy consumption radically, up to about 10.9 and 8.8 times, respectively. The impact of the Intel AVX-512 SIMD extension on the energy consumption and performance is demonstrated. Also, we discover limitations on the usability of CPU frequency scaling as a tool for balancing energy savings with admissible performance losses.

**Index Terms**—CFD, MPDATA , intel xeon scalable, performance-energy trade-off, yokogawa WT310, RAPL

## 1 Introduction

The energy consumption of current and emerging supercomputers is still increasing despite improvements in the energy efficiency area. This trend is among the five major challenges that should be overcome on the way to exascale computing [1]. Most of the gain in the total consumption of energy comes from technology [2], [3] that provides a better performance-energy trade-off by offering more and more energy-efficient and compute-intensive hardware. The remarkable example is a family of Intel Xeon Scalable processors [4] that open promising opportunities to reduce the energy consumption of HPC applications, and to explore new trade-offs between energy and performance.

Another direction in energy-aware HPC is related to rethinking the software, including both execution environments and applications themselves. An example of research on modifying the execution environment is work [5], where an energy-aware task management mechanism is designed and evaluated for an iterative CFD (computational fluid dynamic) application – the multidimensional positive definite advection transport algorithm (MPDATA) [6] running on multicore CPUs. This algorithm is one of the main parts of the EULAG geophysical model [7] developed for simulating thermo-fluid flows across a wide range of scales and physical scenarios (https://www2.mmm.ucar.edu/eulag). MPDATA executes a set of heterogeneous stencils and represents a memory-bound application.

The proposed mechanism is based on the dynamic voltage and frequency scaling technique [8] applied for the existing algorithm. This mechanism provides a solution in the case when the execution time of an algorithm is predefined, and the energy consumption is optimized such that the application is executed in the requested fixed amount of time. The experimental results achieved for a 6-core computing platform showed that the proposed mechanism provided the energy savings of up to 43 percent when compared to the default Linux scaling governor.

At the same time, in our previous paper [9], we focused on modifying the MPDATA applications by developing a set of parametric optimizations targeting the high performance of code, along with a 4-step procedure for customization of the MPDATA code. Among the optimizations are: (3+1)D decomposition, islands-of-cores strategy, exploiting data parallelism and simultaneous multithreading, data flow synchronization, and vectorization. The proposed adaptation methodology allows developing the automatic transformation of the MPDATA code, achieving high sustained scalable performance for all tested ccNUMA platforms with Intel processors of last generations.

The set of optimizations developed in [9] improves the operational intensity of code and permits utilizing computing resources more efficiently. This allows us to formulate a working hypothesis that for memory-bound applications

- *L. Szustak, R. Wyrzykowski, and T. Olas are with the Department of Computer and Information Science, Czestochowa University of Technology, 42-201 Częstochowa, Poland. E-mail: {lszustak, roman, olas}@icis.pcz.pl.*
- *V. Mele is with the Department of Mathematics and Applications R. Caccioppoli, University of Naples Federico II, 80138 Napoli, Italy. E-mail: valeria.mele@unina.it.*

*Manuscript received 20 Sept. 2019; revised 30 Mar. 2020; accepted 16 May 2020. Date of publication 28 May 2020; date of current version 10 June 2020. (Corresponding author: L. Szustak)*
*Recommended for acceptance by C. Carothers.*
*Digital Object Identifier no. 10.1109/TPDS.2020.2996314*

optimizing towards the high performance could be a successful strategy for improving the energy efficiency as well. To verify this hypothesis experimentally, we use the MPDATA application as a use case, exploring the correlation between MPDATA performance optimizations and energy consumption for a ccNUMA/SMP platform equipped with two top-of-the-line Intel Xeon Platinum 8180 CPUs, each with 28 cores. These CPUs represent the first generation of Intel Xeon Scalable processors. Besides using the RAPL infrastructure [10], the Yokogawa WT310 digital power meter [11] is utilized to measure the power and energy of the platform. This provides accuracy and reliability of measurements.

As the first step of our research, we evaluate the impact of the performance optimizations steps proposed in [9] on the energy and power consumption. Specifically, the four versions of the MPDATA application are examined. To reveal the correlation between the performance and energy consumption for all MPDATA versions, we provide an insightful analysis of their interaction with the underlying hardware.

All MPDATA versions apply various performance optimization steps, and in consequence, they vary in terms of their performance and energy efficiency. While the performance of the first (original, non-optimized) version is strongly limited by the memory bandwidth, other versions of MPDATA overcome memory, communication, and synchronization constraints permitting a better utilization of available computing resources. The proposed combination of optimization steps allows us to improve radically the efficiency of MPDATA, reducing both execution time and energy consumption.

As the next step of our study, we investigate the impact of the Intel AVX-512 SIMD extension on the performance and energy consumption, for various MPDATA versions. The paper reveals the benefits of utilizing the features of AVX-512 instructions to reduce energy consumption.

This research also explores the interaction of the frequency scaling technique with both performance and energy efficiency. The goal is to evaluate the usability of this technique as a tool for balancing energy savings with admissible performance losses, considering not only the original version of MPDATA, as it was assumed in work [5], but also the optimized code. Finally, we carefully compare energy measurements obtained with the hardware and software approaches, using respectively Yokogawa WT310 and RAPL.

The main contributions of this work are:

1) For a real-world CFD application called MPDATA, we provide a comprehensive study of the impact of performance optimizations on the energy efficiency of application, as well as an insightful analysis of performance-energy interaction of these optimizations with the underlying hardware that represents the first generation of Intel Xeon Scalable processors.
2) Considering the MPDATA iterative application as a use case, we explore the fundamentals of energy and performance analysis for a memory-bound application when exposed to a set of optimization steps that increase the application performance by improving the operational intensity of code and utilizing computing resources more efficiently.
3) We show that for memory-bound applications, optimizing towards the high performance could be a powerful strategy for improving energy efficiency as well. In fact, for the considered performance optimization steps, the energy gain is correlated with the performance gain but with varying degrees. While for the first step, which improves both cache reusing and the data locality, the energy gains are about 10 percent higher than those for the execution time, the two other steps, that reduce the synchronization and communication overheads, lead to smaller benefits in energy than in performance. As a result, these optimizations allow improving both performance and energy consumption radically, up to about 10.9 and 8.8 times, respectively.
4) We demonstrate the impact of the Intel AVX-512 SIMD extension on improving the energy consumption and performance of applications. The lowest benefits of vectorization are obtained for the original, non-optimized version of MPDATA, when the reduced power requirements of the code with enabled SIMD allows reducing the energy consumption by about 20 percent despite the same execution time as its scalar counterpart. The highest profits are achieved for the most-optimized MPDATA version that permits improving both energy consumption and performance about 2.8 times.
5) We evaluate the usability of CPU frequency scaling as a tool for balancing energy savings with admissible performance losses. While for the non-optimized MPDATA code, its execution with the lowest frequency allows decreasing the energy consumption significantly, without any performance degradation, reducing the frequency for the most-optimized version does not permit any improvements not only in performance but also in energy.
6) We provide a preliminary comparison of the accuracy of the software approach to energy/power measurements with on-chip power sensors and RAPL against the hardware approach using the external power meter (Yokogawa WT310). To avoid high discrepancies in energy measurements for long-time simulations, a simple yet efficient method is proposed to correct results obtained with RAPL. This method is based on deriving the final result as an aggregated sum of measurements obtained separately for all packages.

This paper is organized as follows. Section 2 discusses related works, while Section 3 and Section 4 outline respectively the computing platform and energy/power measurement environment used in experiments. The MPDATA application is introduced in Section 5, which also presents the parallelization methodology for shared memory multi- and manycore architectures. The methodology of the experimental part is introduced in Section 6, while the results of experiments are presented and discussed in Section 7. The paper is concluded in Section 8.

## 2 Related Works

The large computing infrastructures are becoming more and more limited by power/energy constraints, so it is understandable why the energy efficiency of HPC parallel

workloads become a focus of attention in recent times [12]. Beside the trend of energy reduction achieved through re-engineering the hardware, we can observe another one with an excellent potential for energy savings that is realized by transforming and completely rethinking the algorithms and software dedicated for HPC platforms [13].

Following this direction, the development of modern HPC software for scientific computing is currently focusing on optimizing scientific codes with carefully analyzing the trade-off between time and power/energy consumption [14], [15]. Similarly to our study, a lot of research efforts aim today to advance the state-of-the-art in power-aware computing [16], focusing on different aspects of the field.

A common technique is DVFS (dynamic voltage and frequency scaling) [8], which allows the hardware to lower the operational voltage and frequency at the cost of possibly higher execution time [17], [18]. Such an approach gives the possibility to optimize the energy efficiency when cycles are typically being wasted since they are stalled on memory resources [17].

Going forward, works [19], [20] studied the use of hardware power capping mechanisms to reduce the energy consumption in the parallel execution of applications with various workloads and memory usage. These works explored the characteristics of the Intel RAPL infrastructure [10] and the frequency scaling approach, constraining the power budget assigned to computing devices while executing HPC applications.

Wlotzka *et al.* [21] addressed key issues of energy-aware high performance computing by offering opportunities for saving energy in computations by energy-aware runtimes on shared-memory multicore platforms. This work outlined some numerical methods which are often used in scientific applications, and presented an energy profiling and tracing technique suitable to analyze the power consumption of applications.

Work [22] revealed various software optimization techniques for reducing the energy consumption without modifying the underlying hardware. The study discussed performing machine-independent optimizations that do not require any knowledge of the underlying hardware architecture.

Hassan *et al.* [16] studied the impact of using different coding styles on achieving a balance between performance and energy efficiency. In [23], Jakobs *et al.* demonstrated the capabilities and limitations of vectorization concerning energy efficiency, considering both automatic and manual vectorization techniques. The influence of vectorization combining with multi-threading on the energy efficiency of program executions is investigated in [24]. In [25], Rojek proposed a method that leverages the mixed precision arithmetic to reduce the energy consumption for applications executed in supercomputing centers.

Works [26], [27] demonstrated bi--objective approaches to distribute the workload among processing cores in heterogeneous parallel platforms. The goal is to benefit from the trade-off between execution time and energy consumption, aiming at reducing the energy without increasing the execution time.

TABLE 1
Specification of the Test Platform Equipped With Two Intel Xeon Platinum 8180 CPUs (https://ark.intel.com)

| | | |
|---|---|---|
| Scalar base / turbo freq. [GHz] | | 3.2 / 2.5 |
| SIMD base / turbo freq. [GHz] | | 2.3 / 1.7 |
| Sockets | | 2 |
| Cores / Threads | | 56 / 112 |
| SIMD | | AVX-512 |
| L1 [MB] | | $2 \times 28 \times 1$ |
| L2 [MB] | | $2 \times 38.5$ |
| Main memory [GB] | | $12 \times 16$ |
| Memory type | | DDR4-2666 |
| Memory bandwidth [MB/s] | | 255.9 |
| Peak performance * [Gflop/s] | Scalar | 358.4 |
| | SIMD | 2060.8 |

*Refers to multiplication instruction and turbo frequency.*

## 3 Architecture of Intel-Based HPC Platform

The performance and power measurements presented in this work are performed on the dual-socket Intel Server System R2208WFTZS equipped with Intel Xeon Platinum 8180 CPUs and 192GB of DDR4 memory operating at 2666 MHz (96 GB for each CPU). The basic software includes the CentOS Linux 7.5 operating system and Intel Parallel Studio XE 2019 update 1. Table 1 summarizes this platform.

Each Intel Xeon Platinum 8180 processor has 28 cores with two SMT (simultaneous multithreading) threads per core, which gives totally 56 physical cores for the test platform, outlining themselves as 112 logical processors. The thermal design power (TDP) of a processor is 205 W.

Compared to the previous generations, one of the new features of Intel Xeon Scalable processors is the Intel Advanced Vector Extensions 512 (Intel AVX-512) instruction set, that doubles the vector register width to 512 bits. The Intel Xeon Scalable processors also support older instruction sets, including AVX/AVX2. Furthermore, each core of this processor [28] enables processing up to two scalar or vector floating-point instructions at every next cycle. For example, since each core has two FP Add execution units assigned to ports 0 and 5, two vector (or two scalar) `add` instructions can be performed per each cycle.

The Intel Xeon Platinum 8180 processors are clocked at the base frequency of 2.5 GHz. In general, the base clock frequency of Intel Xeon Scalable processors just refers (https://ark.intel.com) to frequency for scalar workloads when both all cores are concurrently utilized, and the Intel Turbo Boost technology [4] is disabled.

The Intel Turbo Boost technology enables increasing the base clock frequency when the thermal and power limitations permit this. The clock speed depends on the type and intensity of workload, as well as the number of concurrently utilized cores. In general, the maximum Turbo Boost frequency is attainable for scalar workloads, lower for heavy AVX2 workloads, and even lower for applications that use AVX-512 intensively.

The Intel Xeon Platinum 8180 processors can reach the maximum Turbo Boost speed of 3.2, 2.8, and 2.3 GHz for respectively scalar, AVX2 and AVX-512 workloads defined when all cores are active. At the same time, the minimum

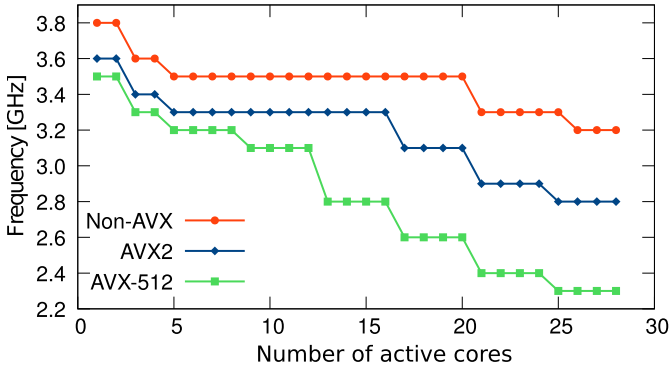


Fig. 1. Intel turbo boost frequency for Intel Xeon Platinum 8180 [4].

possible clock frequency is 1.0 GHz. Fig 1 illustrates the dependency between the number of concurrently utilized cores and maximum turbo frequency for various workloads.

The test platform offers the theoretical peak performance of 358.4 Gflop/s for non-AVX workloads, and 2060.8 Gflop/s for AVX-512 when using double precision floating-point operations (see [29]). These values correspond to Intel Turbo Boost and multiplication instructions. For fused multiply-add (FMA) instructions, the peak performance is twice higher.

## 4 Measurements Techniques for Power and Energy

The accurate measurement of energy consumption during workload execution plays a crucial role in energy-aware HPC computing [30], [31]. There are two popular approaches in this area [12], [32]: (i) hardware-based technique that uses an external AC power meter, and (ii) software-based approach that explores on-chip power sensors.

The first approach enables monitoring of power and energy for the whole computing system, using an external AC power meter [30] such as Yokogawa WT310 digital power meter. The AC power meter is a device that passes power to the server under the load and measures the power and energy consumption in real-time [12]. While this technique is known to be accurate [30], it, however, lacks the ability to perform power and energy analysis for independent elements of a given computing platform.

The second technique is based on on-chip power sensors currently supported by mainstream hardware computing vendors such as Intel, AMD or NVIDIA [30]. For example, Intel CPUs provide Running Average Power Limit (RAPL) [12], [33] to monitor power, as well as control both frequency and voltage. RAPL works based on processor counters and a software calculation model that estimates the energy consumption. The data about energy consumption are collected through model-specific registers (MSRs) to be farther exposed by using specialized tools or general libraries, such as Intel PCM (Performance Counter Monitor) or PAPI (Performance Application Programming Interface) [30], [33]. In RAPL, a given platform is divided into domains for fine-grained energy reports and control. These domains include: (i) all the CPU cores (*PP0*), (ii) a uncore device (*PP1*), (iii) the main memory (*DRAM*), and the entire socket(s) (*Package*.) As a result, RAPL reports the energy consumption for only CPUs and the main memory of a server. In consequence, it lacks the ability for energy monitoring of other essential components, such as the cooling subsystem that is an important contributor to energy consumption.

## 5 MPDATA Application

### 5.1 Background of Application

The MPDATA application implements a general approach to integrating the conservation laws of geophysical fluids on micro-to-planetary scales [34]. The MPDATA algorithm enables solving the continuity equation describing the advection of a non-diffusive quantity $\Psi$ in a flow field [35], namely:

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

where V is the velocity vector.

For a particular time step, the numerical structure of MPDATA is described as an iterated upwind scheme. The first iteration refers to the first-order accurate upwind scheme with the advective velocity given by the physical flow. For the second-order accurate scheme, the advected field is updated by an adequately defined pseudo-velocity designed to reduce the second order of the spatial and temporal truncation errors of the previous iteration. The detailed description of the MPDATA numerical scheme is presented in works [7], [34].

MPDATA corresponds to the group of nonoscilatory forward-in-time algorithms. Typically it is used for long-running simulations executing many thousand time steps solving 2- or 3-dimensional advection problems. In this paper, we consider 3D problems, when MPDATA is defined in a 3D domain of size $n \times m \times l$ according to $i-$, $j-$, and $k-$dimensions.

MPDATA features a stable computation intensity for all time steps. Each of them operates on five input matrices (arrays), and returns a single output array that is used in the next step. The returned array of a given time step requires to perform intermediate computations of 17 MPDATA kernels. In general, these kernels depend on each others, as the outcomes of prior kernels are usually inputs for the subsequent ones. Each kernel is a stencil code that updates elements of its 3D output array, according to a specific pattern. Fig. 2 depicts both data dependencies between MPDATA kernels, and examples of stencil patterns corresponding to kernels 9, 10, and 11.

**Listing 1.** Part of 3D MPDATA basic version, related to the 4-th kernel

```
//Kernel 4
#pragma omp for
for( \ldots ) // i - dimension
 for( \ldots ) // j - dimension
  #pragma omp simd
  for( \ldots ) // k - dimension
  XOut[i,j,k] = XIn[i,j,k] - ((
    (F1[i+1,j,k]-F1[i,j,k]) +
    (F2[i,j+1,k]-F2[i,j,k]) +
    (F3[i,j,k+1]-F3[i,j,k])) / H[i,j,k]);
```

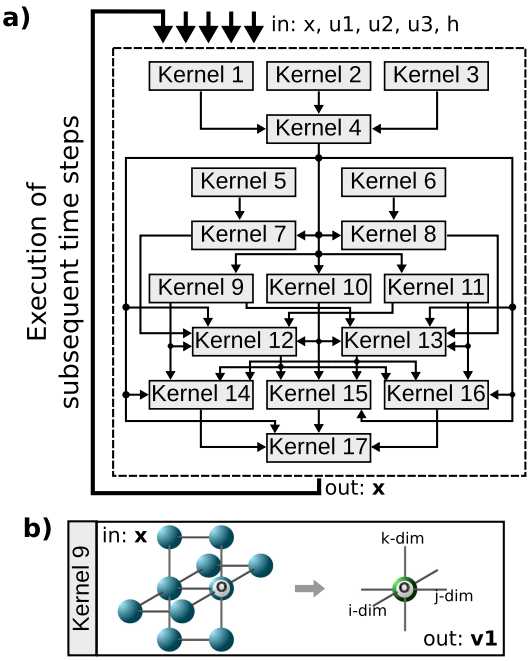


Fig. 2. Structure of MPDATA [9]: a) graph of data dependencies between MPDATA kernels, b) example of heterogeneous stencil patterns.

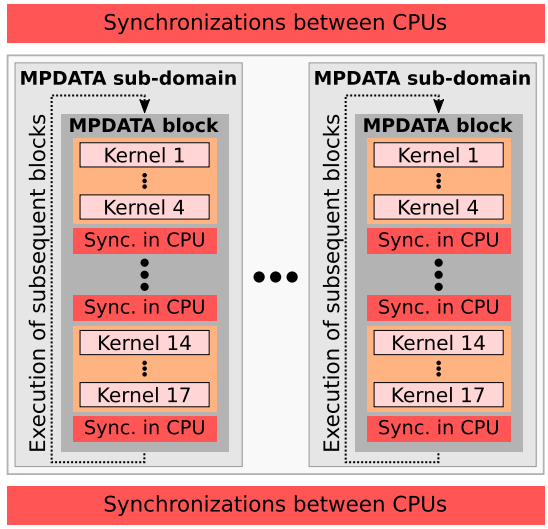


Fig. 3. Executing a single MPDATA time step when using combination of the proposed optimization steps.

### 5.2 MPDATA Parallelization

In the basic implementation of MPDATA (Listing 1), all kernels are executed sequentially, one by one, where each kernel is processed in a parallel way using OpenMP. This version 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.

Every MPDATA kernel reads a required set of arrays from the main memory and writes results to the main memory after computation. The consequence is the intensive traffic to the main memory. This traffic strongly limits the parallel efficiency of the basic version, since the operational intensity of each MPDATA kernel is not high enough ([9]) to efficiently utilize computing resources, and the code is not optimized for cache reusing. Thus, the performance of the basic version of MPDATA is limited by the main memory bandwidth.

To alleviate the memory-bound nature of MPDATA, we developed [9], [36], [37], [38] a parallelization methodology for MPDATA heterogeneous stencil computations. It contributes to ease the memory and communication bounds, and exploits resources of multicore ccNUMA/SMP systems better. This methodology consists of the following parametric optimization steps:

- *(3+1)D decomposition of MPDATA* ([38]) – the prime goal is to take advantage of cache reusing by transferring the data traffic between kernels from the main memory to the cache hierarchy. For this aim, a combination of loop tiling and loop fusion optimization techniques is used, which allows reducing the main memory traffic at the cost of additional computations associated with *halo* areas on boundaries of all intermediate arrays.
- *Partitioning cores into independent work teams* ([36]) – this step enables the flexible management of trade-off between costs of computation and communication, following features of ccNUMA systems. As a result, two scenarios of executing MPDATA kernels are introduced: the first one performs fewer computations but requires more data traffic, while the second scenario allows us to replace the implicit data traffic by replicating some of the computations. As a result, the second scenario is successfully used to reduce inter-processor communications between caches in ccNUMA systems, while the first scenario is applied inside each processor, where the more efficient, local memory hierarchy is utilized for data traffic.
- *Data-flow strategy of synchronization* ([37]) – the main purpose is to synchronize only interdependent threads instead of using the barrier approach that typically synchronizes all threads. This strategy reduces the cost of synchronization since it requires to synchronize radically less number of cores/threads than the barrier approach. Implementing this strategy for MPDATA needs to reveal the scheme of inter-thread data traffic during the execution of MPDATA kernels.
- *Vectorization* – the last step is responsible for ensuring the performance portability of vectorizing MPDATA computations. The main assumption is to specify the process of vectorization entirely and correctly so that a compiler will make vectorization automatically. In paper [9], the 7-step procedure for the MPDATA code transformation was proposed to allow the compiler to perform the vectorization automatically.

Fig. 3 illustrates the execution scheme for a single MPDATA time step according to the proposed methodology. Based on this methodology, we addressed [9] the challenge of performance portable programming by enabling the automatic parameterized transformation of the MPDATA code to achieve high sustained scalable performance for ccNUMA shared-memory systems. As a result, the adaptive MPDATA code follows along with a variety of hardware architectural issues such as memory hierarchy, multi- and manycore, threading, vectorization, and their interaction with the MPDATA code.

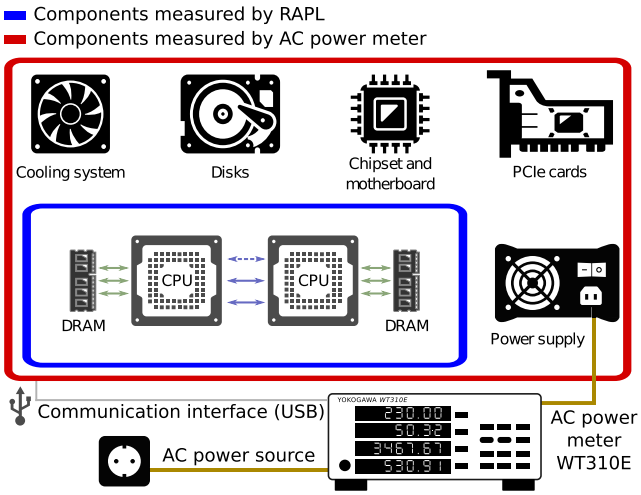


Fig. 4. Hardware- and software-based techniques used for power and energy measurements.

The complexity of the proposed hierarchical decomposition makes it impossible to implement the multithreading parallelization efficiently using OpenMP constructs such as `#pragma omp for`. Instead, we develop a proprietary scheduler responsible for the management of workload distribution and data parallelism [9]. Following the proposed adaptation, this scheduler explicitly defines the scope of work for each thread. Additionally, using the technique known [39] as the first-touch policy with parallel initialization, the memory is allocated *closest* to a physical core on which a given thread is executed. Moreover, the optimal policy for the OpenMP thread affinity interface is selected [9], to reduce the communication paths.

## 6 Methodology of Experiments

We benchmark four versions of the MPDATA application, all using the double precision floating-point format:

- *A* – basic, non-optimized implementation;
- *B* – code with (3+1)D decomposition of MPDATA domain;
- *C* – version B with partitioning cores into independent work teams;
- *D* – version C coupled with data-flow synchronization.

All versions are implemented with both disabled and enabled SIMD vectorization. Four different sizes of MPDATA domains are considered: $1024 \times 512 \times 64$, $1024 \times 512 \times 128$, $2048 \times 1024 \times 64$ and $2048 \times 1024 \times 128$. The Intel icpc compiler (v.19.0.1) is used with the optimization flag -O3 and properly chosen compiler arguments for enabling or disabling auto-vectorization process.

In this study, both hardware and software techniques are employed to measure the power and energy (Fig. 4). To retrieve the energy consumption provided by RAPL, we use a well-known package – PAPI [40], that enables accessing hardware performance counters available on most modern microprocessors. In particular, we utilize *Package* and *DRAM* domains to measure the energy consumed respectively by CPUs and the main memory when executing the MPDATA application.

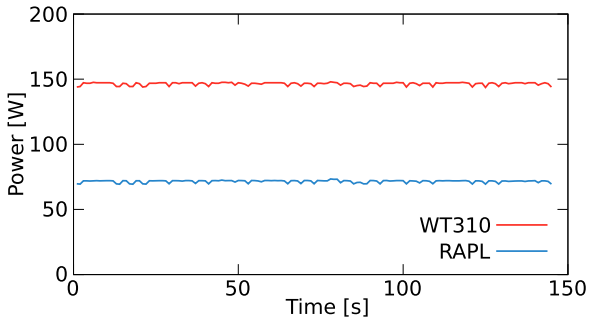


Fig. 5. Power in the idle state (without executing the application) of the test server (every sample is averaged over 1 second interval).

Besides RAPL/PAPI, the Yokogawa WT310 digital power meter is applied to obtain maximally accurate and reliable energy/power measurements for the whole platform. The maximum sampling speed of Yokogawa WT310 is 100k samples per second when the accuracy of measurement is kept at the level of 0.1 percent. This device is equipped with USB interface that enables access to power and energy consumption data, as well as provides management of the measurement process.

As shown in Fig. 4, the Yokogawa WT310 power meter is installed between the input power sockets of the server and the wall AC outlets. The Yokogawa WT310 device transfers the power to the server under the load and measures the power and energy consumed in real-time. This device collects the total power and energy of the platform. To get the measured data, we use the YokoTool command-line tool [41] written in Python, which operates via the serial USB interface. We confirmed that this tool is non-intrusive and does not have any noticeable influence on measurements.

Both the Python-based tool and PAPI interface allow us to start measurements while running the application and then finish them after completing computations. Thus, we provide the following three types of measurements during the application execution: execution time (seconds), average dynamic power (Watts), and total energy consumption (Joules). It is important to note that we perform measurements independently: one run for measuring the execution time, and others for the power and energy consumption that are also separately obtained by Yokogawa and PAPI. Each type of measurements is further repeated at least ten times, and the average values are used to get statistically sound results, with the relative standard deviation (RSD) less than 1 percent. 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.

Fig. 5 shows measurements of asz consumed by the server in the idle state, using both hardware and software power analysis techniques. According to Yokogawa WT310, the power required by the whole server is kept at the average of 147 Watts. RAPL gives the power of about 75 Watts less than the Yokogawa device keeping the power consumption at the average of 72 Watts. This difference is because RAPL determines the power consumed by CPUs and DRAM without considering other components of the platform, such as its cooling system (fans).

TABLE 2
Power/Energy and Performance Results for Various MPDATA Versions Obtained for the Domain of size 2028 x 1024 x 128

| Ver. | Time [s] | Total energy [kJ] | Sust. perf. Gflop/s | % of peak perf. | Avg. power [W] | Energy efficiency [Gflop/J] |
|---|---|---|---|---|---|---|
| **A** | 1055.4 | 489.18 | 52.15 | 2.53 | 460 | 0.11 |
| **B** | 413.5 | 172.92 | 149.62 | 7.26 | 420 | 0.36 |
| **C** | 124.5 | 67.63 | 498.25 | 24.18 | 540 | 0.92 |
| **D** | 99.9 | 59.13 | 620.69 | 30.12 | 590 | 1.05 |

## 7 Evaluation and Experimental Results

### 7.1 Energy/Power and Performance Comparison for Different MPDATA Versions

In the first stage of benchmarks, we evaluate the impact of performance optimization steps on the total energy consumption, which is measured by the Yokogawa power meter. We provide the energy/power and performance comparison for different MPDATA versions while utilizing all cores of CPUs with AVX-512 vector processing enabled. In these tests, the Intel Turbo Boost technology is also enabled to reach the highest possible performance gain.

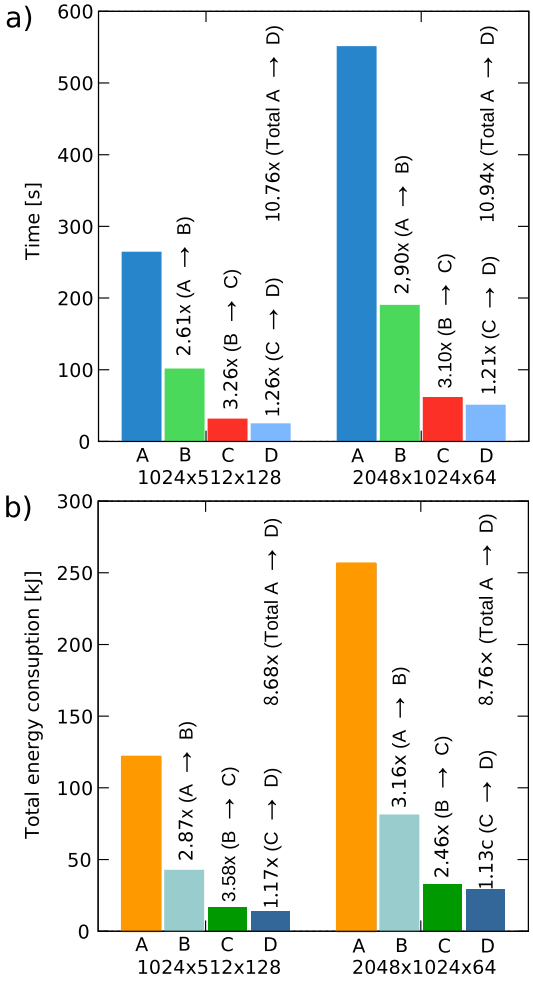


Fig. 6. Impact of optimization steps on reduction in a) execution time and b) total energy consumption, for various sizes of MPDATA domain.

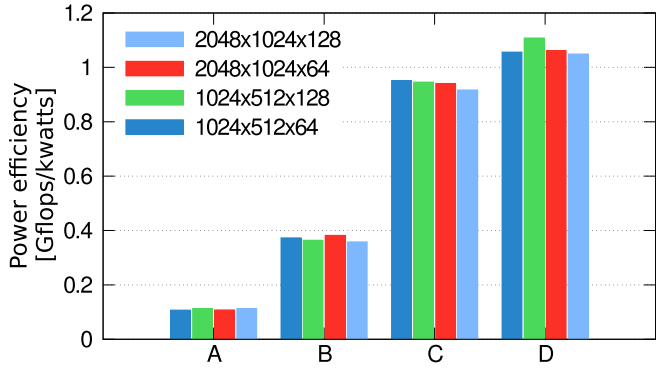


Fig. 7. Energy efficiency achieved for different MPDATA versions and various sizes of MPDATA domain.

Table 2 shows energy/power and performance comparison for different versions of MPDATA codes obtained with the domain of size 2028 x 1024 x 128. Besides the execution time, the total energy consumed by the server is shown for all MPDATA versions. Furthermore, the sustained performance (in Gflop/s) is presented, as well as the performance utilization rate with respect to the theoretical peak performance of the server. Table 2 reports also the average power consumed by the test platform, as well as the energy efficiency (in Gflop/J) of each MPDATA version, expressed as the ratio of the total number of float point operations to the total energy consumption.

Fig. 6 compares the impact of various optimization steps on the gain achieved for both performance and energy, considering different problem sizes. For each version **B**, **C**, and **D**, we also show the reduction in the execution time and energy consumption obtained with respect to the previous version. For example, the value of 3.26x shown for the version **C** means that this version allows decreasing the execution time 3.26 times against the version **B**.

The (3+1)D decomposition (version **B**) permits us to alleviate the memory and communication constraints by increasing both cache reusing and the data locality [9]. This version of MPDATA reduces the execution time for all tested MPDATA sizes, achieving the performance gain in the range from about 2.55x to about 2.9x against the basic code (version **A**). We observe that transformation of the version **A** into **B** also leads to reductions in the energy consumption - in the range from 2.82x to 3.16x. It is worth to mention that the energy consumption gains are slightly higher than those for the execution time – the difference is about 10 percent for a given problem size. Although the performance is increased from 52.15 to 149.62 Gflop/s, this code transformation still yields a relatively small energy efficiency (Fig. 7) at the level of 0.36-0.38 Gflop/J.

The next optimization step (version **C**) fits perfectly into multi-socket architectures [36], [42]. As shown in Fig. 6, the version **C** performs computation up to 3.32 times faster than the pure (3+1)D decomposition (version **B**). This optimization step reduces the energy consumption by a factor of about 2.5, which means a smaller benefit than that for the performance. As a result, the energy efficiency is improved to the level of about 0.95 Gflop/J.

The advantages of using the data flow synchronization (version **D**) – in contrast to the barrier approach – is the synchronization of only interdependent threads [37]. As a result,

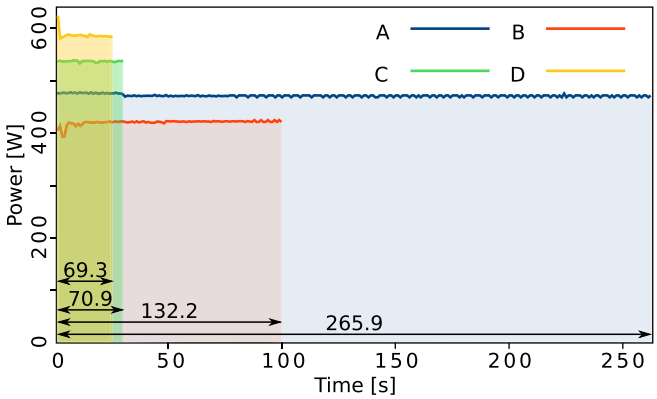


Fig. 8. Power trace for all MPDATA versions with domain of size 1024 x 512 x 128 and 1000 time steps (every sample is averaged over 1 second interval).

this version increases the performance by around 21 percent and reduces the energy consumption by around 15 percent, for all tested MPDATA sizes (Fig. 6). The energy efficiency increases up to 1.1 Gflop/J.

When comparing the version **D** – that involves all optimizations – with the basic version of MPDATA, the main conclusion is that the proposed methodology allows improving both performance and energy consumption radically. The analysis of results for all considered sizes of MPDATA domains show these improvements are in the range of 10.39-10.94 and 8.27-8.76 for performance and energy consumption, respectively.

We also analyze the power required by the platform during the execution of various MPDATA versions. Fig. 8 shows an example of the power trace for all MPDATA versions, corresponding to the domain of size 1024 x 512 x 128. Furthermore, Table 3 presents the average power consumed by the platform during tests.

As shown in Fig. 8, 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, various MPDATA versions correspond to different levels of power, although the AVX-512 capabilities are enabled for all of them.

To explain this behavior, we have to look at the usage of vector units in all MPDATA versions. In our previous work [9], we demonstrated that the version **D** is characterized by a more intensive AVX-512 workload than the version **C**, which uses AVX-512 more efficiently than the version **B**. It can be concluded that a more intensive (efficient) utilization

TABLE 3
Average Power and Average Frequency of the Server, Obtained for Various MPDATA Versions and Domain sizes

| Ver. | Avg. freq. $[GHz]$ | Average power [W] Size of domain 1024 x 512 x 64 | 2048 x 1024 x 64 | 2048 x 1024 x 128 |
|---|---|---|---|---|
| **A** | 2.42 | 474 | 465 | 463 |
| **B** | 2.29 | 433 | 427 | 418 |
| **C** | 2.29 | 528 | 539 | 543 |
| **D** | 2.29 | 563 | 580 | 592 |

TABLE 4
Performance/Energy Comparison of Scalar and Vectorized Versions of MPDATA, for Domain of size 1024 x 512 x 128 and 1000 Time Steps

| Ver. | Metric | Non-AVX | AVX-512 | Gain |
|---|---|---|---|---|
| **A** | Time $[s]$ | 265.9 | 263.8 | 1.01x |
| | Total energy $[kJ]$ | 150.5 | 121.7 | 1.24x |
| | Avg. power $[W]$ | 566 | 461 | 1.23x |
| | Avg. frequency $[GHz]$ | 3.19 | 2.42 | 1.32x |
| **B** | Time $[s]$ | 132.2 | 101.0 | 1.31x |
| | Total energy $[kJ]$ | 73.9 | 42.4 | 1.74x |
| | Avg. power $[W]$ | 559 | 420 | 1.33x |
| | Avg. frequency $[GHz]$ | 3.12 | 2.29 | 1.36x |
| **C** | Time $[s]$ | 70.9 | 31.0 | 2.29x |
| | Total energy $[kJ]$ | 40.1 | 16.4 | 2.45x |
| | Avg. power $[W]$ | 566 | 529 | 1.07x |
| | Avg. frequency $[GHz]$ | 2.99 | 2.29 | 1.31x |
| **D** | Time $[s]$ | 69.3 | 24.5 | 2.83x |
| | Total energy$[kJ]$ | 39.4 | 14.0 | 2.81x |
| | Avg. power $[W]$ | 568 | 572 | 0.99x |
| | Avg. frequency $[GHz]$ | 2.94 | 2.29 | 1.28x |

of vector units requires more power, but the computation are executed faster, so the overall energy consumption is reduced. Finally, the basic version **A** is characterized by the lowest intensity of usage of vector units as they have to wait for loading data from the main memory ([9]). However, in contrast to our previous observation, this version requires more power than the version **B**, despite a less efficient usage of vector units. This effect is explained by a significantly higher main memory traffic for the basic version as compared to the other versions ([9]). Increasing the power required by the basic version results also from a noticeably larger processor frequency during its execution (Table 3). For all tests performed with the versions **B**, **C** and **D**, the processor frequency does not exceed the maximum level of 2.3 GHz dedicated to heavy AVX-512 workloads. In contrast, while executing the version **A**, the thermal and power limitations of the platform permit reaching the processor frequency of up to 2.42 GHz, which also results in higher energy consumption.

### 7.2 Impact of SIMD Processing on Energy Consumption and Performance

The MPDATA code is characterized by vector-friendly data structures that enable us to easily switch each version of code to AVX 2.0, AVX-512, or even non-AVX (scalar) instruction set by selecting compiler arguments properly ([29]). This allows us to evaluate the impact of SIMD processing on energy consumption and performance for all MPDATA versions. In the evaluation, we use the following four metrics: (i) execution time, (ii) total energy consumption measured by the Yokogawa WT310 power meter, (iii) average processors frequency, and (iv) average power of the server. Table 4 presents a collection of all four metrics obtained for the MPDATA domain of size 1024 x 512 x 128 and 1000 time steps. The gain of vectorization is included as well, to show the advantages of AVX-512 for various MPDATA versions. Additionally, Fig. 9 illustrates the effect

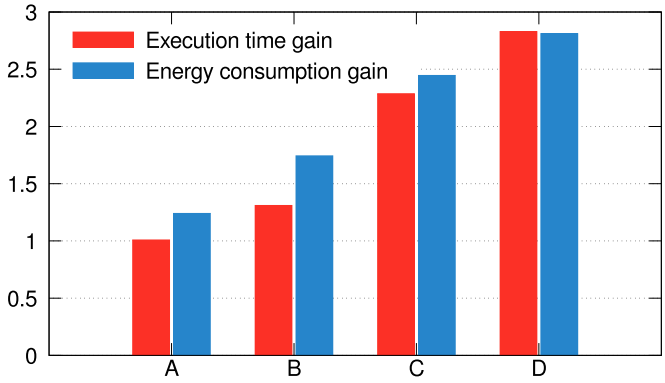


Fig. 9. Comparison of performance and energy gains for MPDATA domain of size 1024 x 512 x 128, achieved by enabling AVX-512.

of using the AVX-512 extension to improve the execution time and energy consumption.

As expected, the highest benefits of vectorization are achieved for the version **D**. In this case, the code with enabled SIMD allows improving both performance and energy consumption about 2.8 times, against its scalar counterpart with disabled vectorization. For the versions **B** and **C**, the vectorization also leads to the reduction in both execution time and energy consumption, but with fewer benefits.

In contrast, the execution times for the version **A** with enabled and disabled vectorization are almost the same. This happens because, as pointed out in Subsection 5.2, the performance of this version is strongly limited by the main memory bandwidth. In consequence, expanding the available computing power by utilizing AVX-512 does not result in performance improvements. At the same time, enabling vectorization allows reducing energy consumption by about 20 percent.

The key to understanding this behavior is the analysis of the processor frequency. It clocks down significantly when performing AVX-512 instructions [4]. In our tests, the clock frequency is reduced from 3.19 GHz to 2.42 GHz. Since the version **A** is not able to utilize vector units efficiently and the CPU frequency is decreased, the average power during the execution of the version **A** with enabled vectorization decreases by about 105 Watts (18.5 percent) compared to the scalar code (Table 4). Concluding, the same execution time and the reduced power requirement result in the lower energy consumption of the basic MPDATA version with enabled vectorization in comparison to its scalar counterpart.

TABLE 5
Execution Time, Total Energy Consumption $E_y$, and Average Power $P_y$ Measured by Yokogawa WT310 for Various CPU Frequencies (1000 Time Steps, Domain of Size 1024 x 512 x 128)

| Version A | | | | | | |
|---|---|---|---|---|---|---|
| **Frequency** | 1.0 | 1.4 | 1.8 | 2.0 | 2.2 | 2.42 |
| Time [s] | 266.5 | 264.9 | 264.4 | 263.8 | 263.7 | 263.8 |
| $E_y$ [kJ] | 94.7 | 99.5 | 109.3 | 111.7 | 115.7 | 121.7 |
| $P_y$ [W] | 356 | 376 | 413 | 424 | 439 | 461 |
| Version D | | | | | | |
| **Frequency** | 1.0 | 1.4 | 1.8 | 2.0 | 2.2 | 2.3 |
| Time [s] | 48.0 | 35.6 | 29.2 | 27.3 | 25.8 | 24.5 |
| $E_y$ [kJ] | 16.7 | 15.1 | 14.6 | 14.8 | 14.8 | 14.0 |
| $P_y$ [W] | 348 | 424 | 502 | 542 | 571 | 572 |

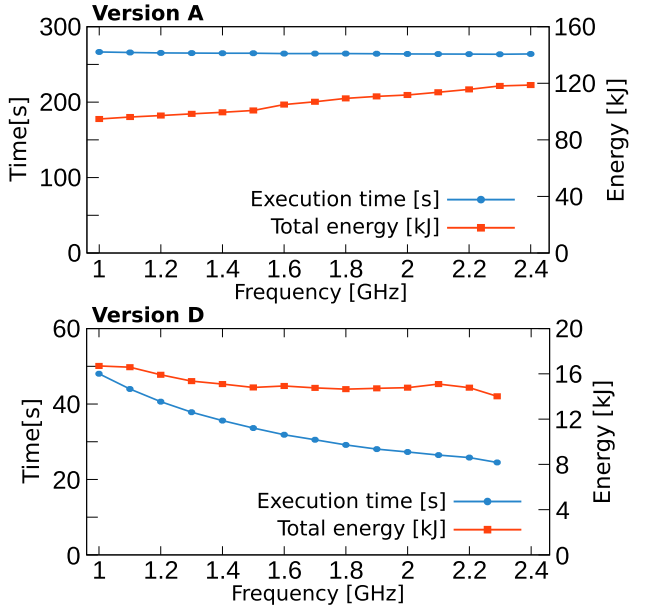


Fig. 10. Execution time and total energy consumption as functions of CPU frequency, measured for versions **A** and **D** with domain of size 1024 x 512 x 128 and 1000 time steps.

### 7.3 Impact of CPU Frequency Scaling on Energy Consumption and Performance

This work also explores the usage of the DVFS technique to optimize the energy efficiency of MPDATA by changing the CPU frequency. This technique is known [17] as an efficient method to save energy for memory-bound applications such as MPDATA, when CPU cycles are being wasted as they are stalled on memory [17]. We use the `CPUfreq` framework [18] to change the frequency of CPUs. 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 2.3 GHz for the versions **B**-**D** (Fig. 1), and 2.42 GHz while executing the version **A**.

The impact of CPU frequency scaling on the execution time, total energy consumption, and power of the test platform is depicted in Table 5. It shows results obtained for the the non-optimized version **A** and the most efficient version **D** (both versions with enabled vectorization). The presented values of the total energy consumption were measured by Yokogawa WT310, which monitors the entire platform. Figure 10 visualizes the monitored values of the total energy consumption and execution time as functions of the CPU frequency.

As shown in Table 5 and Fig. 10, the execution time of the version **A** is practically constant for all CPU frequencies, including even the lowest frequency of 1.0 GHz. The performance of this version depends primarily on memory speed; hence CPU frequency scaling does not affect on the execution time of MPDATA. Nevertheless, the non-optimized version gives us the best chance for reducing energy consumption by CPU frequency scaling. In fact, the execution of this version with the lowest clock frequency permits a maximum reduction in energy consumption, which is decreased by about

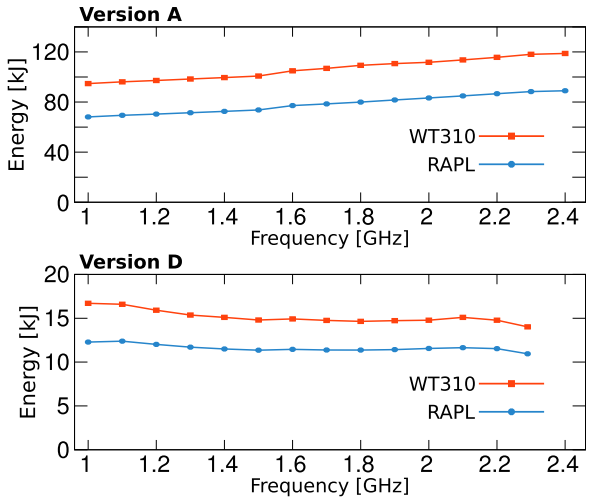


Fig. 11. Comparison of energy consumption [kJ] measured by Yokogawa WT310 and RAPL for versions **A** and **D**, assuming various clock frequencies (domain of size 2028 x 1024 x 128), 1000 time steps.

28 percent compared to the highest clock speed. This mainly results from reducing the power requirement. In fact, the average power is reduced by 105 Watts to 360 Watts while limiting the clock speed from 2.42 GHz to 1.0 GHz, with negligible performance losses not exceeding 1.5 percent.

In contrast, when running the optimized version **D**, reducing the CPU frequency does not lead to any positive effects not only in performance but also in energy. In this case, the optimal performance-energy trade-off corresponds to the highest CPU frequency. This trade-off gives not only the lowest energy consumption but also the highest performance. As shown in Table 5 and Fig. 10, the execution time of the version **D** decreases along with the frequency increase from the lowest to the highest clock speed, accelerating the MPDATA execution about twice and reducing the energy consumption by about 16 percent. The reason for such a little energy profit, in comparison to the performance advantage, is increasing the power requirement with increasing the CPU frequency, resulting in a more intensive utilization of computing resources. In fact, the average power is increased from 348 Watts by 224 Watts.

### 7.4 Comparison of Yokogawa Power Meter and RAPL

Our experiments are completed by evaluating the accuracy of energy measurements performed by RAPL compared to the Yokogawa WT310 meter. The RAPL infrastructure allows us to monitor the energy consumption of MPDATA using 32-bit hardware performance counters and I/O models [43]. RAPL

TABLE 6
Comparison of Energy Consumption Measurements for Yokogawa WT310 ($E_y$) and RAPL ($E_r$) With Different MPDATA versions, for Short-Time Simulations (Domain of Size 2028 x 1024 x 128, 1000 Time Steps)

| Version | A | B | C | D |
|---|---|---|---|---|
| **Time** $[s]$ | 1055.4 | 413.5 | 124.5 | 100.0 |
| $E_r$ $[kJ]$ | 366.8 | 129.3 | 50.9 | 44.4 |
| $E_y$ $[kJ]$ | 489.2 | 172.9 | 67.6 | 59.1 |
| **Ratio** $[\frac{E_r}{E_y}]$ | 1.33 | 1.34 | 1.33 | 1.33 |

TABLE 7
Comparison of Different Methods for Measurement of Energy Consumption Using RAPL ($E_r$) and Yokogawa WT310 ($E_y$) for Long-Time MPDATA Simulations

| **A single measurement for the entire workload** | | | |
|---|---|---|---|
| Time steps | $E_r$[kJ] | $E_y$[kJ] | Ratio $[E_r/E_y]$ |
| 400000 | 434 | 5782 | 13.33 |
| 800000 | 249 | 11619 | 46.57 |
| **Aggregated measurements wit packages of 500 time steps** | | | |
| Time steps | $E_r$[kJ] | $E_y$[kJ] | Ratio $[E_r/E_y]$ |
| 400000 | 4375 | 5782 | 1.32 |
| 800000 | 8754 | 11606 | 1.33 |

determines the energy consumption for CPUs and main memory [33] only. The other components of the platform, such as the cooling system and power supply units, are excluded from RAPL measurements. In contrast, the Yokogawa WT310 power meter provides energy/power measurements for the entire platform.

An exhaustive series of tests is performed with all MPDATA versions, considering the energy consumption and power for a wide range of problem sizes, different numbers of time steps, and a variety of CPU frequencies. The examples of energy consumption profiles measured by Yokogawa and RAPL are presented in Fig. 11 and Table 6.

The main conclusion is that RAPL profiles follow the same patterns as those of Yokogawa WT310, for most of the data points. RAPL energy and power measurements typically match results of the Yokogawa meter. We report the energy consumption measured by Yokogawa WT310 as about 1.3 times higher than results obtained by RAPL. The differences are observed because RAPL provides the energy consumption of CPU and DRAM without considering other components of servers such as the cooling system. The behavior of the latter depends on the temperature of components of a server (such as CPUs). In consequence, the energy consumption of the platform also changes with the change in the speed of fans. The higher the temperature of CPUs, the higher the fan speed and the higher the energy consumption required to cool down the server.

At the same time, we observe high discrepancies in energy/power measurements when running long-time simulations, that execute MPDATA for many thousands of time steps (see the upper part of Table 7). For explaining and solving this problem, we have to look at the accuracy of energy measurements performed by RAPL. This accuracy depends on the accuracy of model-specific registers (MSR) of a given type of CPU [28], [43]. Typically, MSR registers collect measurements on 32 bits, so applications with long execution times can overflow these registers, invalidating measurements for long-time simulations. For iterative applications, this drawback can be easily avoided by an aggregated method consisting of: (i) monitoring energy/power separately for each package with a correctly selected number of time steps, and (ii) deriving the final result as an aggregated sum of measurements obtained separately for all packages.

In our work, we experimentally select 500 times steps as the size of each package, ensuring accurate measurements of power and energy. Our tests show also that the aggregated method does not introduce any significant accuracy

overheads. This result is illustrated in the lower part of Table 7, which corresponds to measuring the total energy consumption using both Yokogawa WT310 and RAPL while executing long-time MPDATA simulations with the number of steps from 400 000 to 800 000.

## 8 Conclusion

The energy/power consumption is one of the main limiting factors for exascale systems [12]. Some supercomputers today are consuming upwards of 15MW (https://top500.org). Because of the scale factor of large cluster installations, even improvement of energy efficiency on a single node can yield significant savings in the total system energy/power.

In this paper, we expose the energy/power consumption of a single node based on a dual-socket Intel Xeon Platinum 8180 processors, using both hardware and software power analysis techniques. Exploiting both Yokogawa WT310 digital power meter and RAPL, we can perform reliable flopper-Joule measurements of the energy efficiency.

This study focuses on verifying the energy and performance efficiency of a set of parametric optimizations recently proposed in [9] for a real-world CFD application called MPDATA. We cover in detail how the proposed combination of optimization steps affects the energy consumption of this memory-bound iterative code. The presented benchmarks reveal that these optimizations improve the efficiency of MPDATA simulations radically, reducing both the execution time and energy consumption. For the considered sizes of MPDATA domains, these improvements are in the range of 10.39-10.94 and 8.27-8.76 for performance and energy consumption, respectively.

Then, we discover the impact of SIMD vectorization on the energy consumption and performance, as well as how enabling vectorization affects clock frequencies of CPUs and their power requirements. As shown in this study, the Intel AVX-512 extension gives us a strong possibility to improve not only performance but also energy consumption of the MPDATA application. In particular, the conducted tests reveal that enabling vectorization can lead to a reduction in energy, even when the memory performance constraints limit the performance of a parallel code.

Going forward, we investigate the usability of CPU frequency scaling as a tool for balancing energy savings with admissible performance losses. It is discovered that for the most-optimized version of MPDATA, reducing the frequency does not permit any improvements neither in performance nor energy. Finally, we carefully evaluate RAPL measurements against the Yokogawa WT310. These two approaches match each other for short-time simulations, but in the case of long-time simulation, we reveal high inconsistencies resulting from overflowing 32-bit MSR registers that collect measurements performed with RAPL. However, these inconsistencies may be successfully overcome by developing the aggregated method for RAPL measurements.

Our further works include an extension of the experimental comparison proposed in this paper over a wide range of current and emerging architectures, as well as across other applications. In particular, we are planning to explore the newest Cascade Lake Intel Xeon processors and the new generation of AMD Rome EPYC processors.

## Acknowledgments

This work was supported by the National Science Centre, Poland under Grants no. UMO-2017/26/D/ST6/00687 and no. UMO-2015/17/D/ST6/04059, as well as by the project financed within the program of the Polish Minister of Science and Higher Education under the name "Regional Initiative of Excellence" in the years 2019–2022 (project no. 020/RID/2018/19, the amount of financing 12 000 000 PLN). The authors were grateful to Intel Technology Poland for granting access to HPC platforms.

## References

[1] K. Bergman *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO)*, Tech. Rep, vol. 15, 2008.

[2] J. V. Ferreira Lima, I. Raïs, L. Lefèvre, and T. Gautier, "Performance and energy analysis of OpenMP runtime systems with dense linear algebra algorithms," *Int. J. High Perform. Comput. Appl.*, vol. 33, pp. 431–443, 2018.

[3] R. Montella *et al.*, "Accelerating Linux and Android applications on low-power devices through remote GPGPU offloading," *Concurrency Comput., Practice Experience*, vol. 29, no. 24, 2017, Art. no. e4286.

[4] "Intel Xeon Processor Scalable Family Specification," Feb. 2018. [Online]. Available: https://www.intel.com/content/dam/www/public/us/en/documents/specification-updates/xeon-scalable-spec-update.pdf, Intel

[5] K. Rojek, A. Ilic, R. Wyrzykowski, and L. Sousa, "Energy-aware mechanism for stencil-based MPDATA algorithm with constraints," *Concurrency Comput., Practice Experience*, vol. 29, no. 8, 2017, Art. no. e4016.

[6] P. Smolarkiewicz, "Multidimensional positive definite advection transport algorithm: An overview," *Int. J. Numer. Meth. Fluids*, vol. 50, no. 10, pp. 1123–1144, 2006.

[7] P. Smolarkiewicz and P. Charbonneau, "EULAG, a computational model for multiscale flows: An MHD extension," *J. Comput. Phys.*, vol. 236, pp. 608–623, 2013.

[8] T. Rauber, G. Rünger, and M. Stachowski, "Performance and energy metrics for multi-threaded applications on DVFS processors," *Sustain. Comput., Inform. Syst.*, vol. 17, pp. 55 –68, 2018.

[9] 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.

[10] K. Khan, M. Hirki, T. Niemi, J. Nurminen, and Z. Ou, "RAPL in action: Experiences in using RAPL for power measurements," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, pp. 9:1–9:26, 2018.

[11] "WT300 Series Digital Power Meter Analyzer," 2018. [Online]. Available: https://tmi.yokogawa.com

[12] J. Jeffers, J. Reinders, and A. Sodani, *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. San Mateo, CA, USA: Morgan Kaufmann, 2016.

[13] P. Ezzatti, E. S. Quintana-Ortí, A. Remón, and J. Saak, "Power-aware computing," *Concurrency Comput., Practice Experience*, vol. 31, no. 6, 2019, Art. no. e5034.

[14] 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, 2017.

[15] Y. Wang, D. Nörtershäuser, S. Le Masson, and J.-M. Menaud, "An empirical study of power characterization approaches for servers," in *Proc. 9th Int. Conf. Smart Grids Green Commun. IT Energy-Aware Technol.*, 2019, pp. 1–5.

[16] H. H. Hassan, A. S. Moussa, and I. Farag, "Performance versus power and energy consumption: Impact of coding style and compiler," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 12, pp. 132–142, 2017.

[17] J. Haj-Yahya, A. Mendelson, Y. B. Asher, and A. Chattopadhyay, *Energy Efficient High Performance Processors: Recent Approaches for Designing Green High Performance Computing*. Berlin, Germany: Springer, 2018.

[18] 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 Electron. Appl.*, vol. 8, no. 2, 2018, Art. no. 18.
[19] D. Cesarini, A. Bartolini, and L. Benini, "Benefits in relaxing the power capping constraint," in *Proc. 1st Workshop Autotuning Adaptivity Approaches Energy Efficient HPC Syst.*, 2017, pp. 1–6.
[20] A. Haidar, H. Jagode, P. Vaccaro, A. YarKhan, S. Tomov, and J. Dongarra, "Investigating power capping toward energy-efficient scientific applications," *Concurrency Comput., Practice Experience*, vol. 31, no. 6, 2019, Art. no. e4485.
[21] M. Wlotzka *et al.*, "Energy-aware high performance computing," in *ICT - Energy Concepts for Energy Efficiency and Sustainability*, G. Fagas, L. Gammaitoni, J. P. Gallagher, and D. J. Paul, Eds. Rijeka, Croatia: IntechOpen, 2017.
[22] A. A. Jerraya and W. Wolf, "Hardware/software interface codesign for embedded systems," *Computer*, vol. 38, no. 2, pp. 63–69, 2005.
[23] T. Jakobs and G. Runger, "Examining energy efficiency of vectorization techniques using a Gaussian elimination," in *Proc. IEEE Int. Conf. High Perform. Comput. Simul.*, 2018, pp. 268–275.
[24] H. Lien, L. Natvig, A. Al Hasib, and J. C. Meyer, "Case studies of multi-core energy efficiency in task based programs," in *ICT as Key Technology against Global Warming*, A. Auweter, D. Kranzlmüller, A. Tahamtan, and A. M. Tjoa, Eds. Berlin, Germany: Springer, 2012, pp. 44–54.
[25] K. Rojek, "Machine learning method for energy reduction by utilizing dynamic mixed precision on GPU-based supercomputers," *Concurrency Computation: Practice Experience*, vol. 31, no. 6, 2019. Art. no. e4644.
[26] H. Khaleghzadeh, M. Fahad, A. Shahid, R. Manumachu, and A. Lastovetsky, "Bi-objective optimisation of data-parallel applications on heterogeneous platforms for performance and energy via workload distribution," *arXiv: 1907.04080*, 2019.
[27] J. J. Escobar, J. Ortega, A. F. Díaz, J. González, and M. Damas, "Energy-aware load balancing of parallel evolutionary algorithms with heavy fitness functions in heterogeneous CPU-GPU architectures," *Concurrency Comput., Practice Experience*, vol. 31, no. 6, 2019, Art. no. e4688.
[28] "Intel 64 and IA-32 architectures optimization reference manual," April 2018. https://software.intel.com
[29] A. Eltablawy and A. Vladimirov, "Capabilities of Intel AVX-512 in Intel Xeon Scalable Processors (Skylake)," Colfax International, 2015.
[30] 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, PP. 1–42, 2019.
[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] J. Rico-Gallego, A. Lastovetsky, and J. Diaz-Martin, "Model-based estimation of the communication cost of hybrid data-parallel applications on heterogeneous clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 11, pp. 3215–3228, Nov. 2017.
[33] A. Haidar, H. Jagode, P. Vaccaro, A. Yarkhan, S. Tomov, and J. Dongarra, "Investigating power capping toward energy-efficient scientific applications," *Concurrency Comput., Practice Experience*, vol. 31, 2018, Art. no. e4485.
[34] P. Smolarkiewicz and L. Margolin, "MPDATA: A finite-difference solver for geophysical flows," *J. Comput. Phys.*, vol. 140, no. 2, pp. 459–480, 1998.
[35] 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.
[36] 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 2018.
[37] 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.
[38] L. Szustak, K. Rojek, T. Olas, L. Kuczynski, K. Halbiniak, and P. Gepner, "Adaptation of MPDATA heterogeneous stencil computation to Intel Xeon Phi coprocessor," *Scientific Programming*, vol. 2015. 2015, Art. no. 10.
[39] D. Unat *et al.*, "Programming Abstractions for Data Locality," 2014. [Online]. Available:http://web.eecs.umich.edu/akamil/papers/padal14report.pdf
[40] "PAPI. Performance Application Programming Interface 5.7.0," 2019. [Online]. Available: https://icl.utk.edu/papi/overview/index.html
[41] *WT310, WT310HC, WT332, WT333 Digital Power Meter: Communication Interface*, 3rd ed., Yokogawa Meters & Instruments Corporation, January 2016.
[42] L. Szustak, R. Wyrzykowski, and O. Jakl, "Islands-of-cores approach for harnessing SMP/NUMA architectures in heterogeneous stencil computations," in *Proc. 14th Int. Conf. Parallel Comput. Technol.*, 2017, vol. 10421, pp. 351–364.
[43] C. B. Navarrete, C. Guillen, W. Hesse, and M. Brehm, "Autotuning the energy consumption," in *Proc. Int. Conf. Parallel Comput.*, 2013, vol. 25, pp. 668–677.

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

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

**Tomasz Olas** received the MSc degree in computer science from the Czestochowa University of Technology, Poland, in 1999, and the PhD degree in computer science for 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.

**Valeria Mele** received the MSc. and PhD degrees in computational science, from the University of Naples Federico II, Italy, where she is a researcher. She has attended the Argonne Training Program on Extreme-Scale Computing and worked for many months with the Argonne National Laboratory. Her research activity is mainly focused on development and performance evaluation of parallel algorithms and software for heterogeneous, hybrid, and multilevel parallel architectures.