Future Generation Computer Systems 177 (2026) 108237

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Editorial

Evaluating AMD EPYC CPU architectures on CFD applications

Marcin Lawenda [a,*], Łukasz Szustak [b], László Környei [c], Flavio Cesar Cunha Galeazzo [d], Paweł Bratek [b]

[a] Poznan Supercomputing and Networking Center, Jana Pawła II 10, Poznań, 61-139, Poland
[b] Czestochowa University of Technology, Dąbrowskiego 69, Częstochowa, 42-201, Poland
[c] Széchenyi István University, Department of Mathematics and Computational Sciences, Egyetem tér 1., Győr, 9026, Hungary
[d] High Performance Computing Center Stuttgart (HLRS), University of Stuttgart, Nobelstraße 19, Stuttgart, 70569, Germany

ARTICLE INFO

Keywords:
New CPU architectures
AMD EPYC processors
Cache size
NUMA
Profiling
Optimization
Memory-bound applications
Task parallelism
Data parallelism
HPC
Co-design

ABSTRACT

In this work, the authors focus on assessing the impact of the AMD EPYC processor architecture on the performance of CFD applications. Several generations of architectures were analyzed, such as Rome, Milan, Milan X, Genoa, Genoa X and Bergamo, characterized by a different number of cores (64-128), L3 cache size (256 - 1152 MB) and RAM type (8-channel DDR4 or 12-channel DDR5). The research was conducted based on the OpenFOAM application using two memory-bound models: motorBike and Urban Air Pollution. In order to compare the performance of applications on different architectures, the FVOPS (Finite VOlumes solved Per Second) metric was introduced, which allows a direct comparison of the performance on the different architectures. It was noticed that local maximum performance occurs at different values of grid element per CPU when utilizing different processor types. Additionally, the behaviour of the models was analyzed in detail using the AMD μProf and LIKWID software profiling analysis tools to reveal the applications' interaction with the hardware. It enabled fine-tuned monitoring of the CPU's behaviours and identified potential inefficiencies in AMD EPYC CPUs. Particular attention was paid to the effective use of L2 and L3 cache memory in the context of their capacity and the bandwidth of memory channels, which are a key factor in memory-bound applications. Processor features were analyzed from a cross-platform perspective, which allowed for the determination of metrics of particular importance in terms of their impact on the performance achieved by CFD applications.

1. Introduction

A comprehensive understanding of the technological advancements shaping modern HPC systems is crucial for developing efficient applications that fully utilize the hardware capabilities. One of the fundamental elements of the HPC architecture is the central processing unit (CPU), which undergoes significant modifications with succeeding generations. In order to adapt the operation of the application to the architecture (co-design) and take advantage of the opportunities it offers, it is essential to understand the impact of the application data flow on performance considering different problem sizes.

The article presents the implementation of cutting-edge HPC solutions, focusing on AMD technologies relevant to pilot applications, which are also compared to previous processor generations to demonstrate performance advancement. It can be noted that processors are eagerly used in the HPC domain, offering a large number of cores, performance and appropriate memory subsystems with large cache capacity. As an example, we can cite the strongest systems in Europe [1] offered by the EuroHPC Joint Undertaking [2] to the scientific and commercial community.

The AMD EPYC 9004 series processors represent the 4th generation of AMD EPYC server-class processors. The design of this generation features the AMD Zen 4 microarchitecture of compute cores, the AVX-512 instruction set, large cache memory, and high memory bandwidth to meet the needs of HPC applications. AMD EPYC 9004 series processors offer a variety of configurations with varying numbers of cores, TDPs, frequencies, and cache sizes.

Simulation tools are commonly used to assess the course of phenomena observed in the surrounding environment, e.g. how air, pollutants, smoke or heat spread in spaces with complex geometry. One of the most popular applications used for this purpose is CFD (Computational Fluid Dynamics). CFD is an intensively developing branch of computational sciences that allows the creation of simulations of fluid-flow phenomena based on the laws governing the movement of fluids. By using dedicated data structures combined with numerical analysis, it is possible to solve problems related to fluid flows. The calculations simulate the flow of the

* Corresponding author.
E-mail addresses: lawenda@man.poznan.pl (M. Lawenda), lukasz.szustak@pcz.pl (Ł. Szustak), laszlo.kornyei@math.sze.hu (L. Környei), flavio.galeazzo@hlrs.de (F.C.C. Galeazzo), pawel.bratek@pcz.pl (P. Bratek).

https://doi.org/10.1016/j.future.2025.108237
Received 21 May 2024; Received in revised form 27 October 2025; Accepted 1 November 2025
Available online 8 November 2025

fluid (liquid and gas) and its interaction with the surfaces defined by the boundary conditions. One of the leading CFD applications is OpenFOAM [3], which is an open-source framework for the solution of Partial Differential Equations (PDE) using the Finite Volume Method (FVM). The work deals with two use cases based on OpenFOAM framework: motorBike, a well-known reference application, and Urban Air Pollution, implemented by Széchenyi István Egyetem-University, simulating air flow and pollution dispersion in an urban area. This choice of applications was consciously made to assess the impact of processing performance from the perspective of two different models and, therefore, slightly different data traffic requirements, which allow for comparative analysis.

The increasing complexity of models and the need to obtain the highest possible accuracy of results result in an increased demand for computing power and its efficient use. In addition to the traditional algorithmic optimization process, the process of increasing software capabilities is also carried out through a co-design procedure, where the software is adapted to better use the available infrastructure features.

This work focuses on benchmarking of new-generation AMD EPYC processors compared to prior generations. In particular, we explore a series of top-of-the-line AMD EPYC CPUs based on Rome, Milan, Milan X, Genoa, Genoa X, and Bergamo architectures (AMD EPYC 7002, 7003 and 9004 series processors). To provide a clear research context, the paper includes a detailed explanation of the application architecture of the new processors and the testing platforms. Special attention is given to the impact of large-cache systems on the parallel performance essential for CFD computational kernels.

Since the work aims to present the features of different generations of processors, a general overview of comparative activities is presented from the perspective of a single node for HPC infrastructures. This made it possible to omit the impact of other factors (e.g. inter-node communication) on the application performance and to highlight the features of systems based on AMD processors (e.g. cache size) along with the findings regarding their profiling results.

The paper is structured as follows. Section 2 provides information on related work in assessing the impact and optimization of cache utilization on application performance. Chapter 3 specifies in detail the infrastructure used for research, focusing on memory-related aspects. The next chapter (4) introduces the application aspect, both in terms of the tools used in tests (compilers, measuring cache parameters) and the description of the framework used to run simulation models. Chapter 5 presents models constituting the basis for the research, characterized by various workflows (cache use): motorBike and Urban Air Pollution. Sections 6 and 7 (respectively for motorBike and UAP) present information on the benchmarking results of different CPU architectures in the context of different data sizes reflected by selected CPU metrics along with performance evaluation. The Chapter 8 analyses the wide range of performance metrics from a cross-platform perspective with reference to the FVOPS metric. The Chapter 9 elaborates a memory-intensive scheme for fully implicit method. The last, 10th Chapter summarizes the research conducted and the results obtained.

2. Related works

The CFD simulation imposes the need to process large amounts of data resulting from the size and density of the computational mesh. Both attributes are important for the modelled phenomenon and the accuracy of the results [4].

Connecting the values achieved in the individual grid nodes with the values in neighbouring nodes implies the need for quick access to the input data necessary to determine the values under the assumed boundary conditions [5].

The effective level of cache usage is influenced by two fundamental aspects: its size (the ability to store a certain amount of data) and the data reading prediction algorithm (data availability on request). This translates into the achieved values on read success (cache hit) or read fail (cache miss), which in turn affects the processing efficiency. Achieving high computational performance requires placing data as close to the computational kernels as possible, ideally by performing efficient prefetching targeting specific cache levels based on dedicated strategies [6].

Table 1
Specification of target platforms.

Platform	CPU	Memory
GIGABYTE H262-Z63-00	2 × AMD EPYC 7742 (Rome)	512GB DDR4-3200
GIGABYTE H262-Z63-00	2 × AMD EPYC 7763 (Milan)	512GB DDR4-3200
Asus RS720A E11 RS12	2 × AMD EPYC 7773X (Milan X)	512GB DDR4-3200
AMD Titanite 4G	2 × AMD EPYC 9554 (Genoa)	768GB DDR5-4800
AMD Titanite 4G	2 × AMD EPYC 9654 (Genoa)	768GB DDR5-4800
AMD Titanite 4G	2 × AMD EPYC 9684X (Genoa X)	768GB DDR5-4800
AMD Titanite 4G	2 × AMD EPYC 9754 (Bergamo)	768GB DDR5-4800

The implementation of HPC nodes with a significant number of cores, on the one hand, allows for running many processes simultaneously and, on the other hand, imposes requirements on the design of collective MPI operations at the intra-node level. One of the elements that should be taken into account is the cache coherence protocol and its impact on the performance of the designed distributed application algorithms. This topic is widely discussed in scientific literature, proving its importance. Some examples are provided below to illustrate.

In a study [7] the authors discuss a scenario where the performance of MPI transmission, based on shared memory, is degraded due to the cache coherence protocol and the multi-socket configuration of the tested platforms. To address this issue, an innovative approach is proposed to reduce the delay of broadcast operations to 1.5 times and implement them in a modified version of the MPI transmission that supports cache coherence.

In another work [8], the impact of new generation of non-uniform memory access (NUMA) on the performance of social sciences applications is elaborated. Performance analysis of an application simulating the structure of a user interaction graph revealed communication limitations that impacted its overall performance. The solution was to implement an improved method of distributing data to ccNUMA domains, which reduced intra-processor data traffic and enabled more efficient use of available cores.

Another approach to optimizing workload in CFD application (ADflow), based on the cache-aware improvements on modern vector processors is presented in the work [9]. It involves dividing computational blocks into smaller, fixed-size blocks that are small enough to completely fit into the available cache size for each core in a given architecture. This enabled higher cache reusing and thus accelerated the basic CFD solver procedures by about 3.27 times when using modern vector instruction sets.

3. Architecture of AMD EPYC CPUs

This work explores a series of top-of-the-line AMD EPYC CPUs with various architectures. In particular, we use seven dual-socket platforms, including four 64-core CPUs with Rome, Milan, Milan X, and Genoa architectures, two 96-core CPUs with Genoa and Genoa X processors, as well as one 128-core CPU based on Bergamo processors. The specifications of these systems are presented in Table 1.

Table 2 summarizes the parameters of AMD EPYC processors family. The studied 2nd, 3rd, and 4th generation of AMD EPYC include the

Table 2
Specification of AMD EPYC CPUs.

CPU type	7742	7763	7773X	9554	9654	9684X	9754
Launch Date	2019	2021	2021	2022	2022	2023	2023
Codename	Rome	Milan	Milan X	Genoa	Genoa	Genoa X	Bergamo
Cores	64	64	64	64	96	96	128
Freq. (Turbo)	2.25 (3.4*)	2.45 (3.5*)	2.20 (3.5*)	3.10 (3.75)	2.40 (3.55)	2.55 (3.42)	2.25 (3.1)
L2 [MB]	0.5	0.5	0.5	1	1	1	1
CCX	4-core	8-core	8-core	8-core	8-core	8-core	8-core
L3 CCX [MB]	16	32	96	32	96	32	16
CCXs in CCD	2×4-core	1×8-core	1×8-core	1×8-core	1×8-core	1×8-core	2×8-core
L3 CCD [MB]	32	32	96	32	32	96	32
CCDs	8	8	8	8	12	12	8
Total L3 [MB]	256	256	768	256	384	1152	256
L3 per core [MB]	4	4	12	4	4	12	2
Memory	8-channel	8-channel	8-channel	12-channel	12-channel	12-channel	12-channel
type	DDR4-3200	DDR4-3200	DDR4-3200	DDR5-4800	DDR5-4800	DDR5-4800	DDR5-4800
NUMAs	4	4	4	4	4	4	4

*maximum frequency achievable by single core

different numbers of cores and are clocked by the base frequency clocks listed in Table 2. The CPU designs of all the platforms feature the out-of-order execution model and support the frequency boost technology, where the maximum turbo frequency depends on the type and intensity of workload and the number of utilized cores. The simultaneous multi-threading (SMT) is turned off for all the systems.

The underlined Rome, Milan, and Milan X chips offer 64-core CPUs and support eight-channel DDR4-3200 main memory per socket. In today's 4th generation of AMD EPYC processors, we address our work to explore 64-, 96- and 128-core processors that incorporate twelve DDR5-4800 memory channels within every socket [10]. Fig. 1 illustrates the block diagram of the 4th generation of AMD EPYC Genoa model with 96 cores. All the platforms represent a group of the ccNUMA shared memory architectures [11,12] combining whole memory regions using 2×4 NUMA domains for dual-socket platforms (4 domains per socket). A single processor uses four NUMA domains with separate quadrants and, in consequence, interleaves memory regions across the eight and twelve memory channels for the previous (2nd and 3rd) and current 4th generations of the AMD EPYC family, respectively.

The most crucial innovation in AMD EPYC processors is the hybrid multi-die architecture first introduced in 2nd generation of EPYC processors [13]. The design of all CPUs consists of a single central I/O hub (or I/O Die) [14] through which all CPU components communicate. Excluding the Bergamo-based CPUs, the tested AMD EPYC CPUs use a collection of 8-core chiplets, called Core Complex Dies (CCDs), connected to the I/O Die through dedicated high-speed Infinity Fabric links. A single processor with 64-core Rome, Milan, Milan X, or Genoa chips consists of eight CCDs, while every 96-core Genoa or Genoa X processor offers configurations with twelve CCDs per socket (see Fig. 1). In contrast, every 128-core Bergamo CPU provides eight CCDs with 16 cores per die.

The AMD EPYC processors family typically offers 32 MB of L3 cache per CCD. It results in a total L3 cache capacity of 256 MB for 8 CCDs AMD chips on 64-core Rome, Milan, and Genoa as well as 128-core Bergamo. The 96-core AMD chip contains 12 CCDs and features 384 MB of aggregated L3 cache size. In processors with AMD 3D V-Cache technology (Milan X and Genoa X), the per-die L3 cache is augmented three times (see Fig. 2). Applying the 3D V-Cache technology enables extending the shared 32 MB L3 cache with 64 MB additional layered above, bringing the per-die total L3 cache to 96 MB. This innovation delivers a large aggregated L3 cache size that reaches 768 MB for 8 CCDs on a 64-core Milan X chip and 1152 MB for 12 CCDs on a 96-core Genoa X chip. A single CCD of AMD EPYC CPUs typically consists of one CCX with eight cores and provides its own 32 MB of L3 cache (96 MB when enabling AMD 3D V-Cache technology). In contrast, every Rome CCD consists of two core complexes (CCXs); each embraces four cores and 16 MB of L3 cache, while the CCD in Bergamo features two core complexes with eight cores each and 16 MB of L3. In contrast, every CCD in Bergamo or Rome combines two core complexes with eight cores each. Each core complex includes the 16 MB shared L3 cache (32 MB per CCD). The new AMD EPYC 9004 series processors have 1MB of L2 cache per core, which is twice as large as prior generations (Rome, Milan, and Milan X).

Table 3
Meshes for motorBike test.

Name	Cell count	Avg. cell count per		
		128 cores	192 cores	256 cores
xsmall	167,555	1309	873	655
small	355,474	2777	1851	1389
smid	603,547	4715	3143	2358
mid	1,897,187	14,822	9881	7411
high	4,166,441	32,550	21,700	16,275
mhigh	6,657,491	52,012	34,674	26,006
uhigh	11,900,363	92,972	61,981	46,486
xhigh	39,400,357	307,815	205,210	153,908

4. Benchmark applications overview

4.1. The motorBike use case

The motorBike use case is one of OpenFOAM's tutorials, that has been with the distribution for a long time. It simulates a motorbike in a wind tunnel and calculates turbulent flow around the vehicle. It showcases the use of snappyHexMesh, OpenFOAM's own unstructured mesh generation tool, and the simpleFoam solver, which solves the steady-state Reynolds-averaged Navier-Stokes (RANS) equations for an incompressible fluid. The motorBike use case was chosen for benchmarking and analysis in this report due to its frequent investigation in assessing OpenFOAM performance and its similarities to the UAP-FOAM solver, working with unstructured meshes and solving the RANS equations for incompressible fluid.

The simulation models the external air flow around a motorbike with a pilot in a volume of $10 \times 8 \times 8$ meters. Boundary conditions for bike and ground are modelled as no-slip walls, inlet air speed head-on is set at 20 m/s, while side and top boundaries are modelled as slip walls. Meshes of several cell counts for the same geometry are generated using OpenFOAM's snappyHexMesh tool, resulting in mesh sizes from 36k to 14M (see Table 3). All meshes were pregenerated to ensure identical results across all platforms.

The parameters of the simulations are mostly as of in the tutorial use case of OpenFOAM com version v2112. The consistent Semi-Implicit Method for Pressure Linked Equations (SIMPLEC) is used to solve the Reynolds-averaged Navier-Stokes equations for incompressible fluids with k-ω-SST turbulence model. A few adjustments were made to fa-

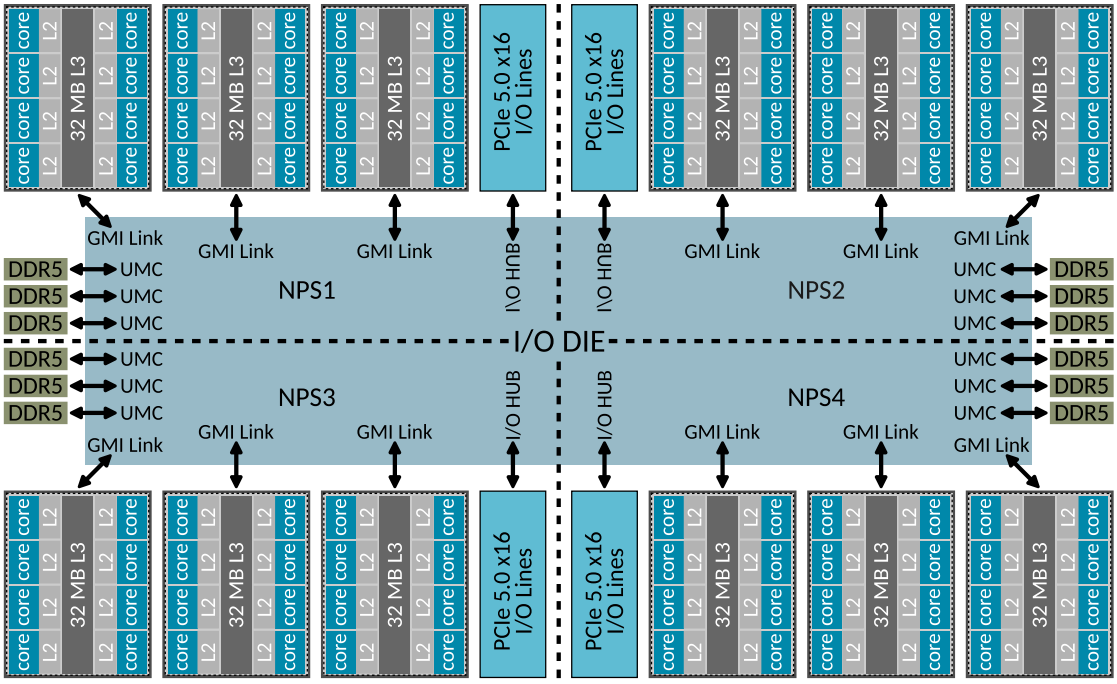

Fig. 1. AMD EPYC Genoa model with 96 cores, 12 CCDs and a central IOD.

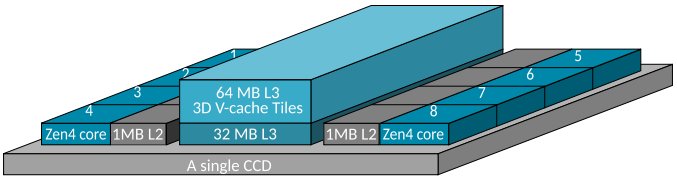

Fig. 2. Layout of CCD with AMD 3D V-cache enabled.

cilitate benchmarking and improve stability. The number of iterations is lowered to 200. The scotch library is used for decomposition [15]. The number of decomposed subdomains was equal to the number of available CPU cores on the underlying architecture. Decomposed data was stored in collated format. The number of non-orthogonal correctors was set to one. The underrelaxation factor for velocity was set to 0.7, while for k and ω to 0.3, and for pressure to 0.3.

4.2. The urban air project use case

The Urban Air Project (UAP) use case is one of the pilots of the HiDALGO2 project [16]. The CFD module simulates air flow and pollution dispersion in an urban area. In the current paper, the OpenFOAM-based implementation of the CFD module is benchmarked. The UAP workflow gathers data, measurements and simulation results with regard to urban geometry, climate conditions, pollution emission and background, which serve as geometry, boundary conditions and source terms for the CFD model. Two solvers of OpenFOAM, simpleFoam and pimpleFoam [17] are utilized to model steady-state and time-dependent behaviour of pollution spread within the city, respectively. In the frame of the current paper only the steady state part is benchmarked using simpleFoam, so results are comparable with the motorBike use case.

The geometry of the use case models external air flow around a geometry resembling the urban area of the Hungarian city of Győr. Boundary conditions use atmospheric wall functions with a roughness factor of $z_0 = 0.4$ for ground and building, slip for top, and a custom inlet-outlet condition for the side walls that set time and height based values obtained from ECMWF weather service interface, polytope [18]. In the

Table 4
Meshes for UAP test.

Name	Cell count	Avg. cell count per 128 cores	192 cores	256 cores
uxlow	36,248	283	189	142
ulow	139,937	1093	729	547
mlow	228,263	1783	1189	892
low	728,162	5689	3793	2844
mid	3,227,275	25,213	16,809	12,607
high	14,332,247	111,971	74,647	55,985

model, pollution is calculated with the COPERT model [19] from traffic data obtained by simulation. Meshes used for these benchmarks are octree based and are generated using the in-house SZE tool octreemesher. All meshes use the same geometry, albeit at different resolutions, and are listed in Table 4. All input data including mesh, weather boundary and pollution source are precalculated and present in files for the benchmark. In this use case the RANS equations for incompressible fluids are solved with k-ϵ turbulence model. The solver simpleFoam is used, which implements the Semi-Implicit Method for Pressure Linked Equations (SIMPLE). Additionally, the RANS equations are coupled with convection-diffusion equations to model pollution spread. A total of 600 iterations are run, except for high mesh size, where 400 iterations are run, and runtime is adjusted afterwards. The Generalized Geometric-Algebraic Multigrid (GAMG) solver with Gauss-Seidel smoother is used to solve the pressure equations. A sample visualization of the results in 3D is shown on Fig. 3.

5. Experiments methodology

In our experiments, all OpenFOAM kernels are compiled with the AOCC compiler [20] and linked against the MPI library pre-installed by platform vendors. We use OpenMPI v4.1 with an open-source optimized communication library UCX (Unified Communication X) for all experiments. The UCX library is configured with the support of xpmem (eXtreme Performance Memory, v2.3) shared memory transports for intra-node communication.

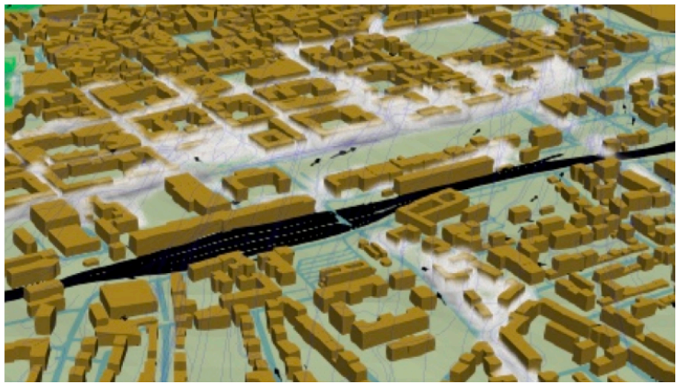

Fig. 3. Visualization of pollution dispersion in the Urban Air Project use case modeling the city of Győr. Streamlines and vectors indicate air flow direction, while white cloud shows pollution spread.

Table 5
Range of clock speed [GHz] measured during benchmark.

Rome	2.8 - 3.1
Milan	2.9 - 3.2
Milan X	2.8 - 3.0
64-core Genoa	≈3.7
96-core Genoa	3.4 - 3.5
96-core Genoa X	3.2 - 3.4
128-core Bergamo	2.8 - 3.1

The AOCC compiler (v.4.1.0) is used with the optimization flag `-O3` and architecture-specific compiler arguments that enable the auto-vectorization process, including `-march=znver2` for Rome, `-march=znver3` for Milan and Milan X, as well as `-march=znver4` for Genoa, Genoa X and Bergamo. To evaluate the impact of the SIMD processing on performance, the underlined kernels are also compiled with `-fno-vectorize` and `-fno-slp-vectorize` compiler arguments that disable auto-vectorization process. However, no performance profit is observed for SIMD processing in any of the benchmarks, including motorBike and UAP use cases. Since target applications use unstructured meshes, the irregularity of elements scattered in memory complicates the SIMD processing that relies on data being stored in contiguous memory locations.

We investigate motorBike and UAP application models on seven platforms by testing different problem sizes. The execution time is measured by extracting the time stamps written by OpenFOAM as "Execution Time" and subtracting the first value from the last value. This way time consumed by the initialization is discarded, albeit the runtime of one less iteration is measured. To ensure the reliability of measurements, every test is repeated 5 times and all measurements are reported and illustrated. All the systems operate with SMT disabled and turbo boost enabled. We observe that the thermal and power limitations of the test platforms allow them to operate at the frequency clock close to the maximum turbo boost speed (Table 5).

To compare the performance between the various systems, the metric FVOPS (Finite VOlumes solved Per Second) is introduced [21]. The FVOPS metric is calculated as follows:

$$FVOPS = \frac{cellcount}{runtimeperiterationortimestep}$$

The value of FVOPS depends on a series of factors, including the simulation type, boundary conditions, and especially the grid size being solved. The results reported here use test cases with fixed conditions in all systems, only varying the grid size. To facilitate the comparison the amount of grid elements per core is the usual metric and is used in the X-axis. This metric often reveals local maxima, which indicates the optimal number of grid points per core per test case and architecture. It is interesting to note that these local maxima occur at different values of grid element per rank when utilizing different processor types.

In addition, we employ AMD μProf (ver. 4.1) software profiling analysis tool [22] to reveal the applications' interaction with the hardware. The AMD μProf performance tool analyses and monitors AMD Zen-based microarchitecture processors. In this work, we use a specific tool called AMDuProfPcm [22], which allows fine-tuned monitoring of the CPU's behaviours and identifies potential inefficiencies in AMD EPYC CPUs. This system analysis utility periodically collects the CPU, core, cache, and memory performance event count values and reports various metrics. In this stage of our work, we select a set of metric groups that includes: *ipc*, *l1*, *l2*, *l3*, *tlb*, *dc* and *pipeline_util*. We distinguish metrics related to all L1, L2 and L3 levels of cache (access/hit/miss), core utilization, CPI (cycle per instruction), IPC (instruction per cycle), CPU pipeline, and others (see full specification in [22]). The selected groups of metrics enable us to indicate the impact of hardware features on performance efficiency. Furthermore, we investigate the MPI trace analysis to estimate the volume of the MPI data traffic between MPI processes. To achieve this goal we use the AMD μProf MPI Light-Weight Tracing tool [22].

The performed analysis is expanded by the LIKWID software profiling tool [23]. It helps us to estimate the aggregated bandwidth of L2 and L3 caches when using `likwid-perfctr`. However, the used version 5.3 of LIKWID does not support the Bergamo-based architecture.

6. The motorBike use case benchmarking

6.1. Platforms comparison

Fig. 4 presents the execution time measurements for the motorBike obtained for seven computing platforms (Table 1) and eight mesh sizes (see Table 3). These results are grouped by the mesh size on different plots. Besides the execution time, this figure also reveals the performance comparison between the selected computing platforms. Notably, if noted, every plot outlines the performance gains for a given platform compared to the prior generation and the highest performance gain obtained between platforms with the best and the worst results.

Table 6 delivers comprehensive performance comparisons across all platforms and different meshes. This table collates the range of speedups that indicate minimum and maximum performance differences for a fixed platform presented in the first row of the table compared to other systems outlined in the left column. Investigating the results, the platform with Rome CPUs exhibits the lowest performance for all mesh sizes except for small mesh sizes, where performance with Milan X is tied (see Fig. 4(b)). The improvement of regular Milan over Rome is about a factor of up to 1.2x for smaller mesh sizes, which diminishes to 1.07x for large ones. Similar to Rome, Milan also uplifts negligible performance over Milan X for the smaller meshes. The advantage of Milan X compared to regular Milan and Rome is strongly noticeable for mid, high, mhigh, uhigh, and xhigh mesh sizes. In this case, the best performance profit equals 1.54x and 1.69x faster than Milan and Rome, respectively, and is observed for the mhigh mesh size. At the same time, the Milan X processors do not improve performance analyzing the remaining smaller meshes.

Considering the AMD EPYC 9004 family (Genoa, Genoa X and Bergamo), we observe significant performance improvement over all previous architectures across all mesh sizes. As shown in Table 6, the system with 64-core Genoa processors performs all the tests faster, accelerating computation about 1.63x-2.22x, 1.35x-2.07x, and 1.24x-1.70x than platforms with Rome, Milan, and Milan X CPUs, respectively. We also identify that 64-core Genoa provides a performance advantage over the 96-core version, accelerating computation by about 1.3x faster for small mesh size. Considering other meshes in this comparison, the larger core count in 96-core Genoa brings a relatively negligible improvement compared to the 64-core version, improving performance in the range of 1.1x to 1.31x.

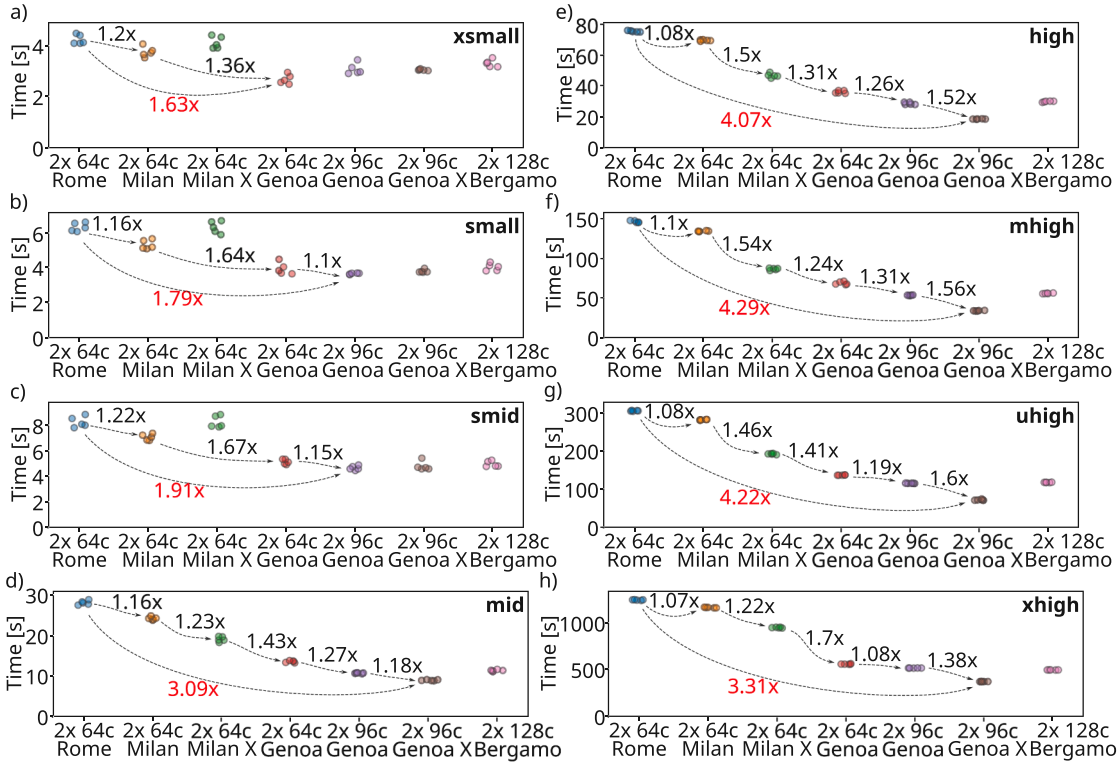

Fig. 4. Performance results obtained for motorBike on a variety of AMD CPUs and different mesh sizes.

Table 6
Aggregated performance gains of motorBike for different problem sizes and on a variety of AMD EPYC CPUs.

	Rome	Milan	Milan X	64-core Genoa	96-core Genoa	Genoa X	Bergamo
Rome	–	1.07x-1.22x	0.97x-1.69x	1.63x-2.22x	1.45x-2.75x	1.49x-4.29x	1.36x-2.62x
Milan	0.82x-0.94x	–	0.81x-1.54x	1.35x-2.07x	1.21x-2.50x	1.24x-3.91x	1.14x-2.39x
Milan X	0.59x-1.04x	0.65x-1.24x	–	1.24x-1.70x	1.50x-1.92x	1.54x-2.67x	1.41x-1.91x
64-core Genoa	0.45x-0.61x	0.48x-0.74x	0.59x-0.81x	–	0.89x-1.31x	0.91x-2.05x	0.84x-1.25x
96-core Genoa	0.36x-0.69x	0.40x-0.83x	0.52x-0.67x	0.76x-1.13x	–	0.98x-1.60x	0.89x-1.04x
Genoa X	0.23x-0.67x	0.26x-0.81x	0.37x-0.65x	0.49x-1.10x	0.62x-1.02x	–	0.61x-0.92x
Bergamo	0.38x-0.73x	0.42x-0.88x	0.52x-0.71x	0.80x-1.20x	0.96x-1.12x	1.09x-1.64x	–

Table 7
Platforms with the best results for motorBike.

Platform	Meshes
2x 64-core Genoa	small
2x 96-core Genoa	small and smid
2x 96-core Genoa X	mid, high, mhigh, high, and xhigh

As shown in Table 6 and Fig. 4, the platform with Genoa X CPUs offers significant performance gain over other systems for mid, high, mhigh, high, and xhigh mesh sizes. This platform overcomes the prior generation, including Rome, Milan, and Milan X CPUs, accelerating computation up to 4.29x, 3.91x, and 2.67x, respectively. This is also worth noting that – similar to Milan X – the new Genoa X model is not faster on smaller mesh sizes over regular Genoa processors. We also reveal that the 128-core Bergamo results are similar to those of the Genoa CPUs but with negligible performance drops over the 96-core model. Table 7 demonstrates the computing platforms that achieve the best results during tests performed for a given mesh size. As expected, the highest performance advantages are noticeable on platforms with Genoa X CPUs for relatively larger mesh sizes. The regular Genoa CPUs, including 64- and 96-core models, bring the best performance improvements for the smaller ones.

6.2. Performance evaluation

We propose using the FVOPS performance metric to compare performance across multiple platforms. This metric helps us estimate computing efficiency for different mesh sizes, indicating better platform utilization for the higher FVOPS level.

Fig. 5 reveals the calculated FVOPS for all platforms and mesh sizes. For every architecture, the calculation performance in FVOPS is plotted versus the per-core cell count.

FVOPS shows a similar trend for all architectures. It increases up to a turning point and decreases afterwards. The best platform utilization is noted for about 4715 cells per core (smid mesh) for Rome and Milan, 14,821 (mid mesh) for Milan X and 64-core Genoa, 7410 (mid mesh) for Bergamo, 9881 (mid mesh) for 96-core Genoa and 21,700 (high mesh) for Genoa X. For small cell count meshes, up to 5000, architectures before Genoa exhibit lower performance than 64-core Genoa. Differences between Rome, Milan and Milan X are not that dominant. Furthermore, the 96-core Genoa and Genoa X with Bergamo show similar performance at a given cell per core value, while outperforming 64-core Genoa. For larger mesh sizes, enabled 3D V-cache technology (Milan X and Genoa X) architectures dominate their own platform. Milan X shows an advantage in the 15k to 100k cell per core range over regular Milan, while Genoa X has an advantage in the 10k to 200k cell per core range over regular Genoa. This makes both products a viable choice for the cell count range of 2M to 13M and 1M to 20M, respec-

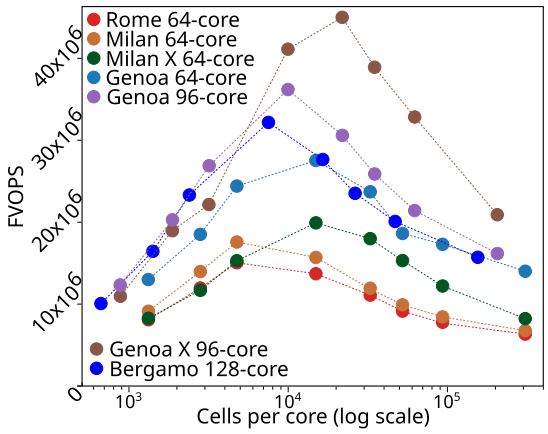

Fig. 5. FVOPS performance metric for motorBike.

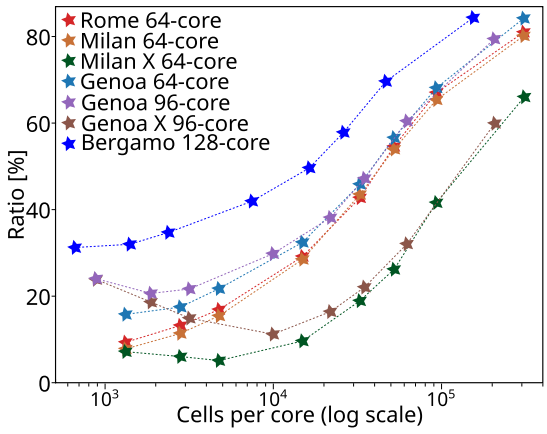

Fig. 6. L3 miss ratio for motorBike.

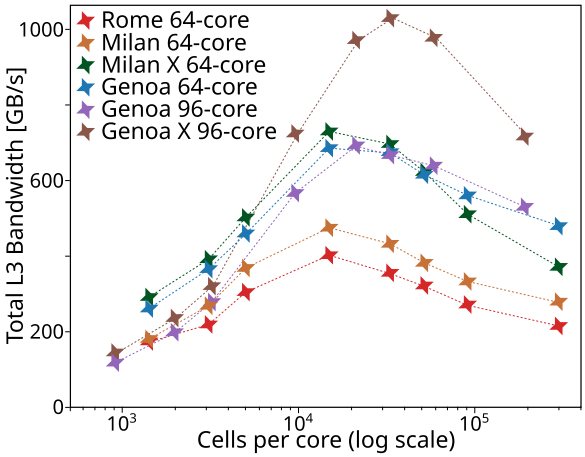

Fig. 7. Aggregated bandwidth of L3 cache for motorBike.

tively. Bergamo shows no advantage on any of the investigated mesh sizes.

To explain the FVOPS results, a closer look at the hardware specification and performance evaluation is required. Hence, in this stage of our work, we perform performance analysis focusing on the impact of L2 and L3 cache capacity on performance and MPI trace analysis. To achieve this goal and collect profiling data we combine the AMD μProf and LIKWID software profiling tools. We start performance evaluation of the impact of L3 cache capacity on performance. Fig. 6 illustrates the L3 cache miss ratio obtained for different meshes and a variety of computing systems by using the AMD μProf profiling tool. Furthermore, this analysis is expanded by using the likwid-perfctr profiling tool to estimate the aggregated bandwidth of L3 cache for motorBike on all tested platforms and all meshes (Fig. 7).

Considering 64-core-based platforms, Fig. 6 shows that the L3 cache miss ratio follows a similar trend for systems with 64-core Rome, Milan, and Genoa CPUs. In this case, the L3 miss rate starts at around 10 %-15 % for the smaller mesh and consecutively grows to 80 % for the larger ones. These three platforms offer 256 MB of L3 for a single processor, thus featuring convergent L3 miss trend. In contrast, the Milan X – which offers 768 MB of L3 cache capacity per CPU – reduces the L3 miss rate for larger mesh sizes. It is mainly noticeable for sizes mid, high, mhigh, and high, where up to 30 percentage points reduce the L3 miss rate over other 64-core CPUs.

When comparing Milan X with regular Milan and Rome, this results in a noticeable increase in performance mainly for larger mesh sizes (mid, high, mhigh, uhigh, and xhigh). However, such a performance advantage decreases when the data volume requirement exceeds the L3 capacity in Milan X, reducing improvement over regular Milan from 1.54x for mhigh to 1.24x for xhigh mesh sizes. We should also note that the large L3 cache in Milan X does not overcome regular Milan or Rome CPUs for relatively smaller sizes where the L3 cache capacity is not a critical performance bottleneck. In this case, the smaller clock frequency of Milan X causes slightly worse performance in comparison to Milan and Rome. Considering platforms with 64-core CPUs, the 4th generation of AMD EPYC processors, thanks to the newest DDR5 and larger 2x 12-channel memory subsystem, offers higher attainable performance than the prior generation Rome, Milan, and even Milan X.

Comparing 96-core Genoa, 96-core Genoa X, and 128-core Bergamo CPUs, the best L3 cache miss ratio trend is notable for the platform with 3D V-cache technology (Genoa X) compared with regular Genoa and Bergamo-based solutions. The large L3 capacity in Genoa X CPUs features a smaller L3 cache miss rate of up to 28 and 37 percentage points over Genoa and Bergamo-based platforms, respectively. It results in noticeable performance profits over other platforms for sizes mid, high, mhigh, uhigh, and xhigh. Like Milan X, the performance advantage of Genoa X is not noticeable for smaller sizes where the L3 capacity is not the performance bottleneck.

It should be emphasized that the platform based in Bergamo exhibits the most unfavorable trend in L3 cache miss ratio among all platforms examined. This is mainly due to the smaller L3 cache size per core compared to other processors (see Table 2). Considering the memory-bound nature of the motorBike, as expected, the L3 cache miss penalties significantly impact performance for the 2x 128 cores of the Bergamo-based platform that do not bring performance advantage over Genoa processors. We also observe that L3 misses occur mainly because the data volume required to transfer through the cache is larger than the total cache capacity. Additionally, for the platforms with Genoa X processors and for rather smaller sizes, we expect that L3 inevitable misses are noticeable where the first time a memory location is read.

As shown in Fig. 7, the aggregated bandwidth of L3 cache features a similar trend for all architectures: it increases to a turning point and decreases afterward. We also observe that the obtained trendlines of the aggregated bandwidth of L3 fit the curves of the FVOPS performance metric (see Fig. 5). Considering relatively larger mesh sizes where the L3 cache size affects performance limits, the measured bandwidth of L3 and FVOPS decreases. This is because the total amount of data actively used by the software is greater than the size of the cache. Consequently, the overall performance is limited by the data traffic data through the L3 cache and main memory. Thus, both L3 cache size and the speed of the main memory subsystem play a key role in attainable performance.

Furthermore, the decreasing trend of the aggregated bandwidth of L3 and FVOPS is also observed considering smaller and smaller mesh sizes. The key to understanding this behaviour is the analysis of the MPI data

Table 8
Total volume of MPI data traffic and the Approximation of MPI Communication Percentage (ACP) for the execution time obtained for motorBike on two-socket platforms with 64-, 96-, and 128-core based CPUs.

Mesh (cells)	MPI data traffic volume [GB]			ACP
	64-core	96-core	128-core	[%]
xsmall (167555)	60.3	73.4	84.1	70
small (355474)	101.9	122.0	136.9	62
smid (603547)	145.7	172.7	195.9	57
mid (1897187)	322.3	384.1	424.6	42
high (4166441)	436.0	519.3	591.4	26
mhigh (6657491)	621.6	737.4	836.8	22
uhigh (11900363)	693.1	855.3	993.6	18
xhigh (39400357)	1276.8	1559.1	1786.0	11

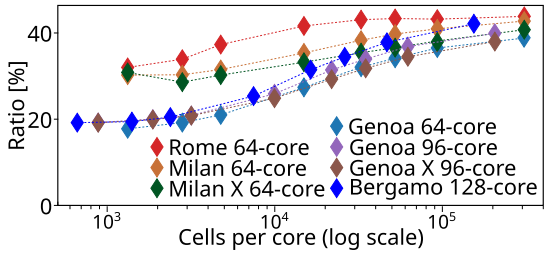

Fig. 8. Total L2 miss ratio for motorBike.

traffic between MPI processes. To reach this goal, we use the AMD μProf MPI Light-Weight Tracing tool [22] to estimate the volume of the MPI data traffic between MPI processes. Table 8 presents the total volume of MPI data traffic established for all platforms and different meshes. The processed analysis shows that, for a given mesh, all 64-core-based platforms (Rome, Milan, Milan X, and Genoa) exchange the same data volume between MPI processes. However, as expected, the volume of data traffic rises when using platforms with 96-core or 128-core CPUs.

Table 8 reveals a comparison between the rise in the number of cells and the increase in MPI data traffic volume for subsequent meshes. This table also indicates the Approximation of the MPI communication costs as a Percent of runtime (ACP), defined as the average value of all platforms for a given mesh size. We note a greater growth in cell count than in MPI data volume required to exchange, comparing subsequent mesh sizes. The computation-to-data-movement ratio, in terms of the number of cells per data traffic volume, is increasingly growing for larger and larger meshes. As a result, platforms process less computation per GB of data moved for smaller meshes than for larger ones. The ACP parameter indicates a larger communication percentage of runtime for smaller meshes than for larger ones. Considering smaller sizes that fit better in the cache, high data movement cost with low computation intensity results in a drop in the FVOPS, as resources are spent on MPI data traffic instead of cell processing. Consequently, MPI data traffic is overtaking computation as the most dominant cost. In addition, we observe that approximately 30 % of the ACP is associated with non-blocking send/receive operations for smaller meshes, increasing to about 70 % for larger ones. In contrast, the cost of global reduction operations starts at around 50 % of the ACP and decreases to roughly 20 % as the mesh size increases.

The new AMD EPYC 9004 series processors have 1MB of L2 cache per core, which is twice as large as prior generations. As shown in Fig. 8, it enables the reduction of the L2 cache miss rates and, hence, performance advantages compared to prior generations. In particular, the total L2 miss rate of Genoa and Bergamo CPUs is reduced by up to 12-14 percentage points over the prior generation of CPUs for smaller mesh sizes (Fig. 8). Considering larger meshes, the total L2 miss trend becomes tied for all platforms with negligible advantage for both Genoa and Geno X in comparison to Bergamo CPUs.

In addition, Fig. 9 illustrates the aggregated bandwidth of L2 cache for motorBike obtained using the likwid-perfctr tool. Generally, the observed trendlines of this profiling metric match the FVOPS curves (see Fig. 5). It demonstrates the benefits of the new L2 cache offered by the AMD EPYC 9004 series processor over prior generations. In particular, the major disadvantage is noticeable for the Milan and Rome processors compared to other platforms. In contrast, the Milan X platform reduces the advantage of the new L2 cache of the AMD EPYC 9004 series processor thanks larger L3 cache that prompts access to required data by L2.

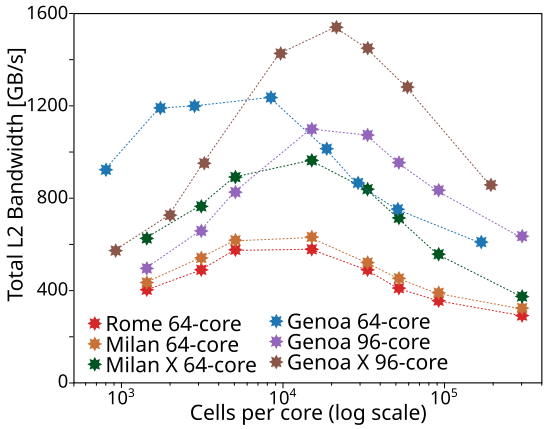

Fig. 9. Aggregated bandwidth of L2 cache for motorBike.

As shown in Fig. 9, considering smaller mesh sizes, the platform with 64-core Genoa features the highest bandwidth of the L2 cache overcoming even 96-core Genoa and Genoa X CPUs. Consequently, we note that 96-core Genoa and Genoa X processors do not offer a performance advantage over 64-core Genoa processors when processing smaller meshes (see Fig. 4(a–c)). This is also affected by the fact that, as shown in Table 11, applying more computing units (cores) uplifts communication overhead (MPI data traffic volume). Consequently, the rise of communication overheads when using more cores offset the performance benefits of using more compute units which is mainly noticeable for smaller meshes. Following larger meshes, the platform with 96-core Genoa X processors achieves the highest bandwidth of the L2 cache (Fig. 9), which together with the advantage of large L3 affects the best performance (see Fig. 4(d–h)).

7. Urban air pollution use case benchmarking

7.1. Platforms comparison

The performance results obtained for the UAP on different computing platforms are presented in Fig. 10. This figure shows execution time measurements grouped by mesh size on various plots. When observed, we also demonstrate the performance improvements identified on a specific platform in comparison to the previous generation, as well as the greatest performance differences achieved across platforms with the best and worst results. Arrows indicate a larger than one speedup, including the actual speedup value.

Additionally, the comprehensive performance report is delivered in Table 9, indicating performance comparisons across all platforms and different meshes. This table collates the range of performance differences determined for a fixed platform presented in the first row of the table in comparison to other systems outlined in the left column.

As expected, the slowest execution time measurements are obtained for the platform with Rome CPUs. The regular Milan CPUs accelerate computation over Rome by around 1.07x-1.18x faster. For smaller meshes, up to the mlow size, we do not observe any performance gains for the system with Milan X CPUs compared to the other platforms, due

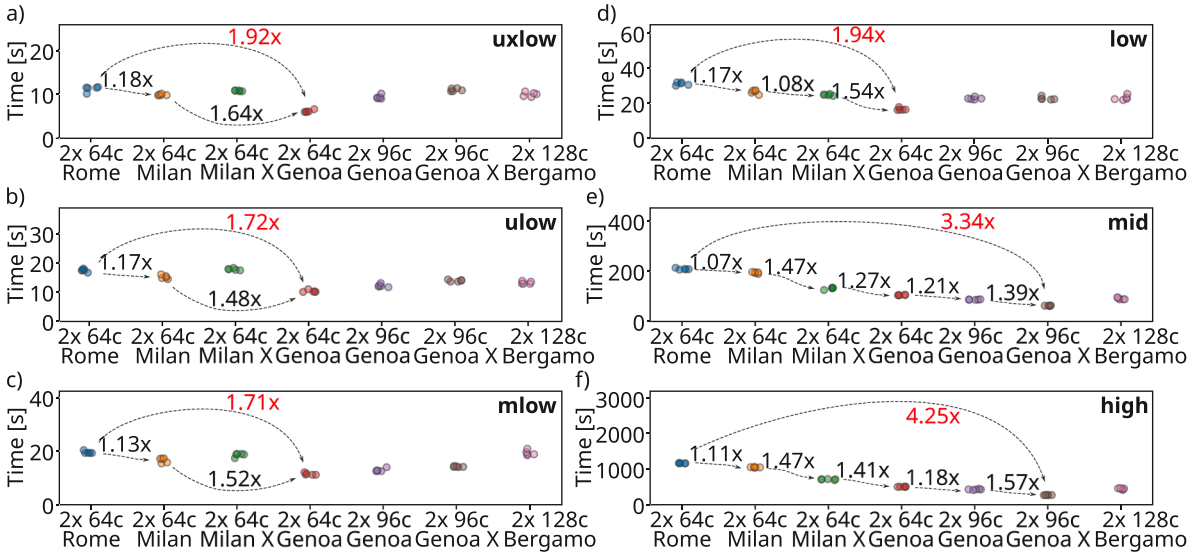

Fig. 10. Performance results obtained for UAP on a variety of AMD CPUs and different mesh sizes.

Table 9
Aggregated performance gains of UAP for different problem sizes and on a variety of AMD EPYC CPUs.

	Rome	Milan	Milan X	64-core Genoa	96-core Genoa	Genoa X	Bergamo
Rome	–	1.07x-1.18x	0.98x-1.63x	1.72x-2.30x	1.24x-2.71x	1.05x-4.25x	1.02x-2.54x
Milan	0.85x-0.94x	–	0.84x-1.47x	1.48x-2.08x	1.06x-2.44x	0.90x-3.83x	0.90x-2.29x
Milan X	0.61x-1.02x	0.68x-1.19x	–	1.27x-1.80x	1.10x-1.66x	0.99x-2.61x	0.99x-1.56x
64-core Genoa	0.43x-0.58x	0.48x-0.68x	0.55x-0.79x	–	0.65x-1.21x	0.55x-1.85x	0.59x-1.18x
96-core Genoa	0.37x-0.80x	0.41x-0.94x	0.60x-0.91x	0.83x-1.54x	–	0.85x-1.57x	0.67x-1.00x
Genoa X	0.24x-0.95x	0.26x-1.11x	0.38x-1.01x	0.54x-1.82x	0.64x-1.18x	–	0.60x-1.09x
Bergamo	0.39x-0.98x	0.44x-1.11x	0.64x-1.01x	0.85x-1.69x	1.00x-1.49x	0.92x-1.67x	–

Table 10
Platforms with the best results for UAP.

Platform	Meshes
2x 64-core Genoa	uxlow, ulow, mlow, and low
2x 96-core Genoa X	mid and high

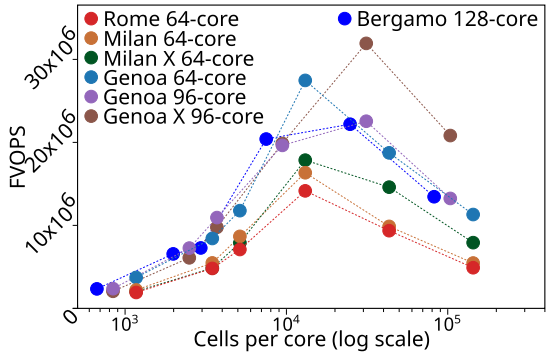

Fig. 11. FVOPS performance metric for UAP.

to their slightly lower clock frequency. In contrast, a noticeable improvement is observed for larger meshes, including low, midm, and high, with a speedup factor of up to 1.47x and 1.63x over regular Milan and Rome, respectively.

As shown in Fig. 10 and Table 9, the 64-core Genoa delivers the best performance for smaller meshes up to the low size over both the prior generation of CPUs as well as 96-core Genoa, Genoa X, and Bergamo CPUs. The highest performance leap is observed for uxlow size, where the 64-core Genoa CPUS accelerates computation with a speedup factor of 2.30x, 2.08x, and 1.80x faster over Rome, Milan, and Milan X. In this case, we also indicate performance gains of about 1.54x, 1.82x, and 1.69x faster than 96-core Genoa, Genoa X, and Bergamo CPUs, respectively. We also reveal that 96-core Genoa and Genoa X do not provide the extra performance seen for smaller meshes, and usually perform worse or similar results in comparison to 64-core CPUs.

In contrast, considering mid and high mesh sizes, the 96-core Genoa provides a circa 20 % benefit over the 64-core Genoa. Furthermore, the high cache 96-core Genoa X shows similar behaviour with a performance improvement of 39 % and 57 % over the 96-core Genoa for mid and high mesh sizes. In this case, Genoa X also features noticeable performance leaps with a speedup factor of up to 1.85x than 64-core Genoa CPUs. Finally, at neither of the mesh sizes does Bergamo deliver the best performance. The list of the best computing platforms that feature the best results for a given mesh size is outlined in Table 10. We reveal that the platform with 64-Genoa CPUs offers the best results when processing relatively small problem sizes. The large L3 capacity of Genoa X CPUs brings the highest performance advantages for larger mesh sizes.

7.2. Performance evaluation

As with the motorBike benchmark, we use the FVOPS metric to indicate computing efficiency for different mesh sizes across multiple platforms. Fig. 11 shows FVOPS for the UAP benchmark.

As shown in Fig. 11, the FVOPS curves show similar behaviour for all platforms, having a maximum value between ca. 6k and 20k cells per core, and decreasing for lower and higher core cell counts. More precisely, for all 64-core CPUs, the peak parallel efficiency is observed at 5688 cells per core (Fig. 11), which corresponds to the low mesh size. In this case, the low mesh size features a higher utilization of 64-core platforms regarding the number of processed cells per second

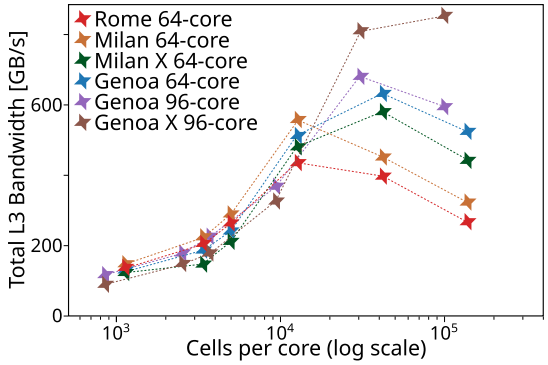

Fig. 12. Aggregated bandwidth of L3 cache for UAP.

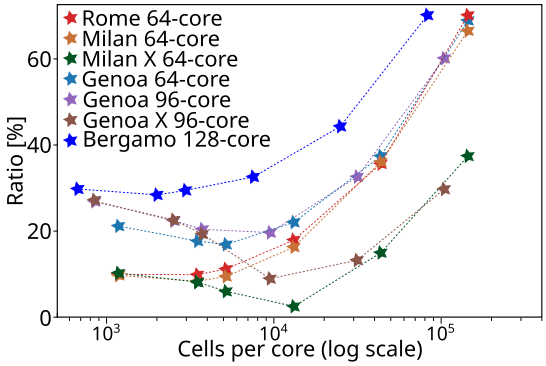

Fig. 13. L3 miss ratio for UAP.

by a single core. Moreover, the Genoa architecture dominates among the 64-core results, while Milan X does have an advantage over Rome and Milan for the largest two meshes. For smaller cell per core counts, Rome and Milan do not exhibit significant differences.

The results for processors with higher core counts are almost identical, except for the Genoa X processor, which outperforms other processors with the largest mesh sizes. The maximum FVOPS is noted for the mid mesh size and refers to 16,808 and 12,606 core cell counts for platforms with 96-core and 128-core CPUs, respectively. These platforms process the mid mesh with higher efficiency - in terms of the number of processed cells per second by core - compared to other meshes. In addition, it can be observed that providing more cores for calculations does not bring performance improvement as 96-core Genoa and 128-core Bergamo cannot show higher performance within this metric.

Fig. 12 illustrates the aggregated bandwidth of L3 cache that features similar trend lines as the FVOPS ones (first it rises to the turn point, then drops down). Comparing all tested platforms, the measurements of aggregated L3 bandwidth reveal similar behaviour for smaller meshes. In contrast, the significant differences in measurements are noticeable when processing larger meshes, including low, mid, and high sizes. In this case, as expected both Rome- and Milan-based platforms offer the lowest L3 bandwidth. The Milan X, thanks to a large L3 cache, reduces the advantage of novel AMD EPYC 9004 processors and features L3 bandwidth close to regular Genoa processors. Both 64-core and 96-core Genoa processors feature similar measurements of L3 bandwidth. Finally, Genoa X outperforms other platforms and offers the highest L3 bandwidth.

Considering relatively large meshes, including mid and high sizes, we observe a high L3 cache miss rate that reaches up to 70 % (Fig. 13). In this case, following the decreasing trends for FVOPS and measured L3 bandwidth, the capacity of the L3 cache plays a key role in the attainable performance. This is because the application requirement for the data volume exceeds cache capacity and generates mainly L3 capacity misses. As a result, the traffic through the L3 and main memory strongly affects overall performance across all sizes.

However, this hardware constraint is alleviated by enabling 3D V-cache technology in Milan X and Genoa X processors. In this case, we note a reduction in the L3 cache miss rate by up to 29 and 30 percentage points over regular Milan and Genoa CPUs, respectively. This results in a performance advantage by reducing the cost of data movement and accelerating the computation of up to 1.47x and 1.56x, respectively, for Milan X and Genoa X compared to Milan and Genoa CPUs.

It should be noted that the Bergamo-based platform features the highest trend for the L3 cache miss ratio (Fig. 13). To explain this behaviour, we have to look at the specification of the Bergamo (see Table 2). This architecture provides a smaller L3 cache size per core and more cores race the same main memory subsystem as other Genoa-based and Genoa X processors. Considering such hardware features, Bergamo processors do not provide a performance advantage for memory-sensitive applications over Genoa and Genoa X processors.

Table 11
Total volume of MPI data traffic and the Approximation of MPI Communication Percentage (ACP) for the execution time obtained for UAP on two-socket platforms with 64-, 96-, and 128-core based CPUs.

Mesh (cells)	MPI data traffic volume [GB]			ACP
	64-core	96-core	128-core	[%]
uxlow (36248)	84.4	110.5	140.3	67
ulow (139937)	168.0	217.0	262.1	57
mlow (228263)	238.8	308.7	360.0	55
low (728162)	220.7	287.0	343.1	54
mid (3227275)	645.4	803.9	902.5	38
high (14332247)	1506.9	1787.9	2061.2	21

As with the motorBike benchmark, we reveal the decreasing trend lines of L3 bandwidth and FVOPS for smaller and smaller mesh sizes. A detailed examination of MPI data traffic between MPI processes is necessary to interpret such a performance drop. To this end, we use the AMD μProf MPI Light-Weight Tracing tool [22] to collect profiling data. Table 11 presents the total volume of the MPI data traffic between MPI processes established for all platforms and different meshes. In addition, Table 11 shows the ACP parameter that estimates the MPI communication costs as a percent of runtime (ACP), taking the average value across all platforms for a particular mesh. As expected, for a given mesh, we observe the same data volume required to exchange between MPI processes in all 64-core-based platforms (Rome, Milan, Milan X, and Genoa). In contrast, using platforms with more cores results in a larger data volume for MPI traffic.

In addition, the performed profiling of MPI data traffic uncovers the computation-to-data-movement ratio, in terms of the number of cells per data traffic volume that increasingly grows for subsequent mesh sizes. As a result, platforms process fewer cells per byte of MPI data for smaller grids than for larger ones. Considering smaller sizes, relatively high data movement costs, and low computation intensity lead to the FVOPS drop, as resources are allocated more to MPI data transfer than cell processing. As a result, for smaller mesh sizes, the ACP parameter values are larger, and consequently, the MPI data traffic becomes the main cost, outweighing the computational cost. Furthermore, we reveal that approximately 25 % of the ACP corresponds to non-blocking send/receive operations for smaller meshes, rising to about 60 % for larger sizes. At the same time, the global reduction operation cost initially refers to about 55 % of the ACP and drops to around 30 % as the mesh size grows.

The Genoa-based processors offer twice larger L2 size per core, reducing L2 cache miss ratio trends (Fig. 14) and improving the bandwidth of L2 cache (Fig. 15) compared to previous generations of AMD EPYC. It results in performance advantages over previous generations of AMD EPYC CPUs (see Fig. 10). As shown in Fig. 14, the total L2 miss rate

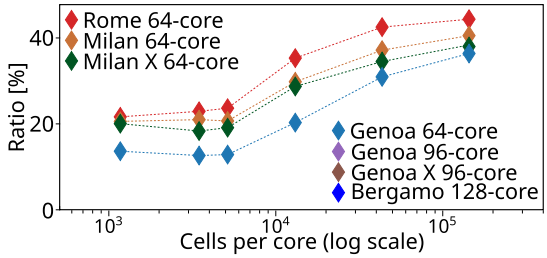

Fig. 14. Total L2 miss ratio for UAP.

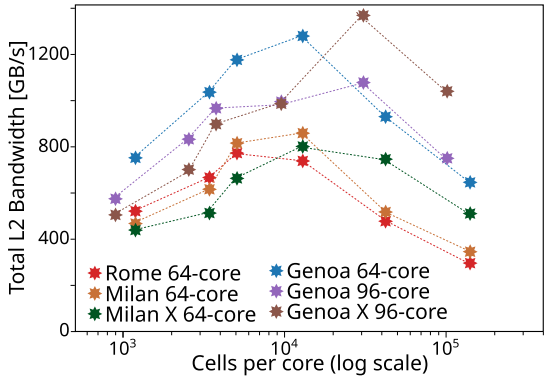

Fig. 15. Aggregated bandwidth of L2 cache for UAP.

of all AMD EPYC 9004 series processors remains tied. The highest level of L2 miss rate is noted for the prior generation of CPUs that uplift the L2 miss rate by up to 10 percentage points over novel CPUs. Considering the previous generation of CPUs, Milan X and Milan feature slightly lower total L2 miss rates than Rome.

In addition, we uncover the advantage of the new L2 cache offered by the AMD EPYC 9004 series processor over prior generations when examining L2 bandwidth measurements (see Fig. 15). The 64-core Genoa features the highest bandwidth of the L2 cache for smaller mesh sizes. In this case, it overcomes even platforms equipped with more cores (96-core Genoa and 96-core Genoa X). In contrast, the 96-core Genoa X achieves the highest bandwidth of the L2 cache for larger meshes. As with Genoa X, Milan X provides better L2 bandwidth than Milan and Rome for larger meshes. Considering smaller meshes, Milan X offers slightly worse measurements of L2 bandwidth than Milan and Rome.

Generally, the novel AMD EPYC 9004 series processors offer higher attainable performance than the prior generation (Rome, Milan, and Milan X). We note the performance uplift for new processors up to 2.5x and 2.7x over regular Milan and Rome CPUs, mainly resulting from the 12-channel DDR5, an extra 32 cores, and larger L2 cache.

We also observe that the 96-core Genoa and Genoa X do not boost performance over 64-core Genoa when processing smaller mesh sizes (see Fig. 10). We should underline here that 64-core Genoa has a higher clock speed (Table 5) that exceeds the frequency of about 200-300 MHz and 300-500 MHz compared to 96-core Genoa and 96-core Genoa X respectively. Additionally, as shown in Table 11, 64-core Genoa transfers less MPI data than the 96-core platforms, resulting in smaller communication overheads for smaller meshes.

8. Cross-platform analysis of metrics

This section aims to assess the impact of a wide range of performance metrics on the FVOPS metric introduced in Section 5. To gain a better understanding of FVOPS behaviour, we focus on two main objectives. Firstly, we analyze the correlation of individual metrics with FVOPS to identify metrics with the greatest influence on overall FVOPS behaviour. Secondly, we delve deeper into the issue by concentrating on two complementary questions: which metrics have the most decisive impact on the growth of FVOPS, and analogously which metrics are most influential in decreasing FVOPS. Addressing these objectives serves as starting point towards developing the FVOPS prediction model.

Table 12
Pearson coefficients between FVOPS and presented metrics for motorBike on different computing platforms.

Platform	DC Fills from same CCX	L2 hit from DC miss	L3 hit	L2 band-width	L3 band-width
Rome	–	0.89	0.80	0.99	0.60
Milan	0.81	0.80	0.85	0.94	0.43
Milan X	0.69	0.74	0.91	0.89	0.96
64-c. Genoa	0.88	0.83	0.84	0.66	0.78
96-c. Genoa	0.71	0.69	0.85	0.92	0.7
Genoa X	0.88	0.88	0.97	0.99	0.94
Bergamo	0.79	0.77	0.82	–	–
Average	0.79	0.80	0.86	0.90	0.74

Table 13
Pearson coefficients between FVOPS and presented metrics for UAP on different computing platforms.

Platform	DC Fills from same CCX	L2 hit from DC miss	L3 hit	L2 band-width	L3 band-width
Rome	–	0.86	0.85	0.44	0.94
Milan	0.92	0.90	0.89	0.66	0.95
Milan X	0.97	0.97	0.99	0.91	0.87
64-c. Genoa	0.83	0.82	0.96	0.65	0.74
96-c. Genoa	0.79	0.78	0.96	0.79	0.82
Genoa X	0.88	0.88	0.96	0.97	0.87
Bergamo	0.80	0.81	0.94	–	–
Average	0.87	0.86	0.94	0.74	0.87

To this end, we analyze the metrics collected using the AMD μProf software profiling tool for both motorBike and UAP use cases across all examined computing platforms presented in Table 2. The number of available performance metrics varies slightly between different computing platforms, but generally includes about fifty different metrics, with a particular focus on those providing details related to various levels of cache memory. Additionally, we extend our analysis to include metrics on aggregated L2 and L3 bandwidth obtained using the likwid-perfctr tool.

For evaluating the correlation between individual metrics and FVOPS, we utilize the Pearson correlation coefficient [24]. The values of this coefficient range from −1 to 1, where −1 indicates a perfect negative linear relationship, 0 indicates no linear relationship, and 1 indicates a perfect positive linear relationship between the two variables being compared. Tables 12 and 13 summarize the analysis, presenting the Pearson coefficient values for five metrics that perform best in both use cases. As can be observed, each of the first three metrics refers to a different cache level. Starting from the L1 cache, we observe that the correlation coefficient for the metric, which determines the total number of DC fills from the same CCX, achieves an average of 0.79 for motorBike and 0.87 for UAP, indicating the significant impact of this metric on shaping FVOPS. Similar values (0.80 and 0.86, respectively) are recorded for the metric concerning the number of L2 hits caused by missing data in the L1 DC level. Even better results in terms of correlation are achieved by the L3 hit metric, with an average of 0.86 for motorBike and 0.94 for UAP. Finally, the obtained average correlation values for aggregated L2 and L3 cache bandwidth confirm our observations from earlier Sections about the significant impact of these metrics on shaping FVOPS.

To address the second objective, which is to identify metrics that have a dominant impact on the increase and decrease in FVOPS, we

Table 14
Metrics that change the most with a increase of FVOPS (group L).

Category	Metric	motorBike	UAP
–	FVOPS	2.70×	9.21×
L1	All DC Fills	2.01×	2.91×
	DC Fills from same CCX	1.93×	2.79×
L2	Access	2.13×	3.06×
	Access from HWPF	2.88×	6.92×
	Access from DC Miss	1.93×	2.85×
	Hit	1.89×	2.58×
	Hit from HWPF	2.27×	5.06×
	Hit from DC Miss	1.88×	2.75×
	Bandwidth	1.77×	1.88×
L3	Access	2.28×	4.17×
	Hit	2.04×	4.02×
	Bandwidth	3.02×	4.89×

Table 15
Metrics that change the most with a decrease of FVOPS (group R).

Category	Metric	motorBike	UAP
–	FVOPS	0.44×	0.48×
L1	DC Fills From Local Memory	9.92×	13.66×
L2	Miss	1.58×	1.28×
	Miss from HWPF	1.60×	1.32×
	Miss from DC Miss	2.13×	1.60×
	Bandwidth	0.50×	0.57×
L3	Miss	2.12×	3.33×
	Bandwidth	0.74×	0.85×
Other	CPI	3.11×	1.71×
	Backend_Bound - .Memory	2.06×	1.71×

investigate the changes in individual performance metrics accompanying the changes in FVOPS. If we look at the behaviour of the FVOPS (presented in Fig. 5 for motorBike and Fig. 11 for UAP) we observe that this metric follows the same typical pattern for all architectures. Initially, it rises with an increase in cells per core, peaks and then begins to decline. For this reason, we divide the analysis into two independent stages. First, for each architecture, we determine the mesh size at which we record the best FVOPS value. This mesh size serves as a reference point from which we create two groups of data: L, containing all data for smaller mesh sizes, and R, including all data for bigger mesh sizes. Then, within each group, we look for the mesh size with the lowest FVOPS value and compare the data concerning this point with the data related to the reference point. This enables us to identify metrics that change the most within each group.

Tables 14 and 15 provide a summary of this analysis for the L and R groups, respectively. Each number in the tables represents the change in a given metric, calculated as an average across all computing platforms. Values greater than 1 indicate an increase, while smaller values indicate a decrease with respect to the reference point.

Starting from the Table 14, we observe that in both use cases, the increase in FVOPS is accompanied by an increase in metrics related to the Fills of L1 DC, as well as Accesses and Hits of L2 and L3 cache levels. For instance, with an average increase in FVOPS by 2.70× in the motorBike use case, we also record an average increase in metrics: All DC Fills by 2.01×, L2 Access by 2.13×, and L3 Access by 2.28×. A similar upward trend is observed for metrics related to the bandwidth of the L2 and L3 caches. The growth in FVOPS is associated with a 1.77× increase in L2 bandwidth for the motorbike use case and a 1.88× increase for the UAP use case. Even larger increases are observed for L3 bandwidth, with growth factors of 3.02× and 4.89×, for the motorbike and UAP, respectively.

Moving on to the Table 15, which summarizes the analysis of group R, we observe an average decrease in FVOPS to 0.44× of the best recorded value for motorbike and 0.48× for UAP. Within this group, we identify 9 metrics that exhibit significant changes. The most notable are observed in the 'DC Fills from Local Memory' metric within the L1 category, with increases of 9.92× and 13.66×, attributed to exceeding the capacities of L3 memory as the size of the analyzed mesh increases. Similarly to the previously discussed L group, significant changes are observed in the L2 and L3 bandwidth metrics, following the same decreasing trend as FVOPS. The last category concerns other metrics that are not directly related to individual cache levels. One of these is CPI, which describes the total number of cycles divided by the number of instructions. Low values of CPI indicate better computational performance. We note that the aforementioned decreases in FVOPS are accompanied by increases in the CPI metric by an average of 3.11× for motorBike and 1.71× for UAP. Additionally, we observe increases for both use cases (2.06× and 1.71×) in the Backend_Bound.Memory metric, which determines the fraction of dispatched slots that remained unused due to stalls in the memory subsystem. The behaviour of this metric confirms the significant impact of memory efficiency on both applications.

Table 16
Meshes for p-U-coupled motorBike test.

Name	Cell count	Avg. cell count per 128 cores	192 cores	256 cores
uxlow	32,897	257	171	129
ulow	110,509	863	576	432
mlow	167,555	1309	873	655
low	355,474	2777	1851	1389
mid	603,547	4715	3143	2358
high	1,897,187	14,822	9881	7411

9. Preliminary results of fully implicit method

In order to investigate additional solvers, the incompressible, steady-state solver pUCoupledFoam from the fork foam-extend version 5.0 was investigated [25]. The solver uses a block-coupled solution, so when constructing the matrix equations, pressure p and all components of velocity (u_x, u_y and u_z) are contained in a single equations, (see e.g. [26]). For this investigation, the motorBike use case was adapted to the block-coupled solver. For turbulence, the coupledKEpsilon method was used, which solves the turbulence equations for $k-\epsilon$ model in a block-coupled way. Additionally, some parameters were adjusted, including under-relaxation factors, to ensure convergence for the meshes investigated. A total of 100 iterations were calculated. Meshes were similarly generated, as for the original motorBike case, from mesh sizes of 33k (uxlow) to 1.9M (high) (see Table 16) to capture the peak FVOPS performance.

Fig. 16 shows the execution times of measurements for the p-U-coupled motorBike, which were obtained across seven computing platforms and various meshes. In addition to the execution times, this figure also highlights the performance improvements observed among the chosen computing platforms.

The Rome-based platform exhibits the lowest performance for almost all mesh sizes except for the smaller one, where performance with Milan X is tied (see Fig. 16(a-b)). The performance advantage of regular Milan over Rome is kept at the level of about 1.08 - 1.19 times faster. The Milan CPUs also uplift negligible performance over Milan X for the smaller meshes due to smaller clock speed. The performance improvements of Milan X compared to regular Milan and Rome are strongly noticeable for low, mid, and high mesh sizes (see Fig. 16(d–f)).

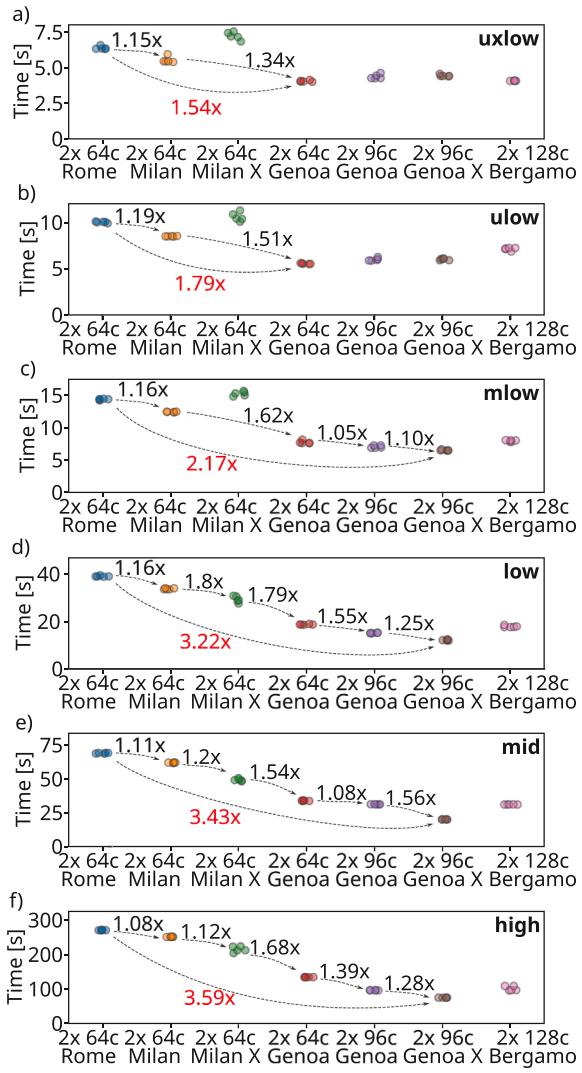

Fig. 16. Performance results obtained for p-U-coupled motorBike on a variety of AMD CPUs and different mesh sizes.

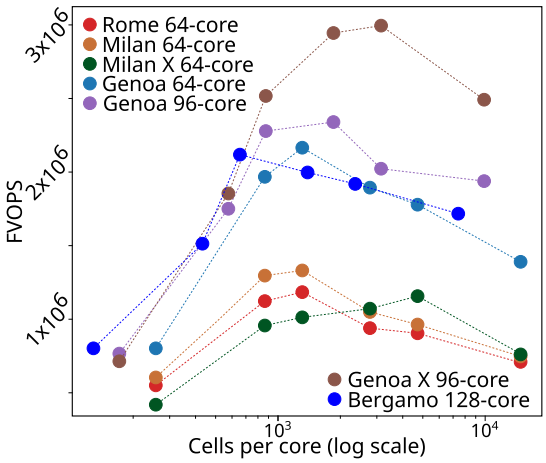

Fig. 17. FVOPS performance metric for p-U-coupled motorBike.

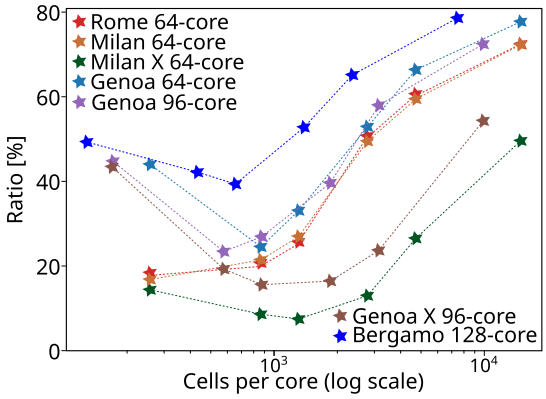

Fig. 18. Total L3 miss ratio for p-U-coupled motorBike.

The AMD EPYC 9004 family offers noticeable performance profit over all previous architectures across all mesh sizes. Considering all 64-core CPUs, the system with 64-core Genoa processors performs computations faster, accelerating execution about 1.5x-2.1x, 1.3x-1.9x, and 1.5x-2.1x than platforms with Rome, Milan, and Milan X CPUs, respectively.

Considering smaller meshes (uxlow and ulow), the 64-core Genoa features shorter execution (see Fig. 16(a-b)). We also reveal that 96-core Genoa brings a relatively negligible improvement compared to the 64-core version, improving performance in the range of 1.05x to 1.55x only for larger meshes. As expected, the platform with Genoa X CPUs improves performance over other systems for larger meshes (see Fig. 16(c–f)). However – similar to Milan X – this platform is not faster on smaller mesh sizes over regular Genoa processors. In addition, the 128-core Bergamo does not bring any performance profits, achieving similar results to those of the Genoa CPUs.

Furthermore, Fig. 17 illustrates the FVOPS calculated for all platforms and mesh sizes when running p-U-coupled motorBike. The FVOPS analysis reveals similar behaviour as for previous tests: it increases up to a turning point and decreases afterward. It should be noted that for the fully implicit solver the FVOPS metric is 10-15x smaller than for the semi-implicit (see Fig. 5). The inflection point (best performance utilization) is much earlier (approx. 60 %) on the X-axis (for a smaller number of cells per core). This should be explained by the much higher requirements for memory capacity and bandwidth of the fully implicit model compared to the semi-implicit solution.

The optimal utilization of Rome, Milan, 64-core Genoa, and Bergamo CPUs is observed for the mlow mesh size (167555 cells). Milan X and Genoa X platforms offer the highest efficiency for the mid mesh (603547 cells). The 96-core Genoa achieves the best performance for the low mesh (355474 cells). For larger mesh sizes, both Milan X and Genoa X architectures dominate their own platforms. Milan X shows an advantage over regular Milan and Rome for low, mid, and high meshes, while Genoa X has an advantage over Genoa-based CPUs from 167,555 to 1,897,187 cells. Considering smaller sizes, differences between Rome, Milan, and Milan X are not that dominant. The tested AMD EPYC 9004 CPUs show similar performance, with a negligible advantage for the 64-core Genoa.

Figs. 18 and 19 demonstrate the performance analysis focusing on the impact of the L3 cache on performance. As expected, both Milan X and Genoa X surpass other architectures, offering lower L3 ratio miss trendlines and higher L3 bandwidth over regular Milan and Rome, as well as regular Genoa and Bergamo, respectively. The Milan X offers the lowest L3 ratio miss while Genoa X reaches the highest L3 bandwidth. Furthermore, all studied CPUs of the 4th gen. of AMD EPYC CPUs feature better L3 bandwidth over Rome, Milan, and Milan X. In line with the FVOPS trendline, the optimal measurements of L3 bandwidth are observed for (i) mlow mesh on Rome and Milan, (ii) for low mesh on Milan X and 64-core Genoa, and (iii) for mid mesh on 96-core Genoa and Genoa X. As we focus on increasingly larger meshes, it becomes evident that L3 misses primarily occur because the volume of data that needs to be transferred through the cache exceeds its total capacity.

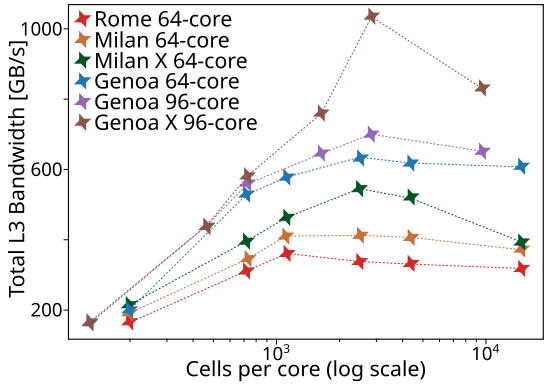

Fig. 19. Aggregated bandwidth of L3 cache for p-U-coupled motorBike.

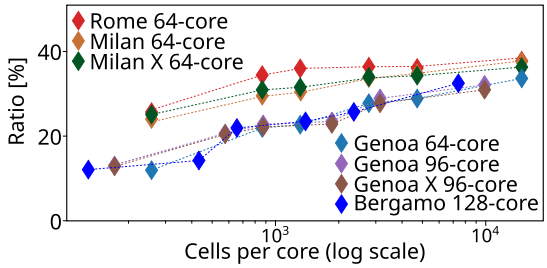

Fig. 20. Total L2 miss ratio for p-U-coupled motorBike.

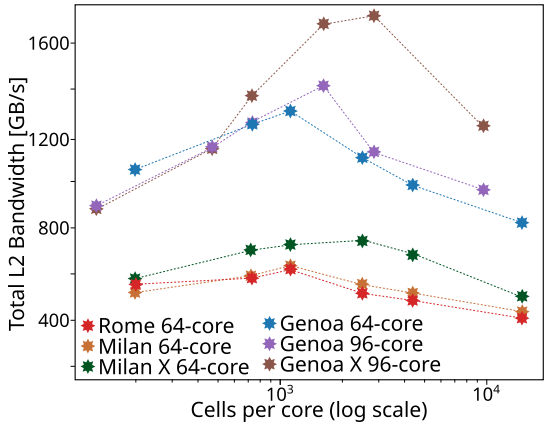

Fig. 21. Aggregated bandwidth of L2 cache for p-U-coupled motorBike.

Consequently, the measured L3 bandwidth shows a decreasing trend, as the memory-cache access pipeline becomes saturated. Considering relatively smaller sizes, we anticipate that the L3 bandwidth becomes limited since the data sizes seem to small to keep the cache-memory pipeline fully occupied and get better transfer efficiency. In addition, as the smallest mesh sizes increase, the unavoidable L3 misses become particularly noticeable when a memory location is accessed for the first time. In this case, as shown in Fig. 18, the Milan X, Genoa, Genoa X, and Bergamo CPUs exhibit a drop in L3 miss ratios. For Milan and Rome, as confirmed by preliminary analysis, a similar trend is expected for smaller mesh sizes that are beyond the scope of this study.

It is also worth noting that the AMD EPYC 9004 series CPUs – thanks to larger L2 – achieve lower L2 cache miss ratio and higher L2 bandwidth compared to previous generations (Figs. 20 and 21). As shown in Fig. 20, the total L2 miss rate remains tied for all AMD EPYC 9004 series processors, reducing it by up to 10 percentage points over novel CPUs. Considering the previous generation of CPUs (Rome, Milan, and Milan X), the L2 miss ratio is kept at a similar level.

As shown in Fig. 21, the highest L2 bandwidth are noted for the Genoa X when processing larger meshes. In contrast, the 64-core Genoa features the highest bandwidth of the L2 cache for smaller mesh sizes (uxlow and ulow). In this case, it overcomes even platforms equipped with more cores, and thus, the 96-core Genoa and Genoa X do not boost performance over 64-core Genoa when processing smaller mesh sizes.

10. Conclusions

The work presents an empirical analysis of the performance of CFD applications based on two selected models (motorBike and Urban Air Pollution) for different generations of AMD processor architectures of the professional EPYC line. Particular attention was paid to the impact of cache (L2 and L3), RAM (DDR4 and DDR5), and the related bandwidth of memory channels, which is important for memory-bound applications. First, the impact of the FVOPS metric was assessed with changing problem size. It was noticed that the application performance increases with decreasing grid size up to a certain point, resulting from the capacity of the cache-memory subsystem. After reaching the inflection point (individual for different processors) resulting from the amount of data per core, a decrease in processing efficiency was recorded. It should be assumed that this is the result of limiting the available memory bandwidth per processing unit and the related deterioration of the effectiveness of the prediction algorithm for the L3 cache, which resulted in higher L3 miss rates (up to 80 %).

The research conducted by means of the AMD μProf profiling application confirmed the assumption that a larger cache is beneficial for achieving higher performance. However, it should be noted that increasing the number of cores and cache size in the CPU does not result in proportionally higher performance, which is due to limitations in data flow and a smaller amount of L3 cache per core. This was particularly visible when measuring the L3 misses metric, which was related to the fact that the data volume required to transfer through the cache is larger than the total cache capacity.

Furthermore, the performed profiling of MPI data traffic uncovers the computation-to-data-movement ratio, in terms of the number of cells per data traffic volume that increasingly grows for subsequent mesh sizes. Consequently, platforms process less cells per byte of MPI data movement for smaller meshes than for larger ones. Relatively high data movement costs and low computation intensity lead to the FVOPS drop for smaller sizes, as resources are allocated more to MPI data transfer than cell processing. As a result, MPI data traffic becomes the dominant cost, surpassing computation for smaller sizes.

We also reveal that the traffic through the L3 cache and the main memory strongly affects overall performance across all sizes. According to FVOPS metric, we identify the mesh size with the highest computing efficiency. Considering all platforms, we observe the average decrease in FVOPS to 0.44× and 0.48× of the best-recorded value for motorBike and UAP, respectively. Such decreases in computational performance can be explained by the increasing demand for significant amounts of data, given the larger mesh sizes that exceed cache capacity.

In summarizing our observations regarding the FVOPS trend lines, we note that FVOPS rises until it reaches a turning point (inflection point) and subsequently declines. It is evident that on the left side of the inflection point, where the number of cells per core diminishes, the costs related to MPI communication increase, thereby restricting performance. Conversely, the rise in the number of cells per core on the right side of the inflection point leads to an escalating demand on the cache and main memory subsystems, which in turn limits performance due to the data traffic between these subsystems. Nevertheless, the turning point of the FVOPS curve, along with the areas surrounding the inflection point, reveals a trade-off among MPI, cache, and main memory in terms of data traffic communication, which promotes improved hardware utilisation.

The performed investigation also reveals that the FVOPS trendline moves up when using the 4th gen. of AMD EPYC CPUs in comparison to

previous models (see Figs. 5, 11, and 17). Consequently, this family of AMD CPUs, thanks to the newest DDR5 and larger 2x 12-channel memory subsystem, as well as doubled L2 cache size, offers higher attainable performance than the prior generation Rome, Milan, and Milan X.

Furthermore, we reveal that the inflection area goes up when using platforms with larger L3 cache sizes. It is noticeable when comparing, for example, regular Milan to Milan X as well as regular 96-core Genoa to Genoa X (see Figs. 5, 11, and 17). We can conclude here that the L3 cache capacity strongly affects attainable FVOPS for the inflection area, moving it up or down when using a larger or smaller L3 cache, respectively.

Since the turning point of the FVOPS curve responds to a trade-off between MPI, cache, and main memory communication constraints, it indicates the optimal number of processed cells per second for a given platform, facilitating better hardware utilisation. As shown in our last paper [27], where we target multi-node investigation, the indicated by the FVOPS ideal number of cells can successfully support selecting the optimal number of computational nodes on the cluster. The performance advantages and energy savings were achieved in [27] by distributing the workload across a number of nodes, ensuring that each node handled an optimal number of cells per platform, based on the inflection point determined through FVOPS.

The cross-platform analysis of memory-related metrics reveals five key metrics from different cache levels that have the strongest correlation to FVOPS across both use cases. Additionally, this section highlights specific metrics that exhibit the most substantial variations accompanying increases or decreases in FVOPS. Depending on the mesh sizes, different groups of factors have a significant impact on the performance. For smaller meshes, metrics related to successful access to data ("access" and "hit" metrics) stand out, while for larger sizes, those related to failed data retrieval ("miss" metrics) from the L2 and L3 cache become more important. The metrics related to the bandwidth of L2 and L3 played a key role, underscoring their importance in shaping FVOPS. In addition to findings on the behaviour of metrics directly related to individual cache levels, the analysis reveals that decrease of FVOPS is accompanied by the growth of CPI and Backend_Bound.Memory metrics.

We uncover that 96-core Genoa/Genoa X do not achieve higher performance for smaller sizes than 64-core platforms. The best performance was measured with the Genoa architecture - EPYC 9554 for small mesh sizes and EPYC 9684X for larger ones. The Bergamo-based platform did not prove to be the most efficient for both smaller and larger mesh sizes.

Additionally, this work covers the preliminary study of testing other, more memory-intensive CFD use cases. To achieve this aim, we tackle the fully implicit methods represented by the p-U-coupled motorBike use case. The performed investigation confirms similar behaviour as for semi-implicit codes.

CRediT authorship contribution statement

Marcin Lawenda: Writing – review & editing, Writing – original draft, Validation, Supervision, Software, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization; **Łukasz Szustak:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **László Környei:** Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization; **Flavio Cesar Cunha Galeazzo:** Writing – review & editing, Methodology, Formal analysis, Conceptualization; **Paweł Bratek:** Writing – original draft, Validation, Methodology, Investigation, Data curation.

Data availability

Data will be made available on request.

Declaration of competing interest

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

Acknowledgements

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking and Poland, Germany, Spain, Hungary, France and Greece under grant agreement number: 101093457. This publication expresses the opinions of the authors and not necessarily those of the EuroHPC JU and Associated Countries which are not responsible for any use of the information contained in this publication. The authors are grateful to AMD company for granting access to the HPC platforms.

References

[1] TOP500 website, 2024, Accessed: 2024-05-17, https://top500.org/.
[2] EuroHPC joint undertaking website, 2024, Accessed: 2024-05-17, https://eurohpc-ju.europa.eu/.
[3] OpenFOAM website, 2024, Accessed: 2024-05-17, https://www.openfoam.com/.
[4] N. M. C. Martins, N. J. G. Carriço, H. M. Ramos, D. I. C. Covas, Velocity-distribution in pressurized pipe flow using CFD: accuracy and mesh analysis, Comput. Fluids 105 (2014) 218–230. https://doi.org/10.1016/j.compfluid.2014.09.031
[5] V. Moureau, P. Domingo, L. Vervisch, Design of a massively parallel CFD code for complex geometries, C.R. Mec. 339 (2011) 141–148. https://doi.org/10.1016/j.crme.2010.12.001
[6] I. Hadade, T. M. Jones, F. Wang, L. d. Mare, Software prefetching for unstructured mesh applications, ACM Trans. Parallel Comput. 7 (1) (2020). https://doi.org/10.1145/3380932
[7] G. Katevenis, M. Ploumidis, M. Marazakis, Impact of cache coherence on the performance of shared-memory based MPI primitives: a case study for broadcast on intel xeon scalable processors, Proceedings of the 52nd International Conference on Parallel Processing (2023) 295-305. https://doi.org/10.1145/3605573.3605616
[8] L. Szustak, M. Lawenda, S. Arming, G. Bankhamer, C. Schweimer, R. Elsässer, Profiling and optimization of Python-based social sciences applications on HPC systems by means of task and data parallelism, Future Gener. Comput. Syst. 148 (2023) 623–635. https://doi.org/10.1016/j.future.2023.07.005
[9] A. Yildirim, C. A. Mader, J. R. R. A. Martins, Accelerating parallel CFD codes on modern vector processors using blockettes, Proceedings of the Platform for Advanced Scientific Computing Conference (2021). https://doi.org/10.1145/3468267.3470615
[10] 4th gen AMD EPYC processors architecture (white-paper), 2023, Accessed: 2023-12-10, https://www.amd.com/en/campaigns/epyc-9004-architecture.
[11] L. Szustak, R. Wyrzykowski, T. Olas, V. Mele, Correlation of performance optimizations and energy consumption for stencil-based application on intel xeon scalable processors, IEEE Trans. Parallel Distrib. Syst. 31 (11) (2020) 2582–2593.
[12] L. Szustak, et al., Architectural adaptation and performance-energy optimization for CFD application on AMD EPYC Rome, IEEE Trans. Parallel Distrib. Syst. 32 (12) (2021) 2852–2866.
[13] AMD EPYC& 7002 series processors, 2024, Accessed: 2024-05-17, https://www.amd.com/en/processors/epyc-7002-series.
[14] I/O controller hub, 2024, Accessed: 2024-05-17, https://en.wikipedia.org/wiki/I/O_Controller_Hub.
[15] SCOTCH library website, 2024, Accessed: 2024-05-17, https://www.labri.fr/perso/pelegrin/scotch/.
[16] HiDALGO2 project website, 2024, Accessed: 2024-05-17, https://www.hidalgo2.eu/.
[17] OpenFOAM standard solvers, 2024, Accessed: 2024-05-17, https://www.openfoam.com/documentation/user-guide/a-reference/a.1-standard-solvers.
[18] M. Leuridan, J. Hawkes, T. Quintino, Polytope: feature extraction for improved access to petabyte-scale datacubes, EGU23-8839 (2023). https://doi.org/10.5194/egusphere-egu23-8839
[19] COPERT - EU standard vehicle emissions calculator, 2024, Accessed: 2024-05-17, https://copert.emisia.com/.
[20] AMD optimizing C/C++ and Fortran compilers (AOCC), 2023, Accessed: 2023-10-22, https://www.amd.com/en/developer/aocc.html.
[21] F. C. C. Galeazzo, R. G. Weiß, S. Lesnik, H. Rusche, A. Ruopp, Understanding super-linear speedup in current HPC architectures, Preprints (2024). https://doi.org/10.20944/preprints202404.0219.v1
[22] AMD uProf user guide 4.1, 2024, Accessed: 2024-01-10, https://www.amd.com/en/developer/uprof.html.
[23] LIKWID performance tools, 2024, Accessed: 2024-04-11, https://hpc.fau.de/research/tools/likwid/.
[24] Pearson correlation coefficient, 2024, Accessed: 2024-05-17, https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[25] K. Jareteg, Block coupled calculations in OpenFOAM, Project within course: CFD with OpenSource software. Chalmers University of Technology (2012).
[26] C. J. G. Henry G. Weller, Notes on computational fluid dynamics: general principles (2022). https://doc.cfd.direct/notes/cfd-general-principles/matrix-construction.
[27] M. Lawenda, L. Szustak, L. Környei, Prediction model of performance-energy trade-off for CFD codes on AMD-based cluster, Future Gener. Comput. Syst. 169 (2025) 107810. https://doi.org/10.1016/j.future.2025.107810