D3.4 Innovative HPC Technologies and Benchmarking

HIDALGO2
CENTRE OF EXCELLENCE

Date: April 18, 2024

EuroHPC
Joint Undertaking

Document Identification			
Status	Final	**Due Date**	31/03/2023
Version	1.0	**Submission Date**	18/04/2024

Related WP	WP3	**Document Reference**	D3.4
Related Deliverable(s)	D3.1	**Dissemination Level (*)**	PU
Lead Participant	PSNC	**Lead Author**	Łukasz Szustak (PSNC)
Contributors	PSNC, SZE, USTUTT	**Reviewers**	József Bakosi (SZE)
			Flavio Galeazzo (USTUTT)

Keywords:

High Performance Computing (HPC), AMD EPYC, Bergamo, Genoa, Genoa-X, Milan-X, profiling, co-design

Document Information

List of Contributors	
Name	**Partner**
Łukasz Szustak	PSNC
László Környei	SZE
Marcin Lawenda	PSNC
Flavio Galeazzo	USTUTT

Document History			
Version	**Date**	**Change editors**	**Changes**
0.1	05/02/2024	PSNC	Table of Contents
0.2	29/02/2024	PSNC	Performance Results (Tables)
0.3	06/03/2024	PSNC	Performance Results (Figures)
0.4	15/03/2024	PSNC, USTUTT	Benchmark methodology
0.5	21/03/2024	PSNC	Architecture overview
0.6	26/03/2024	SZE	Performance Results – Part 1
0.7	26/03/2024	PSNC	Profiling overview – Part 1
0.8	04/04/2024	PSNC	Profiling overview – Part 2
0.9	05/04/2024	SZE	Performance Results – Part 2
0.92	12/04/2024	PSNC	Version for review
0.95	18/04/2024	PSNC, SZE	Addressing reviewers comments
1.0	18/04/2024	PSNC	Final version

Quality Control		
Role	**Who (Partner short name)**	**Approval Date**
Deliverable Leader	Lukasz Szustak (PSNC)	18/04/2024
Quality Manager	Harald Koestler (FAU)	18/04/2024
Project Coordinator	Marcin Lawenda (PSNC)	18/04/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
CFD	Computational Fluid Dynamics
CCD	Core Complex Die
ccNUMA	Cache Coherent Non-Uniform Memory Access
CCX	Core Complex
CoE	Centre of Excellence
CPU	Central Processing Unit
DC	Data Cache (level 1)
FVOPS	Finite VOlumes solved Per Second
GSS	Global System Science
HPC	High Performance Computing
HWPF	Hardware Prefetching
I/O	Input/Output
IC	Instruction Cache (level 1)
IOD	Input/Output Die
L1	Cache level 1
L2	Cache level 2
L3	Cache level 3
MPI	Message Passing Interface
NPS	NUMA Per Socket
NUMA	Non-Uniform Memory Access
SIMD	Single Instruction Multiple Data
SIMD	Single Instruction Multiple Data
SMT	Simultaneous Multithreading
SoC	System on a Chip
TDP	Thermal Design Power
TLB	Translation Lookaside Buffer
UAP	Urban Air Project
UMC	Unified Memory Controllers

Executive Summary

A thorough understanding of the development direction of the technologies on which modern HPC systems are based is the key to building efficient applications that take full advantage of the offered hardware capabilities.

This deliverable outlines the first approach of HiDALGO2 implementations to innovative HPC technologies. Particularly, the report provides information on promising state-of-the-art AMD technologies which apply to HiDALGO2 pilot applications. It defines the HiDALGO2 benchmark suite concerning the profound profiling of AMD EPYC 9004 series processors. The OpenFOAM framework, commonly used to implement use cases in the project, was used as the test base.

For a better understanding of the research context, a detailed description of the application architecture of the new processors and the platforms on which the tests were performed is presented. Particular emphasis is placed on the impact of large-cache systems on the parallel performance of various OpenFOAM computation kernels.

Many tests have demonstrated the optimal balance between memory speed and core count for a Genoa-based system featuring 2x64 cores, particularly evident with smaller mesh sizes. Moreover, the EPYC Genoa processor boasting 96 cores exhibits a performance boost of up to 2.6 times when compared to the standard EPYC Milan. This enhancement is credited to the 12-channel DDR5 memory and an additional 32 cores. The study also highlights that systems equipped with large caches and 3D cache support enabled by AMD processors show significant potential for performance enhancements by addressing memory binding issues in applications like OpenFOAM--based parallel codes. Specifically, substantial performance advantages are realized with the 3D cache systems found in Milan X and Genoa X processors in contrast to their regular Milan and Genoa counterparts. Furthermore, performance improvements were noted in systems featuring innovative DDR5-based compute nodes (Genoa and Bergamo nodes) compared to the previous generation DDR4-based memory subsystem (Rome and Milan nodes).

1 Introduction

1.1 Purpose of the document

Nowadays, we are dealing with dynamic development in the area of hardware and software, which especially applies to the HPC domain. This means that trends in technological solutions for IT equipment and management platforms should be taken into account already when designing software.

In the context of the HiDALGO2 project, this is taken into account for services and methods developed in WP3 and WP4, and for WP5 pilot applications. This will answer the question of what future hardware architectures are suitable for each type of applications and what is the cost of adaptation to effectively and quickly use such an environment for GSS.

This is done by practically analysing promising technologies by having them delivered in the minimum required configuration by hardware manufacturers and drawing conclusions about their specific utility. Benchmark testing is required to understand the prerequisites of each application and the differences from the reference systems.

1.2 Relation to other project work

This report derives from "D3.1 Scalability, Optimization and Co-Design Activities", especially about the benchmarking methodology. Deliverable D3.4 is a living document that outlines the most recent findings in the innovative areas and thus will be updated in D3.5 and D3.6.

1.3 Structure of the document

This document is divided into six primary sections:

- Chapter 2 – outlines an architecture overview of the state-of-the-art AMD products that represent traditional HPC processor vendors,
- Chapter 3 – presents information about the infrastructure used to investigate novel AMD processors,
- Chapter 4 – on benchmarking new-generation AMD EPYC processors, investigating two OpenFOAM-based applications,
- Chapter 5 – presents the performance results obtained for OpenFOAM-based applications including motorBike and UAP use cases,
- Chapter 6 – summarises the presented work and outlines future steps in respect of innovative technologies investigations in the project.

2 Novelty architecture overview

This section outlines an architecture overview of the state-of-the-art AMD products that represent traditional HPC processor vendors. The analysis investigates the top AMD processors, the AMD EPYC 9004 series processors, to reveal trends in innovative HPC architectures considering traditional processors. In addition, the HiDALGO Report D5.8 delivers an overview of prior generations of AMD EPYC processors [2].

We note that AMD EPYC CPUs excel in the HPC domain by offering high core count, performance, and relevant memory subsystems with large cache capacity. This trend is also observed in EuroHPC JU systems used in the HiDALGO2 project, which are mainly built with AMD-based solutions. The new series of EPYC processors offers a wide range of models, making them the right choice for parallelizing demanding tasks in HPC workloads. Since the AMD CPUs are becoming the leaders in HPC, we move our focus on the newest AMD EPYC 9004 series products at this stage of the HiDALGO2 project. Going forward, we plan to explore Intel- and ARM-based CPUs and GPU solutions.

2.1 AMD EPYC 9004 series CPUs

The AMD EPYC 9004 series processors represent the 4th generation of AMD EPYC server-class processors. The design of this generation of AMD EPYC processors features the AMD Zen 4 microarchitecture of compute cores, the AVX-512 instruction set, large cache memory, and higher 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.

Similarly to previous generations, the novel AMD EPYC processors retain the proven multi-chip module chipset architecture [3]. This architecture incorporates compute cores, memory controllers, and I/O controllers into an integrated System on a Chip (SoC). The SoC system includes the Core Complex Dies (CCDs), which contain Core Complexes (CCXs). Every CCX consists of a set of compute cores based on Zen 4 microarchitecture. All CCDs are surrounded by the central high-speed I/O die and interconnect via the AMD Infinity Fabric. The AMD EPYC 9004 series processor distinguishes three different CPU models [5]:

- ***Genoa*** designed for balanced workloads and per-core performance,
- ***Genoa X*** dedicated to memory-bound applications (CPUs with large L3 AMD 3D V-Cache),
- ***Bergamo*** with the highest core density (97x4 series) for compute-intensive workload.

Table 1 summarises the features common to the fourth generation of AMD EPYC processors. The following sections describe each of these models in-depth.

Table 1. General specification of AMD EPYC 9004 Series Processors features by model [5]

Codename	Genoa	Genoa X	Bergamo
Compute cores	Zen 4	Zen 4	Zen 4c
Core process technology	5nm	5nm	5nm
Max number of CCDs	12	12	8
Number of CCXs per CCD	1	1	2
Max L3 cache size (per CCX)	384MB (32MB)	1152MB (96MB)	256MB (16MB)
Max number of cores (threads)	96 (192)	96 (192)	128 (256)
Max # of memory channels	12 DDR5	12 DDR5	12 DDR5
Max memory speed	4800 MT/s DDR5	4800 MT/s DDR5	4800 MT/s DDR5

2.2 AMD Genoa

AMD releases HPC server processors under the brand name Genoa [3]. These processors can provide a higher number of cores, memory channels, and cache sizes to attract HPC applications. The new series of AMD processors is based on the new Zen 4 compute core built with 5nm fabrication technology. Every core features a larger L2 cache and improved cache effectiveness over the prior generation.

The Zen 4 compute core consists of up to 32 KB of 8-way L1 I-cache and 32 KB of 8-way of L1 D-cache [5]. It includes a 1MB private L2 cache unified for instruction and data. All caches also feature a 64B cache line size. The simultaneous multithreading is supported on the Zen 4 core, allowing two separate threads to run independently and share the corresponding core's L2 cache. Every Zen 4 compute core comprises an AVX-512 instruction set based on a SIMD model. It implements an energy-efficient AVX-512 instruction with 256-bit internal data paths that finally uses a 512-bit AVX register file, from which two 256-bit vectors are executed in sequential clock cycles.

The Genoa-based processors [5] incorporate up to eight cores to create a core complex (CCX) with a 32 MB shared L3 cache (Figure 1). This core complex is mapped onto a single core complex die (CCD). Compared to the prior generation, today's 4th Gen AMD EPYC with Genoa CPU combines up to 12 CCDs, which can be configured for up to 96 cores. Enabling SMT technology allows a single processor to support up to 192 concurrent logical cores.

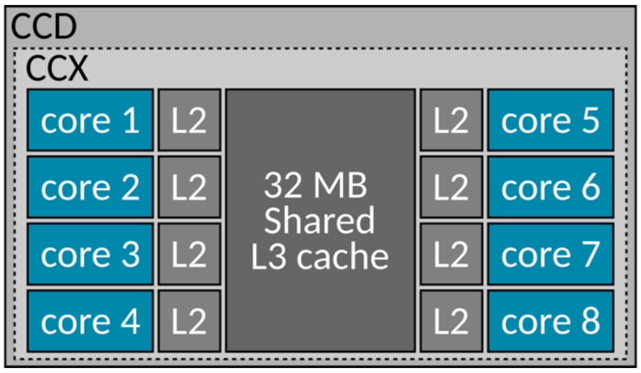


Figure 1. Layout of CCD with eight Zen 4 compute cores sharing an L3 cache

All CCDs connect to memory, I/O, and each other through an I/O Die (IOD) [4]. The CCDs surround the central high-speed IOD and interconnect via the AMD Infinity Fabric. The IOD enables the data path and control support to interconnect CCXs, memory, and I/O. All dies, also called chiplets, are interconnected with each other via AMD Infinity Fabric technology. The IOD maintains cache coherency between CCXs and provides the interface to extend the data fabric to dual-socket servers. Figure 2 illustrates the AMD EPYC 9004 system on the chip block diagram consists of up to 12 CCDs and a central IOD.

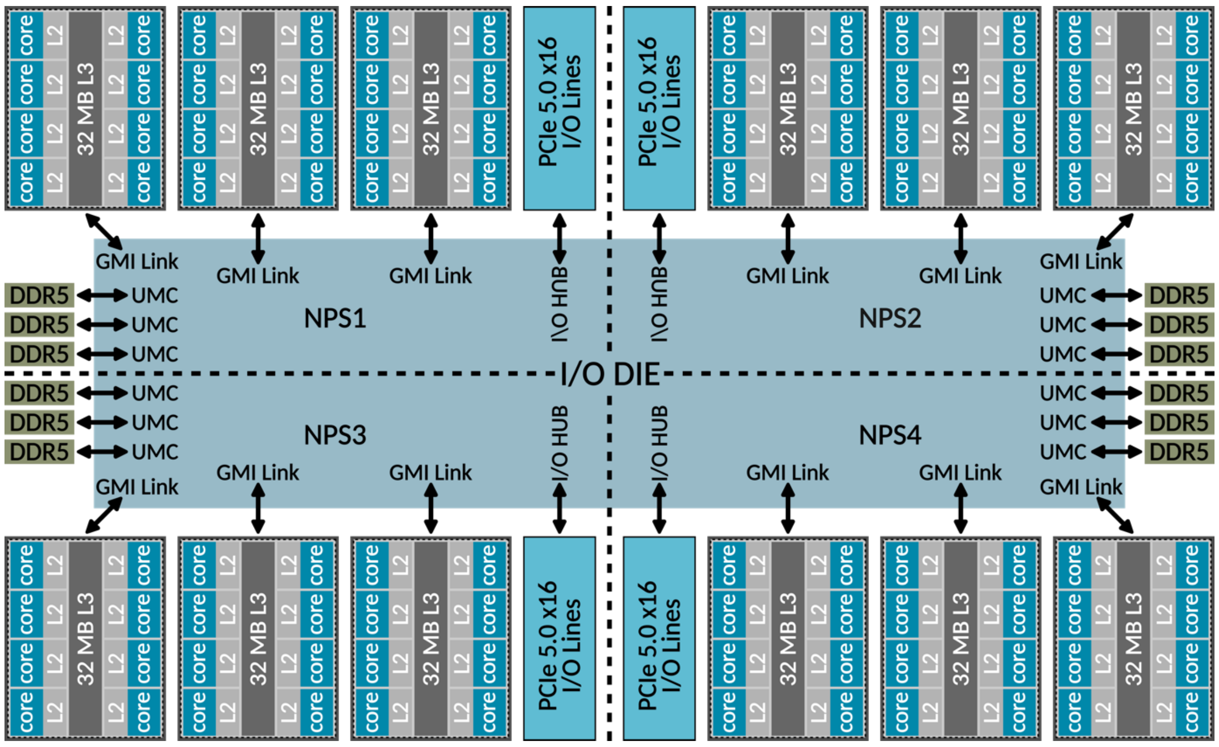


Figure 2. AMD EPYC Genoa model with 12 CCDs and a central IOD

The IOD also exposes DDR5 memory channels, PCIe-5, and Infinity Fabric links (Figure 2). The IOD provides twelve unified memory controllers (UMC) that maintain twelve DDR5 memory channels. Every channel supports up to 2 DIMMs for a maximum of 24 DIMMs per socket. It results in up to 6TB of DDR5 memory [4].

Like prior AMD EPYC CPUs, the 4th generation of AMD EPYC 9004 series processors features a NUMA architecture [4] [5] with separate quadrants, each with 3 CCDs, 3 UMCs, and 1 I/O Hub (Figure 2). Using the cores, memory, and I/O peripherals within the same quadrant provides uniform performance and the closest data path distance. The furthest distance is noticeable for resources in cross-diagonal quadrants and the other processor in a dual-socket configuration. The four logical quadrants allow the processor to be partitioned into different NUMA domains. These domains are designated as NUMA per socket (NPS) and can be adjusted in the BIOS setting to optimize NUMA topology for specific operating environments and workloads. The NPS configuration indicates the interleave pattern of the memory channels within the NUMA domains. For example, a setting of NPS=1 defines a single NUMA node per socket where the memory is interleaved across the twelve memory channels. In NPS=4 mode, a single processor is partitioned into four NUMA domains. In this case, the memory is interleaved across the three memory channels associated with each logical quadrant. Using BIOS settings, the server could be configured as NPS0, NPS1, NPS2, or NPS4, with an additional option to configure L3 cache slices as NUMA domains. More details of NPS configurations are delivered in [4].

Each Genoa-based processor also offers up to eight sets of high-speed PCIe 5.0x16-bit I/O lanes, that is, 128 lanes in single-socket platforms [4]. In dual-socket systems, up to half of the PCIe lanes from each processor are used to communicate two CPUs through AMD’s socket-to-socket interconnect, Infinity Fabric. It leaves each socket with 64 remaining PCIe lanes in a dual-socket platform. In addition, the dual-socket platform can also be customized to reduce socket-to-socket interconnect links to 48 lanes per socket, allocating the remaining 160 lanes for PCIe I/O devices (80 lines per socket).

2.3 AMD Genoa X

The AMD EPYC 9004 series processors include a Genoa X model with the AMD 3D V-cache technology [4]. This technology uses the same 3D die stacking concept as the prior generation, Milan X. It stacks additional cache layers on top of every CCD. The extra L3 cache allows the processor to load and store data more efficiently by decreasing the number of times it needs to proceed with the data traffic through the RAM. It results in having more data close to the cores and increases the performance of the memory-bound applications [4] [5].

The novel Genoa X model uses cache die stacking to augment further L3 cache to regular Genoa Zen 4 CCDs. A refresh of the AMD EPYC Genoa with enabled 3D V-Cache consists of the same cores as EPYC Genoa but with an additional 768 MB of cache stacked onto the compute dies. The AMD 3D V-Cache technology delivers a large aggregated L3 cache size that reaches 1152 MB for 12 CCDs on a 96-core

Genoa X chip. It leverages AMD's Zen 4 compute core, 12-channel DDR5 memory, and other features available in regular AMD EPYC 9004 series processors.

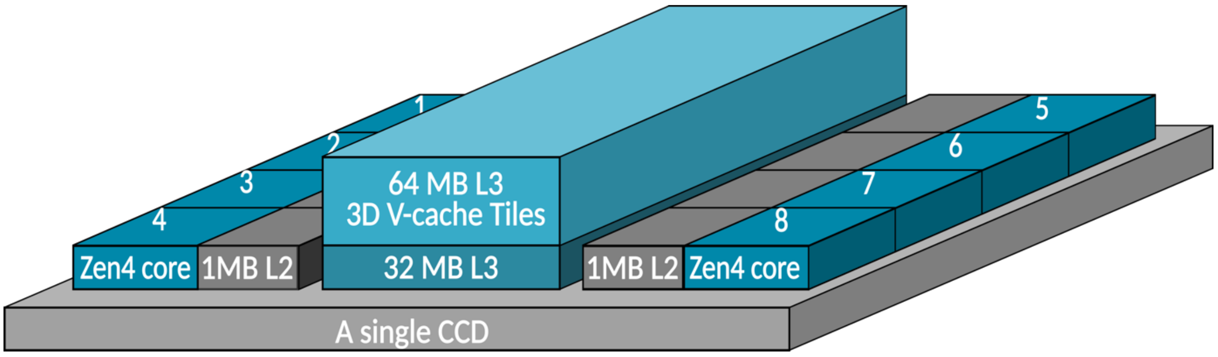


Figure 3. Layout of CCD with AMD 3D V-cache tiles in AMD EPYC Genoa X model

Figure 3 illustrates the layout of a single CCD with AMD 3D V-Cache enabled for the AMD EPYC Genoa X processors model. The processors with AMD 3D V-Cache technology augment the per-die L3 cache three times. This technology allows for extending the shared 32 MB L3 cache with 64 MB additional layered above, bringing the per-die total L3 cache to 96 MB in Genoa X. 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 – its successors – 96-core Genoa X chip with 12 CCDs.

2.4 AMD Bergamo

The fourth generation of AMD EPYC 9004 series also offers a CPU model for compute-intensive workloads called Bergamo – high core count server processors. This model has up to 8 CCDs that each contain two CCXs, as shown in Figure 4.

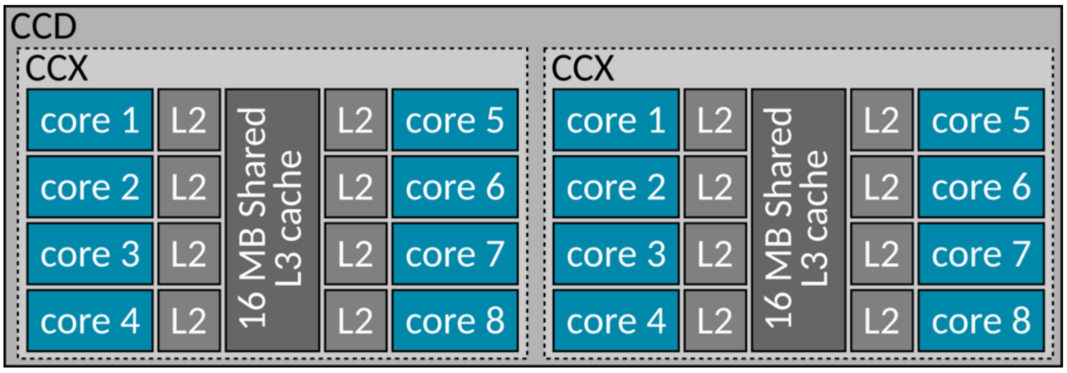


Figure 4. Layout of CCD that includes two CCXs, each with eight Zen 4c compute cores

Two core complexes are combined onto a single CCD with 16 cores and a total of 32 MB of L3 cache per die. Every CCX deploys up to eight density-optimized compute Zen 4c cores and a shared 16 MB L3 cache. The Zen 4c core is optimized for density and features the same register-transfer logic as the Zen 4 available in Genoa, but its physical layout takes less space [5]. It features the same L1I and L1D 32 KB cache size as Zen 4 and the same dedicated L2 cache at 1 MB. In contrast to regular Genoa-based CPUs, the effective L3 cache per core has been reduced to 2 MB from 4 MB on the 8-core Zen 4 CCD. It results in the new 16-core Zen 4c CCD with two 8-core CCXs, each with 16 MB of L3 cache shared among the eight cores of the CCX.

The fourth generation of EPYC 9004 series with Bergamo processors delivers up to 128 cores using eight 16-core Zen 4c CCDs [4]. In comparison, the regular Genoa processor offers up to 96 cores over twelve 8-core Zen 4 CCDs. The new layout of CCD in the Bergamo model is combined with an I/O die, 12-channel DDR5 memory, and other features available in regular AMD EPYC 9004 series processors. Figure 5 illustrates the AMD EPYC 9004 series based on the Bergamo model equipped with 8 CCDs and a central IOD.

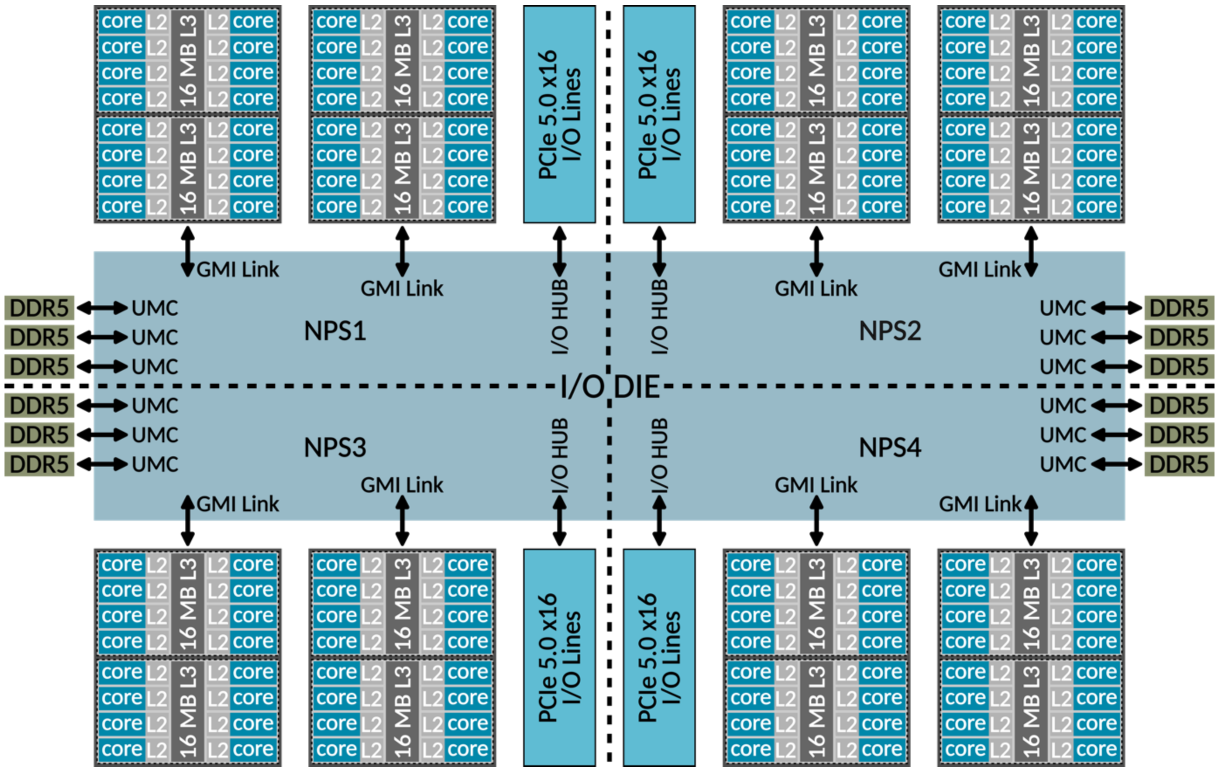


Figure 5. AMD EPYC Bergamo model with 8 CCDs and a central IOD

Although Zen 4c and Zen 4 offer similar performance, the smaller L3 cache can limit performance in bandwidth-sensitive workloads with large data sets compared to a regular Genoa-based model. In contrast, Bergamo brings the compute core density up to 128 cores per processor, delivering performance profits for compute-intensive workloads [3].

3 Target platforms

This chapter presents information about the infrastructure used to investigate the novel AMD processors. To reach this aim, a series of top-of-the-line AMD EPYC CPUs with various architectures is explored. In particular, we use seven dual-socket platforms, including four servers with Rome, Milan, Milan X, and Genoa 64-core CPUs, two servers with Genoa and Genoa X 96-core processors, as well as one server based on 128-core Bergamo CPU architecture. Access to all the tested platforms is provided by the AMD company. A brief specification of the systems is presented in Table 2.

Table 2. Target platforms specification

Platform	Sockets	CPU Type	Cores	Codename	OS	Memory
GIGABYTE H262-Z63-00	2	7742	2x 64	Rome	CentOS 8	512GB DDR4-3200 Samsung
GIGABYTE H262-Z63-00	2	7763	2x 64	Milan	CentOS 8	512GB DDR4-3200 Samsung
Asus RS720A-E11-RS12	2	7773X	2x 64	Milan X	Ubuntu Server 23.10	512GB DDR4-3200 Samsung
AMD Titanite 4G	2	9554	2x 64	Genoa	Rocky 8.7	768GB DDR5-4800 Micron
AMD Titanite 4G	2	9654	2x 96	Genoa	Rocky 8.7	768GB DDR5-4800 Micron
AMD Titanite 4G	2	9684X	2x 96	Genoa X	Rocky 8.7	768GB DDR5-4800 Micron
AMD Titanite 4G	2	9754	2x 96	Bergamo	Rocky 8.7	768GB DDR5-4800 Micron

Table 3 summarizes the parameters of the AMD EPYC processors family. The studied 2nd, 3rd, and 4th generations of AMD EPYC include the different numbers of cores and are clocked by the base frequency clocks listed in Table 3. 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 multithreading (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. All the platforms represent a group of ccNUMA shared memory architectures combining whole memory regions using 2x4 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.

Table 3. A single CPU specification of AMD EPYC family [3]

CPU Type	7742	7763	7773X	9554	9654	9684X	9754
Generation	2nd	3rd	3rd	4th	4th	4th	4th
Codename	Rome	Milan	Milan X	Genoa	Genoa	Genoa X	Bergamo
Architecture	Zen2	Zen3	Zen3	Zen4	Zen4	Zen4	Zen4c
Launch Date	2019	2021	2021	2022	2022	2023	2023
Cores	64	64	64	64	96	96	128
Clock [GHz] (Turbo)	2.25 (3.4*)	2.45 (3.5*)	2.2 (3.5*)	3.1 (3.75)	2.4 (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	32	96	16
#CCX in CCD	2	1	1	1	1	1	2
L3 CCD [MB]	32	32	96	32	32	96	32
#CCD in CPU	8	8	8	8	12	12	12
Total L3 [MB]	256	256	768	256	384	1152	256
L3 per core	4	4	12	4	4	12	2
Memory Type	8-channel DDR4 3200	8-channel DDR4 3200	8-channel DDR4 3200	12-channel DDR5 4800	12-channel DDR5 4800	12-channel DDR5 4800	12-channel DDR5 4800
NUMAs	4	4	4	4	4	4	4

*Maximum frequency achievable by any single core

The most crucial innovation in AMD EPYC processors is the hybrid multi-die architecture first introduced in 2nd generation of EPYC processors. The design of all CPUs consists of a single central I/O hub (or I/O Die) 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. In contrast, every 128-core Bergamo CPU provides eight CCDs with 16 cores per die.

Every ROME CCD consists of two core complexes (CCXs); each of them embraces four cores and 16 MB of L3 cache. As a result, each CCD provides 32 MB of L3 cache. A single core contains the L2 inclusive cache of 512 KB size and the L1-D cache of 32KB size. The core complex die of the MILAN chip holds a single core complex with eight cores and 32 MB of shared L3 cache. Additionally, the configuration of MILAN CCD can be augmented with 3D V-Cache technology to bring the L3 cache capacity to 96 MB in the Milan X processor. Similarly to Rome, every Milan or Milan X core offers 512 KB size and the L1-D cache of 32KB size.

The Genoa CCD used in 64- and 96-core processors consists of one CCX with eight cores, a dedicated 1 MB L2 cache per core, and a 32 MB L3 cache shared between the eight cores. Similarly to Milan X, applying the 3D V-Cache technology in the 4th generation of AMD EPYC CPU enables extending the shared 32 MB L3 cache with 64 MB additional layered above, bringing the per-die total L3 cache to 96 MB in Genoa X. In contrast to Genoa and Genoa X, the CCD in Bergamo combines two core complexes with eight cores each. Every core complex includes a 16 MB shared L3 cache (32 MB per CCD).

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, the per-die L3 cache is augmented three times, bringing it 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.

4 Benchmark methodology

In this stage of the HiDALGO2 project, we focus on benchmarking new-generation AMD EPYC processors, investigating two OpenFOAM-based applications. 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. In this section, we present the general overview of single-node benchmarking activities for AMD-based HPC infrastructures and initial findings on their profiling results.

4.1 Benchmark procedure

The core component of the benchmark is a set of the OpenFOAM v2212 [13] kernels, which helps us indicate the impact of new HPC hardware features on the final performance efficiency. We distinguish two groups for the benchmarking based on OpenFOAM use cases. In the first group of performance experiments, the solver simpleFoam is tested and run in the well-known test case motorBike, which is available in all OpenFOAM variants and versions. For the second group of the tests, the simpleFoam and pimpleFoam solvers are investigated as more computationally expensive parts of the UAP pilot application. In the next sections, we outline the underlined computing kernels in depth.

In our experiments, all OpenFOAM kernels are compiled with the AOCC compiler 4.1.0 and linked against OpenMPI 4.1.5 pre-installed by platform vendors. The AOCC compiler is used with the optimization flag -O3 and corresponding compiler arguments that support the given AMD microarchitectures. Table 4 outlines the software environments for the OpenFOAM kernels on AMD EPYC platforms.

Table 4. Software configuration for OpenFOAM kernels

Platform	OpenFOAM	Scotch	Compiler	Comp. flags	MPI
2x 64-core 7742 (Rome)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver2	OpenMPI 4.1.5
2x 64-core 7763 (Milan)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver3	OpenMPI 4.1.5
2x 64-core 7773 (Milan X)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver3	OpenMPI 4.1.2
2x 64-core 9554 (Genoa)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver4	OpenMPI 4.1.5
2x 96-core 9654 (Genoa)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver4	OpenMPI 4.1.5
2x 96-core 9684X (Genoa X)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver4	OpenMPI 4.1.5
2x 128-core 9754 (Bergamo)	v2212	6.1.0	AOCC 4.1.0	-O3 -march=znver4	OpenMPI 4.1.5

It is worth underlining that we also investigated the impact of different compilers on application performance for AMD CPUs, including GCC, AOCC, and Intel C/C++ solutions. We reveal that for the studied OpenFOAM kernels, the performance comparison between the AOCC and GCC compilers does not differ significantly, with negligible advantages for AOCC. The Intel compiler returns the lowest performance results.

We investigate all OpenFOAM kernels on seven platforms outlined in Table 2 by testing different problem sizes. All the systems operate with SMT disabled, simplifying and increasing the reliability of the platform comparison and performance evaluation processes more reliable. The turbo boost is enabled and NPS4 mode is set on all the systems. 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 (see Table 5).

Table 5. Range of clock speed measured during execution of the OpenFOAM kernels

Platform	Range of Clock [GHz]
2x 64-core 7742 (Rome)	2.77 – 3.10
2x 64-core 7763 (Milan)	2.91 – 3.16
2x 64-core 7773 (Milan X)	2.74 – 2.96
2x 64-core 9554 (Genoa)	3.74 – 3.74
2x 96-core 9654 (Genoa)	3.37 – 3.50
2x 96-core 9684X (Genoa X)	3.24 – 3.40
2x 128-core 9754 (Bergamo)	2.81 – 3.10

Additionally, we select the optimal policy for binding and mapping MPI processes to physical computing resources. As a result, a single MPI process is spawned per physical core and utilizes a memory region close to this core. It helps us avoid performance constraints when many MPI processes are pinned to a given core.

To ensure the reliability of measurements, every test is repeated at least five times for each platform and a given problem size. We note here that all measurements reported in the subsequent sections show the execution time for five runs.

4.2 Profiling methodology

4.2.1 AMD µProf performance tool

We employ AMD µProf [7] software profiling analysis tool to reveal the applications' interaction with the hardware. The AMD µProf performance tool analyses and monitors AMD Zen-based microarchitecture processors, and is used for hotspot measurements, performance counters, and other characteristic data of an application. The AMD µProf 4.1 version is employed for all the tested platforms.

In this work, we use a specific tool called AMDuProfPcm [8], which allows more 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, L3, and DF performance events count values and reports various metrics. Table 6 outlines studied groups of metrics supported by a given CPU architecture. Every group includes a set of metrics addressed for every CPU architecture. For example, the group called l3 consists of four metrics and collects profile data for every CCX, including:

1. *L3 Access* – the L3 cache accesses [PTI],
2. *L3 Miss* – the L3 cache miss [PTI],
3. *L3 Miss (%)* – the L3 cache miss percentage [%],
4. *Ave L3 Miss Latency* – the average L3 miss latency in core cycles.

The detailed description of all metrics is presented in [8].

Table 6. Applied metric groups for the profiling

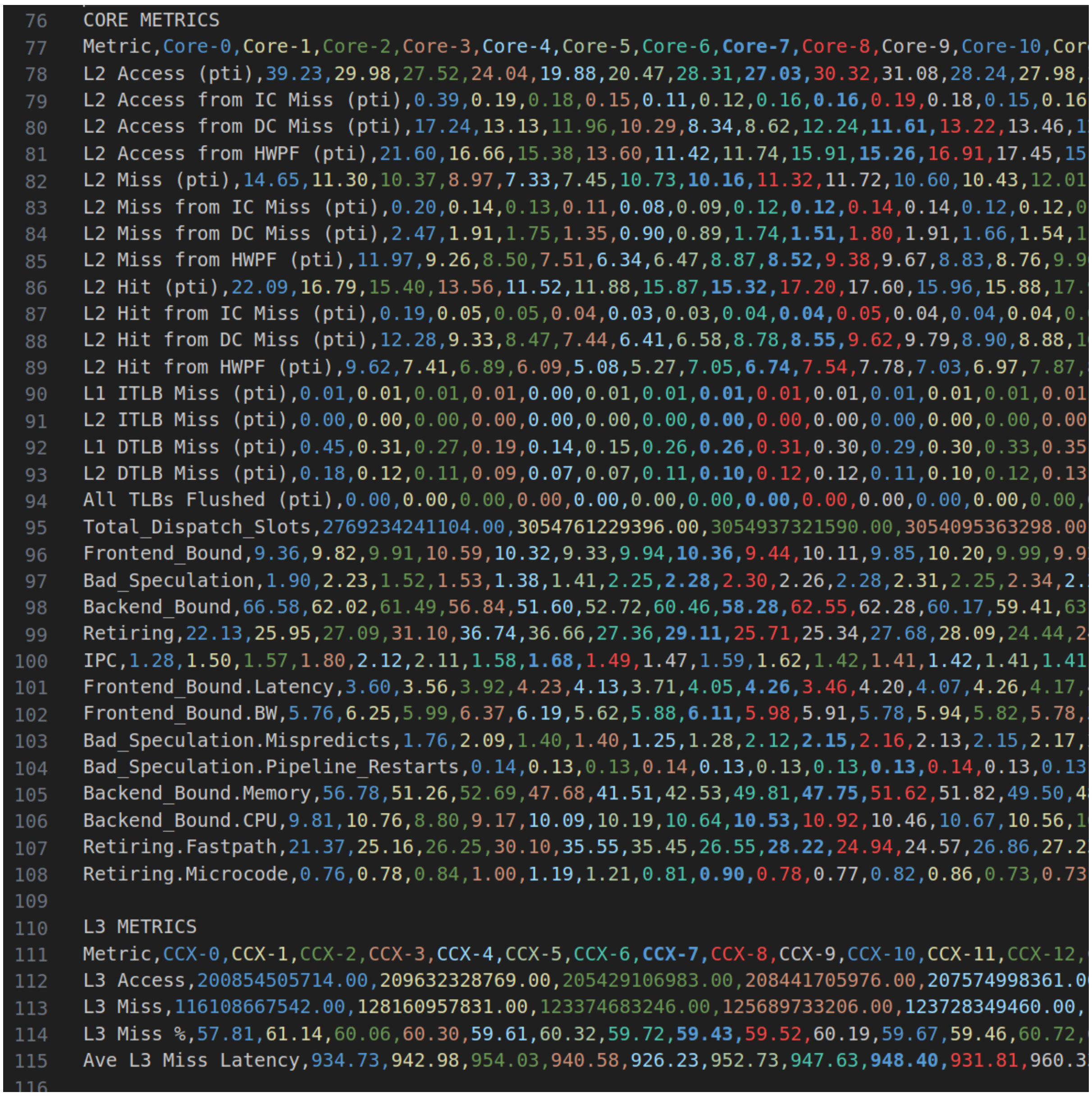
Architectures	Metric Group
Zen 2	ipc,l1,l2,l3,tlb
Zen 3	ipc,dc,l1,l2,l3,tlb
Zen 4	ipc,dc,l1,l2,l3,tlb,pipeline_util

To collect the profile data for a single application run and a given platform, we use the AMDuProfPcm tool with properly selected groups of matrices, for example:

```
$AMDuProfPcm -m ipc,dc,l1,l2,l3,tlb -C -a -o out.csv -- application
```

This example collects IPC, all cache levels and TLB hardware metrics of all the cores and cumulative data at the end of the profile duration for CPUs based on Zen 3 architecture. The profile data are reported in .csv format. Figure 6 shows the partial result of one such analysis. In addition, we observer that the relative cost of using AMDuProfPcm tool is negligible.

A single profile report consists of data for up to 48 metrics of every core or each CCD (see Figure 6). In consequence, the post-processing investigation is required to accomplish the complete profiling process for seven platforms, three OpenFOAM kernels, and different problem sizes.



```
CORE METRICS
Metric,Core-0,Core-1,Core-2,Core-3,Core-4,Core-5,Core-6,Core-7,Core-8,Core-9,Core-10,Cor
L2 Access (pti),39.23,29.98,27.52,24.04,19.88,20.47,28.31,27.03,30.32,31.08,28.24,27.98,
L2 Access from IC Miss (pti),0.39,0.19,0.18,0.15,0.11,0.12,0.16,0.16,0.19,0.18,0.15,0.16
L2 Access from DC Miss (pti),17.24,13.13,11.96,10.29,8.34,8.62,12.24,11.61,13.22,13.46,
L2 Access from HWPF (pti),21.60,16.66,15.38,13.60,11.42,11.74,15.91,15.26,16.91,17.45,15
L2 Miss (pti),14.65,11.30,10.37,8.97,7.33,7.45,10.73,10.16,11.32,11.72,10.60,10.43,12.01
L2 Miss from IC Miss (pti),0.20,0.14,0.13,0.11,0.08,0.09,0.12,0.12,0.14,0.14,0.12,0.12,
L2 Miss from DC Miss (pti),2.47,1.91,1.75,1.35,0.90,0.89,1.74,1.51,1.80,1.91,1.66,1.54,
L2 Miss from HWPF (pti),11.97,9.26,8.50,7.51,6.34,6.47,8.87,8.52,9.38,9.67,8.83,8.76,9.
L2 Hit (pti),22.09,16.79,15.40,13.56,11.52,11.88,15.87,15.32,17.20,17.60,15.96,15.88,17.
L2 Hit from IC Miss (pti),0.19,0.05,0.05,0.04,0.03,0.03,0.04,0.04,0.05,0.04,0.04,0.04,0.
L2 Hit from DC Miss (pti),12.28,9.33,8.47,7.44,6.41,6.58,8.78,8.55,9.62,9.79,8.90,8.88,
L2 Hit from HWPF (pti),9.62,7.41,6.89,6.09,5.08,5.27,7.05,6.74,7.54,7.78,7.03,6.97,7.87,
L1 ITLB Miss (pti),0.01,0.01,0.01,0.01,0.00,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01,0.01
L2 ITLB Miss (pti),0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00
L1 DTLB Miss (pti),0.45,0.31,0.27,0.19,0.14,0.15,0.26,0.26,0.31,0.30,0.29,0.30,0.33,0.35
L2 DTLB Miss (pti),0.18,0.12,0.11,0.09,0.07,0.07,0.11,0.10,0.12,0.12,0.11,0.10,0.12,0.13
All TLBs Flushed (pti),0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,
Total_Dispatch_Slots,2769234241104.00,3054761229396.00,3054937321590.00,3054095363298.00
Frontend_Bound,9.36,9.82,9.91,10.59,10.32,9.33,9.94,10.36,9.44,10.11,9.85,10.20,9.99,9.9
Bad_Speculation,1.90,2.23,1.52,1.53,1.38,1.41,2.25,2.28,2.30,2.26,2.28,2.31,2.25,2.34,2.
Backend_Bound,66.58,62.02,61.49,56.84,51.60,52.72,60.46,58.28,62.55,62.28,60.17,59.41,63
Retiring,22.13,25.95,27.09,31.10,36.74,36.66,27.36,29.11,25.71,25.34,27.68,28.09,24.44,2
IPC,1.28,1.50,1.57,1.80,2.12,2.11,1.58,1.68,1.49,1.47,1.59,1.62,1.42,1.41,1.42,1.41,1.41
Frontend_Bound.Latency,3.60,3.56,3.92,4.23,4.13,3.71,4.05,4.26,3.46,4.20,4.07,4.26,4.17,
Frontend_Bound.BW,5.76,6.25,5.99,6.37,6.19,5.62,5.88,6.11,5.98,5.91,5.78,5.94,5.82,5.78,
Bad_Speculation.Mispredicts,1.76,2.09,1.40,1.40,1.25,1.28,2.12,2.15,2.16,2.13,2.15,2.17,
Bad_Speculation.Pipeline_Restarts,0.14,0.13,0.13,0.14,0.13,0.13,0.13,0.13,0.14,0.13,0.13
Backend_Bound.Memory,56.78,51.26,52.69,47.68,41.51,42.53,49.81,47.75,51.62,51.82,49.50,4
Backend_Bound.CPU,9.81,10.76,8.80,9.17,10.09,10.19,10.64,10.53,10.92,10.46,10.67,10.56,
Retiring.Fastpath,21.37,25.16,26.25,30.10,35.55,35.45,26.55,28.22,24.94,24.57,26.86,27.2
Retiring.Microcode,0.76,0.78,0.84,1.00,1.19,1.21,0.81,0.90,0.78,0.77,0.82,0.86,0.73,0.73

L3 METRICS
Metric,CCX-0,CCX-1,CCX-2,CCX-3,CCX-4,CCX-5,CCX-6,CCX-7,CCX-8,CCX-9,CCX-10,CCX-11,CCX-12,
L3 Access,200854505714.00,209632328769.00,205429106983.00,208441705976.00,207574998361.0
L3 Miss,116108667542.00,128160957831.00,123374683246.00,125689733206.00,123728349460.00,
L3 Miss %,57.81,61.14,60.06,60.30,59.61,60.32,59.72,59.43,59.52,60.19,59.67,59.46,60.72,
Ave L3 Miss Latency,934.73,942.98,954.03,940.58,926.23,952.73,947.63,948.40,931.81,960.3
```

Figure 6. Sample AMDuProfPcm analysis results

Additionally, we use the AMDuProfCLI MPI trace tool [8] to analyse and estimate the communication costs among an MPI application's ranks. The light-weight tracing (LWT) mode is selected to analyse an application for this purpose. The report is generated in CSV format on the fly during the collection stage. This mode traces MPI application activity and reports, communicator summary, rank summary, P2P API summary, and collective API summary. Following is an example of a command to LWT traces an MPI application using AMDuProfCLI:

```
$ mpirun -np <number of processes>./AMDuProfCLI collect -trace
mpi=lwt,openmpi -o <output_directory> <application>
```

4.2.2 FVOPS performance metric

To allow the performance between the various systems to be compared directly, the metric FVOPS (Finite VOlumes solved Per Second) is introduced [6]. The metric FVOPS is similar to the metric FLOPS (FLOating point operations Per Second) commonly used to measure CPU performance, however using the amount of finite volume elements solved using OpenFOAM, instead of the amount of floating-point operations executed by the CPU.

The FVOPS metric is calculated as:

$$\mathrm{FVOPS} = \frac{\text{amount of finite volume elements in the grid}}{\text{time spent per time step or iteration}}$$

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 presented in this report use test cases with fixed conditions in all systems, only varying the grid size. FVOPS is calculated per node (as is usual also for FLOPS). To facilitate the comparison, however, 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 notice that these local maxima occur at different values of grid element per rank when utilizing different processor types.

5 Performance results

This section presents the performance results obtained for OpenFOAM-based applications including motorBike and UAP use cases.

5.1 MotorBike use case

The motorBike use case is one of OpenFOAM's tutorials, which 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.

There are two reasons why the motorbike use case was benchmarked and analysed in this report. First, it is one of the most investigated use cases when assessing the performance of OpenFOAM. Second, it has many similar properties to the UAP-FOAM solver of the Urban Air Project use case, as both work with unstructured meshes and solves the RANS equations for incompressible fluid. Note, that while there is another tutorial, "wind around buildings", which resembles the UAP use case slightly more, we chose motorbike, as it has many more benchmarks in the literature.

5.1.1 MotorBike overview

In this section, an overview of the motorbike use case is presented [9]. First, the geometry of the simulation is presented including boundary condition properties of the simulated domain followed by mesh generation and properties. The governing equations of the incompressible RANS are discussed next, followed by the numerical methods applied for solutions within the OpenFOAM framework.

The geometry of the use case models external airflow around a geometry resembling a motorbike in a volume of 20x8x8 meters. Boundary conditions for ground and motorbike (motorbike group and lowerWall) resemble a wall using "noslip", while domain walls parallel to airflow (upperWall, front and back) use "slip". A fixed velocity of 20 m/s is set at the inlet inwards, perpendicular to the surface. An inlet-outlet condition is set to the outlet patch. See Figure 7 for geometry.

The mesh is generated using the OpenFOAM tool blockMesh and snappyHexMesh. First, blockMesh is used to generate a cartesian base mesh of one-meter cubes. Then, snappyHexMesh is used to refine the mesh around the motorbike, snap the cells to the motorbike surface and add a layered mesh of one layer. This results in a mesh of 355,474 cells with an average cell size of 3.6 litres.

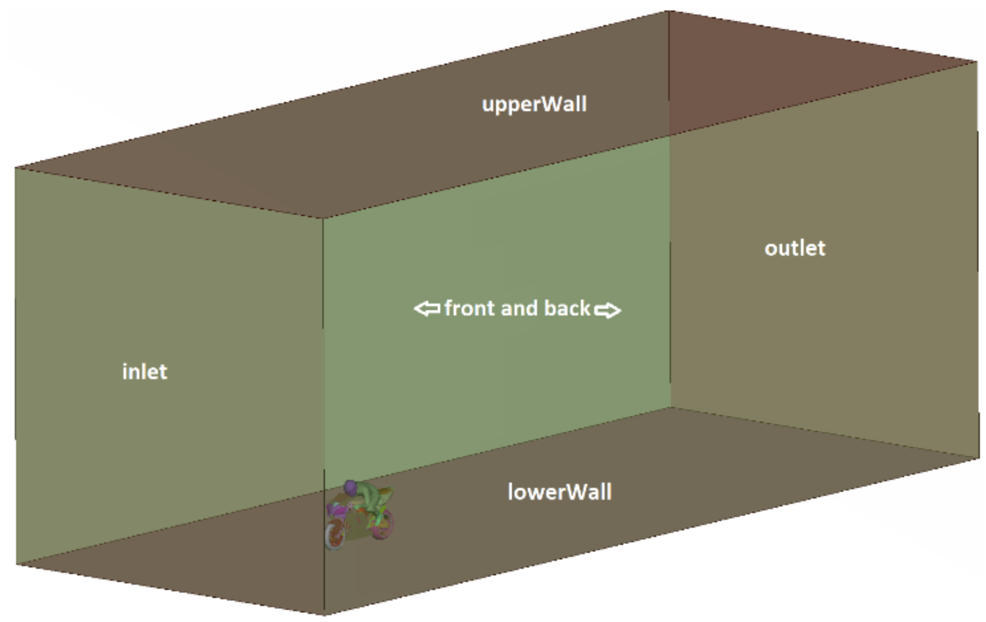


Figure 7. Simulation domain for motorbike use case showing the patch names and bike positioning

The settings, equations and numerical parameters are as of in the tutorial use case of OpenFOAM v2112 [13]. The Reynolds-averaged Navier-Stokes equations with k-ω-SST turbulence model are solved for incompressible fluid [10][11]. The solver simpleFoam is used in the tutorial, which applies the consistent Semi-Implicit Method for Pressure Linked Equations (SIMPLEC method) for solving the continuity and momentum equations [12].

5.1.2 Configuration of motorBike use case

The benchmarks in this section strive to follow the initial settings of the tutorial. However, the following adjustments were made for the performance experiments in this report compared to the original use case to improve stability:

1. Instead of 500 iterations, the simulation is only run for 200 iterations.
2. Instead of hierarchical decomposition, the scotch library is used. Also, domain decomposition was done with decomposePar at the beginning of the simulation after importing the pre-generated mesh with fluent3DMeshToFoam. No multilevel decomposition was used.
3. The number of subdomains was adjusted according to the number of CPU cores of the underlying architecture. One subdomain was generated per CPU core.
4. The collated file format was enabled to limit the number of files that were stored on the storage.
5. The number of non-orthogonal correctors (nNonOrthogonalCorrectors) was set to 1. Non-orthogonality is a measure in which the face normal are not parallel

to the vector that connects the cell centre to the face centre. These correctors improve stability and convergence on non-orthogonal meshes.

6. The under-relaxation factors were adjusted in the following ways for equations: U: 0.7, k: 0.3 and omega: 0.3; and for field p: 0.3. While under-relaxation factors do affect performance, they were selected in a way that they improve stability for finer meshes. Also, these factors are the same across all mesh sizes.

To conduct the benchmarks, various size meshes were generated using blockMesh and snappyHexMesh. All these meshes use the same geometry, albeit with different base cartesian cell sizes and resolutions to get various mesh cell counts detailed in Table 7.

Table 7. Meshes for motorbike use case

Name	Cell count	Volume [m3]	Avg. cell vol. [m3]
xsmall	167 555	1279,68	7,637E-03
small	355 474	1279,68	3,600E-03
smid	603 547	1279,68	2,120E-03
mid	1 897 187	1279,68	6,745E-04
high	4 166 441	1279,68	3,071E-04
mhigh	6 657 491	1279,68	1,922E-04
uhigh	11 900 363	1279,68	1,075E-04
xhigh	39 400 357	1279,68	3,248E-05

Mesh sizes were selected on a wide range so that various performance behaviours may be observed. All meshes were generated beforehand and exported to Fluent MSH format. Meshes are reimported with the fluent3DMeshToFOAM utility. This method is used for the following reasons:

1. Pre-generating meshes will make sure that the same exact mesh is used across all benchmarks and platforms.
2. Fluent MSH is a single-file mesh format that supports polyhedral cells and is supported by various preprocessing tools.
3. Fluent MSH format is also used within the UAP framework.

Performance experiments are conducted in the following manner. For every architecture and mesh size, the simulation described in the previous section is executed 5 times. Execution time is measured in seconds with a resolution of 0.01 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 a runtime of 199 iterations is measured. For purposes of comparison, the median of the five measurements is calculated, effectively dropping the smallest and largest two values.

5.1.3 Performance experiments

Measurements are grouped by mesh size on different plots in Figure 8 and Figure 9, all measured execution time is plotted for different architectures. Arrows indicate a bigger than one speedup, including the actual speedup value. Red values indicate the largest speedup. Unique measurements do fluctuate. The rate of fluctuations decreases with mesh size.

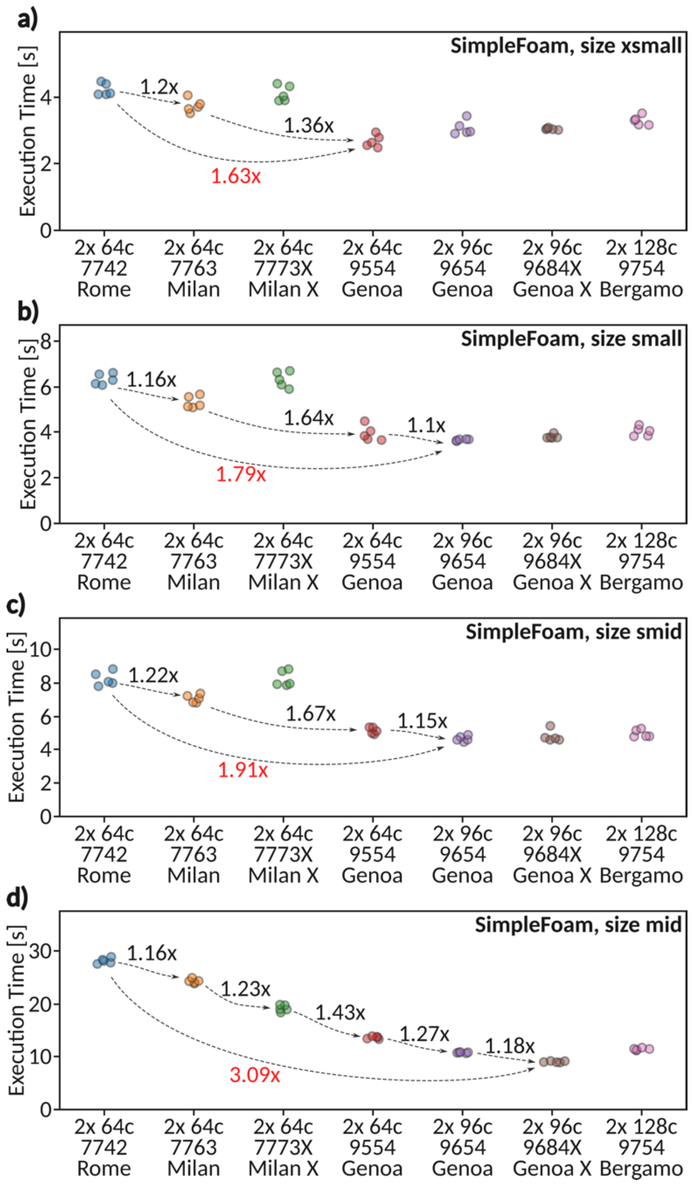


Figure 8. Performance results obtained for motorBike simpleFoam solver on a variety of AMD CPUs and different mesh sizes, including a) xsmall, b) small, c) smid, and d) mid

Investigating the results, for all mesh sizes Rome exhibits the lowest performance, except for small mesh sizes, where performance with Milan-X is tied. The improvement of Milan over Rome is about a factor of 1.2 for smaller mesh sizes, which diminishes, as mesh size gets bigger. The advantage of Milan-X over Milan kicks in at 'mid' mesh size, tops at 'mhigh' with a factor over 1.5, however, diminishes for larger mesh sizes. The 64-core Genoa brings significant improvement compared to all previous architectures across all mesh sizes with 1.2 for mid-size meshes and 1.7 for the largest one. The 96-core Genoa brings a smaller, 1.1-1.3 factor improvement compared to the 64-core version, except for the smallest mesh size, for which it is slower. Genoa-X is not faster on smaller sizes; however, it gets an approximately 1.5-factor speedup for the larger ones. The 128-core Bergamo does not bring any significant improvement. Its performance is on par with the 96-core Genoa.

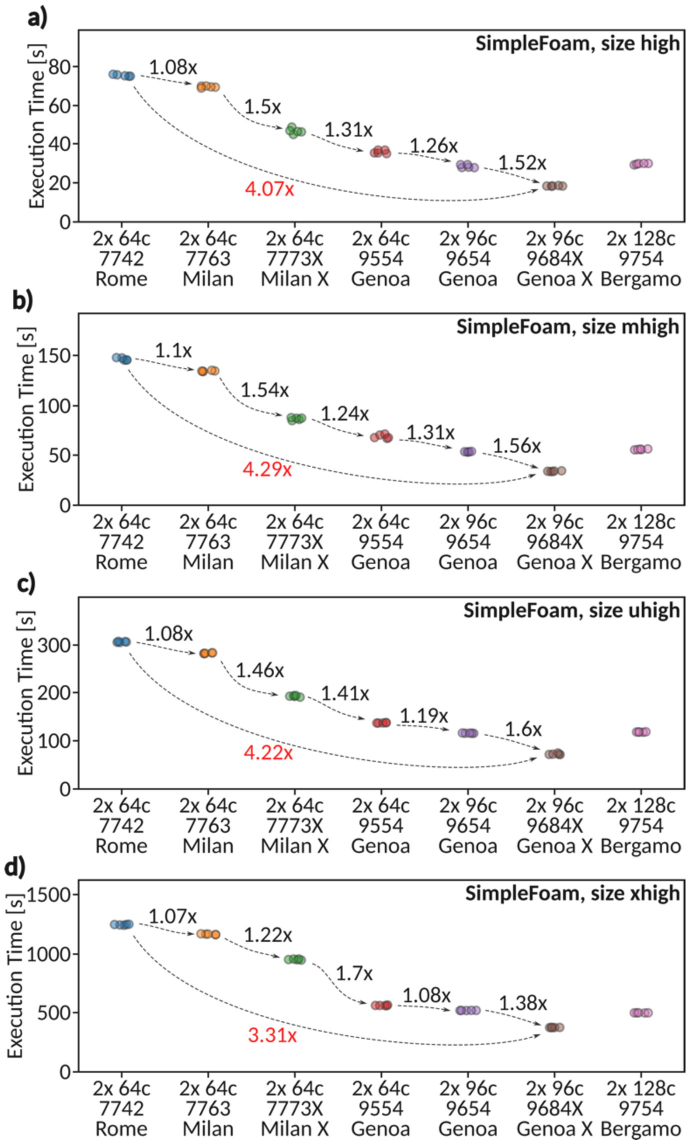


Figure 9. Performance results obtained for motorBike simpleFoam solver on a variety of AMD CPUs and different mesh sizes, including a) high, b) mhigh, c) uhigh, and d) xhigh

5.1.4 Performance analysis

To compare performance across multiple platforms, the Finite Volume Operations Per Second (FVOPS) metric was calculated for all platforms and mesh sizes, effectively. Results are grouped as 64-core architectures (Figure 10a) and Genoa-Bergamo (Figure 10b). For every architecture, the calculation performance in FVOPS is plotted versus the per-core cell count.

The FVOPS curve shows a similar shape for all architectures. It increases up to a turning point and decreases afterwards. The sweet spot for maximum performance is about 5k cells per core for Rome and Milan, 15k for Milan-X and 64-core Genoa, 7k for Bergamo, 10k for 96-core Genoa and 22k for Genoa-X. For small cell count meshes, up to 5k, architectures before Genoa exhibit lower performance than 64-core Genoa. Differences between Rome, Milan and Milan-X are not that dominant. Also, 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, X architectures dominate their own platform. Milan-X shows an advantage in the 15k to 100k cell per core range, while Genoa-X has an advantage in the 10k to 200k cells per core range. This makes both products a viable choice for the cell count range of 2M to 13M and 1M to 20M, respectively. It results, that for larger problems the cache size per core determines attainable performance. In contrast, for the Bergamo-based system, throwing more cores and less L3 cache per core strongly limits the performance advantage over other platforms and consequently shows no advantage on any investigated mesh sizes.

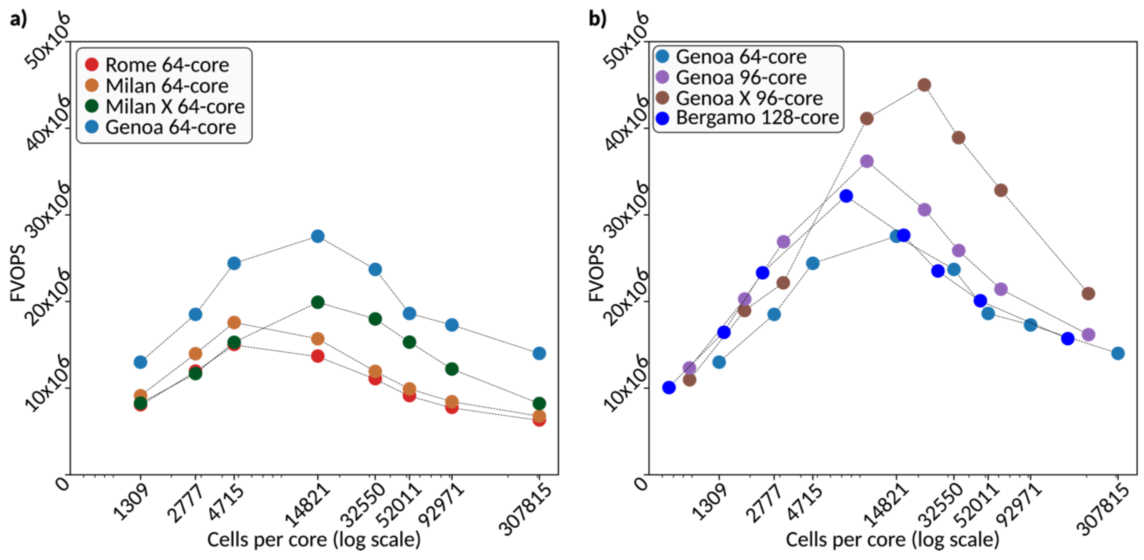


Figure 10. FVOPS performance metrics determined for motorBike simpleFoam solver on a) 2x 64-core and b) Genoa-/Bergamo-based platforms

Our investigation of motorBike is completed by a performance analysis. To reach this aim, we combine AMD µProf (see Chapter 4.2.1) and FVOPS performance metrics (see Chapter 4.2.2) to collect profile data to identify the performance bottlenecks and correlation between the application and the hardware. The profile data are collected for seven computing platforms and eight domain sizes (Figure 11 and Figure 12). In this stage of the HiDALGO2 project, we focus on a set of performance metrics including:

- L2 miss ratio from L1DC [%] – the number of misses from L1 data cache over the total number of accesses from L1 data cache,
- L2 miss ratio from HWPF [%] – the number of misses from L2 cache hardware prefetching over the total number of accesses from L2 cache hardware prefetching,
- L2 miss ratio total [%] – the number of misses from L2 cache over the total number of accesses from L2 cache,
- L3 miss ratio [%] – the number of misses from L3 cache over the total number of accesses from L3 cache,
- MPI communication to computation ratio [%] – the average communication time over the computation time

Figure 11 illustrates the performance report obtained for four computing platforms based on 64-core CPUs, including Rome (Figure 11a), Milan (Figure 11b), Milan X (Figure 11c), and Genoa (Figure 11d) microarchitectures. Figure 11 shows that the L3 cache miss ratio rises consecutively when the mesh size increases. Considering all four platforms, this rate is around 10% for the smaller mesh size. In contrast, for the largest mesh sizes, it reaches around 80% on platforms with 2x 256MB of L3 cache size available in Milan, Rome, and Genoa, as well as 67% for Milan X, which offers a larger L3 cache capacity.

It is also worth noting that, in comparison with Milan X and Milan (Figure 11b and Figure 11c), the three-time larger cache size reduces the L3 cache miss rate by up to 28 percentage points for Milan X over the regular Milan CPUs. It results in noticeable performance profits for sizes mid, high, mhigh, and uhigh (see Figure 8 and Figure 9). As expected, the large L3 cache in Milan X does not overcome regular Milan CPUs for relatively smaller sizes where the L3 cache is not the performance bottleneck. Considering the xhigh mesh size and larger data sets, we observe that the data volume requirement exceeds the L3 capacity in Milan X, and - as a consequence - the performance advantage decreases over regular Milan CPUs.

According to the FVOPS performance metric (Figure 10a), which decreases for relatively larger mesh sizes, the L3 cache size affects the performance limits. 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 volumes between L3 and main memory. As a result, both L3 cache size and the speed of the main memory subsystem play a key role in attainable performance. Considering platforms with 64-core CPUs (Figure 11), 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 Milan X.

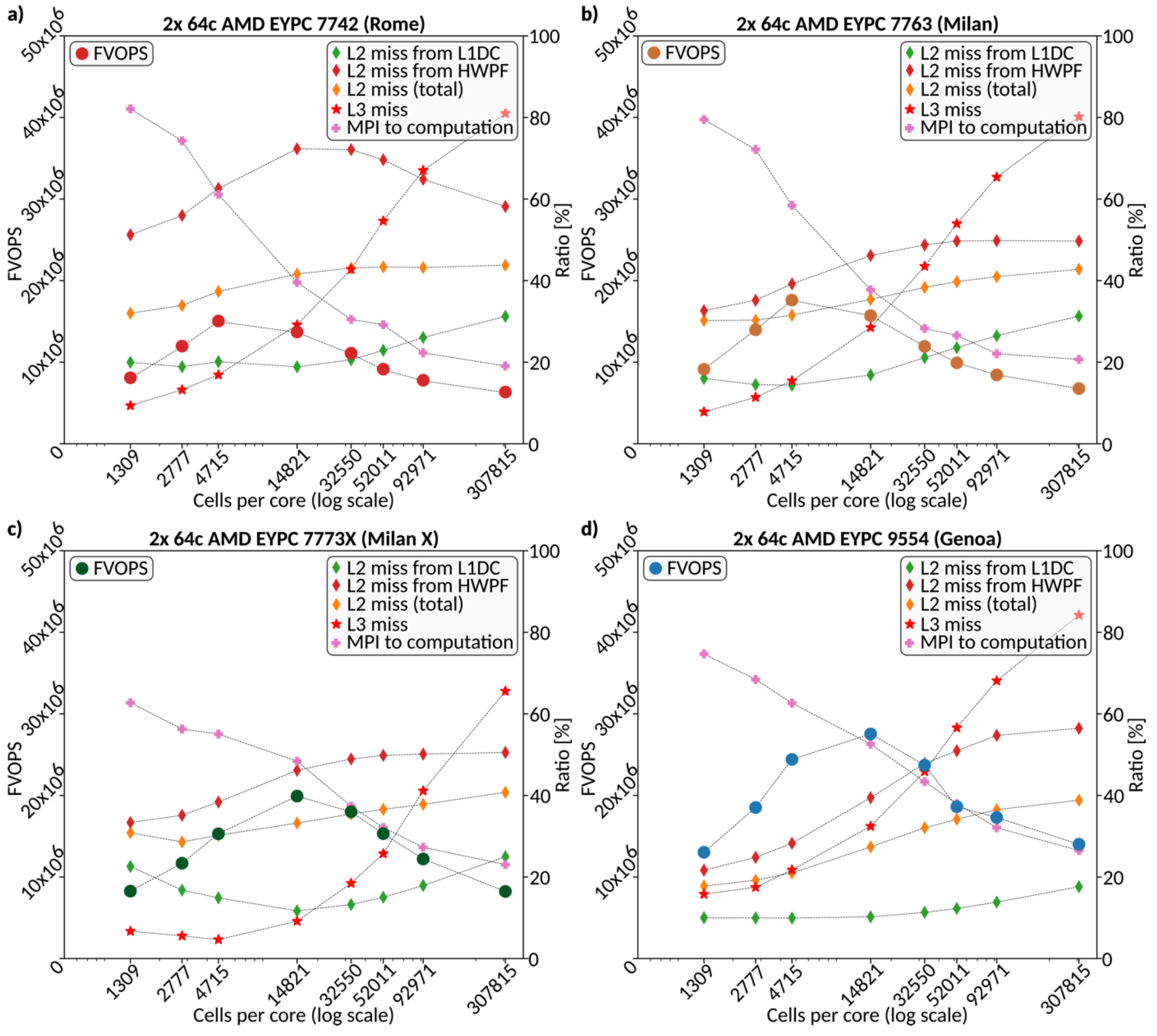


Figure 11. Performance analysis determined for motorBike simpleFoam solver on 2x 64-core HPC platforms, including a) Rome, b) Milan, c) Milan X, and d) Genoa architectures

Additionally, the Genoa-based processors are equipped with 1MB of L2 cache, which is twice as large as that of prior generations. As shown in Figure 11d, it enables miss ratio reduction for all L2-based metrics and performance advantages in comparison to other 64-core platforms (Figure 11a-c). We observe that the expanded L2 capacity in Genoa-based CPUs provides performance advantages for relatively smaller mesh sizes compared to the prior generations of AMD EPYC CPUs (see Figure 10a).

Figure 12 demonstrates the performance report obtained for platforms with 96-core Genoa CPUs (Figure 12a), 96-core Genoa X CPUs (Figure 12b) and 128-core Bergamo CPUs (Figure 12c). As expected, 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. As shown in Figure 12a-c, 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 from a large L3 cache in Genoa X is not noticeable for smaller sizes where the L3 is not the performance bottleneck.

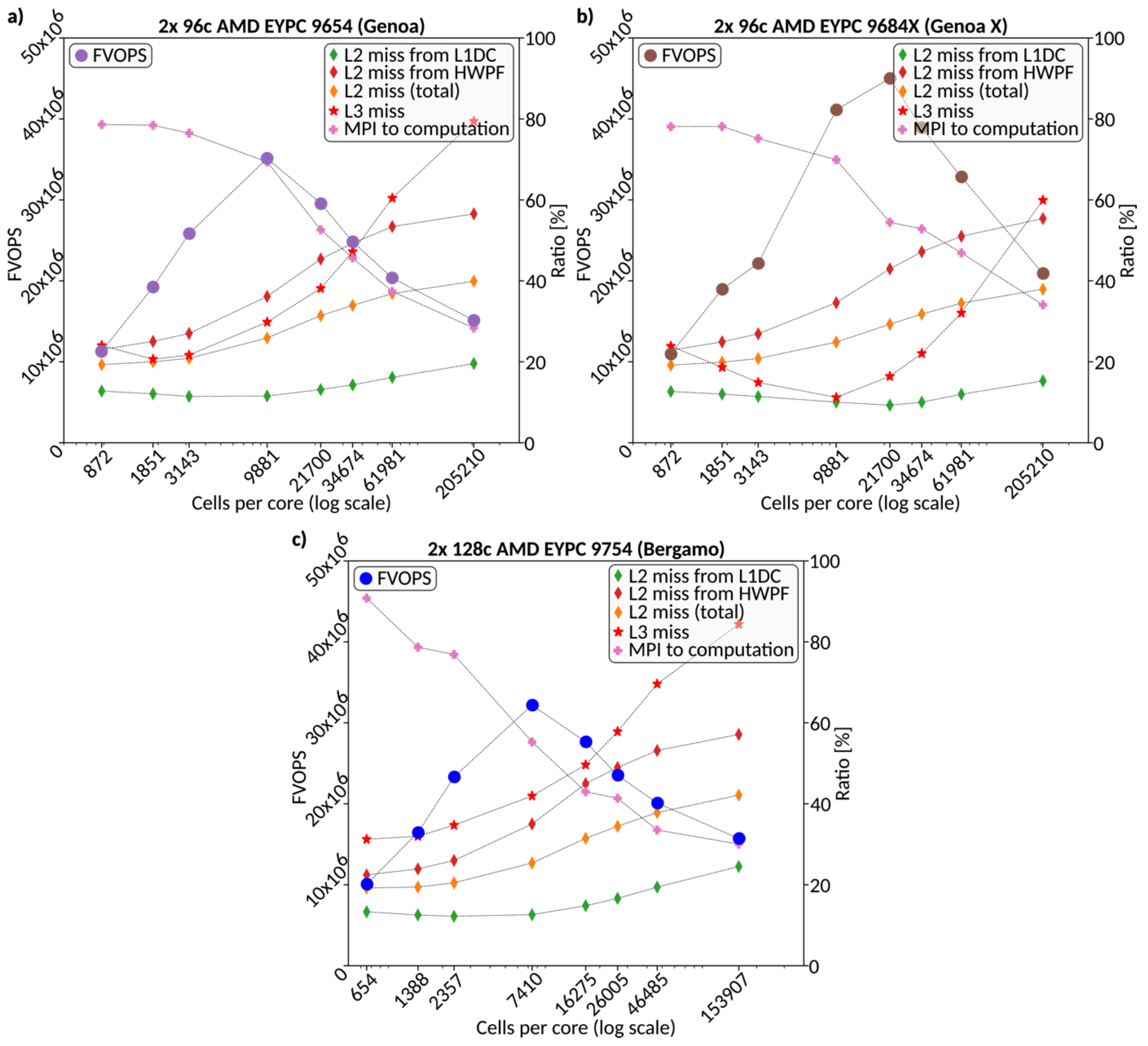


Figure 12. Performance analysis determined for motorBike simpleFoam solver on the 4th generation of AMD EPYC CPUs, including a) Genoa, b) Genoa X, and c) Bergamo architectures

It is also worth noting that the Bergamo-based platform features the worst L3 cache miss ratio trend of all the studied platforms (Figure 11 and Figure 12). This is mainly due to the smaller L3 cache size per core compared to other processors (see Table 3). Considering the memory-bound nature of the simpleFoam kernel, 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 (Figure 9). 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 rather smaller sizes, we expect that L3 compulsory misses are noticeable where the first time a memory location is read.

As shown in Figure 12, the 1MB of L2 cache per core offered by 4th generation AMD EPYC CPUs reduces the L2 miss ratio trend over previous generations. This results in better attainable performance, especially for smaller mesh sizes.

Furthermore, considering rather smaller mesh sizes, we also observe that 96-core processors do not offer a performance advantage over 64-core processors (see Figure 8). Since both processors feature the same memory subsystem, this effect can be explained by the 64-core CPU offering more memory bandwidth per core than 96-core to keep core-memory data movements.

Figure 11 and Figure 12 also outline the MPI communication over computation ratio. This metric reveals that MPI communication mainly limits performance for smaller mesh sizes since the communication cost overcomes computation parts. This fact helps us indicate MPI communication as a crucial performance bottleneck for further co-design activity when using extensive computing resources. In this case, a large computational domain is typically distributed into small sub-domains and processed by a large number of cores, where it is expected – as shown in Figure 11 and Figure 12 – that the MPI communication may play a key role in performance limits.

5.2 UAP use case

The Urban Air Project use case is one of the pilots of HiDALGO2 [1]. The CFD module simulates airflow and pollution dispersion in an urban area. In the current report, the OpenFOAM-based implementation of the CFD module is analysed. The benchmark procedure itself and most of the geometries are based on the benchmarks reported in D3.1, albeit not on EuroHPC JU architectures. The actual differences and the additional meshes, that were benchmarked, are reported here.

All meshes benchmarked are generated with SZE's own in-house mesh generator, octreemesher. Within the simulation, the simpleFoam solver is used to solve the incompressible steady-state RANS equations coupled with steady-state advection-diffusion equations for pollution spread to calculate a steady-state solution applying steady boundary conditions and source terms. Starting from the steady state, the pimpleFoam solver is used to solve the time-dependent URANS equations with coupled pollution spread calculations and time-dependent boundary conditions and source terms.

5.2.1 UAP overview

In this section, an overview of the OpenFOAM-based UAP-FOAM benchmark is presented starting with the geometry, boundary conditions, source terms, and followed by the meshes of various sizes used for the benchmarks and the discussion on governing equations and numerical methods applied within the OpenFOAM framework.

5.2.1.1 Geometry

The geometry of the use case models external airflow around a geometry resembling the urban area of Győr, a city in the northwestern part of Hungary. Boundary conditions for ground and buildings follow atmospheric wall functions with a tuned roughness factor. The top and side boundaries with airflow are separated as “sky” and “inlet”. While “sky” follows a “slip” condition, “inlet” follows a custom inlet-outlet type condition, where inlet wind speed is time and height-dependent and comes from larger scale simulations – in a predictive simulation – or measurements – in an analytic simulation. Inlet condition has been obtained from ECMWF weather service using the polytope interface. Source terms for pollution are obtained from traffic pollutant emissions via separate traffic simulations. The geometry of the simulation domain is shown in Figure 13.

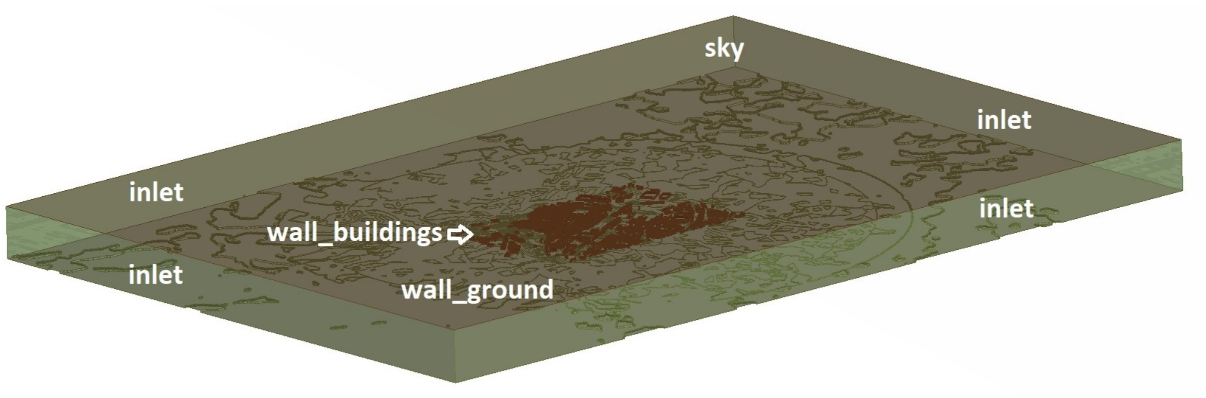


Figure 13. Simulation domain for UAP use case showing the patch names and the position of the buildings

5.2.1.2 Meshes

The meshes for these benchmarks are generated using the in-house SZE tool octreemesher. All meshes use the same geometry, albeit with different resolutions to get various mesh cell counts.

Table 8. Meshes for UAP use case

Name	Cell count	Volume [km3]	Avg. cell vol. [m3]
uxlow	36 248	23,26	641 690
ulow	139 937	23,37	167 004
mlow	228 263	22,40	98 132
low	728 162	22,42	30 790
mid	3 227 275	22,69	7 031
high	14 332 247	21,61	1 508

Mesh sizes were selected on a wide range so that various performance behaviours may be observed (Table 8). Total mesh volume slightly varies, as cubic cells do not precisely follow geometry. All meshes were generated beforehand and saved in Fluent MSH format. Additionally, pollution source term factors are calculated beforehand on a per-cell basis. Meshes are reimported with the fluent3DMeshToFOAM utility.

5.2.1.3 Equations and numerical methods

The Reinolds-averaged Navier-Stokes equations with k-ε turbulence model are solved for incompressible fluid [10][11]. The solver simpleFoam is used for an initial steady-state solution, which applies the Semi-Implicit Method for Pressure Linked Equations (SIMPLE method) for solving the continuity and momentum [10][12]. The RANS equations are coupled with convection-diffusion equations for pollution spread. A total of 600 iterations are run, except for high mesh size, where it is only 400, however, runtimes are adjusted accordingly. The Generalized Geometric-Algebraic Multigrid (GAMG) solver with Gauss-Seidel smoother was used to solve the pressure equation.

In this use case, the steady-state calculations are followed by a transient phase, where the governing equations, boundary conditions and source terms are time-dependent. This part is calculated by the OpenFOAM solver pimpleFoam using the previously calculated steady state as an initial condition. The solver combines the PISO with the SIMPLE algorithm. The Preconditioned Pipelined Conjugate Residuals (PPCR) solver with the Fast Diagonal-based Incomplete Cholesky (FDIC) preconditioner was used to solve the pressure equation.

For domain decomposition, the scotch library is used. Also, domain decomposition was done with decomposePar at the beginning of the simulation after importing the pre-generated mesh with fluent3DMeshToFoam. Multilevel decomposition was used considering the two sockets. Also, the number of subdomains was adjusted according to the number of CPU cores of the underlying architecture. The collated file format was enabled to limit the number of files that were stored on the storage.

5.2.2 Performance experiments

Performance experiments are conducted similarly as with the motorbike use case: every scenario is executed 5 times, and 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. This process is repeated for both steady and transient calculations. Measurements are done separately for the two parts. For purposes of comparison, the median of the five measurements is calculated, effectively dropping the smallest and largest two values.

5.2.2.1 Results for UAP simpleFoam solver

Measurement results for the steady-state simpleFoam solvers are grouped by mesh size on different plots in Figure 14, all measured execution time is plotted for different architectures. Arrows indicate a larger than one speedup, including the actual speedup value. Red values indicate the largest speedup. Unique measurements do fluctuate. The rate of fluctuations decreases with mesh size.

Investigating the results, similar behaviour can be observed as with the motorbike use case. For smaller meshes up to low size, 64-core Genoa delivers the best performance. The biggest leap is between Milan and Genoa with a speedup factor of 1.48x to 1.64x.

For mid and high, the largest leaps are between the non-X and X architectures with a speedup factor of 1.39x to 1.57x. In comparison to the motorbike use case, the number of cells does not reach the range, where the behaviour of runtime is predominantly influenced by memory bandwidth. Also, 96-core Genoa does not provide the extra performance seen for some of the smaller meshes. Across all mesh sizes, Milan provides an improvement of around 10% to Rome. The improvement from Milan to Milan-X can be observed at 47% for mesh sizes mid and high but it is non-existent for mesh sizes below low. Similarly, the 96-core Genoa will provide a circa 20% benefit over the 64-core Genoa for mesh sizes mid and high, while it does perform worse for the smaller mesh sizes. The high cache 96-core Genoa-X shows similar behaviour with a performance improvement of 39% and 57% over the 96-core Genoa. Finally, at neither of the mesh sizes did Bergamo deliver the best performance.

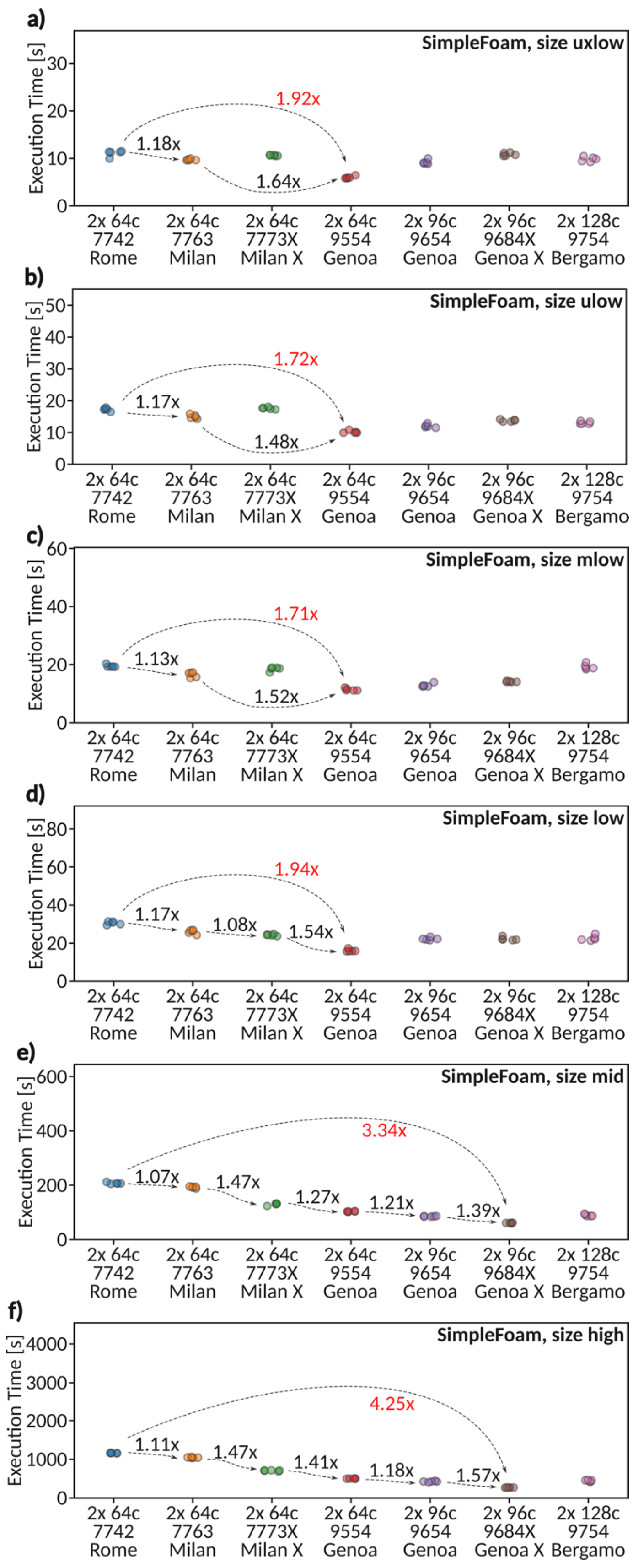


Figure 14. Performance results of UAP simpleFoam solver on a variety of AMD CPUs and different mesh sizes, including a) uxlow, b) ulow, c) mlow, d) low, e) mid, and f) high

5.2.2.2 Results for UAP pimpleFoam solver

Measurement results for the transient pimpleFoam solvers are grouped by mesh size on different plots in Figure 15, all measured execution time is plotted for different architectures. Arrows indicate a larger than one speedup, including the actual speedup value. Red values indicate the largest speedup. Unique measurements do fluctuate. The rate of fluctuations decreases with mesh size.

Small results show similar behaviour for small mesh sizes up to low: Milan is about 10-20% faster than Rome, and 64-core Genoa is about 50% faster than Milan. Other Genoa and Bergamo are not faster than 64-core Genoa for these sizes. The same behaviour was observed for the use cases with simpleFoam.

Larger mesh sizes also show similar behaviour to the use cases with simpleFoam, however, speedup rates are more prominent, leading to a speedup factor from Rome to Genoa-X of 4.22x for mid, and 7.95x for high mesh size. In the latter case, the extra cache size causes a speedup of 2.49x for Genoa-X and 1.69x for Milan-X. Bergamo does not deliver higher performance than the 96-core Genoa. The number of cells does not reach the range, where the behaviour of runtime is predominantly influenced by memory bandwidth.

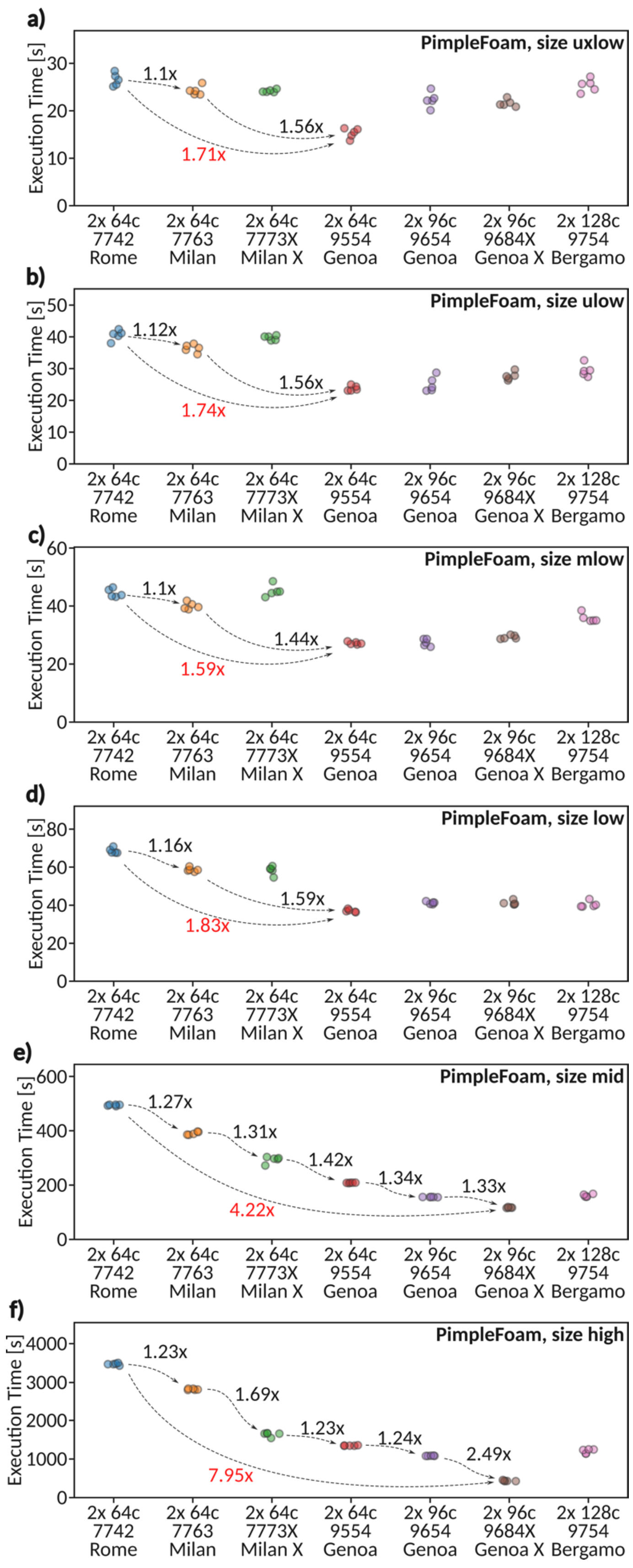


Figure 15. Performance results of UAP pimpleFoam solver on a variety of AMD CPUs and different mesh sizes, including a) uxlow, b) ulow, c) mlow, d) low, e) mid, and f) high

5.2.3 Performance analysis

Using the Finite Volume Operations Per Second (FVOPS) metric, performance across multiple platforms was calculated for the simpleFoam part again, as with the motorbike use case (see Section 5.1). Results are grouped again as 64-core architectures (Figure 16a) and Genoa-Bergamo (Figure 16b), where FVOPS values are plotted again per core cell count.

The curves show similar behaviour, having a maximum value between ca. 6k and 20k, and decreasing for lower and higher values. Values of FVOPS significantly drop – at least a factor of 2-3 – within the next two mesh sizes, so high cell per core value does limit performance. This is in alignment with the findings of D3.1: increasing the number of nodes will decrease cell per core count and will provide greatly increased FVOPS thus leading to super linear speedup. For all 64-core architecture sizes, peak performance is observed at 6k cells per core, and for 96-core and 128-core architectures at ca. 20k cells per core. 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 values, Rome and Milan do not exhibit significant differences.

Results for the higher core count processors are almost identical, save for Genoa-X outperforming others for the largest mesh sizes. It can be observed that simply providing more cores for calculations will not increase performance as 96-core Genoa and 128-core Bergamo cannot show higher performance within this metric. Also, higher core count architectures do not show that drastic performance drop after the FVOPS maximum.

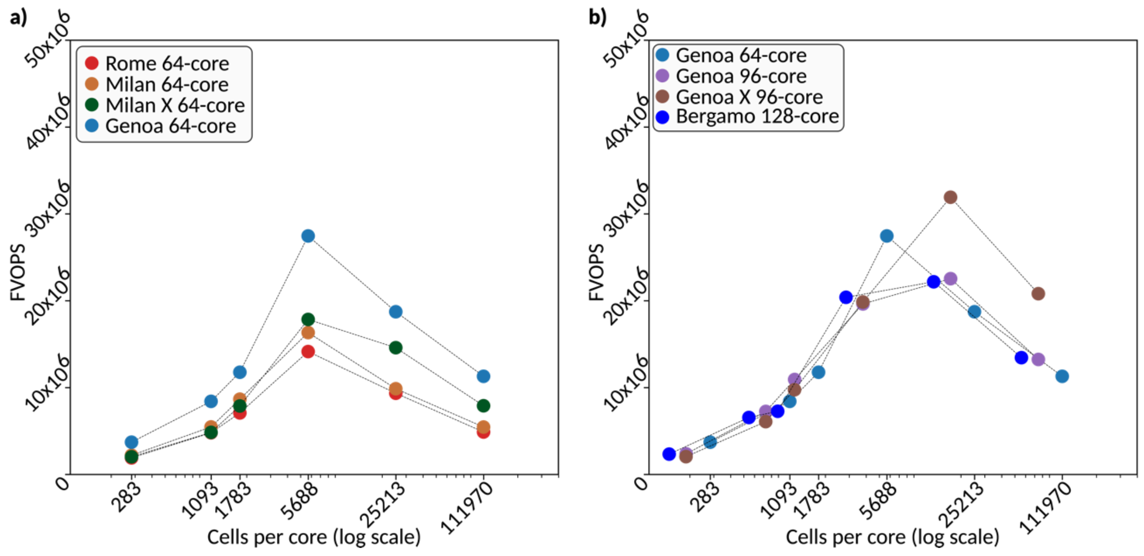


Figure 16. FVOPS performance metrics determined for UAP simpleFoam solver on a) 2x 64-core and b) Genoa-/Bergamo-based platforms

Going forward, we investigate performance analysis for the UAP simpleFoam solver in a similar manner, as with the motorBike use case (see Section 5.1). We combine AMD μProf and FVOPS performance metrics to get a profile report as well as determine the performance limitations and hardware-application correlation. In this stage of the HiDALGO2 project, the following profile data setup is considered: L2 miss ratio from L1DC [%], L2 miss ratio from HWPF [%], L2 miss ratio total [%], MPI communication to computation ratio [%], and FVOPS (see Section 4.2).

Figure 17 and Figure 18 demonstrate profiler reports generated for two groups of processors considering different mesh sizes. The first group outlines profiler data for the 64-core architectures including Rome (Figure 17a), Milan (Figure 17b), Milan X (Figure 17c) and Genoa (Figure 17d). The second group corresponds to 96-core Genoa (Figure 18a), Genoa X (Figure 18b) as well as 128-core Bergamo (Figure 18c) processors.

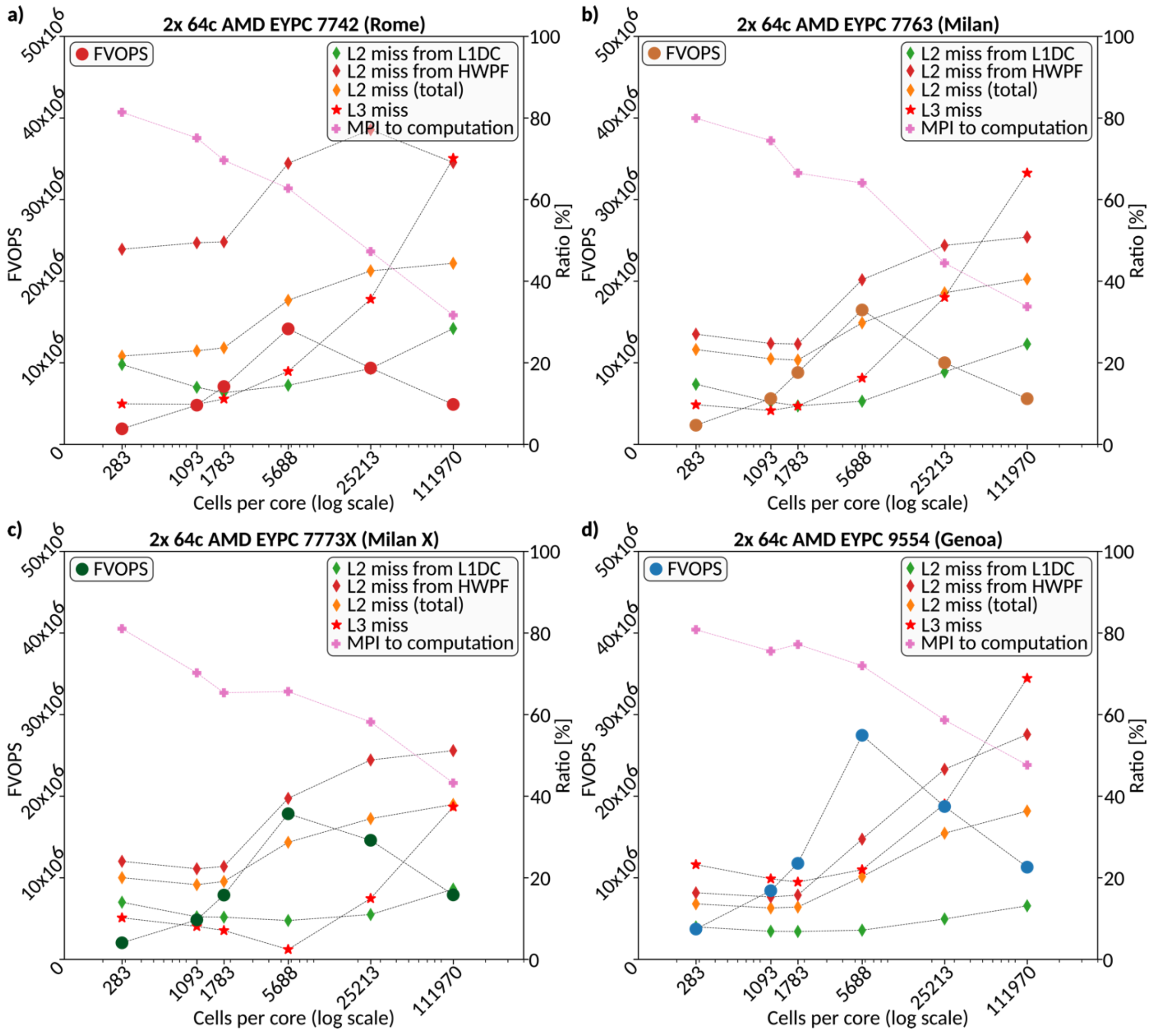


Figure 17. Performance analysis determined for UAP simpleFoam solver on 2x 64-core HPC platforms, including a) Rome, b) Milan, c) Milan X, and d) Genoa architectures

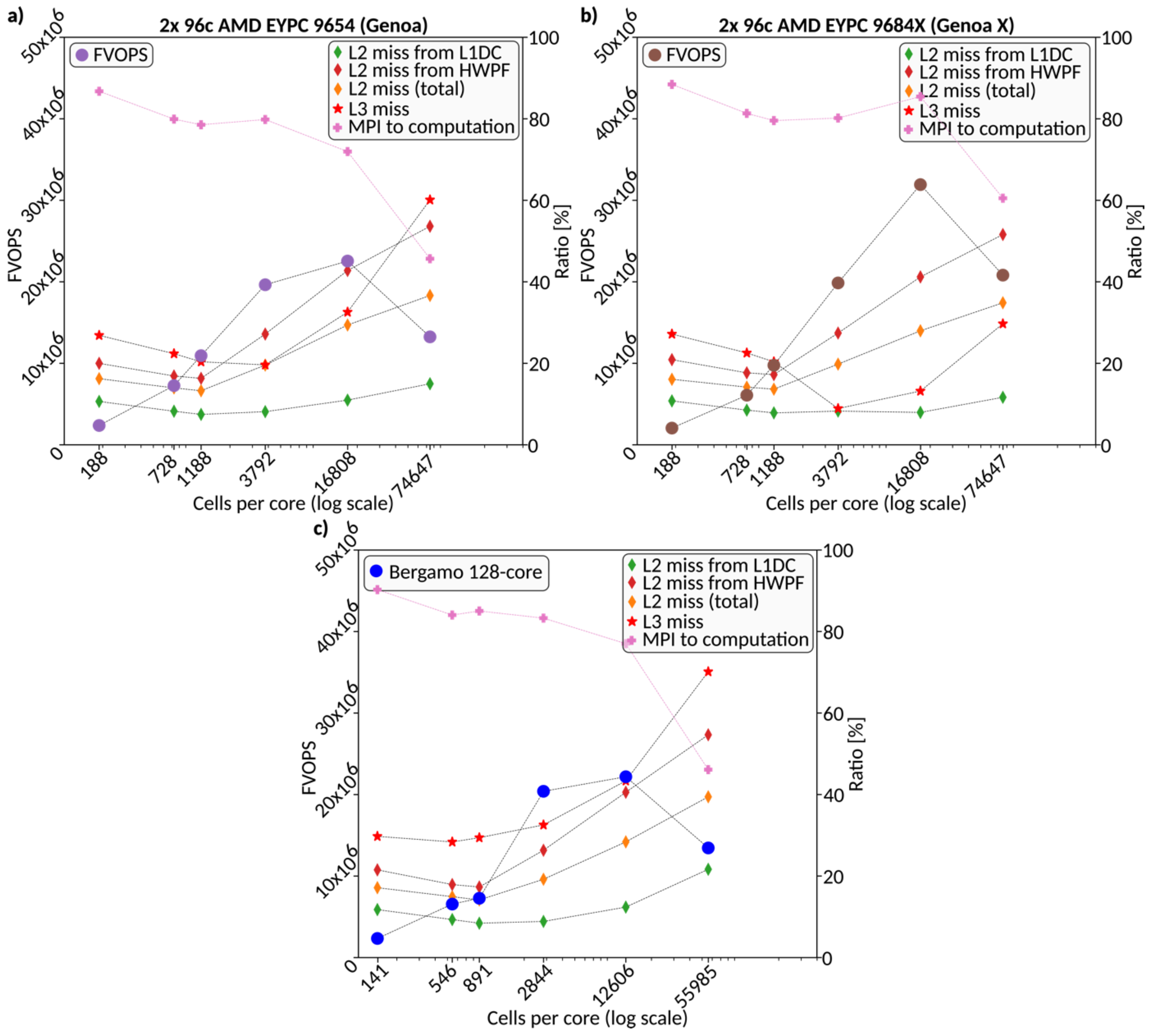


Figure 18. Performance analysis determined for UAP simpleFoam solver on the 4th generation of AMD EPYC CPUs, including a) Genoa, b) Genoa X, and c) Bergamo architectures

The Genoa-based processors offer twice larger L2 size per core, reducing L2 cache miss ratio trends compared to the prior AMD EPYC generations (see Figure 17 and Figure 18). This trend reduction is observed for all the studied metrics, including the L2 miss ratio from L1DC, the L2 miss ratio from HWPF, and the L2 miss ratio total. It results in performance advantages over the prior generations of AMD EPYC CPUs, especially noted for relatively smaller mesh sizes (see Figure 16).

Considering relatively large domain sizes, including mid and high, we observe a high L3 cache miss rate that reaches up to 70% (Figure 17 and Figure 18). In this case, following the trend of FVOPS performance metrics, 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. In contrast, considering relatively smaller domain sizes, we observe that the L3 miss ratio is kept at a smaller rate. We expect here that L3 misses are mainly noticeable where the first time a memory location is read (compulsory 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 the regular Milan and regular 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.

Going forward, the novel AMD EPYC 9004 series processors thanks to the newest DDR5-based and larger 2x 12-channel memory subsystem offer higher attainable performance than the prior generation (Rome, Milan and Milan X). We note the performance uplift for the 96-core EPYC Genoa CPUs of up to 2.5x and 2.7x over regular Milan and Rome CPUs, mainly resulting from the 12-channel DDR5 and an extra 32 cores.

As expected, the Bergamo-based platform features the highest trend for the L3 cache miss ratio (Figure 18). To explain this behaviour, we have to look at the specification of the Bergamo architecture that offers a smaller L3 cache size per core and the same memory subsystem speed compared to other Genoa-based processors (Table 3). Considering this and the fact that memory-intensive parallel codes can suffer from bandwidth saturation as more cores are used, the Bergamo CPUs do not offer a performance advantage over Genoa processors for tested UAP simpleFoam kernel.

Additionally, we analyse the profiler data, which helps us indicate the MPI communication over computation ratio. Similarly to the motorBike use case, we observe that the cost of the MPI communication is higher than the computation for relatively smaller mesh sizes. It helps us underline that MPI communication becomes one of the crucial issues for further co-design activity when using extensive computing resources.

6 Conclusions

In this deliverable we focus on the state-of-the-art AMD products that represent traditional HPC processor vendors. This work investigates the top AMD processors, the AMD EPYC 9004 series processors, to reveal trends in innovative HPC architectures. We observe that AMD EPYC CPUs excel in the HPC domain by offering high core count, performance, and relevant memory subsystems with large cache capacity. The new series of EPYC processors offers a wide range of models, making them the right choice for parallelizing demanding tasks in HPC workloads. Since AMD CPUs are becoming the leaders in HPC, we start our activities by exploring the newest AMD EPYC 9004 series products at this stage of the HiDALGO2 project.

In this stage of the HiDALGO2 project, we focus on benchmarking new-generation AMD EPYC processors, investigating two OpenFOAM-based applications. This work presents the general overview of single-node benchmarking activities for AMD-based HPC infrastructures and initial findings on their profiling results. In particular, we explore a series of top-of-the-line AMD EPYC CPUs with a wide range of CPU products based on Rome, Milan, Milan X, Genoa, Genoa X, and Bergamo architectures. The core component of the benchmark is a set of the OpenFOAM kernels, which helps us indicate the impact of new HPC hardware features on the final performance efficiency.

This study outlines the impact of large cache systems on the parallel efficiency of different computing kernels from OpenFOAM. By testing a wide range of mesh sizes for every computational kernel, we reveal and estimate the performance advantages of the newest 4th generation of AMD EPYC CPUs over the prior generations. Our investigation enables us to better understand the limiters of application performance and evaluate improvements.

Table 9 and Table 10 demonstrate the computing platforms that achieve the best performance results during tests performed for a given computing kernel and fixed mesh size. We identify a good machine balance between memory speed and core count for the system based on Genoa with 2x64 cores, which is especially noticeable for relatively smaller mesh sizes.

Table 9. Platforms with the best performance results for motorBike

Platform	motorBike meshes
2x 64-core 9554 Genoa	xsmall
2x 96-core 9654 Genoa	small and smid
2x 96-core 9684X Genoa X	mid, high, mhigh, uhigh, and xhigh

Table 10. Platforms with the best performance results for UAP

Platform	UAP meshes
2x 64-core 9554 Genoa	uxlow, ulow, mlow, and low
2x 96-core 9684X Genoa X	mid and high

Going forward, we note a 96-core EPYC Genoa with a performance uplift of up to 2.6x over regular Milan, mainly resulting from the 12-channel DDR5 and the extra 32 cores. This work discovers also that large-cache systems with enabled 3D cache from AMD processors offer great opportunities to improve performance by overcoming the memory-bond nature of applications such as OpenFOAM-based parallel codes. In particular, noticeable performance profits are obtained for the systems with 3D cache available in Milan X and Genoa X CPUs in comparison to regular Milan and Genoa, respectively. The performance gains are also observed for systems with novel DDR5-based computing nodes (Genoa and Bergamo nodes) in comparison to - the previous generation - DDR4-based memory subsystem (Rome and Milan nodes).

Our further study includes an extension of the experimental comparison proposed in this work over a wide range of current and emerging architectures, as well as across other applications. The next steps regarding Innovative HPC Technologies within HiDALGO2 include, but are not limited to, the following:

- Intel-based HPC platforms, starting from Intel Xeon Max architecture
- ARM-based solution for HPC
- Nvidia and AMD GPUs

References

[1] Homepage of HiDALGO2 CoE, http://hidalgo2.eu [retrieved: 2024-03-1]

[2] HiDALGO Report D5.8: Final benchmark results for innovative architectures, https://hidalgo-project.eu/sites/default/files/2022-04/HIDALGO_D5.8_Final%20benchmark%20results%20for%20innovative%20architectures_v1.0.pdf [retrieved: 2024-04-8]

[3] Homepage of AMD, https://www.amd.com/ [retrieved: 2024-03-20]

[4] TUNING GUIDE AMD EPYC 9004: High Performance Toolchain: Compilers, Libraries & Profilers, Rev. 1.5, January 2024, https://www.amd.com/content/dam/amd/en/documents/epyc-technical-docs/tuning-guides/58002_amd-epyc-9004-tg-hpc.pdf [retrieved: 2024-03-11]

[5] 4TH GEN AMD EPYC PROCESSOR ARCHITECTURE, 3rd edition, September 2023, https://www.amd.com/content/dam/amd/en/documents/products/epyc/4th-gen-epyc-processor-architecture-white-paper.pdf [retrieved: 2024-03-11]

[6] Galeazzo, F. C. C., Weiß, G. R. and Ruopp, A. Understanding Superlinear Speedup in Current HPC Architectures. 18th OpenFOAM Workshop (OFW18), Genova, Italy, 11-14 July 2023

[7] Homepage of AMD uProf, https://www.amd.com/en/developer/uprof.html [retrieved: 2024-03-11]

[8] AMD uProf User Guide, version 4.2 https://www.amd.com/content/dam/amd/en/documents/developer/version-4-2-documents/uprof/uprof-user-guide-v4.2.pdf [retrieved: 2024-03-11]

[9] Original repository of motorbike tutorial in OpenFOAM, https://develop.openfoam.com/Development/openfoam/-/tree/master/tutorials/incompressible/simpleFoam/motorBike, [retrieved: 2024-03-10]

[10] Description of simpleFoam with continuity and momentum equation, https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-incompressible-simpleFoam.html, [retrieved: 2024-03-10]

[11] Description of k-ω-SST turbulence model description, https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html, [retrieved: 2024-03-10]

[12] Description of SIMPLE method, https://www.openfoam.com/documentation/guides/latest/doc/guide-applications-solvers-simple.html, [retrieved: 2024-03-10]

[13] OpenFOAM version v2212, https://www.openfoam.com/news/main-news/openfoam-v2212 [retrieved: 2024-03-10]