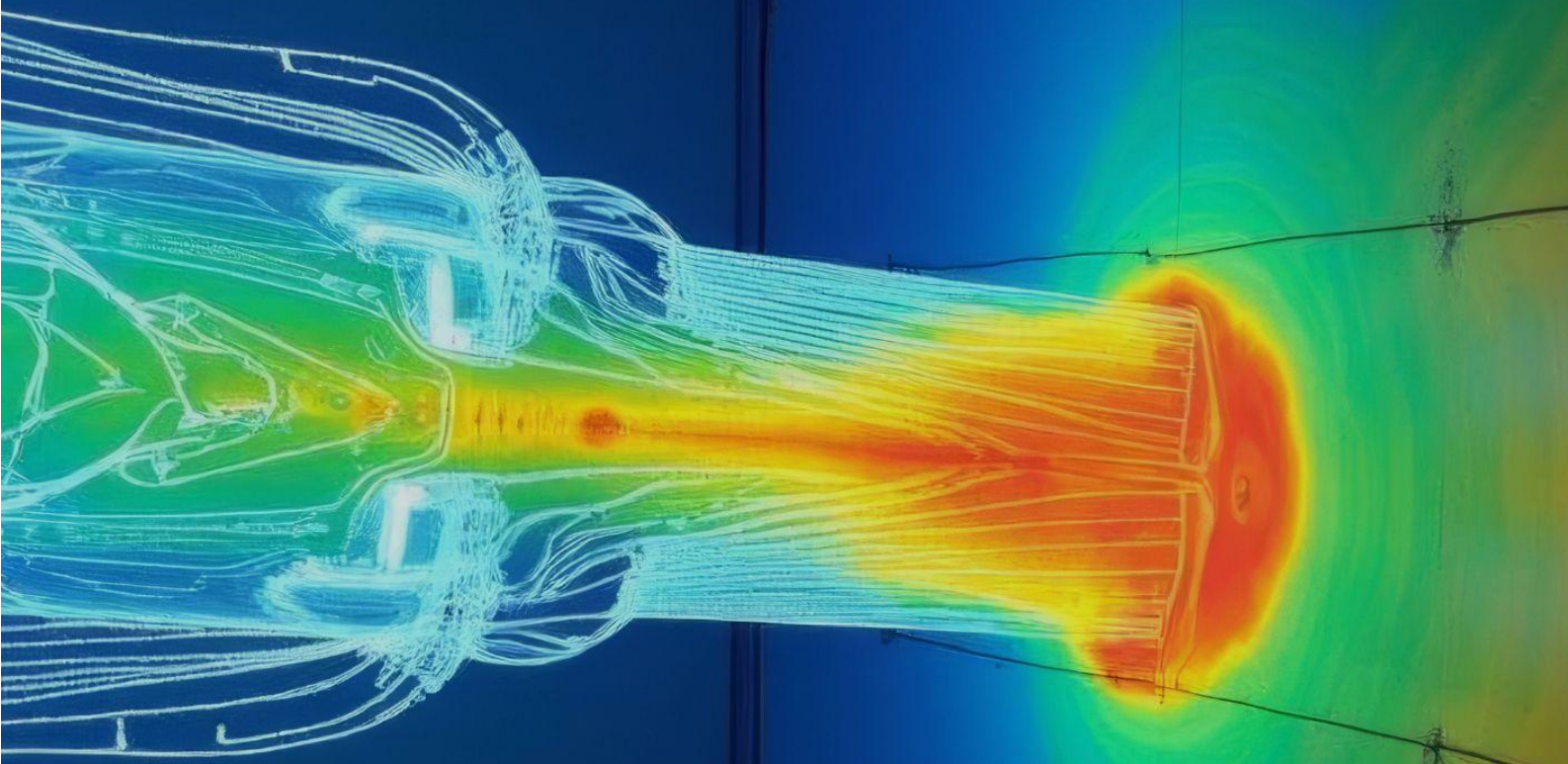
EUROCC ITALY
HPC Competence Center

SPUMA:

the power of open-source CFD running on GPUs



Authors: Simone Bnà, Giuseppe Gianquinto, Tommaso Zanelli, Alessio Piccolo, Francesco Bottau, Ettore Fadiga, Vittoria Sibilia, Eric Pascolo

SPUMA: the power of open-source CFD running on GPUs

Author: Simone Bna, Giuseppe Gianquinto, Tommaso Zanelli, Alessio Piccolo, Francesco Bottau, Ettore Fadiga, Vittoria Sibilia, Eric Pascolo

Index

Introduction 3

OpenFOAM: An Overview 4

HPC enabling technologies for High-Fidelity simulations 5

SPUMA: the first production-ready version of OpenFOAM running on GPU . 7

Accelerating CFD simulations with SPUMA: bigger simulations with less hardware 10

The Agile advantage: reshaping human interaction workflows for a faster and high-quality CFD code 12

Showcase of SPUMA potential: automotive industry grade case 16

SPUMA's potential for time and cost reductions..... 20

Introduction

This white paper presents SPUMA, a port of the popular open-source computational fluid dynamics software OPENFOAM®[1] to GPU accelerators, developed by Cineca.

The first two chapters give an overview of OpenFOAM and its relevance for industrial applications of High-Performance Computing (HPC). The following two describe the methodology used to port of the software to GPU, and the advantages that using accelerators can bring to simulation workflows. The fifth chapter is dedicated to the adoption of the Agile methodology for the development of SPUMA, while the sixth chapter reports its performance while running a test case representative of industrial applications. Finally, an overview of the benefits of SPUMA over traditional CPU-based simulation software, as well as the positive effects of adopting Agile for its development, is given in the conclusions.

[1] OPENFOAM® is a registered trade mark of OpenCFD Limited, producer and distributor of the OpenFOAM software via www.openfoam.com. This offering is not approved or endorsed by OpenCFD Limited, producer and distributor of the OpenFOAM software via www.openfoam.com, and owner of the OpenFOAM and OpenCFD® trade marks.

OpenFOAM: An Overview

The physics of fluid flow affects virtually every sector of industrial and civil engineering, as well as taking part in a wide variety of more complex physical phenomena. Since the advent of digital computers in the second half of the 20th century, Computational Fluid Dynamics (CFD) has been an invaluable tool in the hands of scientists and engineers, to aid in design processes and reduce reliance on costly experiments.

OpenFOAM is an HPC-driven CFD software developed primarily by OpenCFD Ltd since 2004. It is free, open-source, and it has a wide international community of users in both industry and academia. OpenFOAM is a tool to solve Partial Differential Equations (PDEs) in Science and Engineering, ranging from complex fluid flows involving chemical reactions and turbulence to solid mechanics and electromagnetics, among many other applications. OpenFOAM has been applied extensively to study external aerodynamics in the aerospace and automotive industries, in the turbomachinery field to evaluate the efficiency and the torque, and in HVAC simulations to predict the thermal exchange, flow field and humidity transport/generation.

OpenFOAM uses the 2nd order collocated Finite Volume method to discretize PDEs in conservative form like the Navier-Stokes equations. This is the same algorithm found in commercial software (e.g. Ansys Fluent, StarCCM+) used by companies in their CFD workflows. It works with unstructured meshes of generic polyhedral cells that can be generated both using the built-in meshers (e.g. snappyHexMesh) and commercial solutions (e.g. CFMesh+, Pointwise, ANSA).

HPC enabling technologies for High-Fidelity simulations

OpenFOAM can run in parallel using the MPI (Message Passing Interface) paradigm and domain decomposition techniques, and it scales from the few cores of a laptop/workstation up to the tens of thousands of cores of an HPC cluster, a collection of many separate servers connected via a fast network.

Due to its free and open-source nature, OpenFOAM has no license costs. It can be used on High-Performance Computing (HPC) clusters paying only for the computational resources required by the simulation. The parallel scalability of the code depends on many factors, some algorithmic and others related to hardware characteristics, but we can reasonably assume that a minimum number of 10,000 computational cells per CPU core is necessary. Such features allow OpenFOAM to be a cost-effective alternative to commercial software for complex simulations characterized by a high number of cells.

The amount of computational power available to researchers and engineers has continuously grown thanks to the introduction, in recent years, of hybrid CPU - GPU clusters. This has led many to look beyond traditional Reynolds-Averaged Navier–Stokes (RANS) approaches, towards high-fidelity large-eddy simulations (LES) or hybrid RANS-LES simulations for industrial flow configurations. However, OpenFOAM was designed in the 1990s, in a context where the hardware was different from today as it consisted of single-core CPU configurations with access to much less RAM memory and no GPU (Graphical Processing Unit) acceleration for mathematical operations.

CPUs are general purpose computing units implemented on integrated circuit microprocessors that have always been used for scientific computing, among other things. Only in recent times GPUs have been adopted to accelerate the computationally intensive part of a simulation. The architecture of a GPU is different from its CPU counterpart since it was

designed to accelerate digital image processing and computer graphics. To run software developed for CPUs on GPUs, it requires to be “ported” to the new architecture, a task that can be overwhelming if the code is made of millions of lines like OPENFOAM®.

Nowadays, GPUs are the devices that provide most of the computational power of HPC clusters, measured in FLOPS (Floating Point Operations Per Second). As they are a crucial enabling technology for the development of AI, GPUs are evolving rapidly. They are characterized by a high memory bandwidth (in the order of Terabytes per second), that is particularly relevant for memory-bound applications like OPENFOAM®. A GPU-enabled CFD code can have a dramatic impact in the industrial sector: high fidelity simulations could be executed using a limited number of GPUs in the same amount of time of low fidelity simulations using large amounts of CPU cores.

SPUMA: the first production-ready version of OpenFOAM running on GPU

SPUMA (Simulation Processing on Unified Memory Accelerators) is a fork of OpenFOAM released by CINECA that enables the software to run on GPUs.

In recent years, vendors of commercial CFD software have started developing versions of their products able to run on GPU. There is a considerable interest by both developers and users of simulation software to exploit the considerable performance gains that accelerated hardware can offer. These GPU-enabled versions are often complete rewrites of the original software. The open-source community that maintains OpenFOAM does not have the resources necessary for such an endeavour.

An important reason for OpenFOAM's success is the modularity of its frontend: a high-level syntax for writing arbitrary sets of partial differential equations into executable solvers, allowing for the simulation of a wide variety of physical phenomena. On the other hand, its success as open-source software is also due to the relative rigidity of its backend: low-level math routines are seldom changed, reducing the maintenance effort required by the code.

There have been several previous attempts at porting OpenFOAM to GPU over the past decade, with varying degrees of success. These attempts did not consider a full rewrite, but rather focused on porting certain compute-intensive parts of the code or integrating it with already existing frameworks for GPU acceleration (i.e. Thrust, Kokkos...). The former approach usually focused on the solution of systems of linear equations with GPU-ready linear algebra libraries. Since the integration of these libraries could be limited to self-contained plugins, it has found acceptance by the OpenFOAM community, however, the performance that can be gained by this approach is limited. The latter approach, while more promising in terms of performance, can significantly impact the low-level backend of the code with the introduction of API calls. For this reason, none of the porting

attempts using existing frameworks have ever been included in official releases of OpenFOAM and often the small teams developing them do not have the resources to maintain them as an independent fork of the code.

CINECA started working on a GPU porting of OpenFOAM in the context of the EU-funded exaFOAM project that ran from 2021 to 2024. The aim of the project was to investigate ways to increase the performance of OpenFOAM on modern hybrid high-performance computing platforms.

CINECA's contribution to the project was a proof-of-concept, GPU-ready, partial clone of OpenFOAM named zeptoFOAM, which showed that considerable performance gains can be achieved without significantly altering the high-level algorithms and data structures of the code. With the experience gained working on this prototype and analysing the shortcomings of previous porting attempts, CINECA identified the following requirements for a successful port of the entire code:

- **Minimal impact on the code base**: radical changes to OpenFOAM are less likely to be accepted by the community that maintains the code, and they make a hypothetical fork more difficult to maintain.
- **No dependency on external frameworks**: aside from the problems already mentioned above, third-party frameworks generally do not allow for the same performance gains achieved using low-level APIs.
- **Efficient memory management**: OpenFOAM creates short-lived data structures continuously, which impacts GPUs more than traditional architectures due to the higher cost for memory allocation. This problem can be solved with implicit memory management, using a memory pool.
- **Compatibility with multiple platforms**: the philosophy behind OpenFOAM aims to ensure accessibility by the widest possible user base. This, in the past, was ensured using standard C++, however, no similar vendor-agnostic approach has yet emerged for GPUs.

Leveraging on these insights, CINECA started the development of SPUMA, with the aim of creating a full GPU port of OpenFOAM.

The approach finally chosen by CINECA was inspired by the recent GPU porting of the code NEKO[2], a modern implementation of the well-known high order CFD code Nek5000: several backends can be added to the framework in order to run on different hardware (e.g. NVIDIA, AMD, INTEL) but a unique frontend is used everywhere else in the code. The front-end has no dependencies on third-party APIs, which are confined in the various backends. Since the number of backends required to cover all possible parallel operations is limited, the amount of third-party, hardware specific code is kept to a minimum.

This layer of abstraction was combined with the adoption of unified memory, a hardware/software technology feature available on recent GPU accelerators that allows the allocation of a memory address space accessible by both CPUs and GPUs. The result is a minimally invasive, vendor-agnostic implementation. The solution we propose can run on both discrete GPUs and on more recent APUs, which implement unified memory at a deeper level. At the same time, this approach can be adapted to any avaliable GPU paradigm (CUDA, HIP, OpenMP, OpenCL, etc...).

More details on the methodology used for the development of the SPUMA code can be found in the authors' recent publication[3].

[2] N. Jansson, M. Karp, A. Podobas, S. Markidis, P. Schlatter, Neko: A modern, portable, and scalable framework for high-fidelity computational fluid dynamics (Computers & Fluids, 2024) https://doi.org/10.1016/j.compfluid.2024.106243

[3] S. Bnà, G. Giaquinto, E. Fadiga, T. Zanelli, F. Bottau, SPUMA: A minimally invasive approach to the GPU porting of OPENFOAM® (Computer Physics Communications, 2026) https://doi.org/10.1016/j.cpc.2025.110009

Accelerating CFD simulations with SPUMA: bigger simulations with less hardware

One of the main issues in parallel computing is to have “almost perfect” scalable algorithms, i.e. algorithms that maintain the same efficiency by varying the number of processors. For very large problems this can be a challenge due to the communication overhead related to the huge number of messages. In OpenFOAM the situation is exacerbated by the serial operation of mesh decomposition, an operation that can take days to complete or even be unfeasible for larger scale simulations.

A huge number of cells are required to accurately describe complex problems, such as High-Fidelity simulations. In these cases, it can be necessary to decompose the computational domain into tens of thousands of partitions, and each one is assigned to a different CPU core. As mentioned, increasing the number of processes above a certain threshold decreases the parallel efficiency of a code like OPENFOAM®, and additional computational resources bring diminishing returns. Additionally, Input/Output operations in OpenFOAM are segregated, meaning each process writes to a different set of files. For simulations running on thousands of cores, this causes significant strain on the file system.

While CPUs perform better on smaller portions of the computational domain, GPUs, with their high memory bandwidth and large number of computational units, operate best when they are given large amounts of data. While for CPUs the optimal number of cells is in the order of thousands, for GPUs it is in the order of millions. This means that for large, high-fidelity simulations running on GPUs the number of partitions required to decompose the computational domain is considerably reduced, alleviating the problems related to parallel scalability mentioned above. In this regard, a GPU-ready version of OpenFOAM extends the range of applications of the software to much larger problems that would have been impractical on CPUs.

So far, large scale high-fidelity simulations require specialized high-order, high-performance simulation tools, developed mostly in academia and limited in their range of applicability to simple geometries and specific cases. Often, these limitations make them unsuitable for industrial applications. The ability, made possible by this GPU port, to use a versatile general-purpose tool like OpenFOAM for large-scale simulations brings them within reach of a much wider user base. Smaller industrial players that would not have been able to run simulations of this scale can now include them in their workflows. This creates vast opportunities for collaboration between the European SME ecosystem and the continent's HPC infrastructure.

The Agile advantage: reshaping human interaction workflows for a faster and high-quality CFD code

During the development of SPUMA, the organizational framework was changed to adhere to Agile principles, providing the team with a more structured approach to managing their activities.

In Agile SCRUM, an iterative workflow designed to deliver incremental improvements while maintaining high adaptability, work is organized into short sprints that enable teams to continuously refine priorities in line with evolving technological and stakeholder needs, integrate feedback, and reassess objectives as the project evolves. This framework enhances transparency, supports informed decision-making, and cultivates regular communication. After 14 sprints, this approach has proven that redefining human interaction workflows is a key driver in enhancing development processes, ultimately making them faster and more efficient.

The Agile methodology

Each sprint starts with a meeting in which the team defines the sprint goals and organizes tasks, while also updating the product backlog, a prioritized list of all work items and features planned for the project. A 30-minute daily meeting is attended by the Scrum Master (SM) and the CFD developers. The SM, a non-technical figure, ensures that discussions remain goal-oriented and within the allotted time.

Each sprint lasts one week and concludes with a 1-hour meeting during which the team presents completed tasks, discusses encountered issues, and shares the solutions adopted. This meeting serves both as a sprint review and as a retrospective, providing a clear overview of progress while helping identify improvements for the next development cycle. The process ends with a return to the backlog, where new tasks are selected.

In our case, the responsibilities of the Product Owner (PO) are held by the Development Team Coordinator. The PO oversees priorities and ensures

coherence between development and broader project objectives. A senior Project Manager acts as Reviewer, integrating insights from the European projects where SPUMA is used and aligning them with stakeholder needs.

A Non-Conventional, Innovative Approach

The adoption of Agile was motivated by the need to enhance efficiency and support a more structured and transparent development process. Beyond its iterative nature, Agile was introduced to provide clearer organizational structure, increase individual accountability, and encourage a participatory attitude while preserving strong teamwork dynamics. These practices also support conflict management by fostering open communication and offering regular opportunities for external, non-exclusively technical, review. In our context, adopting an Agile approach represents a non-conventional choice.

The SPUMA team is primarily composed of scientific professionals rather than full-time software developers: besides code development, team members support several European research projects, deliver training courses, and apply SPUMA to scientific activities. Introducing Agile into such an environment marks a significant shift toward a more adaptive, collaborative, and iterative way of working, enabling the team to balance software development with scientific duties while maintaining alignment across diverse activities.

An important element of innovation in this workflow is the extension of the Agile approach beyond the internal development of SPUMA to the European research projects in which the software is employed. These projects have traditionally relied on a waterfall-based structure, characterized by linear phases and limited flexibility. Introducing Agile in this context represents a significant shift toward a more adaptive and collaborative model.

Adopting the SCRUM framework allows to:

1. Minimize risks associated with late discovery of issues and misalignment with objectives.

2. Remain flexible in a rapidly changing tech market
3. Adjust to evolving industrial and research priorities
4. Deliver solutions that meet project goals and stakeholder needs

Before and after

Before the adoption Agile, the team followed a seniority-based quasi-pyramidal structure where tasks were mainly dictated by a key individual rather than a clear process that clearly prioritised stakeholder needs. Team interaction primarily occurred during monthly all-hands meetings, with daily tasks largely independent. Activities were predominantly research-driven, and development was mainly performed by senior members with limited collaborative effort. GitLab's issue tracker, although designed for reporting software bugs, was repurposed as a general task management system to track development activities and assign responsibilities.

With the adoption of Agile, team structure and workflow transformed to a fully horizontal organization, where all members of the team can propose work items. Responsibility for stakeholder engagement is shared: team members working on collaborative external projects communicate directly with project partners. New roles, such as Scrum Master and Reviewer, were introduced to ensure oversight and alignment. Team interactions became more structured through regular meetings like sprint planning meetings, daily stand-ups, and sprint reviews, giving rhythm and organisational structure to the workflow and creating regular opportunities for feedback and mutual aid and faster convergence on solutions. As a result, team members gained greater awareness of one another's activities and a clearer understanding of shared objectives. The adoption of a non-strictly-technical Planner tool enhanced development monitoring and coordination by defining tasks and managing their progress and priority.

A poster [4] on the adoption of the agile methodology for SPUMA was presented at the 3rd OpenFOAM Italian Workshop in 2026.

[4] V. Sibilia, E. Pascolo, S. Bnà, T. Zanelli, G. Giaquinto, A. Piccolo, C. D. Arlandini, The Agile Advantage for the SPUMA code Reshaping Human Interaction Workflows for Faster and High-Quality CFD Software Development (2026)

Showcase of SPUMA potential: automotive industry grade case

Parallel scalability

Strong scaling results conducted on Leonardo HPC system (see Table 1) show a significant speed up of SPUMA when compared to standard OpenFOAM (v2412).

Table 1: Hardware specifications of Leonardo HPC system

Partition	**Booster**	**DCGP**
Model	Atos BullSequana X2135 "Da Vinci" single-node GPU blade	Atos BullSequana X2140 three-node CPU blade
Racks	116	22
Nodes	3456	1536
Processors	Single socket 32 cores Intel Ice Lake CPU. 1 x Intel Xeon Platinum 8358, 2.60GHz TDP 250W	Dual socket 56 cores Intel Sapphire Rapids CPU. 2 x Intel Xeon Platinum 8480p, 2.00 GHz TDP 350W
Accelerators	4 x NVIDIA Ampere GPUs/node, 64GB HBM2e NVLink 3.0 (200GB/s)	-
Cores	32 cores/node	112 cores/node
RAM	512 (8x64) GB DDR4 3200 MHz	512 (16 x 32) GB DDR5 4800 MHz

The test case chosen for these simulations is the DrivAer in the open-closed cooling configuration taken from the OpenFOAM HPC repository, representing a production-like automotive RANS simulation consisting of 236 million cells (see Figure 1). Figure 3 shows the streamlines of the flow around the geometry. Using this test we showcase the ability of SPUMA to seamlessly interoperate with a third-party linear algebra library, in this case

AMGX (developed and released by NVIDIA), without additional host-device copies overhead, to offload the pressure equation solution to its algebraic multigrid framework.

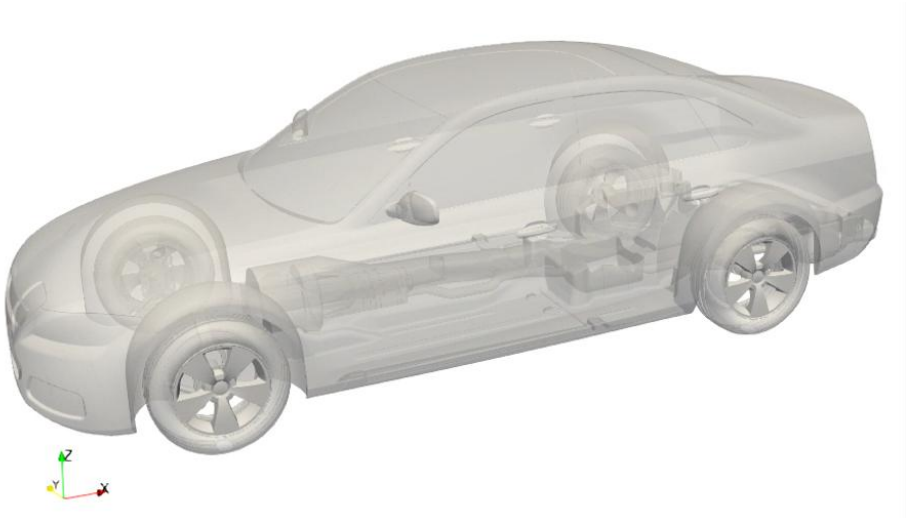


Figure 1: DrivAer Geometry, image courtesy of Charles Mockett, Hendrik Hetmann and Felix Kramer (Upstream CFD GmbH), 2022-2023

Looking at the results shown in Figure 2, it emerges that a single A100 GPU of Leonardo is equivalent to approximately 220 CPU cores of the Leonardo Data Centric General-Purpose partition (see Table 1). It can also be noted that 8 GPUs are faster than 1600 CPU cores.

Multigrid solver tests

The 236 million cells open-closed cooling DrivAer test case was also used to evaluate the performance of OpenFOAM's native geometric/algebraic multigrid (GAMG) linear solver on GPU. Unlike a purely algebraic multigrid such as AMGX, this solver uses geometric information from the mesh to build a hierarchy of progressively coarser matrices on which the solution is smoothed. On GPU, this presents unique challenges, as the coarser meshes are too small to exploit resources effectively. Possible solutions include agglomerating domain partitions or switching to CPU for coarser meshes, however, this was left for future work. On the other hand, effort was spent in finding optimal parameters for the GAMG solver on GPU.

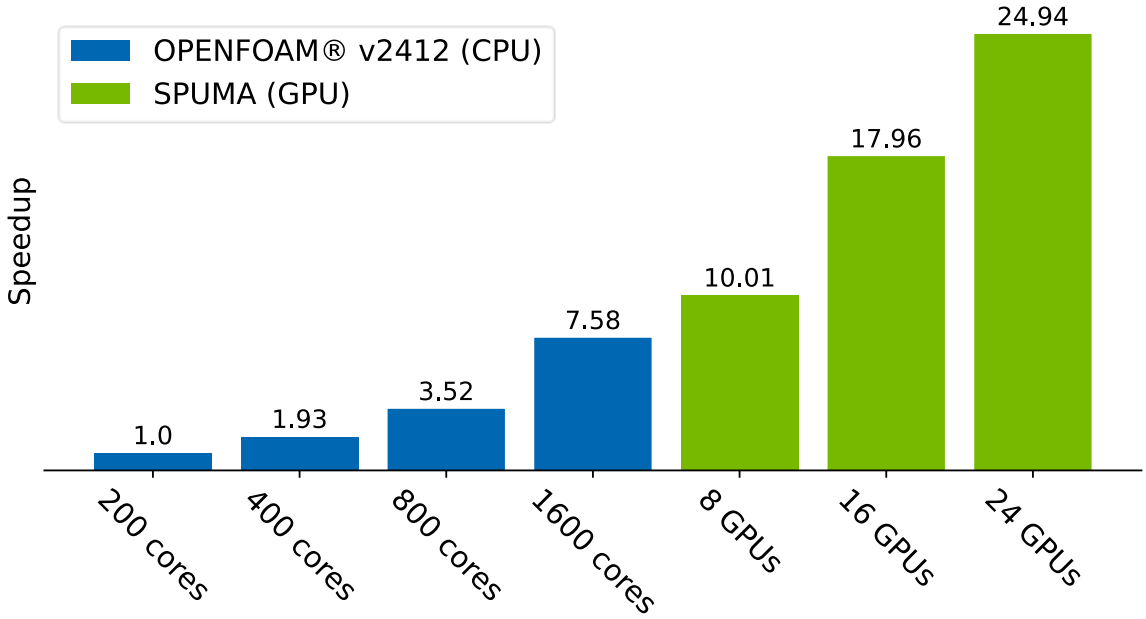


Figure 2: Strong scaling results of the HPC DrivAer test case: speedups relative to the 200 cores CPU run

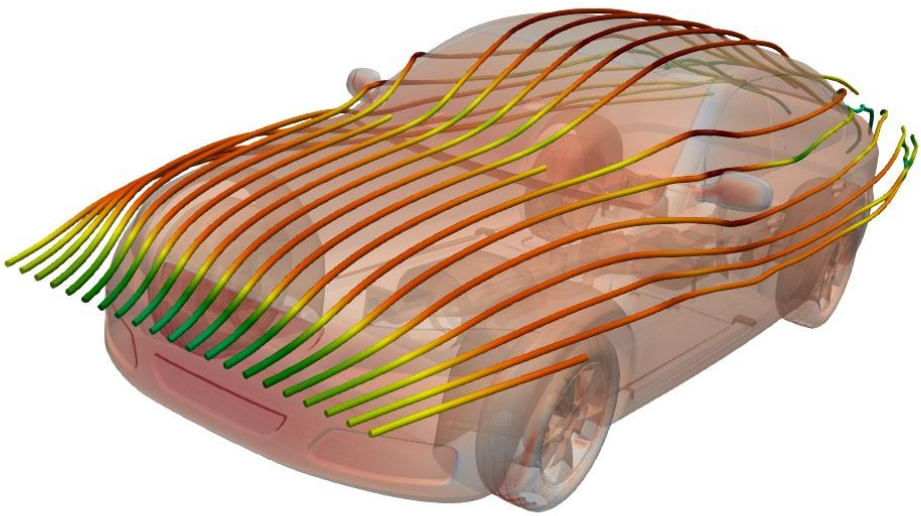


Figure 3: Visualization of streamlines around the DrivAer geometry, coloured with the pressure field

The GAMG tests were run on the Booster partition of the Leonardo cluster (see Table 1) and on the GPU partition of LUMI (see Table 2). Table 3 shows a comparison between different run configurations on the two clusters.

Table 2: Hardware specifications of LUMI HPC system

Partition	**LUMI-G**	**LUMI-C**
Nodes	2978	2048
Processors	single socket 64 cores AMD Trento CPU. 1 x AMD EPYC 7A53, 3.5GHz	Dual socket 64 cores AMD Milan CPU. 2 x AMD EPYC 7763, 2.45 GHz base, 3.5 GHz boost
Accelerators	4 x AMD MI250X GPUs 2nd Gen AMD CDNA architecture 2 x 64GB HBM2e (3200 GB/s)	-
Cores	64 cores/node	128 cores/node
RAM	512 (8x64) GB DDR4	256 (8 x 32) GB (or higher)

Table 3: Total simulation times in seconds for GAMG tests on Leonardo and LUMI

Cluster partition	**Floating point precision**	**Velocity solution**	**2 Nodes (8 GPUs)**	**4 Nodes (16 GPUs)**
Leonardo Booster	Double (DP)	Segregated	7951,89	4780,7
Leonardo Booster	Double (DP)	Coupled	7532,39	4475,62
Leonardo Booster	Mixed (SPDP)	Segregated	6091,25	3701,34
Leonardo Booster	Mixed (SPDP)	Coupled	5086,79	**3413,30**
LUMI-G	Double (DP)	Segregated	8343,20	5276,77
LUMI-G	Double (DP)	Coupled	8919,76	5486,10
LUMI-G	Mixed (SPDP)	Segregated	6608,27	4471,05
LUMI-G	Mixed (SPDP)	Coupled	5697,13	3854,75

On LUMI, each GPU is composed by two Graphics Compute Dies, which are seen as separate devices by the scheduler. Hence, while on Leonardo the domain is partitioned in 8 and 16 parts for the 2 and 4 node runs, on LUMI the domain is partitioned in 16 and 32 parts, affecting scalability.

The tests compare two solution strategies for the velocity, segregated, as in standard OpenFOAM, or coupled, in which each component is solved for together, avoiding the overhead of separate solutions.

Table 3 shows the advantages of using a mixed precision approach: single precision for matrix assembly and double precision for the solution. This is of particular interest for GPU computing, as current hardware trends show a preference for optimizing lower precision floating point operations.

SPUMA's potential for time and cost reductions

The advantages brought by SPUMA are not limited to the increased access to large, high-fidelity simulations. For problems of equal size, GPU hardware allows for considerable speedups compared to CPUs. This is interesting for applications requiring low response times, such as weather forecasting or the racing industry. It should also be noted that GPUs have a much lower energy cost per FLOPS than traditional CPUs, reducing the economic burden as well as the environmental impact of CFD simulations.

Additionally, for common industrial use cases, mesh sizes are generally around a few millions of cells, meaning that a single GPU may suffice for a wide range of applications. Soon, a PC workstation with one or two mid-tier GPUs installed could become a cost-effective alternative to a dedicated CPU-based server for the simulation workflows of many small and medium enterprises.

Adopting the Agile methodology improved SPUMA's development process by increasing transparency, enhancing coordination, and allowing quicker responses to changing requirements. The shift from a seniority-driven model to a collaborative structure fostered better communication and accountability. The Agile methodology thus made the development process more efficient and adaptable, demonstrating its usefulness for the often-small teams working on scientific software.