EUROCC ITALY
HPC Competence Center

SPUMA:

the power of open-source CFD running on GPUs

Authors: Simone Bnà, Giuseppe Giaquinto, Tommaso Zanelli, Francesco Bottau, Ettore Fadiga

SPUMA the power of open-source CFD running on GPUs

Index

What OpenFOAM is

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®[1] 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, the same algorithm that we find 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).

[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.*

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 a 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, working in the industrial sector, 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 a cluster, 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®. The GPU-enabling 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

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 such as GPUs can offer. SPUMA (Simulation Processing on Unified Memory Accelerators) is a fork of OpenFOAM released by CINECA that enable the software to run on GPUs. The open-source community that maintains OpenFOAM does not have the same amount of resources to dedicate to 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 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 integration (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 run 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 started working on 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.

The approach finally chosen by CINECA was inspired by the recent GPU porting of the code NEKO, 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 in the code. The front-end has no dependencies on third-party libraries which are confined in the various backends.

This layer of abstraction, combined with the adoption of unified memory, a hardware/software technology feature available on recent GPU cards that allows to allocate a memory address space accessible by both CPUs and GPUs, produces 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 it guarantees a neutral approach towards the adopted GPU paradigm (CUDA, HIP, OpenMP, OpenCL, etc...).

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.

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.

Showcase of SPUMA potential: automotive industry grade case

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.

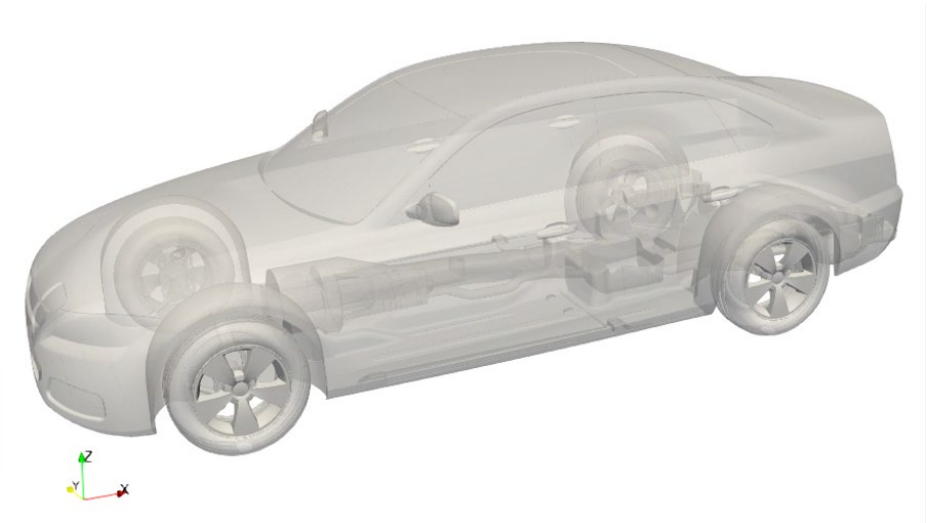

Figure 1: DrivAer Geometry, image courtesy of Charles Mockett, Hendrik Hetmann and Felix Kramer (Upstream CFD GmbH), 2022-2023

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), obtaining that 8 GPUs are faster than 1600 cores.

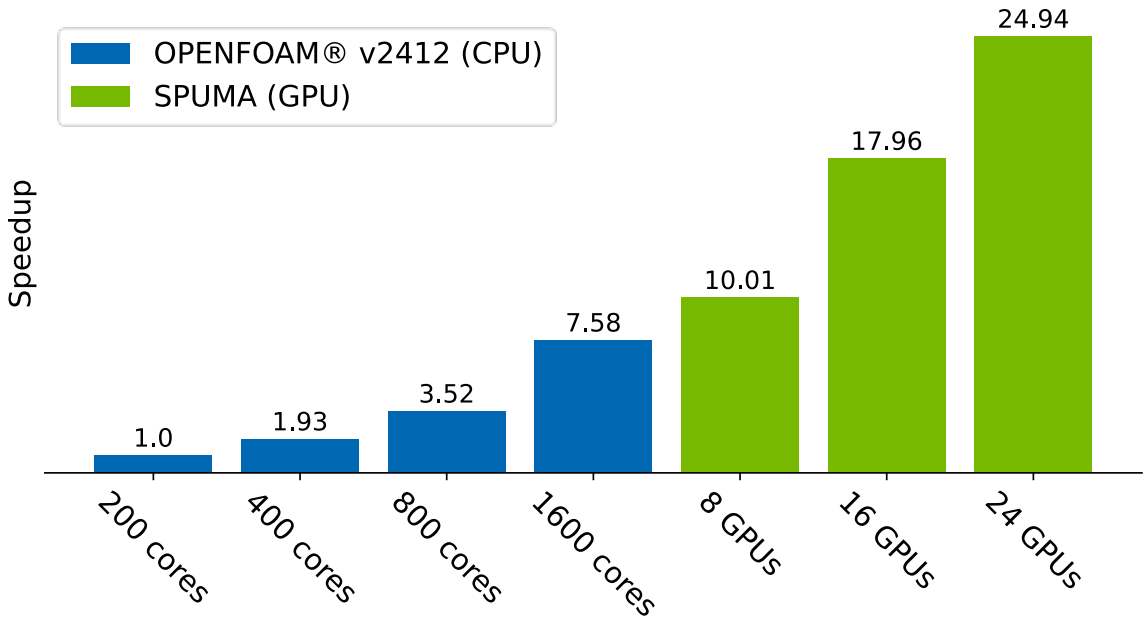

Figure 2: Strong scaling results of the HPC DrivAer test case: speedups relative to the 200 cores CPU run

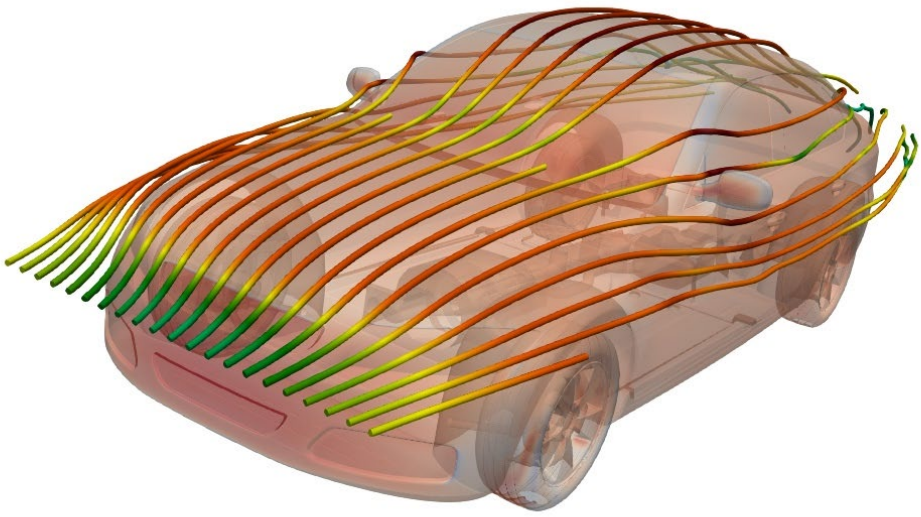

Figure 3: Visualization of streamlines around the DrivAer geometry, coloured with the pressure field

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.