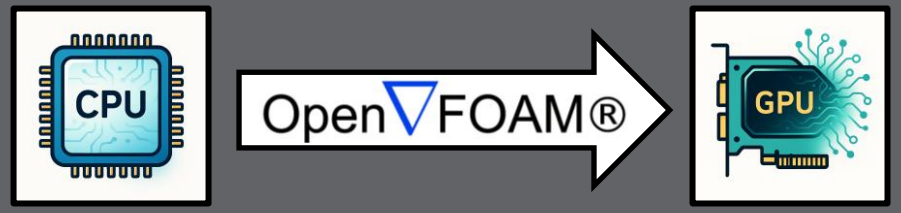
upstreamCFD

Benchmarking different OpenFOAM GPU implementations: Recent progress and challenges ahead

Hendrik Hetmann
hendrik.hetmann@upstream-cfd.com
Marian Fuchs
marian.fuchs@upstream-cfd.com
Charles Mockett
charles.mockett@upstream-cfd.com



15.04.2026

Confidential

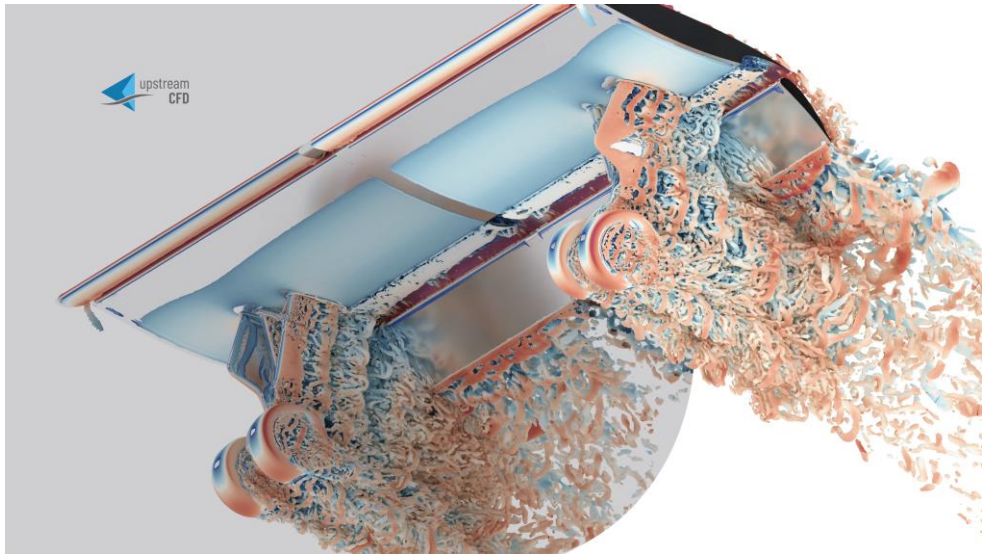
Our motivation at Upstream CFD

- We are focussed on conducting scale-resolving simulations (SRS) for turbulent flows with OpenFOAM:
  - Simulations of 100M+ meshes on thousands of CPU-cores
  - GPU is hence an enabling technology for SRS

- The need for a high-performance GPU solver is perceived by our customers as very urgent:
  - Number of GPU-capable (commercial) CFD solvers has grown rapidly: ANSYS (GPU), StarCCM+ (GPU), Luminary Cloud, Cadence CharLES
  - The pressure to switch from OpenFOAM to a GPU-capable commercial code is growing strong

  → **Bias towards methods available now or soon**

- Most promising OpenFOAM GPU ports currently (in our opinion):
  - SPUMA (CINECA) & NeoN (TU Munich & other contributors)



SPUMA: Simulation Processing in Unified Memory Accelerators

upstream CFD

- **Approach and key features:**
  - Port of OpenFOAM to GPU by identifying and grouping three basic low-level algorithm types and replacing them with GPU optimised variants:
    - For-loop
    - Reduction summation (e.g. scalar product)
    - Reduction comparison (e.g. min/max functions)
  - Unified memory + memory pool approach:
    - Almost all OpenFOAM functions will work without error
    - Non-ported features will keep running on CPU and slow down the simulation significantly
    - Incremental porting possible
- Link to paper pre-print for more details, full published paper here

- **Developed by:** Simone Bnà and team at CINECA High Performance Computing Department

CINECA

- **Status:**
  - Current release 0.1-v2412:
    - Most common solvers GPU ready: *pisoFoam*, *pimpleFoam*, *icoFoam*, *etc.* ...
    - Standard turbulence models available
    - Most common linear solvers, preconditioners and smoothers available
    - For more info see: https://gitlab.hpc.cineca.it/exafoam/spuma/-/wikis/GPU-support
  - Currently working on features for rotating machinery

GitLab Repository: https://gitlab.hpc.cineca.it/exafoam/spuma (GPL3 License)

NeoN: A next-generation framework for computational fluid dynamics

upstream CFD

- **Approach and key features:**
  - Rewrite of modern CFD backend from scratch
    → designed for GPUs from day one, avoid any technical debt from legacy OpenFOAM
  - Leverages best-in-class scientific libraries: no reimplementation of linear solvers
    → use the best available (Ginkgo, PETSc, SUNDIALS)
  - Lazy evaluation removes unnecessary temporary data
    → lower memory use and faster execution
  - Platform portability and vendor independence through Kokkos
  - Sustainable and modern DevOps: C++20, CMAKE build-system, comprehensive CI/CD pipelines, automated testing and quality assurance from day one
  - Modern accessible python frontend: Facilitate coupling with ML/AI and external tools

- **Developed by:** Team of Hartwig Anzt at Chair of Computational Mathematics, TUM
  - Plus numerous community contributors

Technische Universität München

- **Status:**
  - Current release 0.2:
    - *neolcoFoam* (first implementation of a transient incompressible flow solver)
    - All required operators to formulate PDEs for scalar and vector fields
  - Next release 0.3:
    - Performance and memory optimisation
    - Multi-GPU MPI support
    - SA-DDES turbulence model

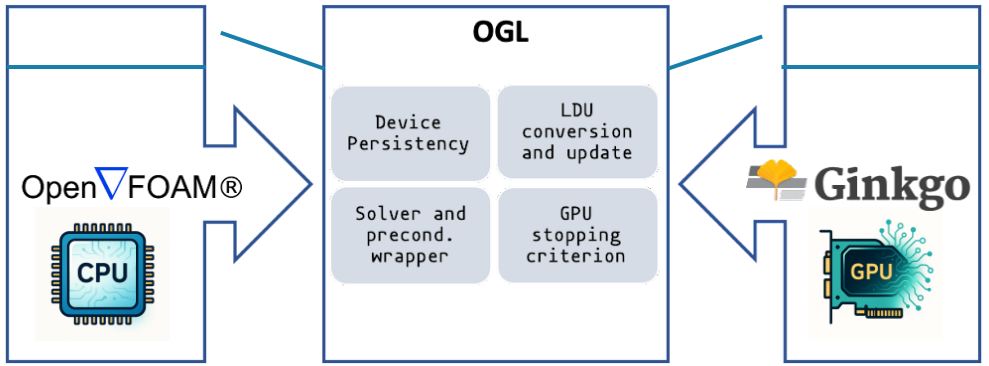
GitHub Repositories: github.com/exasim-project/NeoN (MIT License) github.com/exasim-project/NeoFoam (GPL3 License)

Lessons learned from previous EXASIM project

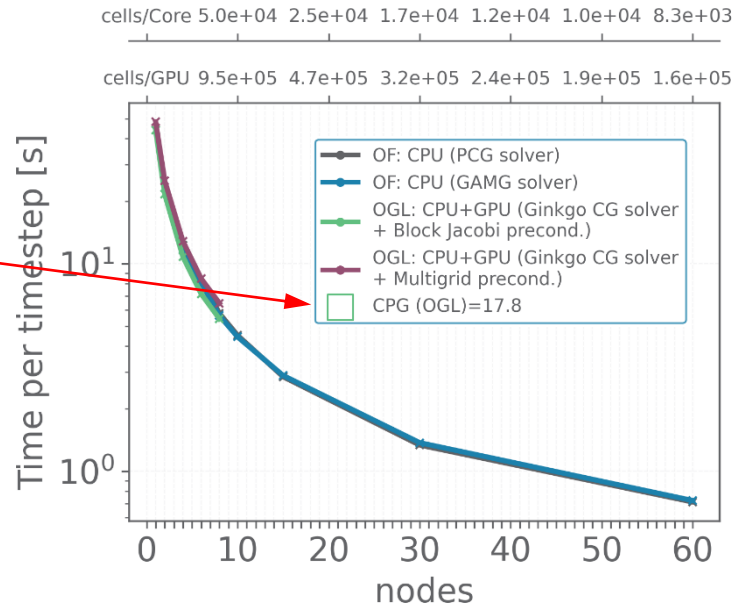
- EXASIM: German-funded project (2022-2025)
  - Joint project between Karlsruhe Institute of Technology (KIT), TU Munich and Upstream CFD

- Main approach investigated in project:
  - Offloading of linear solver(s) to GPU:
    - Matrix assembly remains on CPU
    - Performance inherently limited by Amdahl's law and data transfer between CPU & GPU
  - Comprehensive test case repository for benchmarking was established:
    - 8 cases of different domains (academic, automotive, wind energy)
    - Link: https://github.com/exasim-project/micro_benchmarks

OGL (OpenFOAM-Ginkgo Layer):

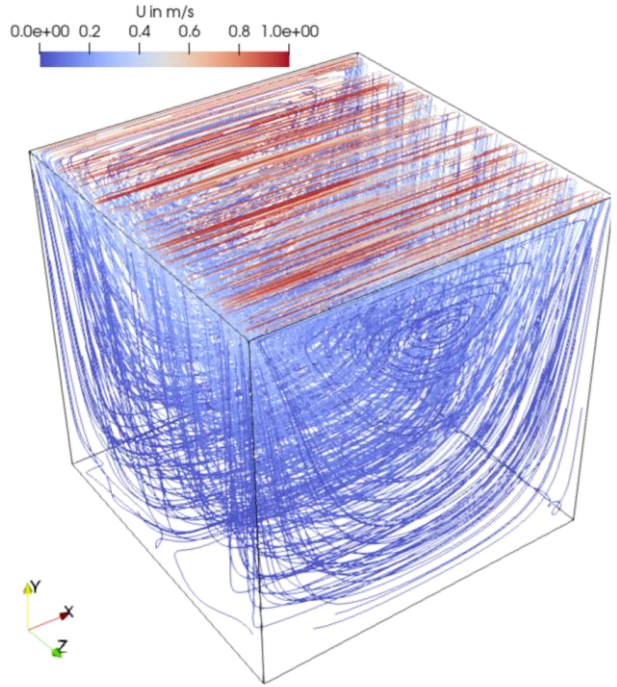


strong scalability study for automotive test case → gains with OGL not significant enough to validate the approach



Considerations for GPU benchmarking

- It's very common to see "marketing" numbers for GPU codes
  - e.g. "10x speedup"
- Without context, these numbers are meaningless:
  - What is the baseline (speedup compared to what?)
  - How many GPUs vs. how many CPU cores?
  - What type/generation of GPU and CPU?
    - And what do they cost to buy and run?
  - Size/type of application case?
  - Numerical settings?
  - Are both the GPU and CPU setups running at maximum efficiency?
  - …
- Transparent data is essential for decision-making
- Users care about minimising Time, Energy and Cost to Solution ("TECS")
- It's unfortunately very complex. We propose a framework/process for extracting meaningful metrics



Is the "pressure-heavy" lid-driven cavity really the most representative benchmark case for CPU vs. GPU?

Overview of approach (no formulas)

Strong scalability testing

Normalisation

GPU vs CPU comparison

Raw data:
- Wall-clock time
- No. devices (GPUs, CPU cores)
- Hardware used
- No. grid cells
- No. iterations

FVOPS
- Finite Volume Operations Per Second
  - 1 FVO = 1 pressure-velocity loop on 1 grid cell
- Throughput

FVOPS / device
- Device efficiency

Grid cells / device
- Degree of parallelisation

COE
- Coefficient Of Equivalence
- No. CPU cores needed to match performance of each GPU
- **Key question:** which data points to compare?
  - Most efficient?
  - Node-to-node?
  - Fixed time-to-solution?

TECS reduction factors:
- Time-to-solution
  - a.k.a. "speedup"
- Energy-to-solution
  - Need: Device power consumption
- Cost-to-solution
  - Need: $ / device-hr

User-oriented quantities

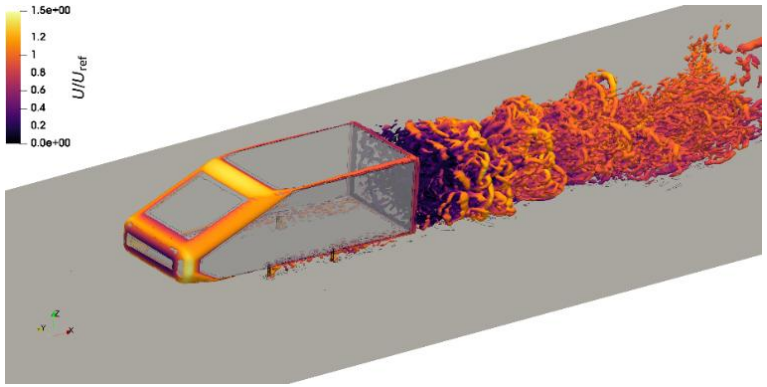
Valid for different problem sizes

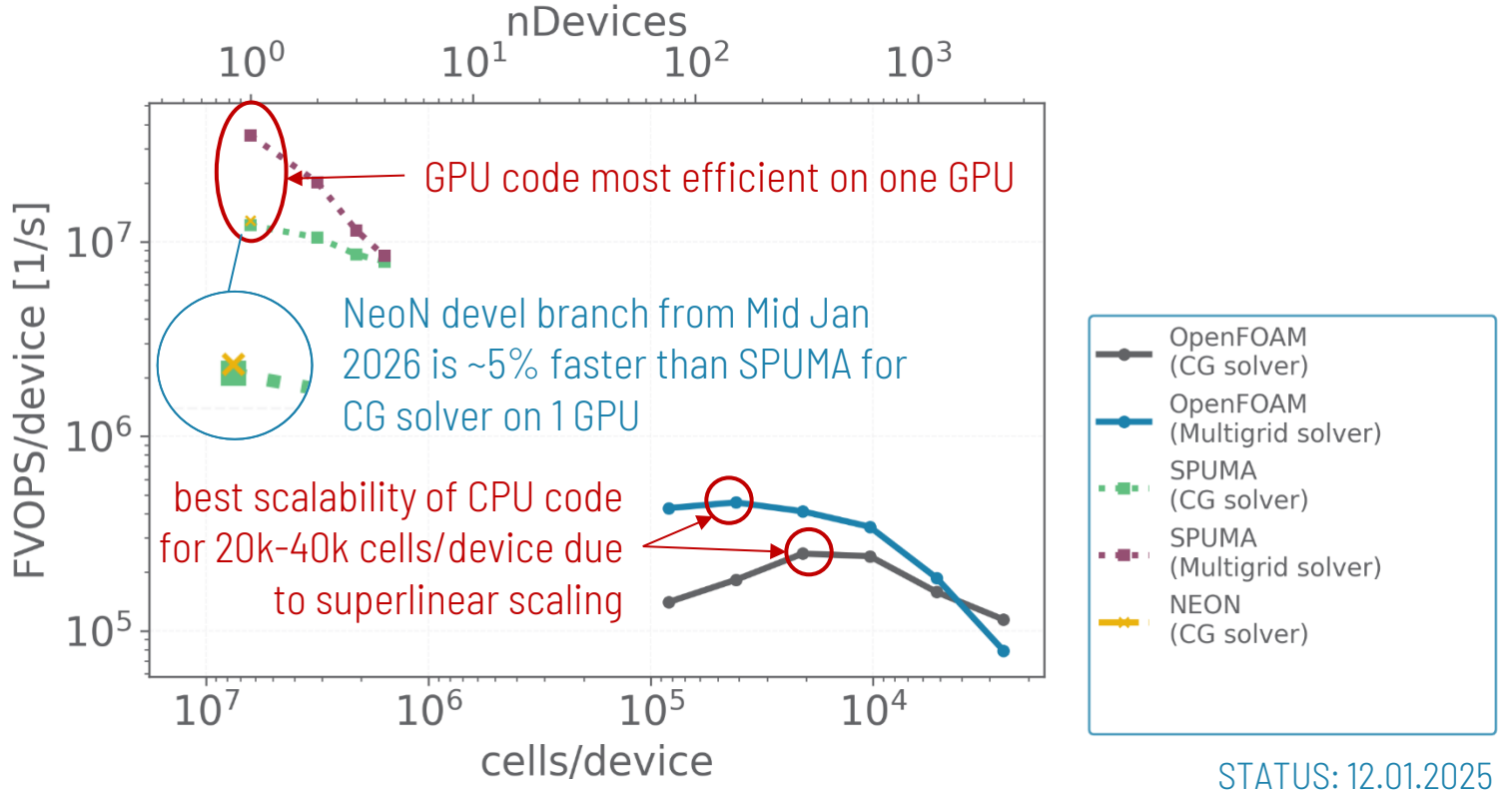
Absolute values

Relative values (baseline must be defined)

Benchmarking results: GPU vs. CPU for NeoN & SPUMA

- Test case: Windsor body
  - Validation case from AutoCFD workshop
  - Solver: pimpleFoam (explicit)
  - No turbulence model
  - Fixed grid size: 6.3M cells
  - Hardware:
    - CPU: Intel Xeon Platinum 8368 (gen 2021)
    - GPU: NVIDIA A100-40GB (gen 2020)

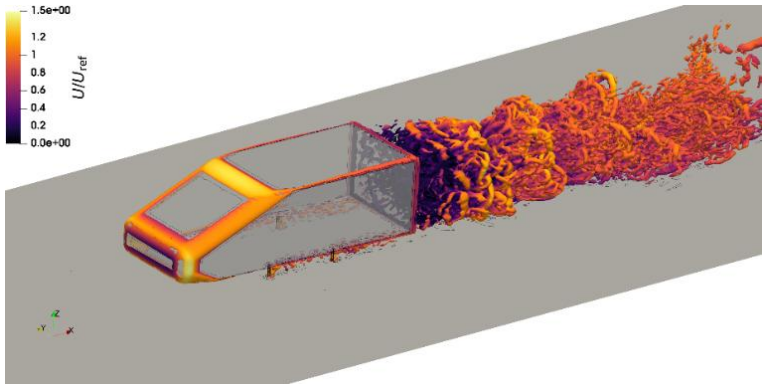


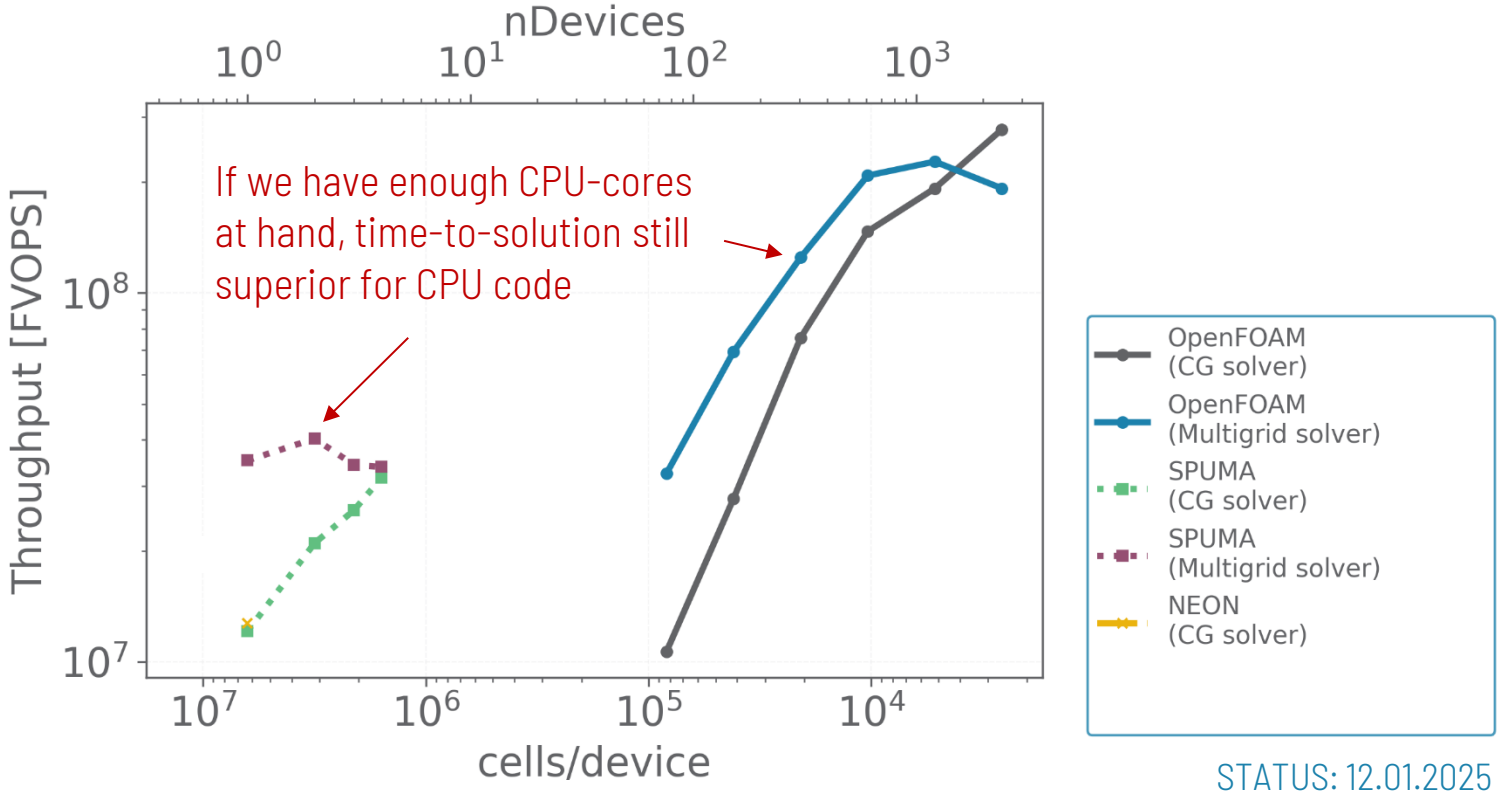


STATUS: 12.01.2025

Benchmarking results: GPU vs. CPU for NeoN & SPUMA

- Test case: Windsor body
  - Validation case from AutoCFD workshop
  - Solver: pimpleFoam (explicit)
  - No turbulence model
  - Fixed grid size: 6.3M cells
  - Hardware:
    - CPU: Intel Xeon Platinum 8368 (gen 2021)
    - GPU: NVIDIA A100-40GB (gen 2020)

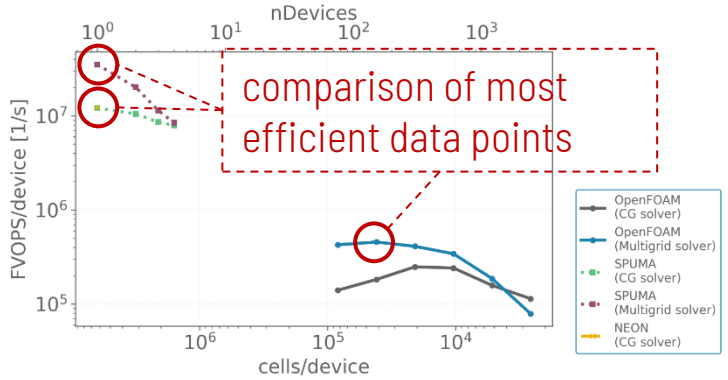




STATUS: 12.01.2025

Benchmarking results: GPU vs. CPU for NeoN & SPUMA

- Test case: Windsor body
  - Comparison of most efficient data points
  - NeoN devel branch from Mid Jan 2026 slightly faster than SPUMA with CG solver
  - Poor metrics for GPU solvers, probably due to small test case size



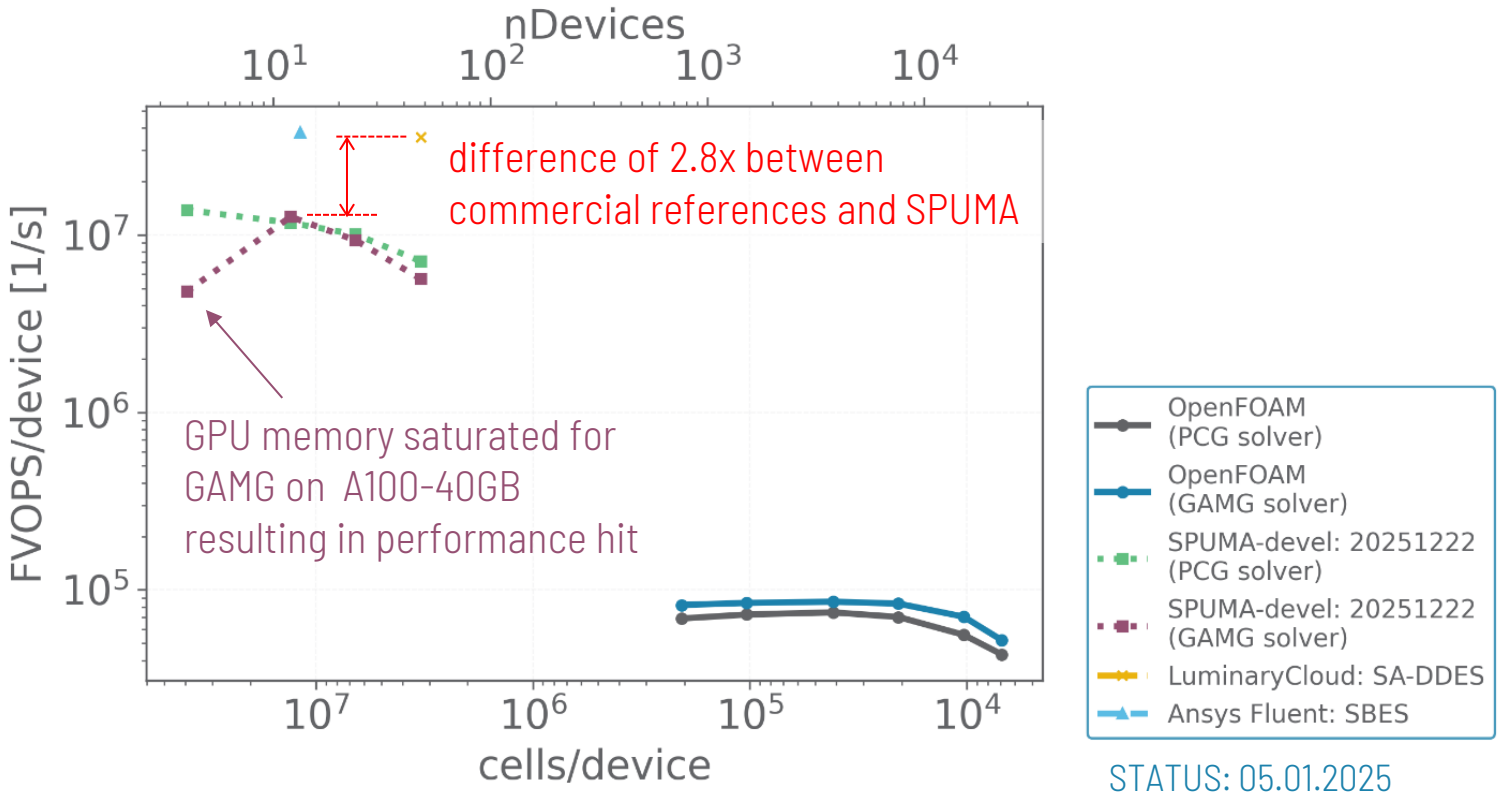
Code & solver	Simulation case	Compute hardware	Equivalent CPU cores per GPU COE_{opt}	Time-to-solution reduction factor $TTSF$	Cost-to-solution reduction factor $CTSF$
OpenFOAM-MG	Windsor body (RV1)	152 CPU-cores	-	-	-
NeoN-CG	Windsor body (RV1)	1x A100 (40GB) GPUs	28.0	0.19x	0.87x
SPUMA-CG	Windsor body (RV1)	1x A100 (40GB) GPUs	26.5	0.17x	0.82x
SPUMA-MG	Windsor body (RV1)	1x A100 (40GB) GPUs	77	0.50x	2.38x

CG – Conjugate gradients solver

MG – Multigrid solver

Benchmarking results: GPU vs. CPU for NeoN & SPUMA

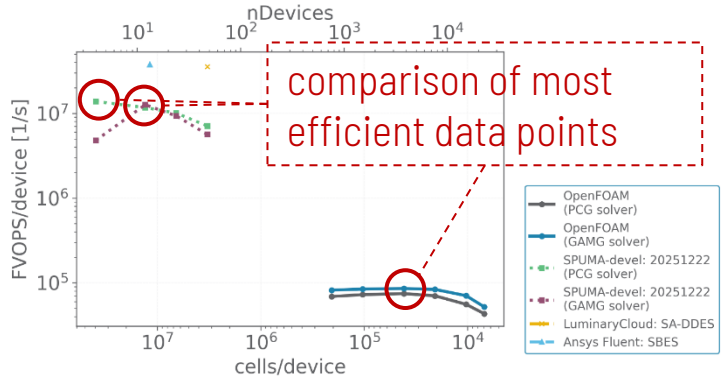
- Test case: DrivAer notchback
  - Validation case from AutoCFD workshop
  - Solver: pimpleFoam (implicit)
  - SPUMA: SA-DDES turbulence model
  - Fixed grid: 157M cells
  - Hardware:
    - CPU: Intel Xeon Platinum 8368 (gen 2021)
    - GPU: NVIDIA A100-40GB (gen 2020)



STATUS: 05.01.2025

Benchmarking results: Performance results SPUMA

- Test case: DrivAer notchback
  - Comparison of most efficient data points
  - Direct comparability only to Luminary Cloud (same case, mesh, turbulence model, GPU)



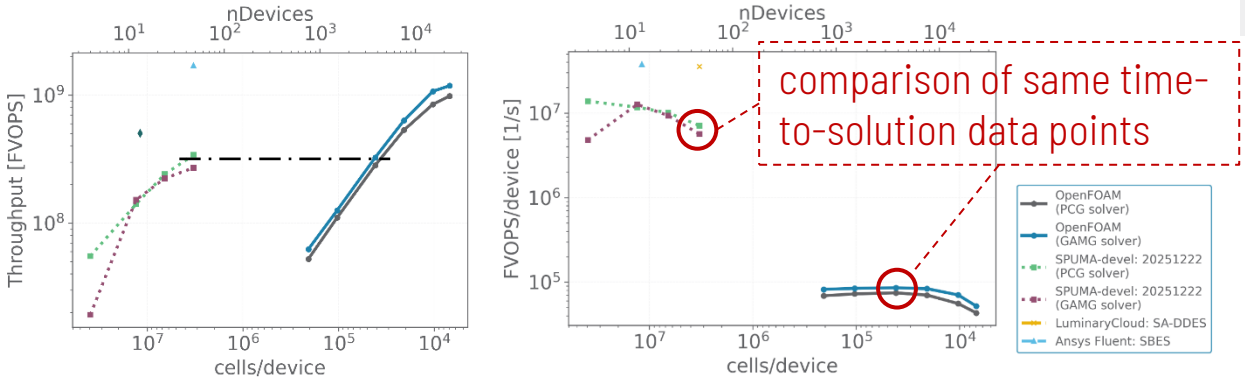
Code & solver	Simulation case	Compute hardware	Equivalent CPU cores per GPU COE_{opt}	Time-to-solution reduction factor $TTSF$	Cost-to-solution reduction factor $CTSF$
OpenFOAM-MG	AutoCFD4 DrivAer	3800 cores	-	-	-
SPUMA-CG	AutoCFD4 DrivAer	4x A100 (40GB) GPUs	161	0.17	4.98
SPUMA-MG	AutoCFD4 DrivAer	12x A100 (40GB) GPUs	148	0.47	4.57
Luminary Cloud	AutoCFD4 DrivAer	48x A100 (80GB) GPUs	415	5.24	12.86
Ansys	Custom DrivAer (rescaled to AutoCFD4)	13.3x A100 (40GB) GPUs	441	1.54	13.66

CG – Conjugate gradients solver

MG – Multigrid solver

Benchmarking results: Performance results SPUMA

- Test case: DrivAer notchback
  - Comparison of GPU data points with same time to solution as CPU reference
  - Direct comparability only to Luminary Cloud (same case, mesh, turbulence model, GPU)



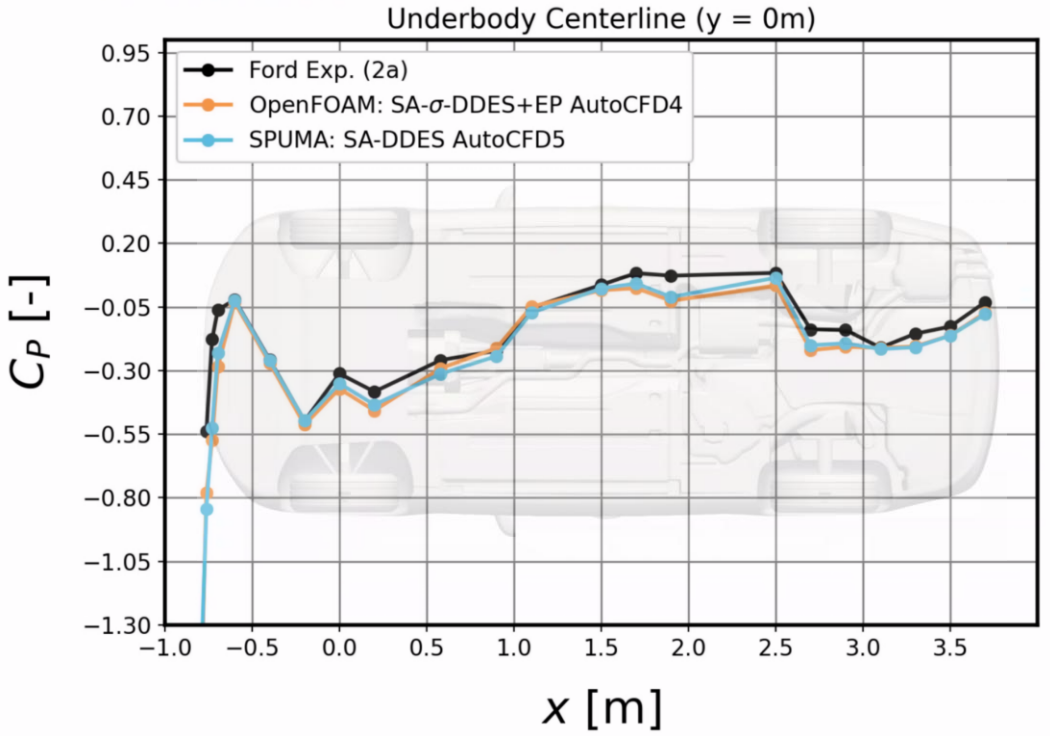
Code & solver	Simulation case	Compute hardware	Equivalent CPU cores per GPU COE	Time-to-solution reduction factor $TTSF$	Cost-to-solution reduction factor $CTSF$
OpenFOAM-MG	AutoCFD4 DrivAer	3800 cores	-	-	-
SPUMA-CG	AutoCFD4 DrivAer	48x A100 (40GB) GPUs	83	1	2.57
SPUMA-MG	AutoCFD4 DrivAer	48x A100 (40GB) GPUs	66	1	2.04
Luminary Cloud	AutoCFD4 DrivAer	48x A100 (80GB) GPUs	415	5.24	12.86
Ansys	Custom DrivAer (rescaled to AutoCFD4)	13.3x A100 (40GB) GPUs	441	1.54	13.66

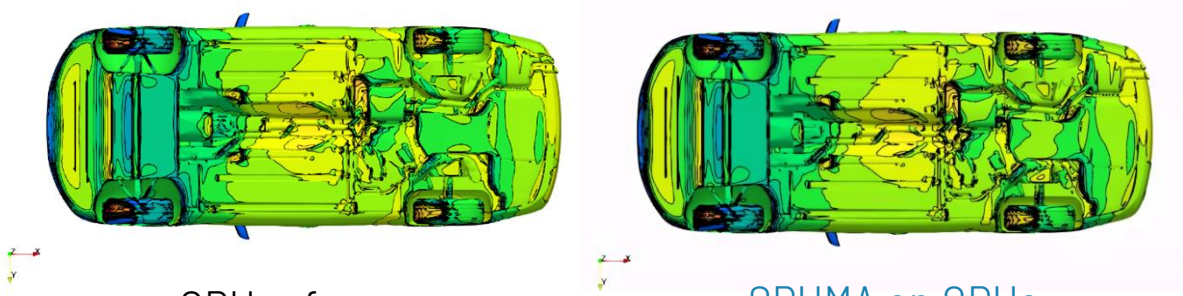
CG – Conjugate gradients solver

MG – Multigrid solver

Automotive proof-of-concept study for SPUMA

- Entire automotive simulation executed on GPUs:
  - Test case: DrivAer notchback from AutoCFD workshop
  - Hardware: 48x A100-40GB
  - SPUMA version:
  - Grid size: 123M
  - Turbulence model: SA-DDES
  - All necessary simulation outputs available (e.g. force coefficients)
  - Turnaround time:
    - Explicit solver setup: 11.3h
    - Implicit solver setup: 27.6h
  - Time-to-solution equivalent to OpenFOAM simulation on ~4000 CPU-cores
  - In terms of energy-to-solution, the GPU solver is roughly a factor of 2.5x more efficient





CPU reference

SPUMA on GPUs

Conclusions and outlook

- SPUMA has shown impressive performance, complex industrial cases (DrivAer DES) already possible:
  - 4.6x to 5.0x cost saving vs. CPU estimated
  - However, comparing most efficient points leads to speed-up <1 due to opposite scaling behaviour of CPUs vs GPUs (Importance of clear performance metrics)
  - 2.8x performance gap to commercial codes (i.e. Luminary Cloud & ANSYS GPU solver)

- NeoN devel version gave similar performance SPUMA for simplified (laminar, single-GPU):
  - Can the benefits of re-writing the code close the gap to the commercial competition with the next devel versions?

- Will the OpenFOAM community converge on one GPU port?
  - E.g. we know that OpenCFD is working with other partners from the UK on another OpenFOAM GPU port "gpuFOAM" (leveraging modern C++ Standard Template Library), which is designed to be minimal invasive in nature
  - We will probably have the opportunity to test the gpuFOAM implementation with the next months and will provide an update on our benchmarking activities at this year's OpenFOAM Workshop (29th June – 3rd July in Guimarães)

upstreamCFD

Thank you for your attention