# A portable algebraic implementation for reliable overnight industrial LES

**M. Mosqueda-Otero**[1], À. Alsalti-Baldellou[1,2], X. Álvarez-Farré[3], J. Plana-Riu[1], G. Colomer[1], F.X. Trias[1], A. Oliva[1]

[1] Heat and Mass Transfer Technological Centre, Universitat Politècnica de Catalunya, Spain
[2] Department of Information Engineering, University of Padova, Italy
[3] High-Performance Computing Team, SURF, The Netherlands

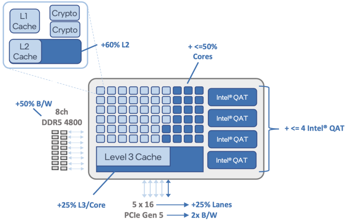
35$^{th}$ Parallel CFD International Conference 2024
Sep 2$^{nd}$ - 4$^{th}$, 2024
Bonn, Germany

Centre Tecnològic de Transferència de Calor
Universitat Politècnica de Catalunya

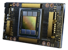
# Motivation

The continuous evolution of hardware, coupled with the widespread adoption of accelerators across various tech domains, has driven the development of modern hybrid HPC architectures.

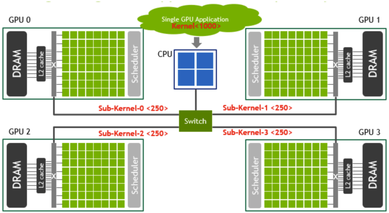
**Intel Xeon $4^{th}$ gen CPU architecture**[1]







**GPU-CPU hybrid architecture**[2]



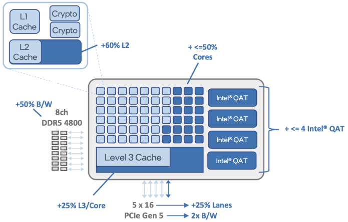

[1] D. Coyle et al. *Maximizing vCMTS Data Plane Performance with 4th Gen Intel® Xeon® Scalable Processor Architecture*. July 2023

[2] U. Milic et al. "Beyond the socket: NUMA-aware GPUs". In: *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. 2017. DOI: 10.1145/3123939.3124534

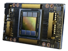
# Motivation

The continuous evolution of hardware, coupled with the widespread adoption of accelerators across various tech domains, has driven the development of modern hybrid HPC architectures.

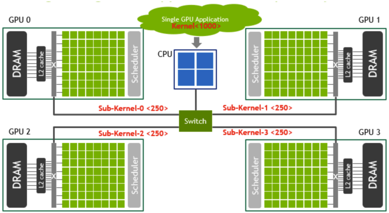
**Intel Xeon 4$^{th}$ gen CPU architecture**[1]







**GPU-CPU hybrid architecture**[2]




- How can we achieve portable CFD codes for different architectures and hardware vendors?

[1] Coyle et al., *Maximizing vCMTS Data Plane Performance with 4th Gen Intel® Xeon® Scalable Processor Architecture*

[2] Milic et al., "Beyond the socket: NUMA-aware GPUs"

# TFA+HPC$^2$

TFA+HPC$^2$ presents thoroughly conservative discretization methods[3] with a set of algebra-dominant kernels[4], easily portable to modern HPC architectures

**Main HPC$^2$ kernels**

| Kernels | Operation |
|---|---|
| `axpy` | Linear combination of vectors |
| `axty` | Element-wise product of vectors |
| `dot` | dot product of vectors |
| `SpMV` | Sparse matrix-vector product |

[3] F. X. Trias et al. "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured meshes". In: *Journal of Computational Physics* 258 (2014), pp. 246–267

[4] X. Álvarez-Farré et al. "HPC$^2$ – A fully portable algebra-dominant framework for heterogeneous computing. Application to CFD". In: *Computers & Fluids* 173 (2018), pp. 285–292

# TFA+HPC$^2$

TFA+HPC$^2$ presents thoroughly conservative discretization methods[3] with a set of algebra-dominant kernels[4], easily portable to modern HPC architectures

**Main HPC$^2$ kernels**

| Kernels | Operation |
|---|---|
| axpy | Linear combination of vectors |
| axty | Element-wise product of vectors |
| dot | dot product of vectors |
| SpMV | Sparse matrix-vector product |

- Do algebra-based implementations enable efficient portability of industrial CFD?

[3] F. X. Trias et al., "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured meshes"

[4] X. Álvarez-Farré et al., "HPC$^2$ – A fully portable algebra-dominant framework for heterogeneous computing. Application to CFD"

# Framework

## Base case

- Turbulent channel flow
- Using a Conjugate Gradient solver with a Jacobi preconditioner
- With an explicit time integration scheme
- Variable time-step
- Solving 10 time steps with a maximum of 800 iterations

## CPU system

- Strong and Weak scalability
- Performed in Marenostrum 5 GPP at BSC
  - CPU: Intel Xeon Platinum 8480+ (2×)
- Focus on MPI-Only vs. MPI+OpenMP

## GPU system

- Strong and Weak scalability
- Performed in Snellius GPU at SURF
  - CPU: Intel Xeon Platinum 8360Y (2×)
  - GPU: NVIDIA A100-40 GiB HBM2 (4×)
- Focus on OpenCL

# [CPU] Strong Scalability

## Considerations

- OpenMP config: 2 MPI processes with 2 comms threads and 54 computation threads
- Analysis performed
  - 1-node baseline: 112 CPU-cores
- Base workload of 525k CVs per CPU-core
  Mesh sizes:
  - 1-node: 350 × 480 × 350 - 58.8M

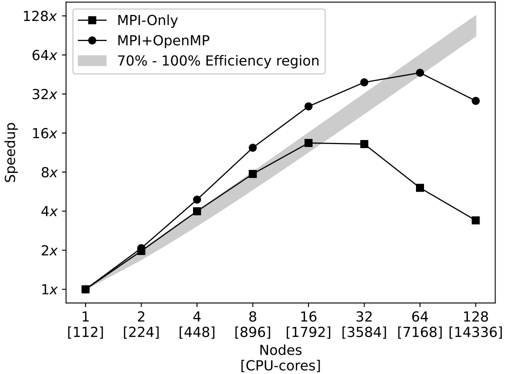



**Figure:** MPI-Only vs. MPI+OpenMP strong scalability on Marenostrum 5 GPP (1-node baseline)

# [CPU] Strong Scalability

## Considerations

- OpenMP config: 2 MPI processes with 2 comms threads and 54 computation threads
- Analysis performed
  - 1-node baseline: 112 CPU-cores
  - 16-node baseline: 1792 CPU-cores
- Base workload of 525k CVs per CPU-core

  Mesh sizes:
  - 1-node: 350 × 480 × 350 - 58.8M
  - 16-nodes: 800 × 1470 × 800 - 940.8M

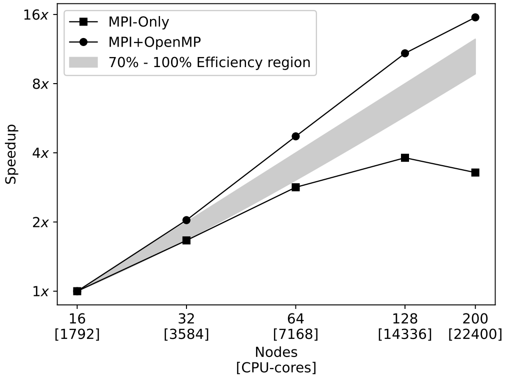



**Figure:** MPI-Only vs. MPI+OpenMP strong scalability on Marenostrum 5 GPP (16-node baseline)

# [CPU] Weak Scalability

## Considerations

- OpenMP config: 2 MPI processes with 2 comms threads and 54 computation threads
- Starting with 16 nodes (1792 CPU-cores) up to 200 nodes (22400 CPU-cores)
- Base workload of 525k CVs per CPU-core

  Mesh sizes:

  - 16 nodes: 800 × 1470 × 800 - 940.8M
  - 32 nodes: 1200 × 1960 × 800 - 1.88B
  - 64 nodes: 1200 × 3136 × 1000 - 3.76B
  - 128 nodes: 2000 × 2352 × 1600 - 7.52B
  - 200 nodes: 2000 × 2940 × 2000 - 11.76B

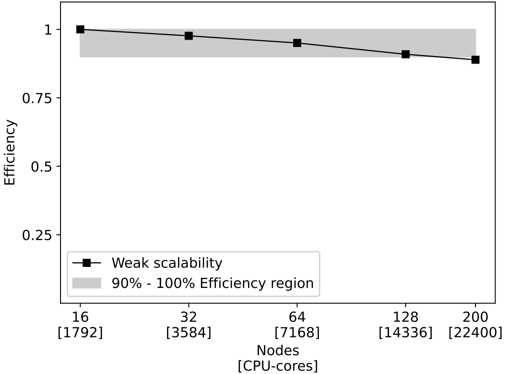



**Figure:** MPI+OpenMP weak scalability on Marenostrum 5 GPP

# [GPU] Strong Scalability

## Considerations

- Config: 4 MPI processes per node (1 MPI per GPU card)
- Analysis performed
  - 1-node baseline: 4 GPUs
- Base workload of 25.6M CVs per GPU
  Mesh size:
  - 1-node: $400 \times 640 \times 400$ - 102.4M

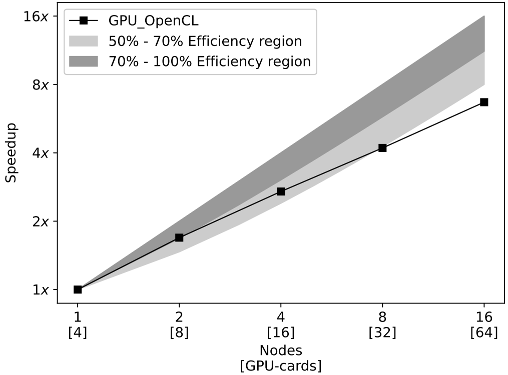



**Figure:** GPU strong scalability analysis on Snellius GPU island

# [GPU] Weak Scalability

## Considerations

- Config: 4 MPI processes per node (1 MPI per GPU card)
- Starting with 1 node (4 GPUs) up to 16 nodes (64 GPUs)
- Base workload of 25.6M CVs per GPU

  Mesh sizes:

  - 1 node: $400 \times 640 \times 400$ - 102.4M
  - 2 nodes: $800 \times 800 \times 320$ - 204.8M
  - 4 nodes: $800 \times 800 \times 640$ - 409.6M
  - 8 nodes: $800 \times 1280 \times 800$ - 819.2M
  - 16 nodes: $1280 \times 1600 \times 800$ - 1.64B

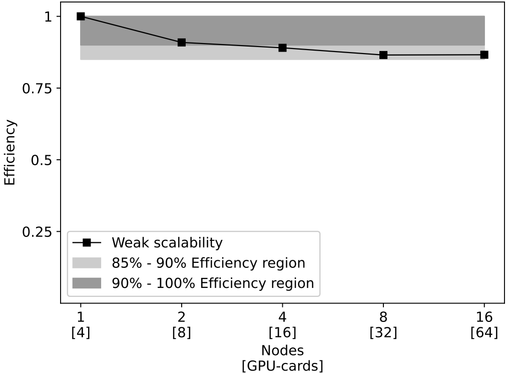



**Figure:** GPU weak scalability analysis on Snellius GPU island

# Parameters

### Equivalent Arithmetic Intensity ($AI_{eq}$)

$$AI_{eq} = \frac{\sum_{k \in K} \alpha_k FLOPS_k}{\sum_{k \in K} \alpha_k BYTES_k}$$

### Equivalent Performance ($P_{eq}$)

$$P_{eq} = \sum_{k \in K} \alpha_k P_k$$

### Data Throughput ($DT_{eq}$)

$$DT_{eq} = \sum_{k \in K} \alpha_k DT_k$$

Where $K$ refers to a set of $HPC^2$ kernels

| Kernels | Operation |
|---|---|
| axpy | Linear combination of vectors |
| axty | Element-wise product of vectors |
| dot | dot product of vectors |
| SpMV | Sparse matrix-vector product |

and

$$\alpha_k = \sum_{k \in K} \frac{N_k}{N_{total}}$$

# CPU Performance

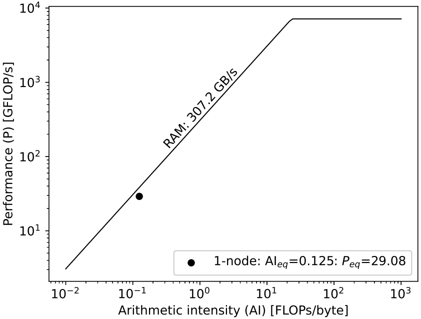



**Figure:** Roofline analysis on Marenostrum 5 GPP; baseline 1 node (112 CPU-cores) with 58.8M CVs grid

# CPU Performance

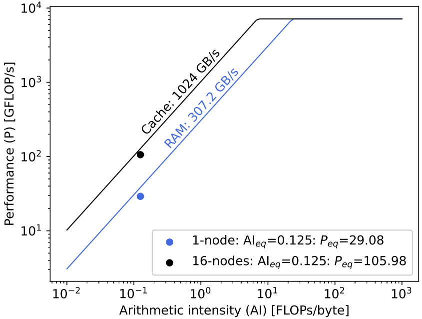



**Figure:** Hierarchical roofline analysis on Marenostrum 5 GPP; showing 1 node (112 CPU-cores) with 58.8M CVs grid, and 16 nodes (1792 CPU-cores) for MPI+openMP strong scalability results

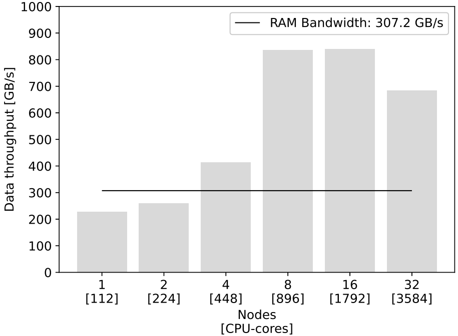



**Figure:** Memory bandwidth roofline analysis on Marenostrum 5 GPP; MPI+OpenMP strong scalability data transfer ($BW_{eq}$)

# GPU Performance

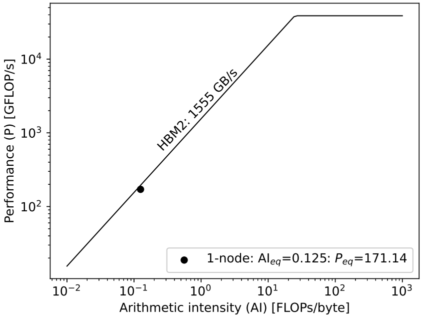



**Figure:** Roofline analysis on Snellius GPU; baseline 1 node (4 GPU-cards) with 102.4M CVs grid

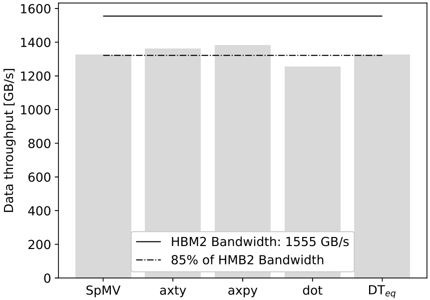



**Figure:** Memory bandwidth roofline analysis on Snellius GPU; kernels and equivalent data transfer for 1-node baseline case

# GPU Performance

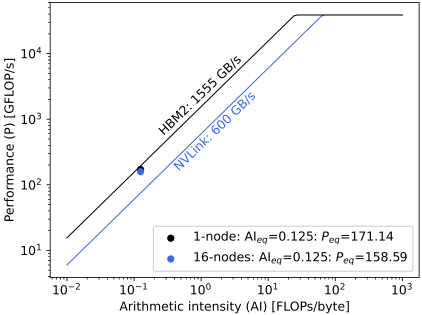



**Figure:** Hierarchical roofline analysis on Snellius GPU; showing 1 node (4 GPU-cards) with 102.4M CVs grid, and 16 nodes (64 GPU-cards) with 1.64B CVs grid

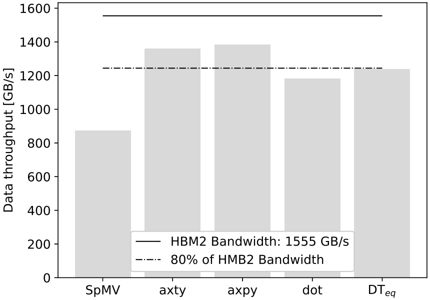



**Figure:** Memory bandwidth roofline analysis on Snellius GPU; kernels and equivalent data transfer for 16-nodes weak scalability solution

# Energy consumption

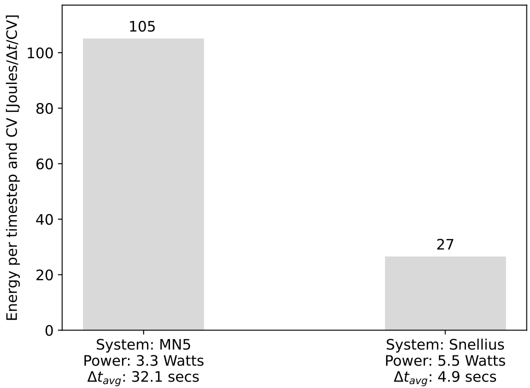



**Figure:** Energy consumption in Watts-sec per time step; normalized by the number of operations and problem size solved

# Conclusions

- TFA+HPC$^2$ design improves portability into different HPC architectures.
- Efforts to increase the arithmetic intensity are required to improve its memory-bound behavior.

# Conclusions

- TFA+HPC$^2$ design improves portability into different HPC architectures.
- Efforts to increase the arithmetic intensity are required to improve its memory-bound behavior.
- CPU systems exhibit superior strong scalability, with the hybrid paradigm (MPI+OpenMP) delivering higher performance than MPI-only, primarily due to the benefits of cache utilization and reduced communication overhead.

# Conclusions

- TFA+HPC² design improves portability into different HPC architectures.
- Efforts to increase the arithmetic intensity are required to improve its memory-bound behavior.
- CPU systems exhibit superior strong scalability, with the hybrid paradigm (MPI+OpenMP) delivering higher performance than MPI-only, primarily due to the benefits of cache utilization and reduced communication overhead.
- Despite high power requirements, execution on GPU systems results in energy consumption that is approximately 74% lower than CPU systems.

# Conclusions

- TFA+HPC$^2$ design improves portability into different HPC architectures.
- Efforts to increase the arithmetic intensity are required to improve its memory-bound behavior.
- CPU systems exhibit superior strong scalability, with the hybrid paradigm (MPI+OpenMP) delivering higher performance than MPI-only, primarily due to the benefits of cache utilization and reduced communication overhead.
- Despite high power requirements, execution on GPU systems results in energy consumption that is approximately 74% lower than CPU systems.
- Finally, weak scaling analysis delivers great efficiency, showing the capability of this implementation to scale to demanding Industrial applications.

# Future work

- To perform large-scale urban simulations leveraging spatial regularities[5]
- Continue exploring strategies to increase GPU computation.

[5] À. Alsalti-Baldellou et al. "Lighter and faster simulations on domains with symmetries". In: *Computers & Fluids* 275 (2024), p. 106247. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2024.106247

# Thanks for your attention!!!