ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep 02-04 2024, Bonn, Germany

A PORTABLE ALGEBRAIC IMPLEMENTATION FOR RELIABLE OVERNIGHT INDUSTRIAL LES

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

[1] Universitat Politècnica de Catalunya - BarcelonaTech, Heat and Mass Transfer Technological Center, Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
e-mail: marcial.francisco.mosqueda@upc.edu

[2] Department ICEA, University of Padova, Via Francesco Marzolo, 9, 35131 Padova PD, Italy

[3] High-Performance Computing Team - SURF, Science Park 140, 1098 XG Amsterdam, The Netherlands

Key words: MPI+OpenMP, OpenCL, Heterogeneous computing, Large-Eddy Simulation, Overnight industrial applications

Abstract. The present work aims to assess the feasibility of large-scale simulations focusing on industrial applications. A symmetry-preserving discretization method for unstructured collocated grids is applied for LES simulations of turbulent flows. It ensures stability without introducing artificial dissipation and maintains cross-platform portability by relying on a minimal set of algebraic kernels. Challenges such as the low arithmetic intensity of sparse linear algebra and efficient computation are addressed. Finally, a scalability analysis of the algorithm under MPI-only vs. MPI+OpenMP parallelization paradigms and GPU architectures is presented, validating its effectiveness in enhancing parallel computational efficiency

1 INTRODUCTION

The continuous development of novel numerical methods and the rapid evolution of high-performance computing (HPC) systems has led to the increasing integration of computational fluid dynamics (CFD) into different industrial processes. However, CFD's progress faces obstacles, as early implementations struggled with the formerly compute-bound nature of processors, leading to the application of compute-centric programming models. Processor design evolved to overcome this limitation, leading to a disparity between computational power and memory bandwidth; as a result, the former grows faster, creating the need for intricate memory hierarchies that complicate traditional program optimization. Moreover, emerging APIs like CUDA, OpenCL, and HIP have made it challenging to transfer legacy codes to modern hardware; hence, porting algorithms and applications have become essential. On the other hand, conventional numerical methods in use throughout the industry, mainly centered in RANS models, struggle to meet the demands of more precise but expensive models, like LES or DNS. In this regard, the thoroughly conservative discretization method for unstructured grids proposed by [1] is adopted using TermoFluids Algebraic (TFA), our in-house code, constituting a novel and robust HPC^2 approach that can be easily implemented in existing open-source codes [2, 3] and hybrid supercomputers.

While computational power has improved, the time and resources required for detailed simulations remain a significant bottleneck. Achieving large-scale simulations is crucial for meeting industry demands for rapid decision-making, product development cycles, and broadening CFD industrial application fields. Therefore, our research aims to ensure the integration of modern CFD methodologies into the industry, allowing precise and accurate simulations of complex processes while efficiently harvesting available resources and reducing simulation costs under a limited timeframe.

2 Portability for CFD

Constructing codes based on a minimal set of kernels has become essential for ensuring portability, optimization, and code maintenance, especially given the diverse range of computational architectures and vendors. Additionally, the hybrid nature of modern HPC systems imposes further challenges due to the need to efficiently utilize processors and parallel accelerators, often requiring heterogeneous computations and complex data exchanges between them. However, traditional CFD codes typically rely on intricate data structures and computational routines, posing significant barriers to achieving portability. Therefore, a response centered on algorithms that rely on algebraic kernels, such as the sparse matrix-vector product (`SpMV`), the linear combination of vectors (`axpy`), an element-wise product of vectors (`axty`), and the `dot` product of vectors, emerge as solutions [3].

Code portability, optimization, and scalability are simplified by requiring minimal algebraic kernels. Nevertheless, this approach introduces two challenges and constraints: (i) computational challenges, including the low arithmetic intensity of the `SpMV` operation. This issue can be mitigated by employing the more computationally intensive sparse matrix-matrix product (`SpMM`), which proves beneficial in various scenarios whenever dealing with matrices, $\hat{\mathsf{A}} \in \mathbb{R}^{N \times N}$, decomposable as the Kronecker product of a diagonal matrix, $\mathsf{C} \equiv diag(\boldsymbol{c}) \in \mathbb{R}^{K \times K}$, and a sparse matrix, $\mathsf{A} \in \mathbb{R}^{N/K \times N/K}$, *i.e.*, $\hat{\mathsf{A}} = \mathsf{C} \otimes \mathsf{A}$. Hence:

$$\boldsymbol{y} = \hat{\mathsf{A}}\boldsymbol{x} \quad \Longrightarrow \quad (\boldsymbol{y}_1, \ldots, \boldsymbol{y}_K) = \mathsf{A}\left(c_1\boldsymbol{x}_1, \ldots, c_K\boldsymbol{x}_K\right), \tag{1}$$

where $\boldsymbol{x}_i, \boldsymbol{y}_i \in \mathbb{R}^{N/K}$. Replacing the `SpMV` with `SpMM` in such situations significantly reduces memory accesses and the memory footprint of operators by recycling matrix coefficients, and (ii) algorithmic challenges emerge, requiring, for instance, the re-definition of boundary conditions, which can be naturally addressed by defining them into an affine transformation, such as:

$$\boldsymbol{\varphi}_h \to \mathsf{A}\boldsymbol{\varphi}_h + \boldsymbol{b}_h, \tag{2}$$

facilitates an algebraic treatment suitable for explicit and implicit time integration methods [4].

As proposed by [5], a suitable approach to accelerate any Poisson's solver by exploiting domain symmetries. Where an adequate ordering of the unknowns led to the replacement of `SpMV` operations with `SpMM`, with an overall increase of compute-intensive kernel performance by 2.5x and a considerable reduction of the solver‘s footprint and setup costs, which potentially expanded its application to large-scale simulations.

3 Algorithm scalability analysis

Numerical experiments were designed to study TFA's base algorithm performance under different parallel paradigms by comparing MPI-only, centered on assigning one task per CPU-core, vs. MPI+OpenMP, based on 2 MPI process and 56 multithreaded executions, which allows for reducing the communication overheads due to its shared memory concept. In addition, a scalability analysis on hybrid HPC systems is possible due to TFA's underlying structure (based on minimal algebraic kernels), which delivers extended code portability for broad GPU hardware. In addition, a roofline model presents the implementation performance analysis.

A turbulent channel flow is solved by a conjugate gradient solver with a Jacobi preconditioner for Poisson‘s equation with an explicit time integration scheme with a variable time step. As the main focus relies on measuring TFA+HPC^2 kernels scalability, only 10 time-steps are performed for each case with a constraint of a maximum of 800 iterations. Finally, the solution considers the entire domain without exploiting symmetries, leaving `SpMV`, `axpy`, `axty`, and the `dot` product as the main algebraic kernels.

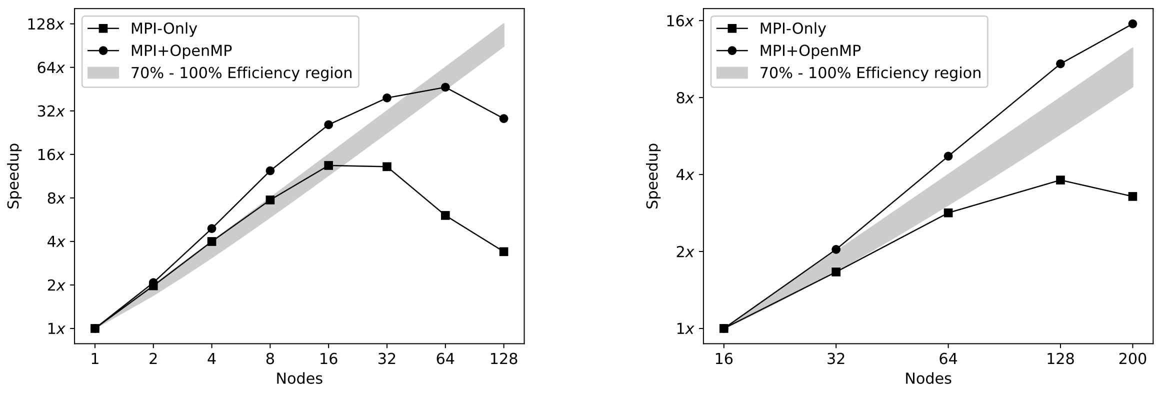

Figure 1: MPI-only vs MPI+OpenMP strong scalability; with a 350 × 480 × 350 - 58.8M CVs - grid (left plot) and a 800 × 1470 × 800 - 940.8M CVs - grid (right plot)

MPI-only and MPI+OpenMP tests were conducted on the MareNostrum 5 GPP supercomputer at BSC. Experiments run on nodes equipped with two Intel Xeon Platinium 8480+ (56 cores, 2 GHz, 105 MB L3 cache, and 307.2 GB/s memory bandwidth) with a total of 256 GB of RAM and interconnected through ConnectX-7 NDR200 Infinity Band.

Preliminary results analyze TFA's strong and weak scalability under both parallel paradigms (MPI-only and MPI+OpenMP). Figure 1 shows the strong scalability analysis for two baselines: (i) 1 node (left plot) with a $305 \times 480 \times 350$ grid and (ii) 16 nodes (right plot) with a $800 \times 1470 \times 800$ grid. Delivering a base workload of 525k control volumes (CV) per CPU-core. The analysis presents a significant super-linear speedup for hybrid processes with a fairly repeatable pattern over different baselines due to assumed cache effect. Additionally, Figure 2 presents a weak scalability analysis for the MPI+OpenMP test, showing an 11% drop in performance for 200 nodes (concerning a 16-node baseline).

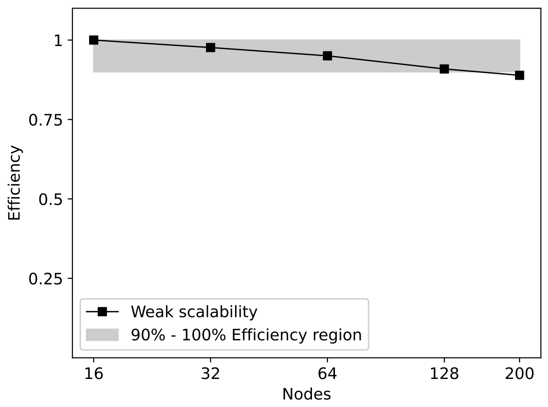

Figure 2: MPI+OpenMP weak scalability with 525k CV per CPU-core; starting with 16 nodes (1792 CPU-cores) up to 200 nodes (22400 CPU-cores)

In order to measure the implementation performance, an equivalent arithmetic intensity (AI_{eq}) and equivalent Performance (P_{eq}) were defined by applying a weighted average:

$$\mathrm{AI}_{eq} = \frac{\sum_{k\in K} \alpha_k \mathrm{FLOPS}_k}{\sum_{k\in K} \alpha_k \mathrm{BYTES}_k}, \tag{3}$$

and

$$\mathrm{P}_{eq} = \frac{\sum_{k\in K} \mathrm{P}_k \mathrm{N}_k}{\sum_{k\in K} \mathrm{N}_k}, \tag{4}$$

where K is the set of kernels ($K = \{\texttt{SpMV}, \texttt{axpy}, \texttt{axty}, \texttt{dot}\}$), N_k and P_k corresponds with the number of operations and the performance of each kernel, respectively, while α_k, FLOPS_k, and BYTES_k represents the operations ratio, the number of floating-point operations, and the number of memory transfers for each kernel, respectively.

Further, $\mathrm{AI}_{\texttt{SpMV}}$ is computed by the expression proposed by [5]:

$$\mathrm{AI}_{\texttt{SpMV}} = \frac{2nnz(\mathsf{A}) + 1}{8nnz(\mathsf{A}) + 4nnz(\mathsf{A}) + 4(n+1) + 8n + 8m + 8}, \tag{5}$$

where $nnz(\mathsf{A})$, n and m correspond with the number of non-zeros, 7 in the current implementation, and the number of rows and columns of matrix A, respectively.

Figure 3 presents the roofline model analysis for the TFA+HPC^2 equivalent performance. Showing that the performance lies in the memory-bound region with an AI_{eq} of

≈ 0.125 and a marked gap between the supercomputer peak performance (P_{peak}) and the P_{eq} of the implementation. On the other hand, the equivalent performance shows how efficiently TFA+HPC^2 uses the resources for its arithmetic intensity as the performance point is closely located to the memory line bound.

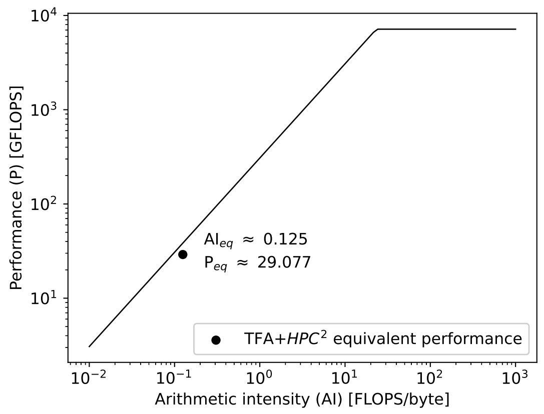

Figure 3: Roofline model analysis of TFA+HPC^2 on Marenostrum 5 GPP supercomputer; using 1 node (112 CPU-cores) and $350 \times 480 \times 350$ - 58.8M CVs - grid

This work will be expanded by executing large scalability tests on GPU-accelerated nodes (under the OpenCL framework). In addition, large-scale urban simulations will measure performance and simulation time, exploiting domain symmetries to improve the solver's performance by replacing `SpMV` operations with `SpMM`. Moreover, GPU-accelerated nodes test will be conducted on the Snellius supercomputer at SURF on nodes equipped with two Intel Xeon Platinium 8360Y (36 cores, 2.4 GHz, 54 MB L3 cache, and 204.8 GB/s memory bandwidth) with 512 GB of RAM, and interconnected through two ConnectX-6 HDR100 cards.

Acknowledgements

This work is supported by the Ministerio de Economía y Competitividad, Spain, SIMEX project (PID2022-142174OB-I00). M.MO. is supported by the Catalan Agency for Management of University and Research Grants (2024 FI-1 00684). In addition, J.PR is also supported by the Catalan Agency for Management of University and Research Grants (2022 FI_B 00810). Calculations were performed on the MareNostrum 5 GPP supercomputer at the BSC. The authors thankfully acknowledge these institutions.

References

[1] F. X. Trias, O. Lehmkuhl, A. Oliva, C.D. Pérez-Segarra, and R.W.C.P. Verstappen. "Symmetry-preserving discretization of Navier-Stokes equations on collocated unstructured meshes". In: *Journal of Computational Physics* 258 (2014), pp. 246–267.

[2] E. Komen, J. A. Hopman, E. M. A. Frederix, F. X. Trias, and R. W. C. P. Verstappen. “A symmetry-preserving second-order time-accurate PISO-based method”. In: *Computers & Fluids* 225 (2021), p. 104979.

[3] X. Álvarez-Farré, A. Gorobets, F. X. Trias, R. Borrell, and G. Oyarzun. “HPC^2 – A fully portable algebra-dominant framework for heterogeneous computing. Application to CFD”. In: *Computers & Fluids* 173 (2018), pp. 285–292.

[4] À. Alsalti-Baldellou, G. Colomer, J. A. Hopman, X. Álvarez-Farré, A. Gorobets, F. X. Trias, C. D. Pérez-Segarra, and A. Oliva. “Reliable overnight industrial LES: challenges and limitations. Application to CSP technologies”. In: *14th International ERCOFTAC Symposium on Engineering, Turbulence, Modelling and Measurements: 6th-8th September 2023, Barcelona, Spain: proceedings*. European Research Community on Flow, Turbulence, and Combustion (ERCOFTAC). 2023.

[5] À. Alsalti-Baldellou, X. Álvarez-Farré, F. X. Trias, and A. Oliva. “Exploiting spatial symmetries for solving Poisson’s equation”. In: *Journal of Computational Physics* 486 (2023), p. 112133.