A PORTABLE ALGEBRAIC IMPLEMENTATION FOR RELIABLE INDUSTRIAL LES

***M. Mosqueda-Otero*[1], *À. Alsalti-Baldellou*[1,2], *J. Plana-Riu*[1], *X. Álvarez-Farré*[3], *G. Colomer*[1], *F. X. Trias*[1], *A. Gorobets*[4] *and A. Oliva*[1]**

[1] ***Universitat Politècnica de Catalunya - BarcelonaTech, CTTC, Carrer de Colom 11, 08222 Terrassa, Spain***

[2] ***University of Padova, Department of Information Engineering, Via Giovanni Gradenigo, 6b, 35131 Padova PD, Italy***

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

[4] ***Keldysh Institute of Applied Mathematics, 4A, Miusskaya Sq., Moscow 125047, Russia***

marcial.francisco.mosqueda@upc.edu

Abstract

This work provides a performance evaluation of the feasibility of large-scale simulations for industrial applications using a symmetry-preserving discretization method for unstructured collocated grids in LES of turbulent flows. A method that ensures stability without artificial dissipation and maintains portability through minimal algebraic kernels. The present analysis evaluates its performance accross MPI-only, MPI+OpenMP, and GPU architectures using OpenCL. Demonstrating the method's effectiveness in enhancing parallel execution while supporting large-scale simulations.

1 Introduction

The continuous development of novel numerical methods, coupled with the rapid evolution of high-performance computing (HPC) systems, has significantly expanded the role of computational fluid dynamics (CFD) in various industrial applications. Despite these advancements, the development of CFD faces persistent challenges. Early implementations were hindered by the compute-bound limitations of processors, which led to the adoption of compute-centric programming models. Over time, processor designs have evolved, addressing these limitations and resulting in a mismatch between computational power and memory bandwidth. This, in turn, forces the creation of complex memory hierarchies, complicating the optimization of traditional programs. At the same time, the widespread use of accelerators in diverse technological fields has driven the rise of hybrid architectures, offering greater computational throughput while improving power efficiency in large-scale applications (Witherden et al. 2014). However, this shift introduces a new challenge: ensuring the portability of legacy applications. This challenge requires versatile software architectures and the development of specialized APIs, such as CUDA, OpenCL, and HIP (Edwards et al. 2014; Zhang et al. 2013).

In this context, the conservative discretization method for unstructured grids, as proposed by Trias et al. (2014), has been adopted and implemented under TermoFluids Algebraic (TFA)—our in-house code based on an innovative algebra-dominant framework, HPC^2(See Álvarez-Farré et al. 2018). This robust framework facilitates seamless integration into open-source codes and hybrid supercomputing environments, as shown by Komen et al. (2021).

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, shorter product development cycles, and expanding CFD's industrial application fields. Thus, our research seeks to ensure the integration of modern CFD methodologies into industry practices, enabling precise and accurate simulations of complex processes while efficiently utilizing available resources and reducing simulation costs within limited timeframes.

2 Portability for CFD

The construction of codes based on a minimal set of algebraic kernels has become essential for ensuring portability, optimization, and ease of maintenance, particularly in light of the growing diversity of computational architectures and hardware vendors. The hybrid nature of modern high-performance computing systems presents additional challenges, as effective utilization of both processors and parallel accelerators often requires heterogeneous computations and complex data exchanges. Traditional CFD codes, however, rely on intricate data structures and specialized computational routines, which complicates portability. To address this, algorithms centered on algebraic kernels, such as the sparse matrix-vector product (`SpMV`), linear combination of vectors (`axpy`), element-wise product of vectors (`axty`), and the `dot` product, emerge as promising solutions (See Álvarez-Farré et al. 2018).

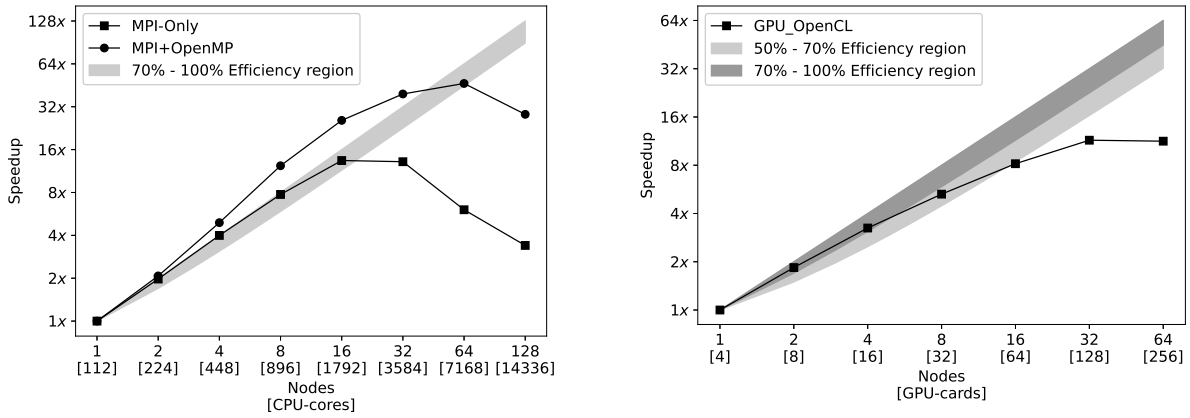

Figure 1: Strong scalability analysis; CPU-based system with $350 \times 480 \times 350$ - 58.8M CVs - grid comparing MPI-only vs. MPI+OpenMP paradigms (left plot) and GPU-based system with a $370 \times 400 \times 370$ - 54.76M CVs - grid (right plot)

However, this approach introduces a computational challenge linked with the low arithmetic intensity of `SpMV` operations. This limitation can be mitigated using the more computationally intensive sparse matrix-matrix product (`SpMM`). Substituting `SpMV` with `SpMM` significantly reduces memory access demands and the memory footprint by reusing matrix coefficients (See Alsalti-Baldellou et al. 2023b).

Beyond portability, algebra-based CFD implementations offer distinct numerical benefits. For example, they facilitate the development of efficient CFL-like conditions, reducing computational cost by up to 4x compared to classical approaches (See Trias et al. 2023). Additionally, algebraic frameworks allow for the streamlined incorporation of advanced techniques such as flux limiters, yielding compact and efficient implementations by utilizing incidence matrices and local operations to control gradient ratios, thus reducing the number of computing kernels required for porting (See Valle et al. 2022).

Building on these principles, Alsalti-Baldellou et al. (2023a) proposed an effective approach to accelerate Poisson solvers by exploiting domain symmetries. By carefully ordering the unknowns, `SpMV` operations could be replaced with `SpMM`, leading to a 2.5x increase in the performance of compute-intensive kernels while significantly reducing the solver's memory footprint and setup costs.

3 Algorithm scalability analysis

A numerical test evaluates the performance and scalability of TFA's base algorithm under different parallel computing paradigms to ensure its capability to address large-scale problems. Specifically, we compare the performance of an MPI-only configuration—where each CPU core is assigned a single task—against a hybrid MPI+OpenMP configuration, which utilizes MPI processes and multi-threaded executions per node. Furthermore, the scalability of the code on hybrid HPC systems is analyzed, taking advantage of TFA's underlying structure, which is based on minimal algebraic kernels. This architecture enables broad portability across diverse GPU hardware using OpenCL.

The numerical test case solves a turbulent channel flow using a conjugate gradient solver with a Jacobi preconditioner for Poisson's equation, combined with an explicit time integration scheme and a variable time step. Since the primary objective is to assess the scalability of TFA+HPC^2 kernels, each case is limited to 10 time steps, with 800 solver iterations per step. The test problem is solved over the entire domain without exploiting symmetries, thereby isolating the key algebraic kernels —`SpMV`, `axpy`, `axty`, and `dot` product— for performance measurement and analysis.

CPU-based system tests were conducted using MPI-only and MPI+OpenMP configurations on the MareNostrum 5 GPP supercomputer at BSC. The experiments run on nodes equipped with two Intel Xeon Platinum 8480+ processors (56 cores, 2 GHz, 105 MB L3 cache, and 307.2 GB/s memory bandwidth) with 256 GB of RAM, interconnected via ConnectX-7 NDR200 InfiniBand. Additionally, GPU-accelerated tests were conducted on the MareNostrum 5 ACC supercomputer at BSC, using nodes with two Intel Xeon Platinum 8460Y processors (40 cores, 2.63 GHz, 105 MB L3 cache, and 307.2 GB/s memory bandwidth), 512 GB of RAM, four NVIDIA Hopper H100 with 64GB HBM2, and interconnected through four ConnectX-7 NDR200 InfiniBand cards.

Figure 1 illustrates the strong scalability results for both HPC architectures: (i) CPU-based system (left plot) applying a 1-node baseline with a $350 \times 480 \times 350$ grid and (ii) GPU-based system (right plot) using a 1-node baseline with a $370 \times 400 \times 370$ grid. The results show a marked super-linear speedup for hybrid MPI+OpenMP processes, which can be attributed to enhanced cache utilization; moreover, the MPI-only solution exhibits significant communication over-

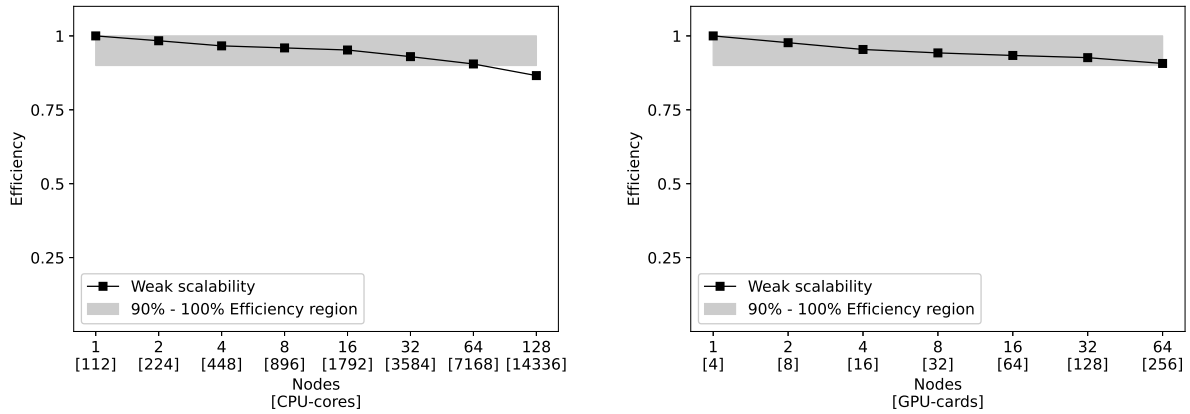

Figure 2: Weak scalability analysis; MPI+OpenMP paradigm with 525k CVs per CPU-core, up to 128 nodes (left plot) and GPU-based implementation with 13.69M CVs per GPU card, up to 64 nodes (right plot)

head, particularly when using 16 or more computational nodes. In addition, the strong scalability for TFA+HPC^2 on a hybrid architecture presents a steady speed-up as the problem scales. However, despite the memory-bound nature of CFD applications, the efficiency remains between 50% and 70% up to 16 nodes. For a 32-node implementation, the efficiency drops to approximately 40%, indicating the increasing impact of communication overhead in hybrid architectures.

On regards to the weak scalability analysis[1], Figure 2 presents both CPU-based (focus on MPI+OpenMP results) and GPU-based tests. The hybrid parallel paradigm (left plot) shows a 13% drop in performance when scaling up to 128 nodes and a 9% drop when scaling up to 64 nodes, referenced to the 1-node baseline. Similarly, the GPU-accelerated weak scalability (right plot) displays a 9% drop in performance when scaling up to 64 nodes, referenced to the 1-node baseline.

4 Performance analysis

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

$$\mathrm{AI}_{eq} = \frac{\sum_{k\in K} \alpha_k \mathrm{FLOPS}_k}{\sum_{k\in K} \alpha_k \mathrm{BYTES}_k}, \tag{1}$$

$$\mathrm{P}_{eq} = \frac{\sum_{k\in K} \mathrm{P}_k \mathrm{N}_k}{\sum_{k\in K} \mathrm{N}_k}, \tag{2}$$

and

$$\mathrm{DT}_{eq} = \frac{\sum_{k\in K} \mathrm{DT}_k \mathrm{N}_k}{\sum_{k\in K} \mathrm{N}_k}, \tag{3}$$

[1]The main objective of this study is to assess the scalability of TFA+HPC^2 kernels. Thus, the number of iterations per time step was fixed to ensure consistent kernel calls as the problem scales.

where K is the set of kernels ($K = \{\texttt{SpMV}, \texttt{axpy}, \texttt{axty}, \texttt{dot}\}$), N_k, P_k and DT_k corresponds with the number of operations, the performance, and data throuhgput 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 Alsalti-Baldellou et al. 2023a:

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

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 illustrates the roofline model and data throughput analysis of TFA+HPC^2 performance on two distinct HPC architectures. On one hand, the simulation features an equivalent arithmetic intensity of approximately 0.125, with a noticeable gap between the theoretical peak performance (P_{peak}) of the supercomputers and the achieved performance, highlithing the memory-bound behaviour of CFD simulations. On the other hand, results show that TFA+HPC^2 efficiently utilizes the available resources for its given arithmetic intensity, as performance points for both architectures lands near the memory bandwidth limit, showing a data throughput of the 75% of the available memory bandwidth. The latter results lies close to memory benchmarks like STREAM (McCalpin 1995) which depicts values between 80-90% for different architectures (Deakin et al. 2015; Špet'ko et al. 2021). Finally, the GPU-based implementation (figure 3 lower row) exhibits higher efficiency than the MPI+OpenMP configuration (figure 3 upper row).

Furthermore, while both architectures are constrained by memory bandwidth, the GPU-accelerated

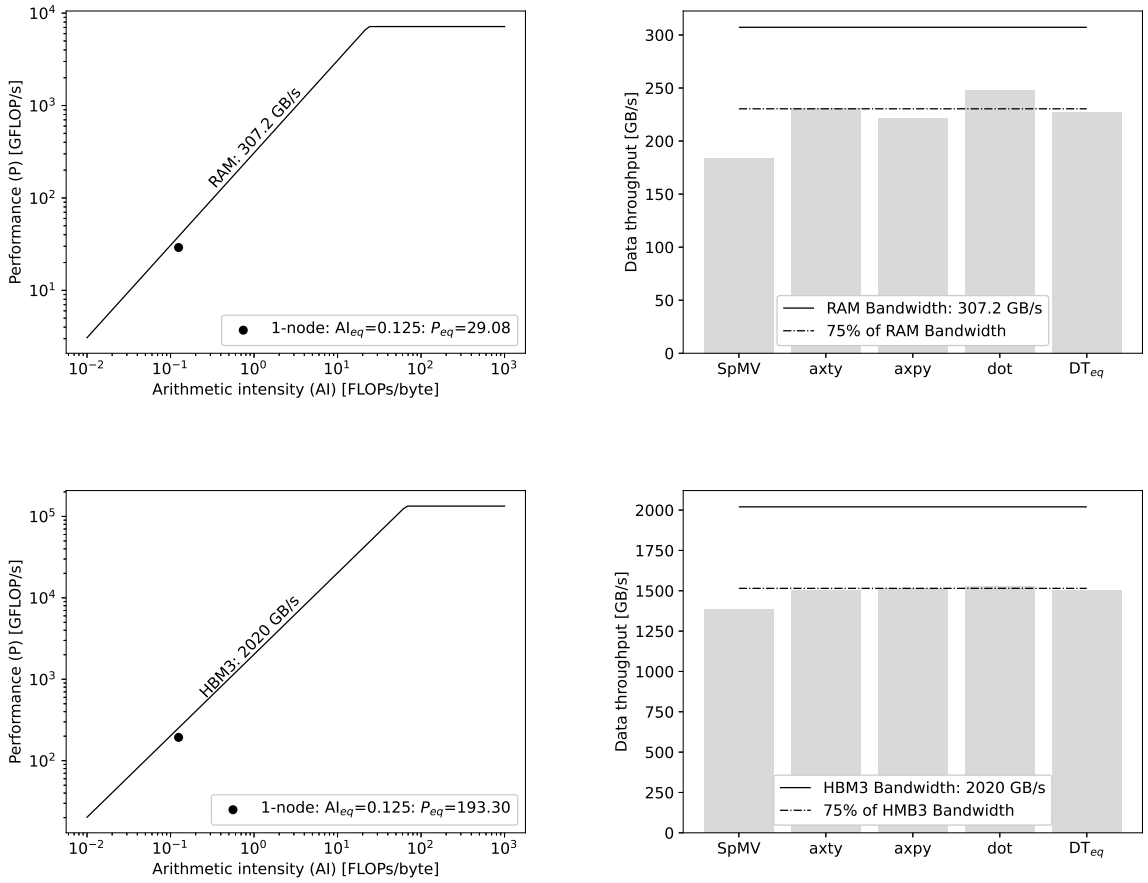

Figure 3: Roofline and data throughput models; MPI+OpenMP paradigm on 1 node (112 CPU-cores) with a $350 \times 480 \times 350$ - 58.8M CVs - grid solution (upper row) and GPU-based implementation on 1 node (4 GPU cards) with a $370 \times 400 \times 370$ - 54.76M CVs - grid solution (lower row)

system consistently outperforms the MPI+OpenMP solution at lower arithmetic intensities, benefiting from higher computational throughput and more optimized parallel execution routines. This highlights the advantage of GPU architectures in achieving better performance despite the inherent memory-bound limitations.

5 Closing remarks

The TFA+HPC^2 framework demonstrates strong portability across various HPC architectures, leveraging its minimal algebraic kernel design to ensure broad compatibility and efficiency. The MPI+OpenMP hybrid paradigm shows superior strong scalability over MPI-only, benefiting from better cache utilization and reduced communication overhead, making it ideal for large-scale CPU-based simulations. While the GPU-accelerated implementation is promising, it faces performance limitations due to inter-node communication overhead, indicating the need for increased computational load per GPU. Nonetheless, the current implementation is highly optimized, with performance primarily constrained by memory bandwidth.

Future work will focus on increasing the arithmetic intensity of TFA+HPC^2 to overcome its memory-bound limitations, e.g., by exploiting domain symmetries in large-scale urban simulations; replacing `SpMV` operations with `SpMM` to enhance solver performance. The weak scaling analysis shows excellent efficiency, confirming the framework's robustness and scalability for demanding industrial applications.

Acknowledgments

M.MO, A.AB, J.PR, X.AF, G.C, F.X.T and A.O are supported by the Ministerio de Economía y Competitividad, Spain, SIMEX project (PID2022-142174OB-I00). In addition, M.MO. is supported by the Catalan Agency for Management of University and Research Grants (2024 FI-1 00684) and 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 and ACC supercomputers. The authors thankfully acknowledge these institutions.

References

Alsalti-Baldellou, À., X. Álvarez-Farré, F. X. Trias, and A. Oliva (2023a). "Exploiting spatial symmetries for solving Poisson's equation". In: *Journal of Computational Physics* 486, p. 112133.

Alsalti-Baldellou, À., G. Colomer, J. A. Hopman, X. Álvarez-Farré, A. Gorobets, F. X. Trias, C. D. Pérez-Segarra, and A. Oliva (2023b). "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*.

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

Deakin, Tom and Simon McIntosh-Smith (2015). "GPU-stream: Benchmarking the achievable memory bandwidth of graphics processing units". In: *IEEE/ACM SuperComputing*, pp. 3202–3216.

Edwards, H. C., C. R. Trott, and D. Sunderland (2014). "Kokkos: Enabling manycore performance portability through polymorphic memory access patterns". In: *J. Parallel Distrib. Comput.* 74.12, pp. 3202–3216.

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

McCalpin, John D. (Dec. 1995). "Memory Bandwidth and Machine Balance in Current High Performance Computers". In: *IEEE Computer Society Technical Committee on Computer Architecture (TCCA) Newsletter*, pp. 19–25.

Špet'ko, Matej, Ondřej Vysockỳ, Branislav Jansík, and Lubomír Říha (2021). "Dgx-a100 face to face dgx-2—performance, power and thermal behavior evaluation". In: *Energies* 14.2, p. 376.

Trias, F. X., X. Álvarez-Farré, À. Alsalti-Baldellou, A. Gorobets, and A. Oliva (2023). "An Efficient Eigenvalue Bounding Method: Cfl Condition Revisited".

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

Valle, N., X. Álvarez-Farré, A. Gorobets, J. Castro, A. Oliva, and F. X. Trias (2022). "On the implementation of flux limiters in algebraic frameworks". In: *Computer Physics Communications* 271, p. 108230. ISSN: 0010-4655.

Witherden, F. D., A. M. Farrington, and P. E. Vincent (2014). "PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach". In: *Comput. Phys. Commun.* 185.11, pp. 3028–3040.

Zhang, Y., M. Sinclair, and A. A. Chien (2013). "Improving Performance Portability in OpenCL Programs". In: *Supercomputing*, pp. 136–150. ISBN: 978-3-642-38750-0.