X International Conference on Computational Methods for Coupled Problems in Science and Engineering
COUPLED PROBLEMS 2023
M. Papadrakakis, B. Schrefler and E. Oñate (Eds)

ACCELERATING THE FLOWSIMULATOR: IMPROVEMENTS IN FSI SIMULATIONS FOR THE HPC EXPLOITATION AT INDUSTRIAL LEVEL

Marco Cristofaro*†, Jonathan Alexander Fenske†, Immo Huismann†, Arne Rempke† and Lars Reimer‡

† Institute of Software Methods for Product Virtualization
German Aerospace Center (DLR), Zwickauer Straße 46, 01069 Dresden, Germany
e-mail: Marco.Cristofaro@dlr.de, web page: https://www.dlr.de/sp

‡ Institute of Aerodynamics and Flow Technology
German Aerospace Center (DLR), Lilienthalplatz 7, 38108 Brunswick, Germany

Key words: CFD, CSM, High–Performance Computing, Mesh Deformation, Mesh Partitioning

Abstract. High–performance computing systems enable the use of computationally intensive models in aircraft design simulations, but challenges arise with communication times and memory consumption as the number of cores scales up. This paper presents improvements to the mesh partitioner and mesh deformation methods for highly parallelized simulations within the *FlowSimulator* framework. The results are obtained for a simple wing test case, a full–aircraft configuration and an aircraft research model. Improvements to the mesh partitioner are presented that allow for even larger and more parallelized simulations than was previously possible in the framework. In addition, scaling results from a mesh deformation method based on the elastic analogy are presented, which shows superior scalability compared to the conventional radial basis function method. This improves the overall suitability of the toolchain for large parallelization and highlights its potential for industrial applications. The results demonstrate a step forward towards an optimal exploitation of high performance computing systems for solving coupled simulations in the aerospace industry.

1 INTRODUCTION

The European Green Deal aims to achieve climate neutrality by 2050, with an interim target of reducing greenhouse gas emissions by at least 55% by 2030. In 2017, aviation accounted for 3.8% of total EU CO_2 emissions and 13.9% of transport emissions, second only to road transport[1]. Despite these seemingly small percentages, the increasing number of flights, largely driven by developing countries, emphasizes the importance of innovations in cleaner aircraft to contribute significantly to reducing aviation pollution.

The new aircraft need to be designed and thereafter certified. The aircraft design process of today relies heavily on simulation tools to reduce the need for expensive and time–consuming

[1] `https://climate.ec.europa.eu/eu-action/transport-emissions/reducing-emissions-aviation_en` [retrieved July 14, 2023]

prototyping. The availability of accurate and fast simulation results can drastically improve the time–to–market and the quality of final designs. For this reason, industrial–grade simulation tools, together with a robust computing infrastructure, should provide high–fidelity solutions with acceptable run times. Continuous advances in hardware and software enable increasingly accurate results, with the same time–to–solution. These advances include improvements in computing power, memory bandwidth, software scalability, and node–level performance, as well as evolution of existing algorithms and development of entirely new ones.

This paper presents recent method improvements aimed at increasing the scalability of standard pipelines for aircraft simulation, using the *FlowSimulator* environment maintained by DLR (*Deutsches Zentrum für Luft– und Raumfahrt e.V.*). In particular, the new developments in the repartitioning steps allow to obtain a number of partitions larger than what was previously possible. These developments are demonstrated using a CFD mesh of an aircraft research model provided by NASA. Furthermore, the simulation of fluid–structure interaction exposes new challenges when a high degree of parallelization is applied, in particular in regards to the CFD mesh deformation process. The steady aeroelastic fluid–structure interaction simulation toolchain is applied on a simple wing and on a full aircraft configuration. The scaling results of the mesh deformation method based on the elastic analogy are then compared to the conventional radial basis function method. The results about the LANN wing test case show how the elastic analogy method is faster for extreme parallelism ($\approx$ 300 nodes/MPI–rank and below) due to the superior scalability characteristics. When applied to the larger XRF1 mesh, although the elastic analogy method is much slower compared to the conventional radial basis function method, substantial improvements come from algebraic multi–grid methods and from the use of hybrid parallelization techniques (i.e. MPI + OpenMP). These reduce the gap in the computational cost, while keeping the method robustness even for complex cases.

2 THE FLOWSIMULATOR

Since 2005, Airbus and DLR have been working together to develop a unified environment for parallel flow simulations. This is called the *FlowSimulator* and it aims to improve the interoperability between different simulation methods with a common simulation layout supported by a shared data–exchange layer.

The *FlowSimulator DataManager* (*FSDM*) serves as a library within the *FlowSimulator* environment to provide storage and access to mesh–based data for all simulation blocks, while also offering a range of useful functionalities [1]. This allows different simulation tools to exchange information in both fluid and structural domains, as well as the interpolation between different regions. Each instance of *FSDM* can store multiple meshes and each mesh can contain multiple datasets based on node, face or cell elements. For coupled Fluid–Structure Interaction (FSI) simulations, two separate meshes can be stored in a single *FSDM* instance, along with interpolation matrices for the interface. In addition, *FSDM* offers a wide range of utilities that can be used in any part of a simulation toolchain to simplify and extend the capabilities of each simulation block. Among others, common tasks such as mesh import, export, and partitioning (for structured, unstructured and hybrid meshes), geometry and boundary conditions handling and dimensionalization, are available with an increasing variety of options and continuous support. While the underlying functionalities are implemented in C++ for fast processing of large data (e.g. mesh data), a Python wrapper using SWIG is adopted for all higher–level operations and the surrounding plugins in order to speed–up the users learning

curve. The combination of a Python wrapper and the C++ sub–layer, allows easy access for the implementation of simulation control scripts, without affecting the performance of critical processes. Among them, the partitioning of mesh data structures is at the core of the parallelization concept for distributed memory systems: runtime is drastically reduced by dividing the whole simulation domain among MPI–ranks and allowing them to work as independently as possible on a reduced portion of the domain. This is important on any hardware, but it becomes critical when exploiting High Performance Computing (HPC) systems.

Several high–fidelity flow solvers are directly integrated in the *FlowSimulator* (e.g. CODA [2], DLR TAU [3], and TRACE) and plugins are available for accessing proprietary structural codes (e.g. Nastran). A recent alternative for structural simulation is the direct access of DLR's B2000++ [4] to the *FlowSimulator*. The direct access to the various simulation data structures avoids the cost of writing and reading input files from disk for each coupling step, by relying on in–memory data transfer.

Furthermore, for fluid–structure interaction simulations, a mesh deformation tool is required to adapt the computational grid to the new deformed solid state. Within the *FlowSimulator*, two methods are available: one based on radial–basis functions and one that implements the elastic analogy method [5] using PETSc [6] and Spliss [7] as underlying linear solver libraries. Although the radial–basis functions method is the most widely employed, elastic analogy methods may be advantageous when using HPC systems due to the availability of linear solvers that are already optimized to run highly parallelized.

3 PARTITIONING

Efficient execution of large–scale simulations relies on workload balancing across processes. Although *FSDM* distributes mesh data during import for faster processing, it may not achieve optimal load balancing. Consequently, repartitioning the mesh data becomes necessary to ensure optimal workload distribution. *FSDM* offers built–in methods such as Recursive Coordinate Bisection (RCB) [8], a direct interface to the library ParMETIS [9], as well as a plugin providing an interface to the library Zoltan [10], for mesh repartitioning. The standard approach for repartitioning the mesh within *FSDM* is composed by three steps: prepartitioner, graph extractor and graph partitioner.

To enhance the efficiency of the graph extraction process and mitigate potential memory and communication overhead, prepartitioning the mesh is essential. In this regard, *FSDM* employs the RCB method, which rapidly partitions the grid by recursively bisecting it based on vertex coordinates. To address excessive memory usage with a large number of processes, the RCB method has been improved by leveraging the `MPI_Allreduce` routine to simultaneously collect and distribute cut data, ensuring efficient memory utilization. In the following step, a mesh graph is extracted that accurately represents the grid's connectivity. *FSDM* incorporates a built–in functionality for mesh graph extraction, which makes the graph accessible to the partitioning library. In this study, the graph extraction process has been enhanced by storing data from only one other process at a time and promptly deleting the data after checking for neighborhood relationships. This significant reduction in memory requirements during graph extraction helps avoid the risk of encountering out–of–memory errors. Finally, given the generated mesh graph, the graph partitioning methods provided by ParMETIS or Zoltan libraries are accessed for generating the final partitioning.

4 THE STEADY AEROELASTIC TOOLCHAIN

The *FlowSimulator* applications range from steady–state flow simulations to complex unsteady simulations of maneuvering aircraft. The Python user interface allows complete simulation chains to be defined by automatically updating the simulation parameters that change during the toolchain. This work highlights the utilization of the *FlowSimulator* for fluid–structure interaction through the simulation of steady aeroelastic equilibrium of an aircraft. Figure 1 illustrates the iterative toolchain process involving two distinct domains: the surrounding flow simulated using Computational Fluid Dynamics (CFD), and the internal structure modeled with Computational Structural Mechanics (CSM). These domains are alternately simulated, exchanging load and displacement information.

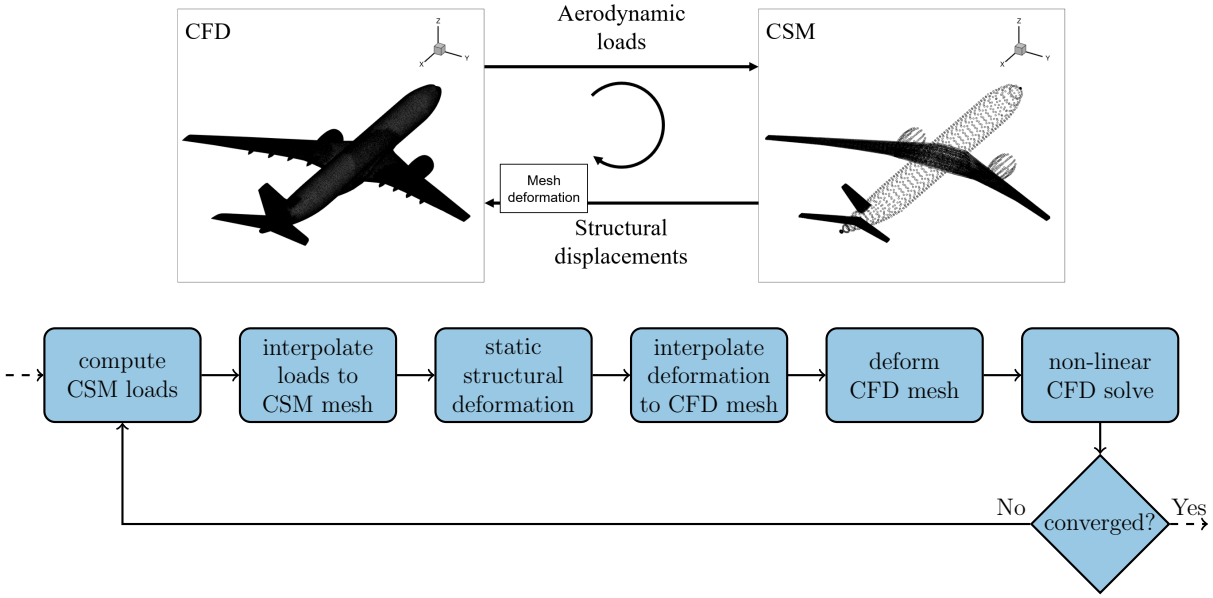

Figure 1: Iteration of the fluid–structure interaction simulation: graphical representation (above) and flow chart (below).

As preliminary steps, the CFD and CSM meshes are imported and repartitioned among the number of processes. Then, after an initial CFD solution is obtained, the pressure acting on the aircraft surface is interpolated to the structural nodes of the CSM model with the Finite Interpolation Elements (FIE) approach [11]. The displacements computed from the CSM simulation, are then interpolated back to the CFD mesh with the same FIE approach. The CFD volume mesh is then deformed according to the new structural state, employing the elastic analogy method [5]. Unlike its conventional use of repairing the bad quality cells (e.g. negative volumes) arising from the Radial Basis Functions (RBF) method [12], the elastic analogy method serves as the primary deformation method in this study, providing a more robust mesh deformation method and offering potential scalability on HPC systems. Following the mesh deformation, the flow is recalculated based on the modified mesh, using the previous solution as a initial condition. This process is repeated until the relative change in forces, moments, and displacements falls below a specified threshold. In order to stabilize the convergence rate, a under–relaxation factor of 0.7 is applied to the displacements before interpolation, while respecting the principle of preserving virtual work.

5 TEST CASES

5.1 The LANN wing

The LANN wing is a well–established research test case originally developed to analyze transonic flow regimes in wind tunnels. The wing has a semispan of 1 m, an area of 0.25 m^2, and a chord varying between 0.361 m at the root and 0.144 m at the tip. It also features a 25° sweep angle at 1/4–chord and a linear twist of 4.8° [13]. In this work, the Reynolds–Averaged Navier–Stokes (RANS) equations are used to model the turbulent flow around the wing, while the structure of the aircraft is modeled with linear elasticity.

The structural deformation is computed using a modal CSM solver. The modeshapes and eigenfrequencies are computed in an offline preprocessing step using Nastran from a cantilever wing model with $1,260$ nodes, and the lowest 13 elastic modes are considered during the aeroelastic coupling. An extra load of 2 kN in the direction of the lift force is applied on the wing tip during the FSI coupling to increase the overall wing bending. The flow simulation considers fixed upstream Mach number $M = 0.82$, angle of attack $\alpha = 0.6°$, and Reynolds number $Re = 5.43 \times 10^6$ based on a characteristic length of 0.36 m. The flow solution is computed with DLR TAU using the Spalart–Allmaras turbulence closure model (with negative formulation) [14]. The overall CFD mesh counts 1.85×10^6 cells, from which 1.05×10^6 are prisms in the boundary layer refinement zone and 0.80×10^6 are tetrahedra in the remaining external volume. A symmetry boundary condition is defined on the plane perpendicular to the wing span direction (xz–plane).

The CFD solver is set to run 300 outer iterations of the non–linear multigrid solver of DLR TAU for the initial CFD solution, while 200 iterations for the intermediate CFD simulations during the coupling steps and the final solution. The iterations number of the CFD solution also defines the switching frequency between simulation domains. This should be a trade–off choice between not computing unnecessarily accurate intermediate CFD solutions, while not spending too much computation time in the interpolation and mesh deformation procedures. In this case, 12 CFD–CSM coupling steps are necessary to reach a steady aeroelastic converged solution (i.e. a relative change of the forces, moments and displacements below the set residuum of 1×10^{-6}). Figure 2 presents the CFD and CSM meshes together with the aerodynamic pressure coefficient at the final steady aeroelastic equilibrium deformation state. The values are plotted on the wing surface and on the symmetry plane. The undeformed wing surface is also shown as reference.

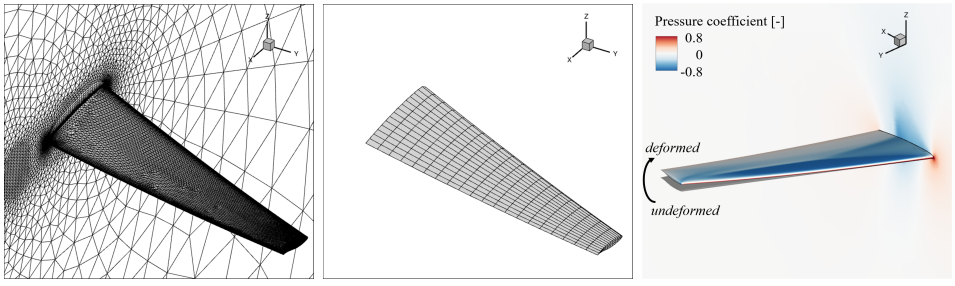

Figure 2: LANN wing case: CFD mesh (left), CSM mesh (center), and final pressure coefficient (right).

5.2 The XRF1 aircraft

A further case used in this work has been provided by Airbus as an industry relevant model and it represents a "standard" configuration for a single–deck long haul passenger aircraft. Industrially relevant geometries are essential to link the research topics with the real needs of industry and to enable the collaboration between differently founded–parties in the development and application of physical modeling techniques. The geometry used in this work is called the eXternal Research Forum (XRF1) [15]: it is a full aircraft configuration, including powered engines. A similar setup to the LANN wing case presented in the previous section is adopted solving the RANS equations for the flow, and a linear elasticity model for the structure.

For the XRF1 case, modeshapes and eigenfrequencies of the unrestrained (free–free) structural model made of $18,362$ nodes are computed and the lowest 30 elastic modes are considered during the aeroelastic coupling. The main flow parameters are the upstream Mach number $M = 0.85$, the angle of attack $\alpha = 2°$, and Reynolds number $Re = 60.7 \times 10^6$ based on the mean aerodynamic chord length of 9.096 m. From the 104.8×10^6 cells that make the CFD mesh, 38.6×10^6 are prisms in the boundary layer region and 65.4×10^6 are tetrahedra modeling the remaining volume. The wing chord is discretized with ≈ 100 elements and the first cell layer close to the wing as a characteristic height of 5×10^{-6} m, corresponding to a $y^+ < 2$ (approximated from the RANS solution as $y^+ = y/\ell_\tau \simeq y \cdot u_\tau/\nu$, where u_τ is the friction velocity, y is the distance from the wall, and ν is the kinematic viscosity of the fluid). Three different images with increasingly zoomed view of the CFD mesh are shown in Fig. 3.

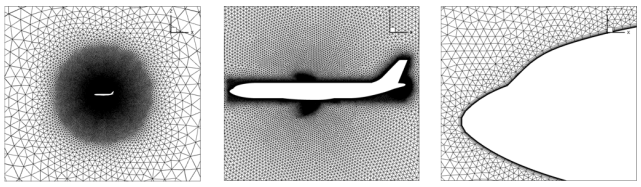

Figure 3: Longitudinal cuts of the CFD mesh of the XRF1 aircraft with increasing zoom level.

In this case $1,000$ outer iterations are set for the initial and final CFD solution, and 250 during the coupling steps. The aeroelastic equilibrium converged state is reached after 14 CFD–CSM coupling steps. Figure 4 shows the resulting equilibrium state: on the left the deformation in the normal axis (z–direction) and, on the right, the aerodynamic pressure coefficient on the external surface of the aircraft and on a longitudinal plane–cut.

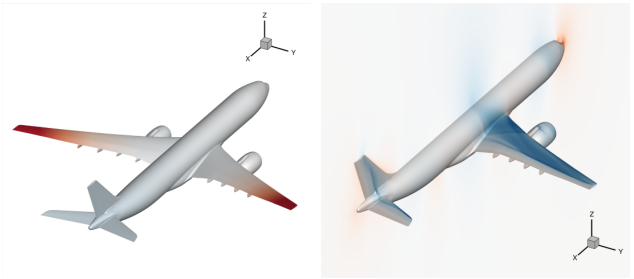

Figure 4: Final results of aircraft in static aeroelastic equilibrium: deformation in the normal direction (on the left) and pressure coefficient computed by CFD (on the right).

5.3 The CRM aircraft

The numerical grid used to present the improvements about the partitioning is based on the high–lift version of NASA's Common Research Model (CRM). This mesh poses a significant challenge during the repartitioning phase due to the large cell and node counts. In this work, it serves as a demonstration of the capabilities of the repartitioning method when dealing with very large number of processes. Geometrical details were provided at the 4th High Lift Prediction Workshop [16]. The considered mesh family is the *103–ANSA–Unstructured–hiA–Yplus1* for external aerodynamic analyses and the mesh files are provided by BETA CAE Systems SA. Among the available options, the largest mesh was selected, with about 723 million cells and 629 million nodes, as depicted in Fig. 5.

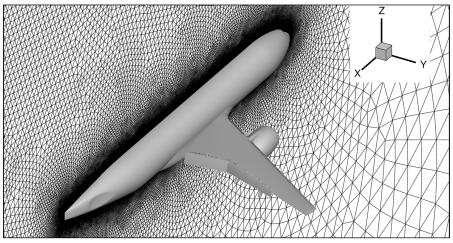

Figure 5: High–lift CRM mesh.

6 RESULTS

6.1 Computational setup

The performance measurements are conducted on DLR's high–performance computer CARO. Each computational node consists of two AMD EPYC 7702 CPU with 64 cores each (comprising 16 dies with shared L3 caches) and 256 GB DDR4 RAM. To prevent frequency scaling (e.g. boost mode or thermal throttling) from affecting the measurements, a fixed clock frequency of 1.8 GHz is imposed during all simulations. Time measurements of the different toolchain simulation blocks are obtained by summing the values over all coupling steps. This reduces the impact of initialization procedures time over the whole measurements, and it provides the actual simulation block contribution to the overall toolchain runtime.

6.2 High–lift CRM: partitioning

In previous studies, the partitioning of the mesh was identified as a bottleneck of the simulation chain [12]. Not only did the runtime of the partitioning increase with the number of processes, the memory requirements also became prohibitive. Depending on the mesh, using the RCB method was not possible for more than $2,000$ processes, precluding the usage of a capable prepartitioner and further raising the overall partitioning runtime.

To study the gains from the improvements in *FSDM* partitioning, the CRM mesh from Figure 5 was repartitioned with *FSDM* – once with version 2022.07 and once with version 2023.03. As the partitioned mesh can be employed in simulation toolchains that can take advantage from hybrid parallelization (e.g. during mesh deformation, see Sec. 6.4, and with the flow solver CODA) and because the dies with shared L3 caches in CARO's architecture consist of four cores, four threads are used per process. This raises the amount of available

memory per process compared to previous studies by a factor of four [12]. Figure 6 depicts the runtimes of the partitioning process decomposed into three phases: the RCB prepartitioning, the extraction of the graph, and lastly the graph partitioning with ParMETIS using the default `ParMETIS_V3_PartGeomKway` algorithm. A missing datapoint in the plot stands for a failing run – in most cases due to memory requirements.

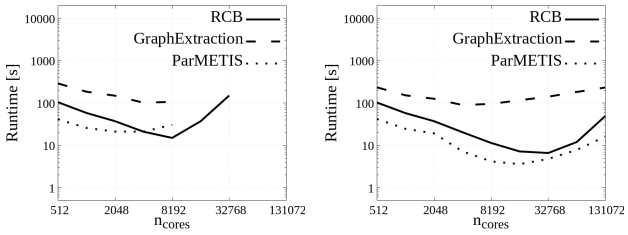

Figure 6: Partitioning runtime comparison between the old (on the left) and new (on the right) *FlowSimulator* versions when scaling from 512 cores to 131,072 cores. Missing data points stand for failing runs.

The old version of FSDM was capable of running the RCB method up to around 32,768 cores, and the graph extraction up to around 8,192. The improvements presented in Section 3 lower the memory requirements from an accidental $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$. This allows to pass the previous limit from $\approx 10,000$ cores and perform partitioning up to $\approx 130,000$ – it is to be noted that this limit is only of currently available hardware, CARO was simply not large enough to allow larger runs. Furthermore, the overall partitioning time was reduced due to changes in the RCB implementation, as seen in the data for 8,192 and 16,384 cores. However, it is important to acknowledge that limitations persist: here, a setup with four threads per process was shown. With pure MPI, however, the mesh partitioning still encounters out–of–memory errors for a number of processes near 30,000 cores.

6.3 LANN wing: steady aeroelastic toolchain

The scaling performances of the LANN wing steady aeroelastic simulation toolchain are presented in this section. Figure 7 shows the contribution of the main simulation blocks (i.e. CFD, CSM and mesh deformation) to the overall toolchain runtime and cost (in CPU hours) of the LANN wing test case using the RBF mesh deformation method. The simulations are executed MPI parallel only. As expected, CFD takes the longest time in all cases, but, for large number of cores, the relative contribution of the RBF mesh deformation method becomes relevant.

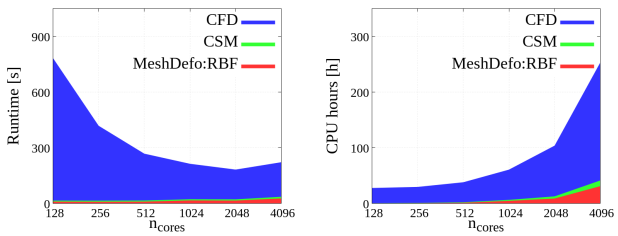

Figure 7: Steady aeroelastic toolchain scalability with RBF mesh deformation method in the LANN wing case: runtime (on the left) and CPU hours (on the right).

The corresponding analyses using the Elastic Analogy (EA) mesh deformation is also presented in Fig. 8. The linear problem of the elastic analogy method is solved with the PETSc library using the BiConjugate Gradient Stabilized (BiCGStab) as iterative method with Successive OverRelaxation (SOR) as preconditioner [6]. When applying the EA mesh deformation method, the time spent in the deformation process is significantly larger than RBF with low number of cores, but it scales well with increasing parallelization: at $4,096$ cores, corresponding to ≈ 300 nodes/MPI-domain from the CFD mesh, less than 3% of the total runtime is spent in the mesh deformation process.

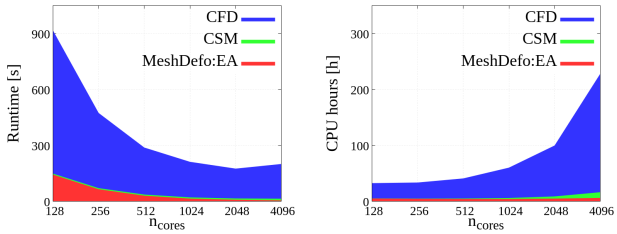

Figure 8: Steady aeroelastic toolchain scalability with EA mesh deformation method in the LANN wing case: runtime (on the left) and CPU hours (on the right).

Figure 9 compares the two mesh deformation methods alone in terms of runtime, CPU hours, speed–up, and parallel efficiency. From these results, it can be seen that the RBF method requires ≈ 15 s regardless of the number of cores, while the EA method scales almost perfectly, providing a uniform cost of ≈ 5 CPU hours, near–ideal speedup, and parallel efficiency consistently above 80% and peaking at 121%. Superlinear speed–up (i.e. efficiencies above 100%) can appear due to cache effects (less memory per process is required for increased numbers of cores) or a different load balance obtained from the partitioner, while still not being affected by large communication overhead. Although the provided case shows a time benefit of using EA only above $1,024$ cores, this can be expected to out–pace the RBF method for larger parallelization.

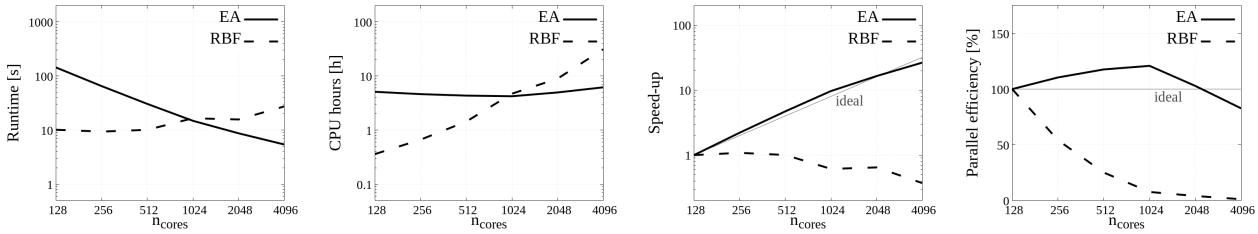

Figure 9: Scaling performance of the elastic analogy mesh deformation method compared to RBF in the LANN wing case. From left to right: runtime, CPU hours, speed–up, and parallel efficiency.

6.4 XRF1: mesh deformation

This section investigates the scaling performances of the mesh deformation part of the XRF1 steady aerolastic simulation toolchain. For large and complex cases, the newly developed algebraic multigrid linear solver implemented in the Spliss library provides an ideal solution. It simultaneously reduces the computational cost of solving sparse linear systems, improves the overall solution procedure's robustness, and utilizes hybrid parallelization. Since recent efforts in enhancing CFD solver performances in HPC have concentrated on taking advantage from hybrid parallelization techniques (e.g. CODA [17]), it is crucial to have mesh deformation methods that can also take advantage of such techniques. This becomes highly important in improving the overall performance of FSI toolchains. This section presents then the benefits of using MPI + OpenMP compared to pure MPI for the mesh deformation process. The results are presented with reference to the number of cores, corresponding to the number of MPI–ranks multiplied by the number of OpenMP threads.

Figure 10 compares the RBF method against the EA method with pure MPI and MPI + OpenMP parallelization. The EA linear system is solved using the iterative method BiCGStab preconditioned by an algebraic multigrid procedure, which applies Gauss–Seidel as smoother on each level. The usage of hybrid parallelization significantly improves the scaling characteristics for a large number of cores. However, in this case, the EA method consistently exhibits slower runtimes and higher computational cost than the RBF method. It should be noted that using the RBF method for mesh deformation around complex geometries can lead to problems such as the appearance of poor–quality cells, which may cause the CFD solver to diverge and make the simulation impossible to proceed [1]. In contrast, the EA method, which assumes the mesh volume and its constituent cells to behave as elastic solid bodies, rarely presents such problems once the linear system solution has converged. Therefore, the elastic analogy mesh deformation method may still be preferred for industrial applications due to its robustness. Furthermore, at $4,096$ cores, the hybrid parallelized EA solution scales with a parallel efficiency of 70%, while the RBF method achieves only 17%. It is worth mentioning that, as the RBF method does not implement multithreading (OpenMP parallelization), the runtimes are solely dependent on the number of MPI–ranks. For the sake of brevity, these results are not presented here. Similar to the results obtained for the LANN wing, these findings suggest that the EA method may deliver better performance when large degrees of parallelization are employed.

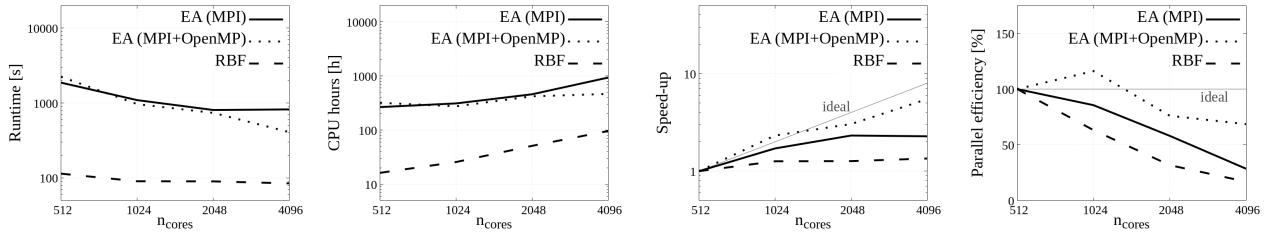

Figure 10: Scaling performances of the elastic analogy mesh deformation method with the algebraic multi–grid approach using pure MPI and MPI + OpenMP (4 threads) in comparison with RBF in the XRF1 case. From left to right: runtime, CPU hours, speed–up, and parallel efficiency.

Compared to a single–grid approach, the multigrid method offers a drastic reduction in the number of iterations required by the linear solver, resulting in decreased runtime and CPU

hours, while providing an equally converged solution to the linear problem. However, the multigrid method requires additional computational effort during the initial steps of the linear solver to generate the coarse grid levels. Figure 11 illustrates the runtime scaling performances of the EA mesh deformation method, divided into the preparation phase and the actual iterative solution phase ("Solving"), with pure MPI and MPI + OpenMP parallelization. The graphs demonstrate that the solving runtime scales with the number of cores for both parallelization techniques, with parallel efficiencies always above 60%, with a negligible effect from hybrid parallelization. In contrast, the preparation time significantly increases with large number of cores in the pure MPI case, whereas it scales better when 4 OpenMP threads are utilized, achieving a parallel efficiency of 86% at $4,096$ cores. The reason for this discrepancy lies in the fact that, in this case, the preparation time does not scale well for a number of MPI–ranks above $1,024$, thus the runtime significantly improves when these are reduced due to the OpenMP threads.

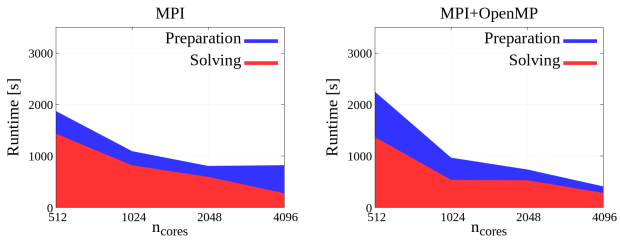

Figure 11: Preparation and linear solver runtime of the elastic analogy method in the XRF1 case. Pure MPI parallelization approach (on the left) and MPI + OpenMP with 4 threads (on the right).

The overall result is a significant improvement in the scaling performances of the EA mesh deformation method when MPI + OpenMP is employed. Consequently, hybrid parallelization extends the maximum number of cores for which the simulation runtime continues to decrease, thereby enhancing the utilization of HPC infrastructures.

7 CONCLUSIONS

The work presented here demonstrates recent developments in *FlowSimulator* toolchains aimed at improving method scalability in HPC systems. The recent improvements in the repartitioning of meshes make it possible to run simulations with above $100,000$ cores when hybrid parallelization, MPI and OpenMP, is used. This opens up the possibility to greatly reduce the runtime of numerical simulations due to a much greater degree of parallelization. The runtime of the repartitioning takes in the order of a few minutes, even for a large number of processes, which can be considered acceptable in comparison with typical simulation runtimes. Furthermore, the elastic analogy method for mesh deformation has been implemented in the steady aeroelastic toolchain, and the scaling performance has been analyzed for two test cases. Compared to the radial basis function method, this approach exhibits good scaling characteristics, making it a promising candidate for highly parallelized simulations. Additionally, since it relies strictly on the solution of a linear problem, it can take advantage of developments in linear solver libraries such as Spliss, with respect to algebraic multigrid, hybrid parallelization, as well as GPU exploitation (not presented here). In the near future, this method may represent a superior choice for mesh deformation in terms of both robustness and speed.

Acknowledgements The authors thank Airbus for supplying the industrial–relevant XRF1 testcase and BETA CAE Systems SA for providing the CRM mesh. This research was funded in parts via the DLR–internal projects COANDA and HighPoint.

REFERENCES

[1] L. Reimer, R. Heinrich, S. Geisbauer, T. Leicht, S. Görtz, M. R. Ritter, and A. Krumbein, "Virtual aircraft technology integration platform: Ingredients for multidisciplinary simulation and virtual flight testing," in *AIAA SciTech Forum*, Januar 2021.

[2] T. Leicht, J. Jägersküpper, D. Vollmer, A. Schwöppe, R. Hartmann, J. Fiedler, and T. Schlauch, "DLR-project Digital-X – next generation CFD solver 'Flucs'," in *Deutscher Luft- und Raumfahrtkongress 2016*, 2016.

[3] L. Reimer, "The FlowSimulator – a software framework for CFD-related multidisciplinary simulations," in *NAFEMS European Conference: Computational Fluid Dynamics (CFD) – Beyond the Solve*, 2015.

[4] M. Petsch, D. Kohlgrüber, C. L. Munoz, and T. Rothermel, "Integration of the structural solver B2000++ in a multi-disciplinary process chain for aircraft design." Conference presentation at DLRK2020, In German, September 2020.

[5] R. P. Dwight, "Robust mesh deformation using the linear elasticity equations," in *Computational Fluid Dynamics*, pp. 401–406, 2006.

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing* (E. Arge, A. M. Bruaset, and H. P. Langtangen, eds.), pp. 163–202, Birkhäuser Press, 1997.

[7] O. Krzikalla, A. Rempke, A. Bleh, M. Wagner, and T. Gerhold, "Spliss: A sparse linear system solver for transparent integration of emerging HPC technologies into CFD solvers and applications," in *STAB-Symposium 2020*, Notes on Numerical Fluid Mechanics and Multidisciplinary Design, pp. 635–645, Springer International Publishing, Juli 2020.

[8] M. J. Berger and S. H. Bokhari, "A partitioning strategy for nonuniform problems on multiprocessors," *IEEE Transactions on Computers*, vol. 36, no. 05, pp. 570–580, 1987.

[9] G. Karypis, "METIS and ParMETIS," pp. 1117–1124, 2011.

[10] E. G. Boman, U. V. Catalyurek, C. Chevalier, and K. D. Devine, "The Zoltan and Isorropia parallel toolkits for combinatorial scientific computing: Partitioning, ordering, and coloring," *Scientific Programming*, vol. 20, no. 2, pp. 129–150, 2012.

[11] A. Beckert, "Coupling fluid (CFD) and structural (FE) models using finite interpolation elements," *Aerospace Science and Technology*, vol. 4, no. 1, pp. 13–22, 2000.

[12] I. Huismann, L. Reimer, S. Strobl, J. R. Eichstädt, R. Tschüter, A. Rempke, and G. Einarsson, "Accelerating the FlowSimulator: Profiling and scalability analysis of an industrial–grade CFD–CSM toolchain," in *IX. International Conference on Coupled Problems in Science and Engineering*, Juli 2021.

[13] G. C. Firth, "LANN wing design," *NASA. Langley Research Center Cryogenic Wind Tunnel Models*, 1983.

[14] S. R. Allmaras and F. T. Johnson, "Modifications and clarifications for the implementation of the Spalart-Allmaras turbulence model," in *Seventh international conference on computational fluid dynamics (ICCFD7)*, pp. 1–11, 2012.

[15] N. Kroll, M. Abu-Zurayk, D. Dimitrov, T. Franz, T. Führer, T. Gerhold, S. Görtz, R. Heinrich, C. Ilic, J. Jepsen, *et al.*, "DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods," *CEAS Aeronautical Journal*, vol. 7, no. 1, pp. 3–27, 2016.

[16] "4th AIAA CFD High Lift Prediction Workshop (HLPW-4)," 2022.

[17] I. Huismann, S. Fechter, and T. Leicht, "HyperCODA – extension of flow solver CODA towards hypersonic flows," in *New Results in Numerical and Experimental Fluid Mechanics XIII*, pp. 99–109, 2021.