Scalability Evaluation of the CFD Solver CODA on the AMD Naples Architecture

Michael Wagner

Abstract Computational fluid dynamics (CFD) simulations are an increasingly important part of aircraft design. They allow in-depth insight into the aerodynamic behavior of components and help reducing cost and time in development. CODA is a next-generation CFD solver for aerodynamic simulations of fully equipped aircraft. It is developed by the German Aerospace Center (DLR), the French Aerospace Lab (ONERA), and Airbus, and is one of the key applications represented in the European Centre of Excellence for Engineering Applications (Excellerat). This work evaluates the CODA CFD solver on the CARA HPC system based on the AMD Naples architecture. The evaluation includes an assessment of the scalability on the largest available partition of the production system with the NASA common research model in a strong scaling scenario, a comparison of different hybrid-parallel setups suitable for the specific memory and NUMA layout and a comparison of the results with the Intel Cascade Lake architecture. Furthermore, it demonstrates the impact of node placement and unfavorable network loads on large scale runs.

1 Introduction

One of the key challenges in aviation is the aim for climate-neutral, low-noise air transport by the middle of the century. The European Commission, for instance, defines in its vision for Europe's aviation several goals to, among others, increase affordable and reliable connectivity within the European Union and at the same time mitigate the adverse impact of aviation on society and environment. These goals include a reduction of 75 % of CO_2 emissions, 90 % of NO_x emissions, and 65 % of perceived aircraft noise by 2050; in comparison to a typical new aircraft in 2000 [2].

Michael Wagner
German Aerospace Center (DLR), Institute of Software Methods for Product Virtualization,
e-mail: m.wagner@dlr.de

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023

M. M. Resch et al. (eds.), *Sustained Simulation Performance 2021*,
https://doi.org/10.1007/978-3-031-18046-0_7

To attain these goals, new aircraft have to become significantly lighter and more aerodynamically efficient, in combination with the introduction of innovative flight control and an intelligent mix of alternative propulsion system concepts. This will require a disruptive approach including step-changing aircraft technology and new design principles. Thus, future aircraft designs may be driven by unconventional layouts such as the low noise aircraft model (LNA), the blended wing body aircraft, or the flying wing configuration.

For these unconventional layouts flight characteristics will be dominated by non-linear effects. In this case, high-fidelity numerical simulations become inevitable for the design and assessment of new aircraft designs to provide reliable insight into new aircraft technologies and reach best overall aircraft performance through integrating aerodynamics, structural mechanics and systems design.

Another challenge on the road to climate-neutral aviation is the reduction of development time for new aviation technology. The development, testing and production of new aircraft involve significant time and financial investments and risks. The huge time and financial investment in aircraft development and the resulting long aircraft operation spans slow down the introduction of progressive technology and dynamic improvements. For this reason, the German Aerospace Center (DLR) is putting the virtual product at the heart of its scientific work in its guiding concepts for aeronautics research. The virtual product, i.e., high-precision mathematical and numerical representation of a new aircraft and all its characteristics and components, allows faster development cycles; starting from product development up to approval, production, maintenance and decommissioning [4].

Computational fluid dynamics (CFD) simulations for aircraft aerodynamics are already today imperative in the aircraft design process. Not only do they allow to reduce cost and time of aircraft development by omitting unnecessary prototyping, wind tunnel experiments and real flight tests, but allow a more in-depth insight into components and systems. Especially for future aircraft design driven by step-changing technology, new design principles and, consequently, non-linear effects in flight characteristics, highly accurate and efficient CFD simulations are essential.

CODA is a CFD solver for the solution of the Reynolds-Averaged Navier–Stokes equations on unstructured grids based on second-order finite-volume and higher-order Discontinuous-Galerkin (DG) discretization. The implementation addresses the efficient usage of current and upcoming high performance computing (HPC) systems and emerging technologies such as GPUs. CODA is developed in a joint effort of the German Aerospace Center (DLR), the French Aerospace Lab (ONERA) and Airbus and is one of the key next-generation engineering applications in the European Centre of Excellence for Engineering Applications (Excellerat) [3].

In this work, the CODA CFD solver is evaluated on the German Aerospace Center's CARA HPC system based on the Naples architecture from AMD. The contribution of this work is, first, an assessment of the scalability on the largest available partition of the production system with the NASA common research model in a strong scaling scenario. Second, a comparison of different hybrid-parallel setups suitable for the memory and NUMA layout of the AMD Naples architecture and a comparison of the results with the Intel Cascade Lake architecture. Third, a demonstration of the impact of node placement and network interference on large scale runs. This contribution serves, on the one hand, as best practice recommendation for the CFD solver CODA on the CARA HPC system and, on the other hand, it may provide guidance for researchers and developers in their efforts to execute their applications on the AMD Naples architecture.

The following sections provide background on the CFD solver CODA (Sect. 2), the used test case (Sect. 3) and the CARA HPC system (Sect. 4). Sect. 5 presents the results of the scalability assessment and the comparison of different hybrid-parallel setups. Finally, Sect. 6 summarizes the presented work and draws conclusions.

2 The CFD solver CODA

At the German Aerospace Center (DLR), CFD codes have been developed for decades, many of them in regular industrial use. One of them is the DLR *TAU* code [8], which is in production in the European aircraft industry, research organizations and academia since more than 15 years. It was, for instance, used for the Airbus A380 and A350 wing design. TAU implements a classical MPI parallelization to simulate steady as well as unsteady external aerodynamic flows using a second order finite-volumes discretization.

In 2012 DLR initiated the development of a new, flexible, unstructured CFD solver called *Flucs* [6], which held the opportunity to design a modern, comprehensive concept for HPC from scratch. Next to HPC, the focus was set on algorithmic efficiency using strong implicit solvers, higher-order spatial discretization via the Discontinuous Galerkin method featuring hp-adaptation in addition to finite volumes with maximum code share, and seamless integration into Python-based multi-disciplinary process chains via *FlowSimulator* [7].

Though the Flucs development had been started at DLR, it has become part of a larger cooperation that is driven by Airbus, the French aerospace lab ONERA, and DLR. After Airbus expressed its interest for a new generation CFD solver that is co-developed by ONERA and DLR in 2015, in May 2017 all three parties reached an agreement pursuing the joint effort. The joint development of the CFD solver based on Flucs was named *CODA* (CFD for ONERA, DLR and Airbus) to honor the new collaboration and the involvement of all three partners.

Similar to TAU, CODA implements classical domain decomposition to make use of distributed-memory parallelism via MPI and, additionally, the GASPI [1] implementation GPI-2 as an alternative to MPI. This Partitioned Global Address Space (PGAS) library features efficient one-sided communication to reduce network traffic and latency. Furthermore, CODA features overlapping halo-data communication with computation to hide network latency and, thus, improve scalability. In addition to classical domain decomposition, CODA uses a hybrid two-level parallelization. CODA implements sub-domain decomposition, where each domain is further partitioned into sub-domains, each of which being processed by a dedicated software thread that is mapped one-to-one to a hardware thread to maximize data locality. This allows utilizing shared-memory parallelism and provide a flexible adaption to different hardware architectures (as can be seen in Sect. 5) [11].

An integral part of CODA is the Sparse Linear Systems Solver (Spliss) [5] that is used for solving linear equation systems for implicit time integration methods, e.g. for the test case used in this work. Spliss is a linear solver library that, on the one hand, is tailored to the requirements of CFD applications but, on the other hand, independent of the particular CFD solver. Focusing on the specific task of solving linear equation systems allows for integrating more advanced, but also more complex, hardware-specific optimizations, while at the same time hiding this complexity from a CFD solver such as CODA.

3 The test case for external aerodynamics

The test case for the scalability evaluation is based on the NASA Common Research Model from the fifth AIAA CFD Drag Prediction Workshop [10]. This test case simulates steady airflow at subsonic speed and computes typical characteristics like air velocity and direction, pressure and turbulence. Fig. 1 visualizes the output of the test case with the aircraft configuration and mesh on the left and the airflow around the wing and fuselage with air pressure on the aircraft on the right. It is well studied and provides experimental data as well as numerical solutions by other CFD applications for comparison.

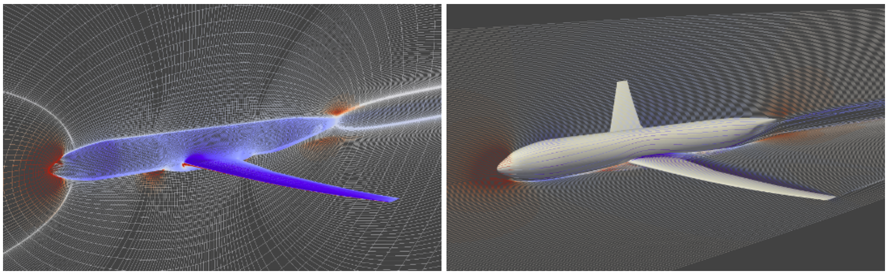

Fig. 1: Visualization of the test case simulation: aircraft configuration with mesh (left) and airflow around wing and fuselage (right); both with air pressure as color gradient.

For the CRM test case, CODA solves the Reynolds-averaged Navier–Stokes equations (RANS) with a Spalart–Allmaras one-equation turbulence model in its negative form (SAneg). It uses a second-order finite-volume spatial discretization with an implicit Euler time integration. For the linear problem, a block-Jacobi solver with LU decomposition is applied. The flow conditions are outlined by the following parameters: the Mach number is set to 0.2, the Reynolds number to 5e6, and a fixed 2.5° angle of attack is set.

The input of the test case is a rather small unstructured mesh with 5.2 million points and 10.2 million prisms that is obtained by splitting each hexahedron in the original mesh into two prisms such that the geometry's surface mesh is purely triangular. Please note that this rather small mesh (one order of magnitude smaller than industrial cases) was chosen to allow a strong scalability analysis at relatively small core counts, i.e., neither the purely prismatic volumes nor the small number of cells allow high CFD accuracy in the boundary layer.

4 The CARA HPC system

The *Computer for Advanced Research in Aerospace* (CARA) is the German Aerospace Center's current main HPC system installed by NEC. It was ranked at 221 in the Top500 list of 11/2019 providing 1.7 TFlop/s out of 2.6 Tflop/s theoretical peak performance [9]. The system is primarily used for production simulations and research in the fields of aerospace and mobility.

The CARA HPC system offers 2280 compute nodes, which are connected by an Infiniband HDR network. Each compute node consists of two AMD EPYC 7601 (32 cores at 2.2 GHz) with four dies of eight cores each. The system has two-way simultaneous multi-threading (SMT) enabled, i.e. there are two hardware threads running on each core. In total, the system offers 145,920 compute cores.

With respect to memory access, the AMD Naples architecture presents rather complex characteristics. The architecture includes eight NUMA (non-uniform memory access) domains and three NUMA distances: first, to the memory of the seven other cores on the same die, second, to the memory on the three other dies on the same chiplet (socket) and, third, to the memory located on the other chiplet. In addition, only four of eight cores on each die share a last level cache (L3 cache) leading to an additional difference in memory access latency depending on the locality of the data; weather it is in the shared L3 cache of the according core or in the adjoining L3 cache on the same die. The complex NUMAness and the split L3 cache per die should be put in consideration when it comes to data locality and memory access, in particular, for shared-memory parallelization and thread synchronization.

5 Evaluation

This section first outlines the measurement setup and then presents scalability results for different hybrid-parallel setups, different mesh sizes, analyzes the impact of node placement and unfavorable network loads, and concludes with a comparison of the AMD Naples and Intel Cascade Lake architectures with respect to their impact on threading performance.

5.1 Measurement setup

Prior to the launch of the CARA system, it was already established for CODA that, in general, a hybrid-parallel execution of the code using MPI and threads provides best performance. In particular for higher core counts, a suitable utilization of shared-memory parallelization via threads reduces the total number of MPI ranks, the number of MPI operations and cost for MPI global communication (e.g. collectives) since less MPI ranks are involved. However, it was also established for CODA that threading performance is impacted by the memory hierarchy, in particular, data locality and the size of NUMA domains.

Therefore, for the scalability evaluation, first, all software threads are bound to a hardware thread to ensure thread affinity. Second, three different hybrid-parallel setups are evaluated to identify the impact of the memory hierarchy:

- 16 MPI processes per node with 4 OpenMP threads each. This way all four threads are in the same NUMA domain and share the same L3 cache.
- 8 MPI processes per node with 8 OpenMP threads each. This way all eight threads are in the same NUMA domain but are split across two L3 caches.
- 4 MPI processes per node with 16 OpenMP threads each. This way the 16 threads are split across two NUMA domains.

Please note that other combinations such as 1 MPI process with 64 threads each, i.e. threads split across two sockets, were tested but not included in the full evaluation since they did perform inferior to the above setups, which was already established before, and did not provide any further inside into the Naples architecture itself.

As stated before, on the AMD Epyc architecture each core can be over-subscribed to use two hardware threads on each core, i.e. two-way simultaneous multi-threading (SMT). This allows running two software threads on each core, scheduled by the operating system, and may help increasing performance by increasing the number of independent instructions in the execution pipeline. In addition to the above setups using one hardware thread per core, the according setups with simultaneous multi-threading are recorded, too. For these setups the number of OpenMP threads per MPI process is doubled, e.g. the version with 16 MPI process and 4 OpenMP threads each is also measured with 16 MPI processes and 8 OpenMP threads each, whereas the 8 OpenMP threads run on the same 4 cores as the 4 OpenMP threads.

All measurements were executed only one time due to the large core counts, according costs and wait times in the queue. This must be kept in mind when discussing the significance of individual data points. In general, the recorded runtimes are consistent in themselves; nonetheless, the data points should not be taken as exact values but rather as basis for general trends. Parallel runtimes are affected, amongst others, by the applied scheduling to nodes and the overall load on the system. In that sense, the recorded runtimes reflect typical behavior that users would see in normal production mode; not isolated benchmark runs in a near-perfect environment.

5.2 Evaluation of different hybrid-parallel setups

Fig. 2 shows the parallel speedup for the different setups of MPI processes to OpenMP threads without and with enabled hyper-threading for 1 to 512 nodes, i.e. 64 to 32.768 cores; whereas 512 nodes was the largest partition that could be reasonably used during normal production of the system. The figure highlights the general scaling behavior of the various setups.

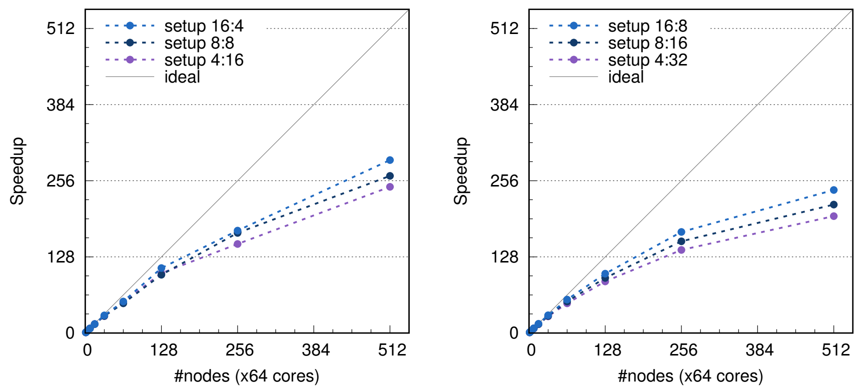

Fig. 2: Speedup for 1 to 512 nodes (64 to 32.768 cores) for different MPI rank to OpenMP thread ratios: without (left) and with simultaneous multi-threading (right).

Without simultaneous multi-threading (Fig. 2, left) CODA achieves about 90 % parallel efficiency at 4096 cores and 59 % parallel efficiency at 32,768 cores. This represents very good strong scaling behavior for such a small mesh, where at 32,768 cores on average only 312 elements are assigned to each software thread; an extreme case that is usually not approached in production simulations. As expected, based on the architecture, the best setup is with four threads per MPI process, so that all four threads are executed on the four cores that share a last level cache. The second-best setup is with eight threads per MPI process, so that all eight threads are executed within a single NUMA domain. The execution of threads across NUMA domains leads to further reduced performance.

With enabled simultaneous multi-threading (Fig. 2, right) CODA achieves about 88 % parallel efficiency at 4096 cores and 47 % parallel efficiency at 32,768 cores. This again represents very good strong scaling behavior for such a small mesh, where on average only 150 elements are assigned to each software thread at 32,768 cores. Consequently, the scalability is slightly reduced since each thread has only half the computational load. In that sense, computing a test case with 10.2 million prisms across 65,536 threads sets an extreme case and highlights the excellent scaling behavior of CODA even on very little computation load per thread. In comparison, typical workloads used in production simulation have one or two orders of magnitude more elements per thread.

Although the setups with enabled simultaneous multi-threading show slightly lower parallel efficiency at scale, they provide significantly better compute performance. Comparing the individual simultaneous multi-threading setups with their non-simultaneous multi-threading counterparts, the setups with enabled simultaneous multi-threading have a 15 - 20 % reduced runtime, which might also be a factor in the slightly reduced scalability.

5.3 Evaluation of different mesh sizes and node placement

To evaluate the scalability relative to the number of mesh elements, the strong scalability of CODA is measured for three different mesh sizes: *tiny* with 1.2 million prisms, *medium* with 10.2 million prisms (same mesh as above) and *fine* with 34.5 million prisms. All use the setup with 16 MPI processes per node and 4 OpenMP threads each (disabled simultaneous multi-threading).

The left side of Fig. 3 shows the parallel speedup for the different mesh sizes. It highlights the general scaling behavior relative to the number of mesh elements. As expected, the larger the mesh size, the better the scaling behavior. However, scalability relative to the number of elements per thread does not increase proportionally. Hence, additional factors, except the decreasing workload per thread, impact overall scalability, especially, for large core counts, were MPI communication becomes an increasingly limiting factor, for instance, the non-linear scaling of global MPI collectives and network interference.

Indeed, on a production system such as CARA, the fluctuating network load can significantly impact application performance for large core counts. The right side of Fig. 3 compares the scalability of CODA with three levels of network interference for the medium mesh: First, the typical network interference for a typical CODA run as seen in the previous results. In this case, the job scheduler places the application on the first available set of nodes (random placement in Fig. 3). Second, reduced network interference that is achieved by using a set of nodes that is connected by a minimal number of network switches (good placement). This can be realized, for

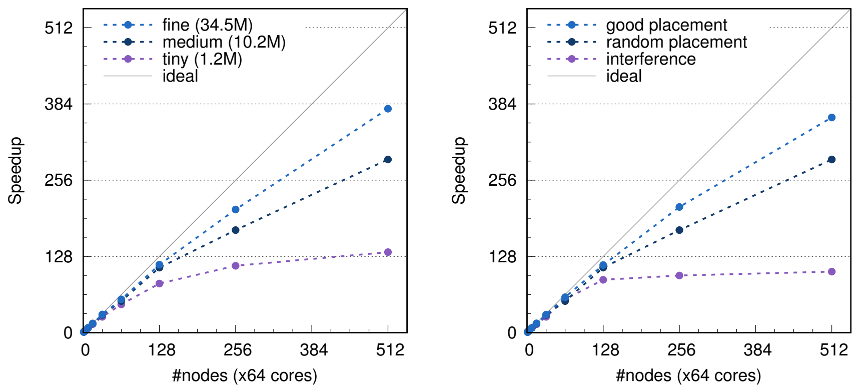

Fig. 3: Speedup for 1 to 512 nodes (64 to 32.768 cores) for different mesh sizes (left) and for different node placements and unfavorable network loads leading to interference (right).

instance, with the according use of the switches option in the Slurm job scheduler. Third, increased network interference that can be reproduced by using the random node placement and running another large-scale network-heavy application at the same time, e.g. another large CODA simulation.

With the improved node placement CODA achieves about 93 % (vs. 90 %) parallel efficiency at 4096 cores and 71 % (vs. 59 %) parallel efficiency at 32,768 cores. However, with default node placement and unfavorable network loads CODA only achieves about 88 % parallel efficiency at 4096 cores and 20 % parallel efficiency at 32,768 cores. The huge span from 20 % (heavy network interference) to 59 % (typical network interference) to 71 % (reduced network interference) underlines the significant impact on application performance for large core counts that can occur unwillingly and possibly unnoticed on a production system. As a consequence, today, Slurm's switches option with a moderate wait time is set by default for all jobs submitted on CARA.

5.4 Comparison of AMD Naples and Intel Cascade Lake architectures

To better understand the threading performance on the AMD Naples architecture, the results are compared to results achieved on the Intel Cascade Lake architecture. The AMD Naples nodes consist of two AMD Epyc 7601 with 32 cores and 64 hardware threads each and has a total power consumption of 360 W. The Intel Cascade Lake architecture consists of two Intel Xeon Platinum 9242 with 48 cores and 96 hardware threads each and has a total power consumption of 700 W. To fairly compare the two architectures, two AMD Naples nodes are set against one Intel Cascade Lake node to match power consumption, which is often a limiting factor in computing centers and mainly influences operational costs. While this comparison based on power gives the AMD a slight benefit of 20 W, it can still be considered fair since the Intel architecture was released almost two years later.

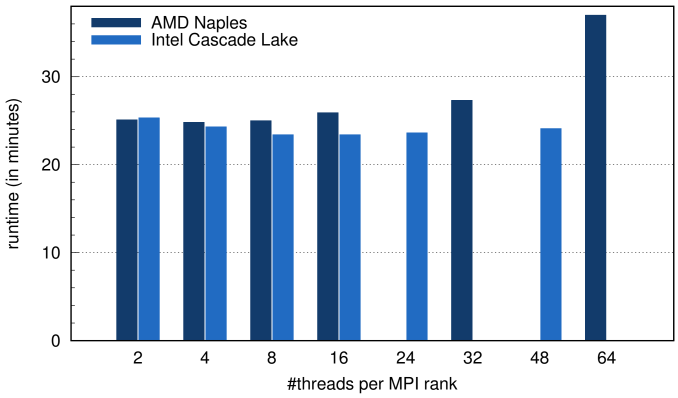

Fig. 4: Threading performance on two AMD Naples nodes vs. one Intel Cascade Lake node.

Fig. 4 shows the runtime for the test case with the tiny mesh of 1.2 million elements for different hybrid-parallel setups with enabled two-way simultaneous multi-threading or hyper-threading, respectively. In general, both architectures achieve very similar performance. However, as seen before, CODA performs less efficiently on the AMD Naples architecture the more threads per MPI process are used; with the optimum being four threads per process and a significant increase towards using one MPI process and 64 threads per socket. For the Intel Cascade Lake architecture, the test case shows much less variance between the different setups; reaching its optimum at 16 threads per MPI rank but comparable performance up to one MPI process and 48 threads per socket.

These results put the AMD Naples architecture at a disadvantage for large scale runs, where good threading parallelism is a crucial factor, since it allows reducing the number of MPI processes and, thus, the impact of MPI communications.

6 Conclusion

This work presents an evaluation of the scalability of CODA, a CFD solver for aircraft aerodynamics. This evaluation includes an assessment of the scalability on the largest available partition of DLR's CARA HPC system. The test case based on the NASA common research model achieves 93 % parallel efficiency at 4096 cores and 71 % parallel efficiency at 32,768 cores in a strong scaling scenario despite running on a very small mesh with very little computational load per thread; an extreme case that is usually not approached in production simulations. Furthermore, the evaluation compares different hybrid-parallel setups suitable for the specific memory and NUMA layout of the AMD Naples architecture. It highlights that best hybrid-parallel performance is reached when using only four threads per MPI process, so that these threads share the same last level cache. This stands in contrast to the Intel Cascade Lake architecture, where comparable performance for all hybrid setups was obtained. Lastly, an assessment of node placement and network interference underlines the significant impact of unfavorable network loads on application performance resulting in up to a factor of 3.5 divergence in parallel efficiency.

Acknowledgements This work has been supported by the EXCELLERAT project, which has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 823691.

References

1. T. Alrutz, J. Backhaus, T. Brandes, V. End, T. Gerhold, A. Geiger, D. Grünewald, V. Heuveline, J. Jägersküpper, A. Knüpfer, O. Krzikalla, E. Kügeler, C. Lojewski, G. Lonsdale, R. Müller-Pfefferkorn, W.E. Nagel, L. Oden, F.-J. Pfreundt, M. Rahn, M. Sattler, M. Schmidtobreick, A. Schiller, C. Simmendinger, T. Soddemann, G. Sutmann, H. Weber and J.-P. Weiss. GASPI – A Partitioned Global Address Space Programming Interface. In: *Facing the Multicore-Challenge III*, LNCS 7686, pp. 135–136 (2013). DOI: https://doi.org/10.1007/978-3-642-35893-7_18
2. Directorate-General for Mobility and Transport (European Commission), Directorate-General for Research and Innovation (European Commission). Flightpath 2050: Europe's vision for aviation: maintaining global leadership and serving society's needs (2012). DOI: https://doi.org/10.2777/15458
3. The European Centre of Excellence for Engineering Applications (EXCELLERAT). http://www.excellerat.eu [Online; accessed 2022-02-08]
4. Guiding concepts for DLR aeronautics research. https://www.dlr.de/EN/research/aeronautics/guiding-concepts.html [Online; acc. 2022-02-08]
5. 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: *New Results in Numerical and Experimental Fluid Mechanics XIII*, pp. 635–645 (2021). DOI: https://doi.org/10.1007/978-3-030-79561-0
6. T. Leicht, D. Vollmer, J. Jägersküpper, A. Schwöppe, R. Hartmann, J. Fiedler and T. Schlauch. DLR-Project Digital-X – Next Generation CFD Solver 'Flucs'. In: *Deutscher Luft- und Raumfahrtkongress* (2016).
7. M. Meinel and G. Einarsson. The FlowSimulator Framework for Massively Parallel CFD Applications. In: *PARA 2010*, (2010).
8. D. Schwamborn, T. Gerhold and R. Heinrich. The DLR TAU Code: Recent Applications in Research and Industry. In: *Proc. of the European Conference on Computational Fluid Dynamics, ECCOMAS CFD* (2006).
9. E. Strohmaier, J. Dongarra, H. Simon and M. Meuer. The 54th Top500 list (2019). https://www.top500.org/lists/top500/2019/11/ [Online; accessed 2022-02-08]

10. J. Vassberg. A Unified Baseline Grid about the Common Research Model Wing/Body for the Fifth AIAA CFD Drag Prediction Workshop. *29th AIAA Applied Aerodynamics Conference* (2011). DOI: https://doi.org/10.2514/6.2011-3509
11. M. Wagner, J. Jägersküpper, D. Molka and T. Gerhold. Performance Analysis of Complex Engineering Frameworks In: *Tools for High Performance Computing*, pp. 123–138 (2021). DOI: https://doi.org/10.1007/978-3-030-66057-4