Evaluating Performance and Scalability of the Sparse Linear Systems Solver Spliss

Michael Wagner, Jasmin Mohnke, Olaf Krzikalla, Arne Rempke

Abstract Solving linear equation systems is an integral part of implicit methods in computational fluid dynamics (CFD). The sparse linear system solver Spliss aims to provide a linear solver library that, on the one hand, is tailored to requirements of CFD applications but, on the other hand, independent of the particular CFD solver. Focusing on the specific task of solving linear systems allows for integrating more advanced, but also more complex, hardware-specific optimizations, while at the same time hiding this complexity from the CFD solver. Spliss enables the execution of the computationally intensive linear solver on GPUs without the necessity of any code adaption. This work evaluates performance and scalability of Spliss using CODA as an example. CODA is the CFD software being developed as part of a collaboration between the French Aerospace Lab ONERA, the German Aerospace Center (DLR), Airbus, and their European research partners. CODA is jointly owned by ONERA, DLR and Airbus. The evaluation includes an assessment of the scalability on an HPC system based on the AMD Naples architecture and demonstrates the seamless integration of GPUs on a cluster based on Nvidia V100 GPUs.

1 Introduction

For future aircraft design, Computational fluid dynamics (CFD) is a key technology on the road to solving global challenges like climate change or mobility. CFD simulations are indispensable to reach European goals of drastic reductions in aviation emissions such as a reduction of 75 % of CO2 emissions, 90 % of NOx emission and 65 % of perceived aircraft noise by the middle of the century while at the same time increasing affordable and reliable connectivity within the European Union, its neighbors and partners and ensuring the competitiveness of European industry [1].

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

For this, newly developed 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.

Computational fluid dynamics simulations for aircraft aerodynamics are already today imperative in the aircraft design process. They allow to reduce cost and time of aircraft development by omitting unnecessary prototyping, wind tunnel experiments and real flight tests and, thus, accelerate the introduction of progressive technology and dynamic improvements. In particular, the progress towards the virtual product, i.e., high-precision numerical representation of an aircraft and all its characteristics and components, leads to faster development cycles; starting from product development up to approval, production, maintenance and decommissioning [2].

Moreover, high-precision numerical simulations are inevitable for the 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. Such simulations require vast computational resources that are met partly by today's high performance computing (HPC) systems. This year marks the advent of the first true exascale HPC system, i.e. a system capable of calculating more than 10^{18} double precision operations per second, consisting of over 8.7 million cores [3]. CFD simulations aiming to utilize these powerful systems need to be, first, highly parallel to take advantage of the increasing number of compute cores and, second, support the usage of heterogeneous systems consisting of new processor architectures and hardware accelerators such as GPUs.

The sparse linear systems solver Spliss [4] is designed to assist with this challenge by offering the efficient solving of large linear equation systems that result from the discretization of the Reynolds-averaged Navier-Stokes (RANS) equations in CFD methods. Since Spliss focuses on the specific task of solving linear equation systems, it integrates more advanced, but also more complex, hardware-specific optimizations, while at the same time hiding this complexity from the CFD solver. Spliss enables the efficient and transparent execution of the computationally intensive linear solver on new architectures and hardware accelerators such as GPUs. This way, the CFD solver can leverage new architectures and hardware accelerators without the necessity of any code adaptation in the CFD solver.

This work evaluates performance and scalability of Spliss using the CODA CFD software as an example. CODA is the CFD software being developed as part of a collaboration between the French Aerospace Lab ONERA, the German Aerospace Center (DLR), Airbus, and their European research partners. CODA is jointly owned by ONERA, DLR and Airbus. It is one of the key next-generation engineering applications represented in the European Centre of Excellence for Engineering Applications (EXCELLERAT) [5].

The contribution of this work is, first, an assessment of the performance and scalability of Spliss with CODA on the largest available partition of the German Aerospace Center's CARA HPC production system based on the AMD Naples architecture using the the NASA common research model in a strong scaling scenario. Second, a demonstration of the seamless integration of hardware accelerators such as GPUs into CODA via Spliss on a cluster based on Nvidia V100 GPUs.

The following sections provide background on the sparse linear systems solver Spliss (Sect. 2) and the CFD software CODA (Sect. 3). Sect. 4 introduces the CARA HPC system, the used test case and presents the results of the scalability assessment. Similarly, Sect. 5 introduces the Nvidia V100 test cluster, the used test case and presents first results of Spliss executing the linear solver of CODA on GPUs. Finally, Sect. 6 summarizes the presented work and draws conclusions.

2 The Sparse Linear Systems Solver

The sparse linear systems solver Spliss is a library that enables a CFD software like CODA to solve large distributed linear equation systems originating from the discretization RANS equations. Such linear equation systems are sparse, since each equation only relates a few of the millions of unknowns. The prescribed couplings however have a structure that results in a block-structured sparse matrix for the linear equation system to be solved.

Spliss allows to specify and solve such linear equations with sparse matrices of dense blocks. A single dense matrix block typically corresponds to the couplings between the different physical quantities within a physical element or on the facette between two elements. For a classical finite volume discretization, the blocks have a size of e.g. 7x7 for a two-equation turbulence model, and for higher order Discontinuous Galerkin approaches these dense blocks can become much larger. Spliss supports both fixed block sizes for the whole matrix (same order approach) and different block sizes within the same equation system (different order approach).

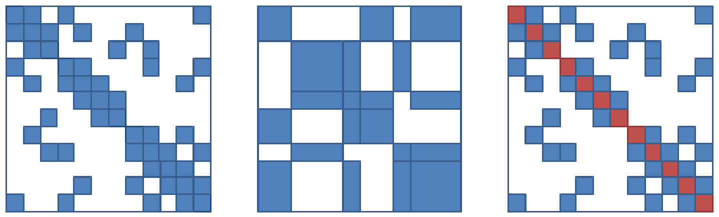

Fig. 1 Supported matrices in Spliss: sparse block-structure with constant (left) or variable (middle) block sizes, different scalar types (right).

Another important feature of Spliss is that matrix entries can be of mixed scalar data type. This is particularly important when solving harmonic balance or time-spectral methods, where part of the matrix (typically the diagonal blocks) contain complex numbers, while other parts remain real-valued, see Fig. 1. Also using different precision scalar types is possible with this approach, which allows saving memory and computational effort, e.g., when evaluating matrix vector products.

In order to solve the sparse linear equation system, Spliss includes a wide range of iterative solvers and preconditioners. These solver components can be stacked together in a flexible way, allowing multiple layers of different solver types operating on potentially different linear operators, see Fig. 2. This permits providing fast preconditioners in the inner hot loop while utilizing more exact higher-level operators in the outer loops. Some solver components are specifically tailored for the demands of CFD simulations, e.g., the LinesInversion for handling the high aspect-ratio cells in the boundary layer of a mesh for flows with high Reynolds numbers [4].

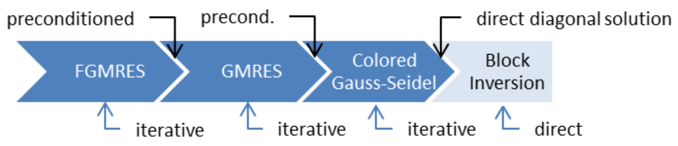

Fig. 2 A possible solver chain in Spliss modeling the inner-outer GMRES method preconditioned by a point-implicit multi-color Gauss-Seidel method.

Spliss is designed and implemented to efficiently leverage the computational resources of contemporary and emerging HPC platforms. This includes techniques such as one-sided communication, hybrid and heterogeneous parallelization. The use of the C++ template mechanism shifts decisions such as the usage of a GPU and the selection of an appropriate memory layout to the compile time. Thus, a shared library object of Spliss compiled and built on a particular HPC cluster respects the properties of that cluster. On the other hand, Spliss is designed in such a way, that it does not expose these properties to the library interface.

By that means, it is sufficient to link against a GPU-enabled Spliss library in order to benefit from available GPUs. An application is not even required to use the GPU framework during its own compilation process thanks to an automatic explicit template instantiation process. If multiple GPUs are available on a compute node, Spliss distributes those GPUs to the processes running on the node. In addition, Spliss supports CUDA-aware MPI implementations [6]. Those implementations combine MPI and CUDA, so that GPU buffers can be directly passed to MPI calls; omitting a detour via the host buffer.

3 The CODA CFD Software

The development of computational fluid dynamics software has a long history at the German Aerospace Center (DLR). Currently, the *TAU* CFD package [7] developed and maintained by DLR is in production in the European aircraft industry, research organizations and academia since more than 20 years. For instance, TAU played a vital role in the Airbus A380 and A350 wing design. As state-of-the-art for its time, TAU implements a classical MPI parallelization to simulate steady and unsteady external aerodynamic flows using a second order finite-volumes discretization.

To design a modern concept for HPC from scratch, in 2012 DLR initiated the development of a new, flexible, unstructured CFD solver called *Flucs* [8]. Thereby, 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* [9, 10]. While the development of Flucs had been started at DLR, it since has become part of a larger cooperation that is driven by Airbus, the French aerospace lab ONERA, and DLR. The joint development of the CFD software based on Flucs was named *CODA* (CFD for ONERA, DLR and Airbus) to honor the new collaboration and the involvement of all three partners pursuing the joint effort and co-development.

The CODA CFD software implements classical domain decomposition to support distributed-memory parallelism via MPI and the GASPI [11] implementation GPI-2. This Partitioned Global Address Space (PGAS) library features efficient one-sided communication to reduce network traffic and latency. Furthermore, CODA improves scalability by allowing the overlap of halo-data communication with computation to hide network latency. In addition to classical domain decomposition, CODA employs a hybrid two-level parallelization to utilize shared-memory parallelism for multi- and many-core architectures [12]. 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. The hybrid approach allows utilizing all parallelism layers and provide a flexible adaption to different hardware architectures [13, 14].

4 Evaluation on an AMD Naples HPC System

This section provides an assessment of the scalability of Spliss with CODA on the largest available partition of the German Aerospace Center's CARA HPC production system based on the AMD Naples architecture using the the NASA common research model in a strong scaling scenario. The *Computer for Advanced Research in Aerospace* (CARA) is one of the German Aerospace Center's main HPC systems providing 1.7 TFlop/s peak performance. The system offers 2280 compute nodes, whereas each compute node consists of two AMD EPYC 7601 (32 cores at 2.2 GHz) with four dies of eight cores each. In total, the system offers 145,920 compute cores.

The AMD Naples architecture within this system 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 the eight cores on each die share a last level cache (L3 cache), which presents an additional difference in memory access latency depending on the locality of the data; whether it is in the shared L3 cache of the according core or in the adjoining L3 cache on the same die.

4.1 The Test Case

The test case for the scalability evaluation is based on the NASA Common Research Model (CRM) from the fifth AIAA CFD Drag Prediction Workshop [15]. It simulates the external airflow at subsonic speed and computes typical characteristics like air velocity and direction, pressure and turbulence via a one-equation turbulence model. Fig. 3 visualizes the computed air pressure and shows aircraft configuration and mesh on the left and the airflow around the wing and fuselage on the right. The CRM test case is well studied and provides experimental data as well as numerical solutions by other CFD applications for comparison.

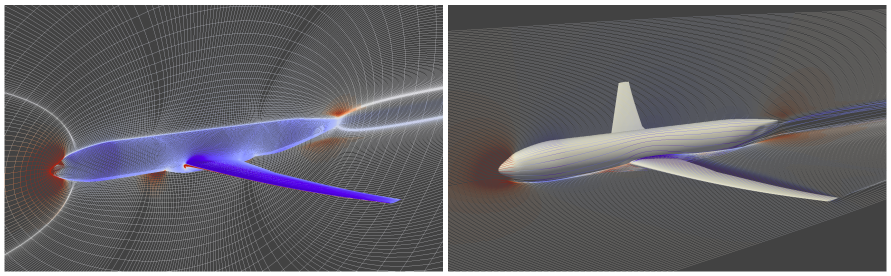
Fig. 3 Visualization of the test case simulation: aircraft configuration with mesh (left) and airflow around wing and fuselage (right); both with air pressure as color gradient.

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 and solved via Spliss. For this case, the vast majority of the iteration phase is spent in the linear solver, i.e. Spliss, thus, measuring the iteration time of CODA provides a very close estimation of the performance and scalability of Spliss within a real-world example. While results may be biased by CODA, the measurements show the minimum performance and scalability of CODA together with Spliss since performance degrading effects accumulate. In this sense, CODA and Spliss may achieve better performance and scalability individually.

The test case operates with a 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. This mesh is about one order of magnitude smaller than typical industrial cases and was chosen to allow a strong-scaling analysis at relatively small core counts, i.e., neither the purely prismatic volumes nor the small number of cells allow for high CFD accuracy in the boundary layer.

4.2 Measurement Setup

For the scalability evaluation all software threads are bound to a hardware thread to ensure thread affinity and three 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.

In addition to the above setups using one hardware thread per core, the according setups with two-way 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 processes 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 four cores as the 4 OpenMP threads.

All measurements were executed only a single 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, data points should be considered as general trend rather than exact values. Parallel runtimes, in particular, can be affected by the specific scheduling to nodes and the overall load on the system, among others. In that sense, the recorded runtimes reflect typical behavior that users can expect in production mode; not isolated benchmark runs in a close-to-perfect environment.

4.3 Results

Fig. 4 shows the general scaling behavior for the different setups of MPI processes to OpenMP threads without and with enabled two-way simultaneous multi-threading for 1 to 512 nodes, i.e. 64 to 32.768 cores. On the CARA system 512 nodes was the largest partition that could be reasonably used during normal operation.

Without simultaneous multi-threading CODA with Spliss achieves about 90 % parallel efficiency at 4096 cores and 59 % parallel efficiency at 32,768 cores within the iterative phase. 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. As expected for the AMD Naples architecture, the best setup is with four threads per MPI process, so that all four threads are executed on the four cores that share the same 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 (16 threads per MPI process) leads to further reduced performance.

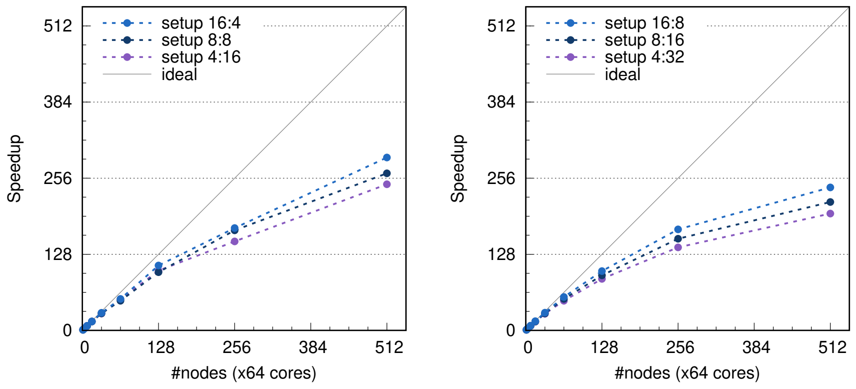

Fig. 4 Speedup for 1 to 512 nodes (64 to 32.768 cores) for different MPI rank to OpenMP thread ratios: without (left) and with simultaneous multi-threading (right).

With enabled simultaneous multi-threading CODA with Spliss achieves about 88 % parallel efficiency at 4096 cores and 47 % parallel efficiency at 32,768 cores in the iterative phase. Again, this 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 computational load per thread. In comparison, typical workloads used in production have at least 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 Evaluation on an Nvidia V100 GPU System

This section highlights the benefits of using Spliss to incorporate accelerators, in this case GPUs, for solving large, sparse linear equations systems while no changes need to be implemented in the CFD software, in this case CODA. For this assessment, a test case is run on an Nvidia V100 GPU system, which has compute nodes consisting of two Intel Xeon 6230 CPUs with 20 cores each and four Nvidia Tesla V100 GPUs.

5.1 The Test Case

The test case for the evaluation is based on the NASA 3D Onera M6 wing test case in version 2308 [16]. It is widely used for research purposes and well-studied. Both experimental data and numerical solutions are available for validation of results.

The test case solves the Reynolds-averaged Navier-Stokes equations (RANS) with a Spalart-Allmaras one-equation turbulence model in its negative form (SAneg). A Spliss Block Inversion preconditioned Block-Jacobi Solver is applied to solve the linear system. The mesh used for the test case consists of about 8.65 M volume elements which is large enough to achieve peak performance per single V100 GPU with the chosen linear solver components for the scale of multi-GPU results presented.

5.2 Measurement Setup

The evaluation compares a single- and multi-GPU setup with the Spliss linear solver offloaded to GPUs on one and multiple compute nodes to a CPU-only setup on the same system. To compare one GPU against one CPU only two of the four GPUs per compute node are used. For the parallel setups on CPU, two MPI proecesses per node, i.e one MPI processes per socket, with 20 OpenMP threads each are launched and bound to a core. For the GPU accelerated execution the CPU setup is identical and additionally each process offloads to one GPU. Switching from the CPU-only setup to taking advantage of Spliss GPU acceleration merely requires linking to a Spliss variant that is compiled for GPU, no changes on the CODA side are necessary.

Both the runtime for the entire time integration iteration phase with a fixed number of 100 iterations and solely the time spent in the linear solver are measured. The latter includes any host-to-device or device-to-host memory transfers that are necessary when offloading this section of computation to GPUs.

5.3 Results

Acceleration of the Spliss linear solver by offloading computations provides performance gain with a single GPU compared to a single 20-core CPU on the test system compute node. With the use of multiple GPUs both intra- and inter-node MPI communication required for halo updates can prove to be a bottleneck when being carried out via host memory despite all data is residing in the GPU memory during the computation. In order for a gain seen with GPU acceleration to translate to a multi-GPU case, CUDA-aware MPI with GPUDirect acceleration and, thus, optimized transfers from host to device buffers and vice versa need to be taken advantage of (currently work in progress). This gain can be observed both for the intra-node case as depicted in Fig. 5 with two MPI processes as well as inter-node communication with four MPI processes distributed on two compute nodes.

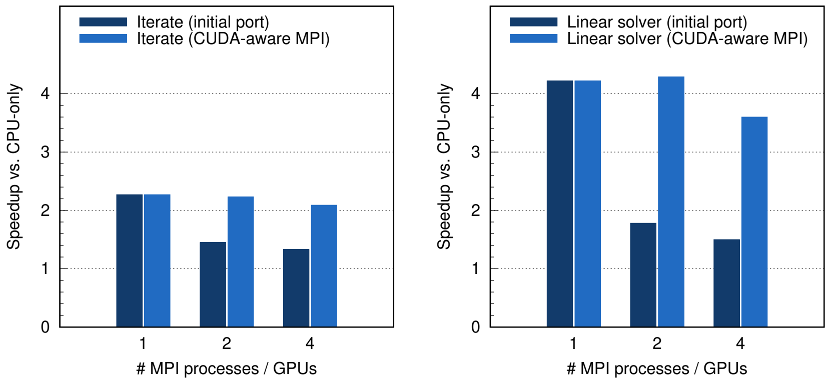

Fig. 5 GPU speedup of the entire iteration phase (left) and solely the linear solver (right) with MPI communication via host buffers (initial port) and using CUDA-aware MPI.

For the parallel configuration, the number of available GPUs predetermines the number of used MPI processes that are used. Though this is generally the preferred setup from a GPU performance perspective it is not a fixed constraint given by Spliss and can be circumvented by using Nvidia MPS (Multi Process Service) [17], which enables the efficient offloading of work from multiple processes to a single GPU. Therefore, the seamless integration of GPU usage in application codes via Spliss also extends to the parallel configuration setup.

6 Conclusion

This work presents an evaluation of performance and scalability of the sparse linear systems solver Spliss with the CODA CFD software for aircraft aerodynamics. The test case based on the NASA common research model achieves 90 % parallel efficiency at 4096 cores and 59 % 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 rarely approached in production simulations. Furthermore, the assessment 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, which is specific to the memory layout of the AMD Naples architecture and stands in contrast to other architectures. A second test case based on the 3D ONERA M6 wing shows that Spliss allows leveraging GPUs without the necessity of any code adaption in the CFD solver. By providing the seamless integration of GPUs, Spliss accelerates the test case by a factor of 2.2 in a node-wise comparison.

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. 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
2. Guiding concepts for DLR aeronautics research. https://www.dlr.de/EN/research/aeronautics/guiding-concepts.html [Online; acc. 2022-07-29]
3. Erich Strohmaier, Jack Dongarra, Horst Simon and Martin Meuer: The 59th Top500 list (2022) https://www.top500.org/lists/top500/2022/06/ [Online; accessed 2022-07-29]
4. Olaf Krzikalla, Arne Rempke, Alexander Bleh, Michael Wagner and Thomas 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_60
5. The European Centre of Excellence for Engineering Applications (EXCELLERAT). http://www.excellerat.eu [Online; accessed 2022-07-29]
6. Jiri Kraus: An Introduction to CUDA-Aware MPI. (2013) https://developer.nvidia.com/blog/introduction-cuda-aware-mpi [Online; acc. 2022-07-29]
7. Dieter Schwamborn, Thomas Gerhold, Ralf Heinrich: The DLR TAU Code: Recent Applications in Research and Industry. In *Proc. of the European Conference on Computational Fluid Dynamics, ECCOMAS CFD* (2006)
8. Tobias Leicht, Daniel Vollmer, Jens Jägersküpper, Axel Schwöppe, Ralf Hartmann, Jens Fiedler and Tobias Schlauch: DLR-Project Digital-X – Next Generation CFD Solver 'Flucs'. Deutscher Luft- und Raumfahrtkongress 2016
9. Michael Meinel and Gunnar Einarsson: The FlowSimulator Framework for Massively Parallel CFD Applications. In *PARA 2010*
10. Immo Huismann, Lars Reimer, Severin Strobl, Jan Eichstädt, Ronny Tschüter, Arne Rempke and Gunner Einarsson: Accelerating the FlowSimulator: Profiling and Scalability Analysis of an Industrial-grade CFD-CSM Toolchain. In: *9th Edition of the International Conference on Computational Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2021)* (2021) DOI: https://doi.org/10.23967/coupled.2021.008
11. Thomas Alrutz, Jan Backhaus, Thomas Brandes, Vanessa End, Thomas Gerhold, Alfred Geiger, Daniel Grünewald, Vincent Heuveline, Jens Jägersküpper, Andreas Knüpfer, Olaf Krzikalla, Edmund Kügeler, Carsten Lojewski, Guy Lonsdale, Ralph Müller-Pfefferkorn, Wolfgang E. Nagel, Lena Oden, Franz-Josef Pfreundt, Mirko Rahn, Michael Sattler, Mareike Schmidtobreick, Annika Schiller, Christian Simmendinger, Thomas Soddemann, Godehard Sutmann, Henning Weber and Jan-Philipp 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
12. Jens Jägersküpper and Daniel Vollmer: On Highly Scalable 2-level-parallel Unstructured CFD. In: *8th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2022)* (to be published)
13. Michael Wagner, Jens Jägersküpper, Daniel Molka and Thomas 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_6
14. Michael Wagner: Scalability Evaluation of the CFD Solver CODA on the AMD Naples Architecture. In: *Sustained Simulation Performance 2021* (to be published)
15. John 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
16. Daniel Destarac, Antoine Dumont: ONERA M6 Wing Test-Case, Original and TMR. (2016) https://turbmodels.larc.nasa.gov/onerawingnumerics_val.html [Online; accessed 2022-07-29]
17. Nvidia: Multi-Process Service. (2021) https://docs.nvidia.com/deploy/pdf/CUDA_Multi_Process_Service_Overview.pdf