The CFD Solver CODA and the Sparse Linear Systems Solver Spliss: Evaluation of Performance and Scalability

HLRN CFD Workshop Day, July 29th 2021

Michael Wagner
German Aerospace Center (DLR)
Institute of Software Methods for Product Virtualization
High Performance Computing

Background and Motivation

Flightpath 2050: Europe‘s Vision for Aviation

Europe’s Vision for Aviation

- Maintaining global leadership
- Serving society’s needs

Goals (relative to typical aircraft in 2000)

- CO2 emissions reduced by 75%
- NOx emissions reduced by 90%
- Perceived aircraft noise reduced by 65%
- Long term: zero emission and low noise

Consequences

- Heavy demands on future product performance
- Step changes in aircraft technology required
- New design principles mandatory

Background and Motivation (2)

Numerical Simulation – Key Enabler for Future Aircraft Design

Future aircraft

- Goals: drastic reduction CO_2, NOx and noise emissions
- Step changes in aircraft technology and new designs

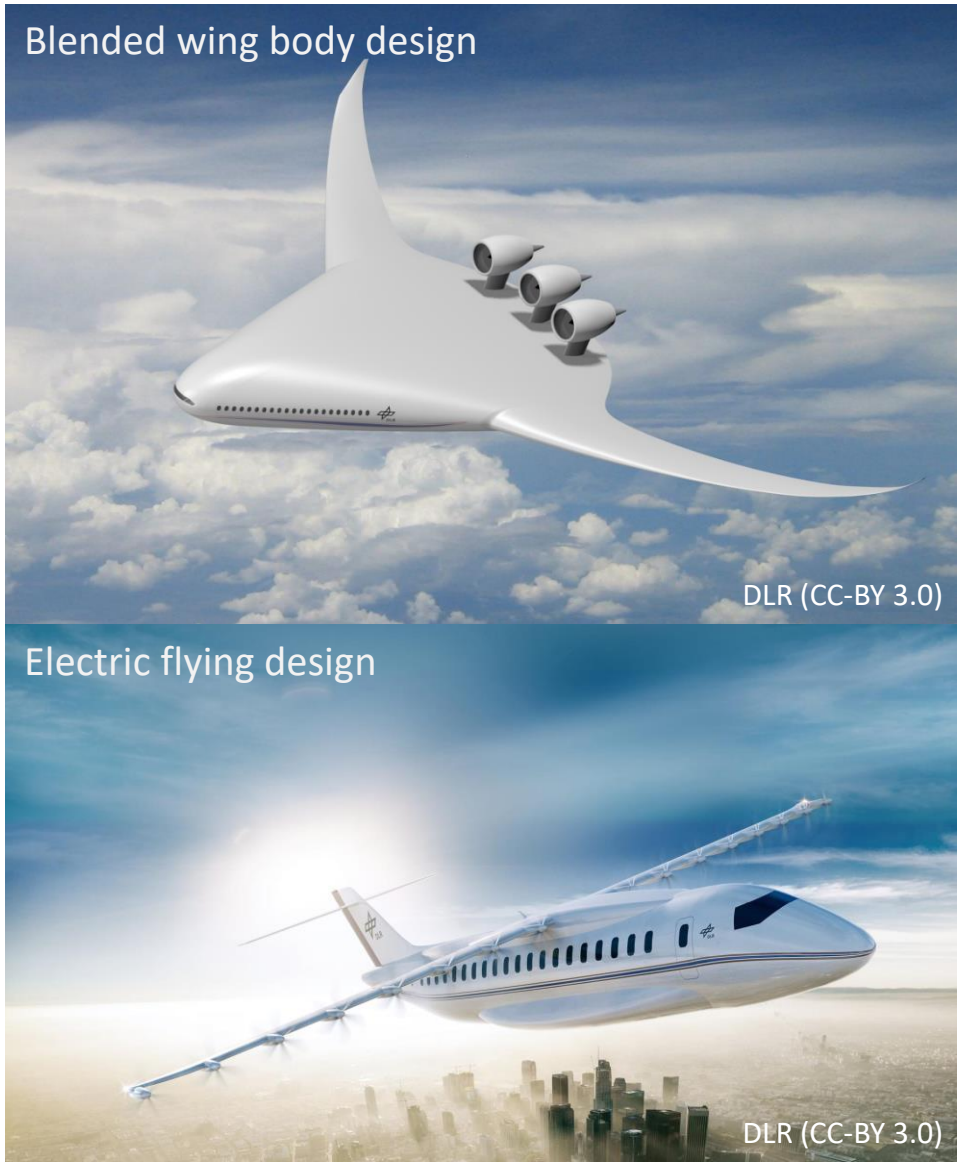
High-fidelity CFD methods indispensable

- Flight characteristics dominated by non-linear effects
- Reliable insight to new aircraft technologies
- High-fidelity CFD simulation of aircraft aerodynamics

Efficient linear system solving important

- CFD requires solving of large linear equation systems
- Linear systems solving makes up majority of time

Further improvement of simulation capabilities, computational efficiency and scalability necessary.



DLR CFD Code Strategy TAU and CODA

Next Generation CFD Solver for European Aircraft Industry

DLR TAU Code – CFD Solver for external aerodynamics

- 2nd Order Finite Volume method for unstructured grids & compressible flows
- Production code in European aircraft industry, research organizations and academia

Next Generation Solver Flucs

- Start from scratch at DLR
- New code design
- Actively started mid 2012
- Supported by several prototype codes

Since 2018 cooperation between DLR, Airbus and ONERA: Flucs → CODA

Next Generation Solver CODA

Goal: Secure Quality of CFD for Virtual Aircraft Requirements

New code design started from scratch

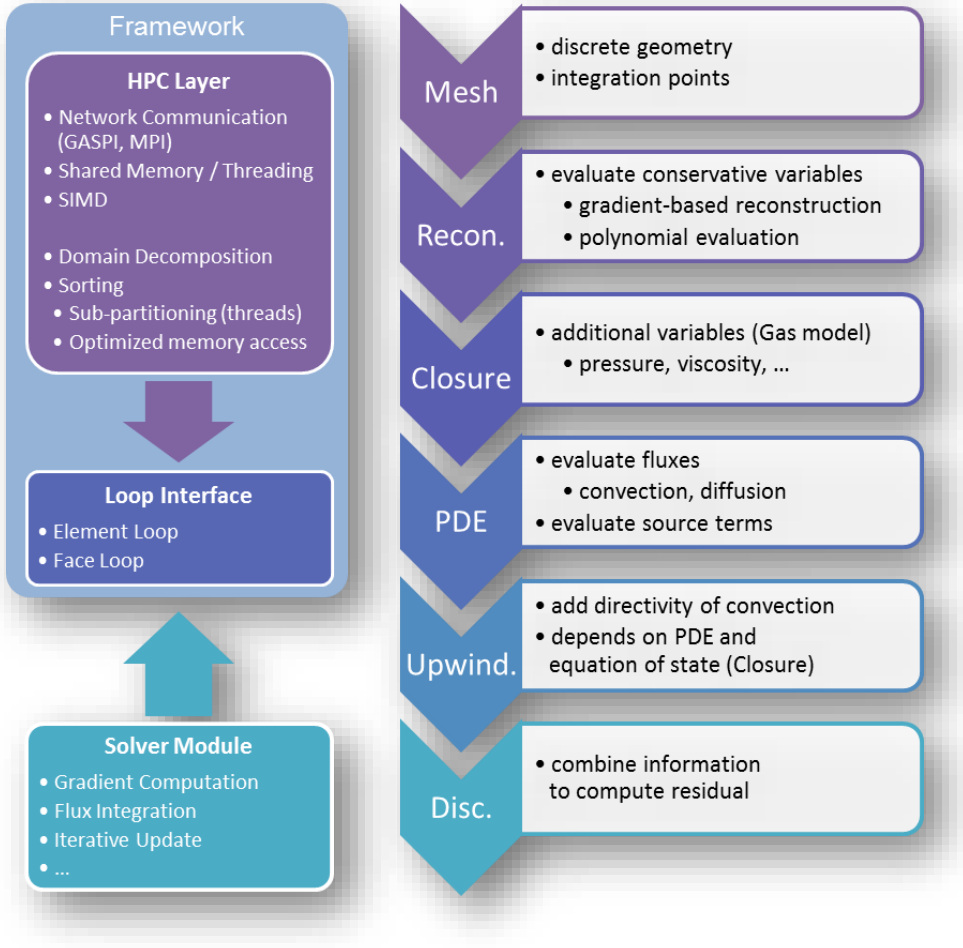
- C++ 11 (now C++ 17) + Python interface
- Finite Volume & HO Discontinous Galerkin & zonal coupling
- Focus on algorithmic efficiency: use of strong implicit solvers
- Seamless integration into multidisciplinary simulation

Focus on HPC and Scalability

- MPI (or GASPI) + OpenMP (2 level parallelization)
- 2-level partitioning: partition among processes + sub-partition among threads, thread sync avoided wherever possible
- Supports overlap of communication & computation

Spliss: Sparse Linear Systems Solver

- Linear systems solving for implicit methods
- Focus on high efficiency and flexibility for CFD



Key Requirements for Spliss by CFD Solvers (e.g. CODA, TRACE)

Why Yet Another Linear Algebra Package?

Sparse matrices

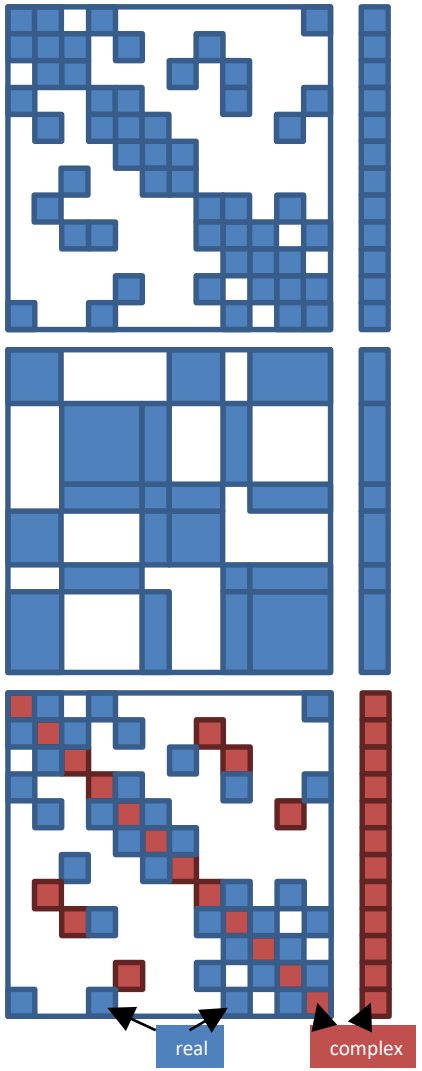
- Dense blocks with a fixed block size, e.g. 5x5, 6x6 … (RANS, RANS-SAneg)
- Variable block sizes (mixed-order DG[1])
- Mixed data types, e.g. real and complex (time-spectral, e.g. LFD[2], HB[3])

Solver

- Different components should be combinable (as preconditioner)
- Robust methods for stiff CFD problems:
  - Direct inversion of (generalized) diagonal blocks
  - Jacobi, Gauss-Seidel, GMRES, linear multigrid, …

Efficient parallelization for HPC

- Distributed memory (GASPI, MPI)
- Shared memory (Threading)
- Vector instructions (SIMD)
- Accelerators (GPU)



1) Discontinous Galerkin method | 2) Linear frequency domain | 3) Harmonic balance

Main Benefits of Spliss for CFD Solvers (e.g. CODA, TRACE, HYDRA)

Tailored to CFD applications but independent of the particular CFD Solver

Flexibility

- Sparse matrices with fixed or variable block size
- Mixed data types
- Different solver components are combinable (as preconditioner)
- Numerical independence from used parallelization

Full access for CFD application

HPC Efficiency

- Access to current and future HPC systems
- Scalable parallelization (hybrid GASPI/MPI + threading)
- Efficient computation (SIMD vectorization)
- Utilize emerging technologies (GPU, vector machines/accelerators)

Hidden from the CFD application

Demonstration of CODA with Spliss

- For implicit methods linear systems solving in CODA is done by Spliss
- Typically 80-95% of iteration time spent in linear solver (Spliss)
- Results may be biased by effects in Spliss and vice versa*
- Two use cases:
  - Use Case 1: HPC scalability on CARA (main DLR cluster)
  - Use Case 2: Seamless GPU usage on GPU test cluster

* Results show the minimum performance and scalability of CODA and Spliss since performance degrading effects accumulate. CODA and Spliss may achieve better performance and scalability individually.

Use Case 1: HPC Scalability

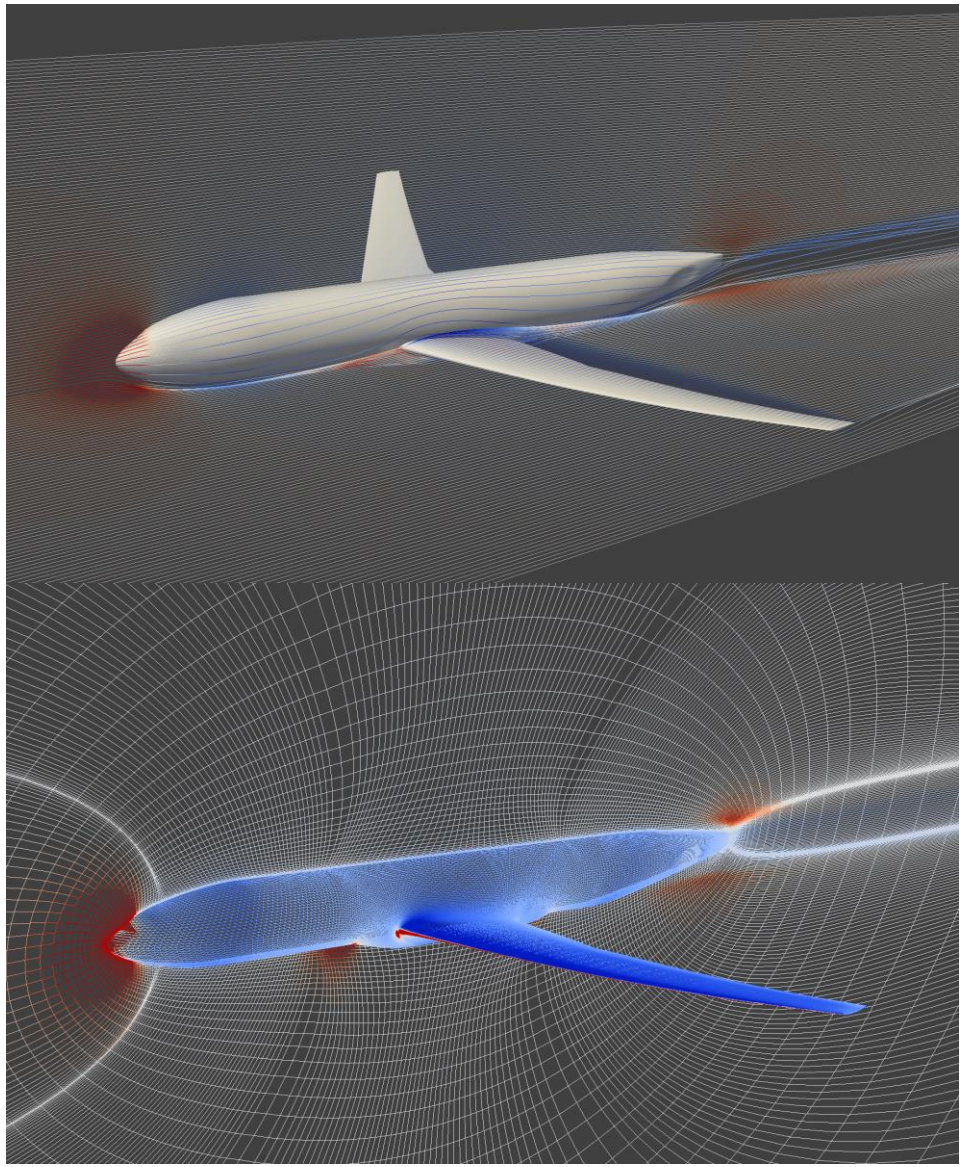
External Aerodynamics Simulation on CARA

Airflow for steady forward flight at subsonic speed

- Reynolds-averaged Navier-Stokes equations (RANS) with Spalart-Allmaras turbulence model (SA-neg)
- Re = 5e6 | Mach = 0.2 | angle of attack = 2.5°
- Finite volume spatial discretization with implicit Euler time discretization (CFL number: initial 10 and growing)
- Linear systems solving via Spliss (Block Jacobi)

Small mesh used for strong scalability analysis

- NASA Common Research Model: wing body configuration
- Unstructured prism mesh with 5M points (10M prisms)
- Production meshes >20x larger with better scaling



CARA

Cluster for Advanced Research in Aerospace

Overview

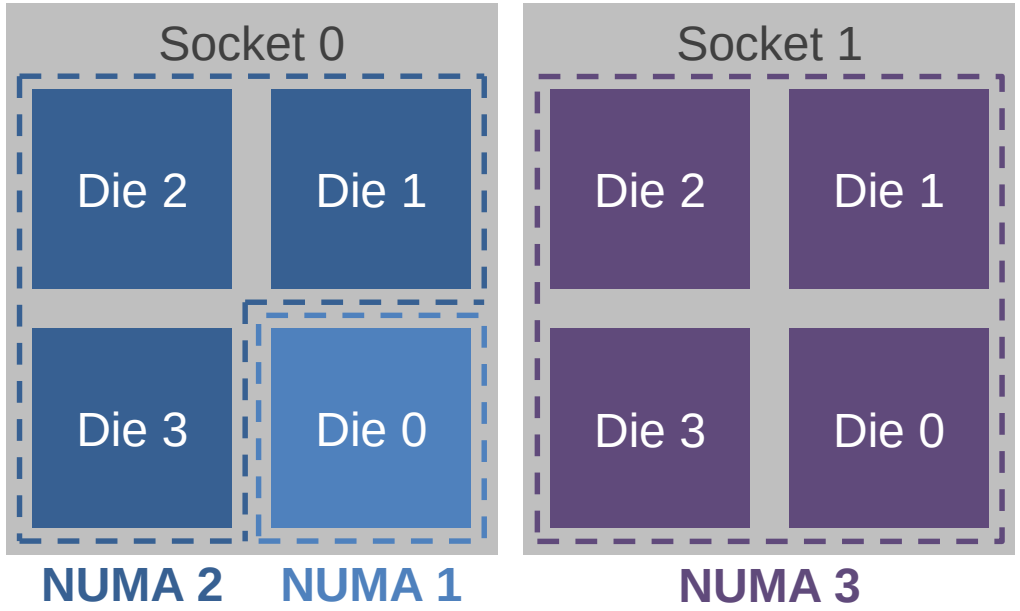
- DLR's main HPC cluster from NEC
- Ranked 221 in 11/2019: 1.7 TFlop/s (2.6 Tflop/s peak)

Compute nodes

- 2280 compute nodes
- 2x AMD Epyc 7601 (32 cores @ 2.2 GHz) per node
- 145,920 cores

Complex NUMAness

- 8 NUMA domains
- 3 NUMA distance (on die, on socket, 2nd socket)
- 4 cores per die share L3



Evaluation of Hybrid Setups

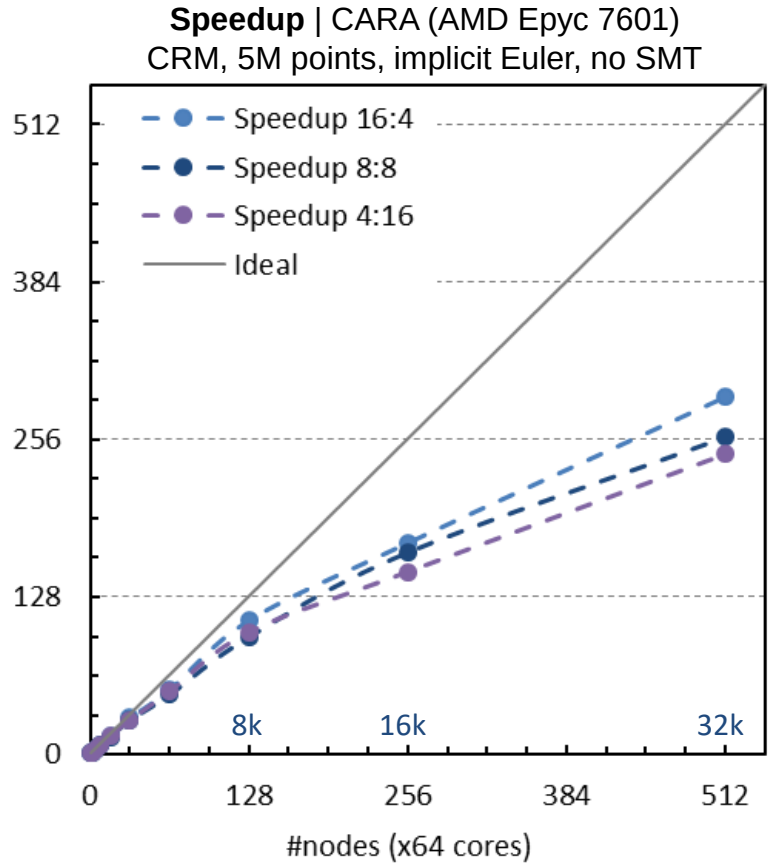
Scalability Evaluation without Simultaneous Multithreading (SMT)

Expectation (early 2020, before any tests)

- In general, more threads per process = better scalability (less impact by MPI global communication)
- But, sensitivity to NUMA domains
- Thus, the setup 8:8 (8 MPI ranks with 8 OpenMP threads each) should deliver best scaling
- L3 cache should have additional impact

Observation

- 90% efficiency at 4k cores, 59% efficiency at 32k cores
- Very good scaling for small mesh (300 elements/thread at 32k)
- Setup 16:4 scales better than 8:8

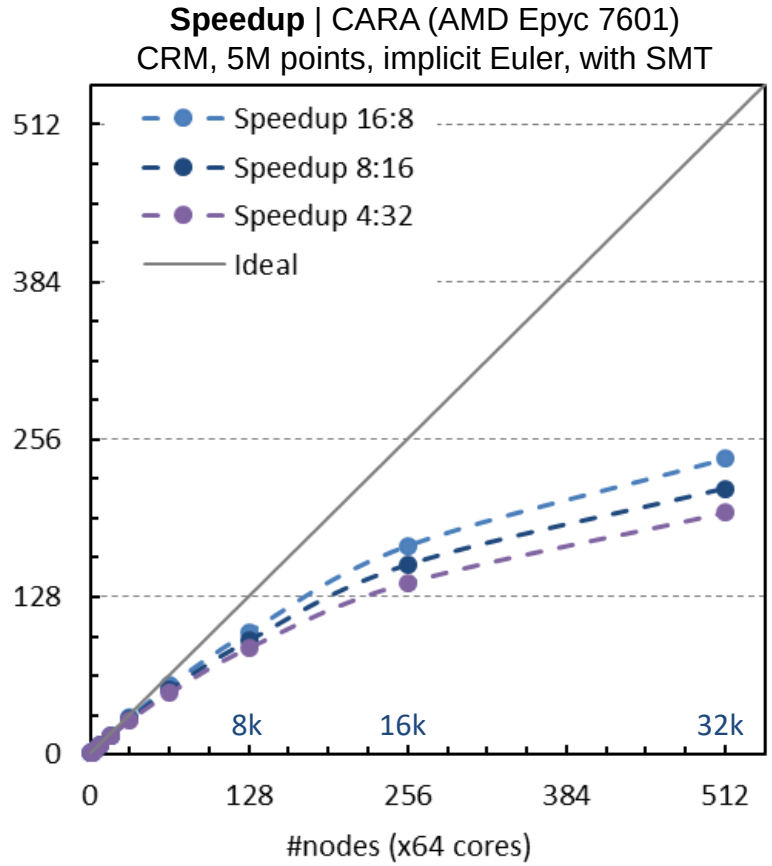


Evaluation of Hybrid Setups (2)

Scalability Evaluation with Simultaneous Multithreading (SMT)

Observation

- 88% efficiency at 4k cores, 47% efficiency at 32k cores
- Very good scaling for small mesh (150 elements/thread at 32k)
- Lower scaling due to half computational load per threads
- 47% at extreme case: 150 elements/thread is very little

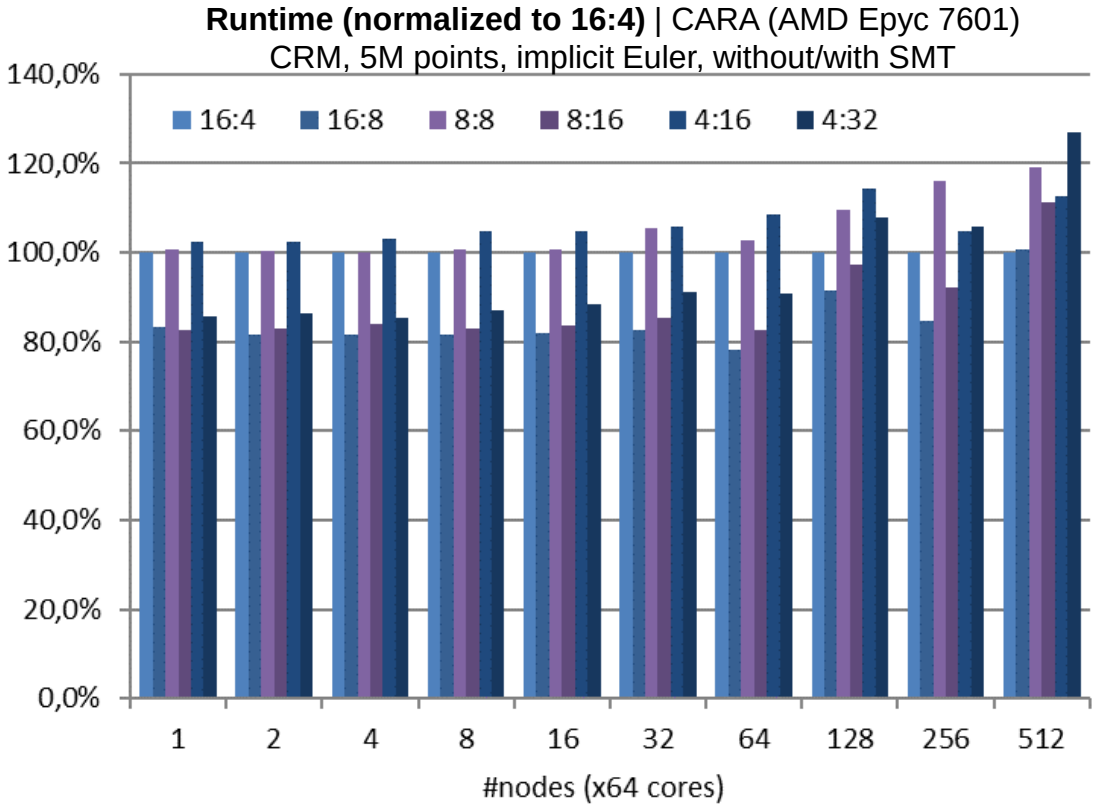


Evaluation of Hybrid Setups (3)

Comparison without vs. with Simultaneous Multithreading (SMT)

Observation

- Setups with SMT provide significantly better performance: 15-20%
- Improved sub-partitioning (threads) is expected to provide better SMT scaling*



* The shown measurements are recorded with a 2020 version of CODA. In current versions the sub-partitioning among threads has been improved.

Evaluation of Hybrid Setups on Different Architectures

Comparison of AMD Epyc and Intel Cascade Lake

Evaluation of processor architectures of AMD Epyc and Intel Cascade Lake

- AMD Epyc 7601: 2x 32 Cores, 360W, AVX-2
  → 2 nodes for comparable power
- Xeon Platinum 9242: 2x 48 Cores, 700W, AVX-512

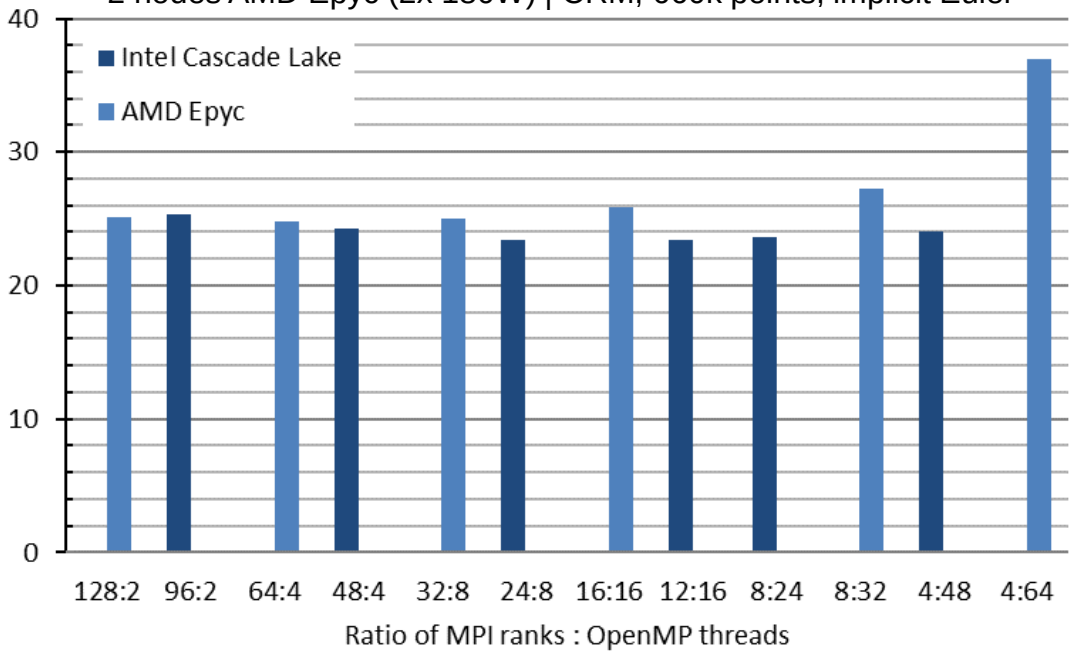
Comparing code performance on both architectures

- Performance very similar
- So far, no effect from the 512bit vector registers

Evaluation of hybrid setups

- AMD Epyc becomes less efficient with more threads
- Good hybrid parallelization on Intel Cascade Lake
- CODA shows good thread parallelization (4 thread optimum only on AMD), hybrid setup adjustable

Runtime (in seconds) | 1 node Intel Cascade Lake (350W) vs. 2 nodes AMD Epyc (2x 180W) | CRM, 660k points, implicit Euler



Network and System Load Interference

Random Node Placement / Normal System Load

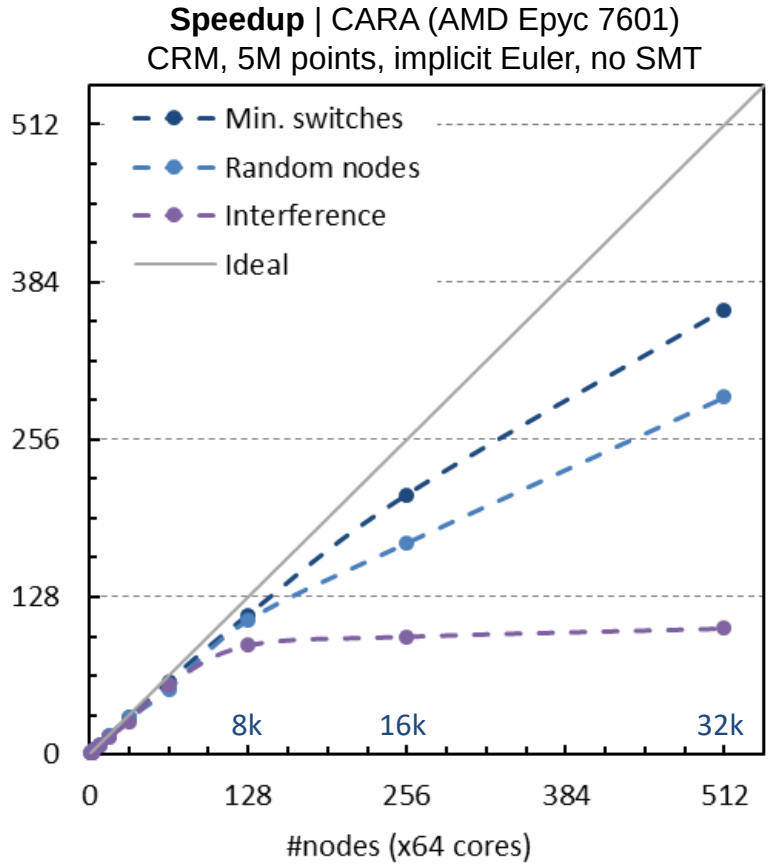
- CARA is a production system with according usage and load
- Above measurements are all in “default” mode without any specific benchmarking setup

Improved node placement

- Improved placement with minimum network switches
- Improved but not ideal placement (unfeasible in production mode)
- 93% (vs. 90%) efficiency at 4k cores
- 71% (vs. 59%) efficiency at 32k cores

System load interference

- Large scale runs are more vulnerable to network load (higher communication-to-computation ratio)
- Heavy system/network load can drastically impact performance

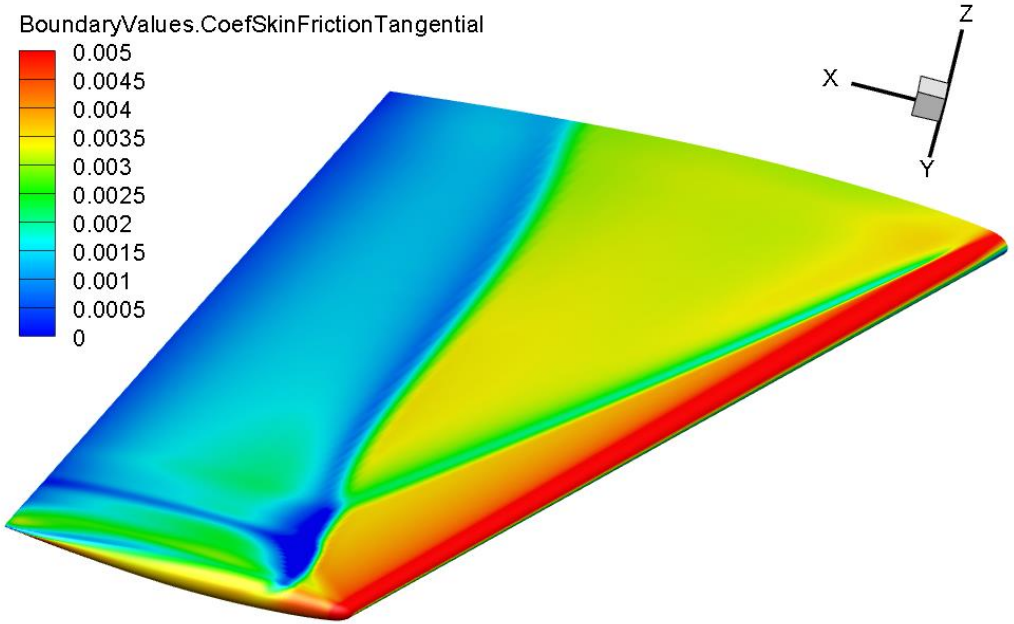


Use Case 2: Seamless GPU Usage

External Aerodynamics Simulation on GPU Test Cluster

Airflow for steady forward flight at subsonic speed

- Reynolds-averaged Navier-Stokes equations (RANS) with Spalart-Allmaras turbulence model (SA-neg)
- Re = 14.6e6 | Mach = 0.86 | angle of attack = 3.08°
- Finite volume spatial discretization with implicit Euler time discretization (CFL number: initial 5.0 and growing)
- Linear systems solving via Spliss on GPUs



Small mesh used for strong scalability analysis

- NASA 3D Onera M6 test case
- Unstructured prism + hexa mesh with 1.1M elements

Usage of GPUs without any modification to the CODA source code or installation

Evaluation CPU vs. GPU

System – GPU test cluster

- 2x Intel Xeon 6230 (20 cores @ 2.1 Ghz) per node
- 4x Nvidea Tesla V100 per node

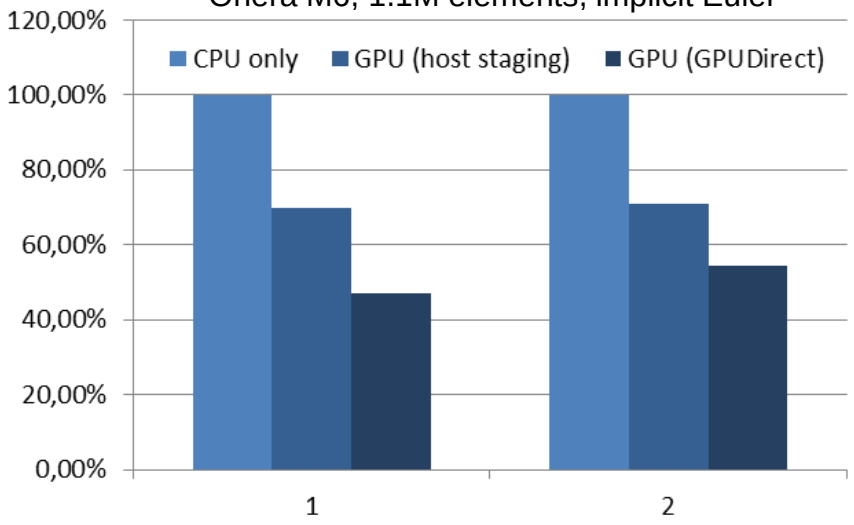
Setup

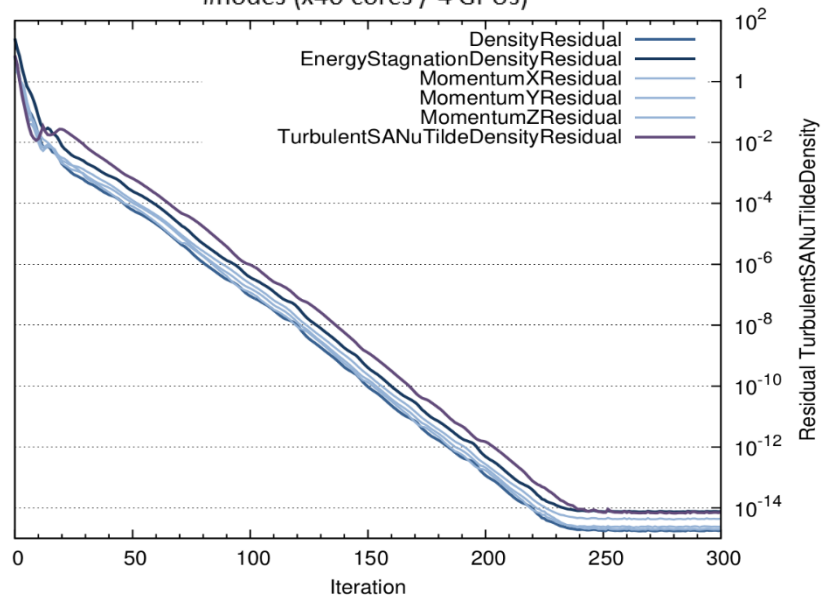
- Load Spliss module with GPU support
- No modifications to CODA source code or installation
- GPU (host staging): Initial version of Spliss
- GPU (GPUDirect): optimized Spliss version using CUDA-aware MPI and GPUDirect to allow direct communication from GPU to GPU

Observations

- Numerical stability and correctness for GPU version validated
- Speedup 1.8 to 2.1 for node-wise comparison (work in progress)
- Easy testing of CODA on GPUs

Runtime (normalized to CPU) | Intel Xeon vs. Nvidea V100
Onera M6, 1.1M elements, implicit Euler





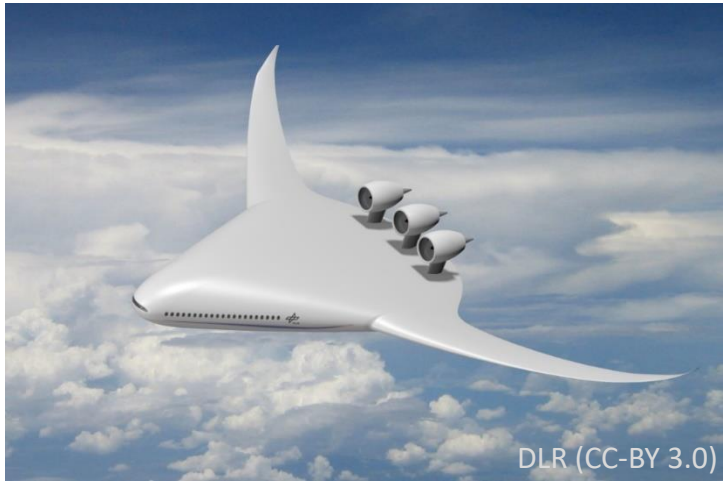
Summary

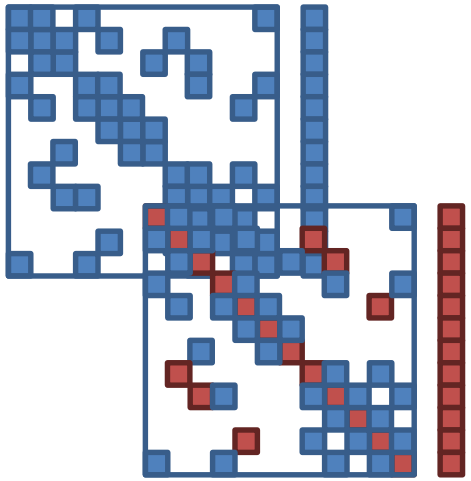
- Numerical simulation is important for future aircraft design
- CODA is a new CFD solver with strong focus on HPC and scalability
- CODA shows very good strong scaling on AMD Epyc cluster
- Best (hybrid) setup is individual for every system: flexibility is important
- Scalability is significantly influenced by node placement and network load

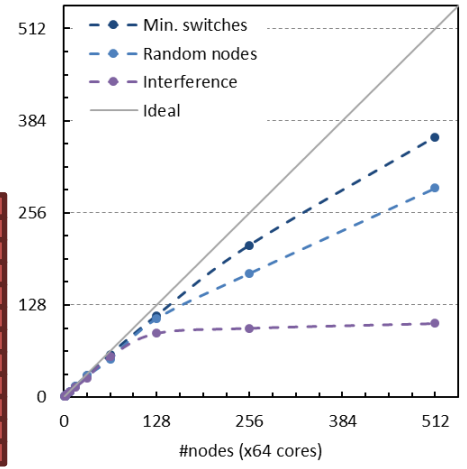
EXCELLERAT
www.excellerat.eu

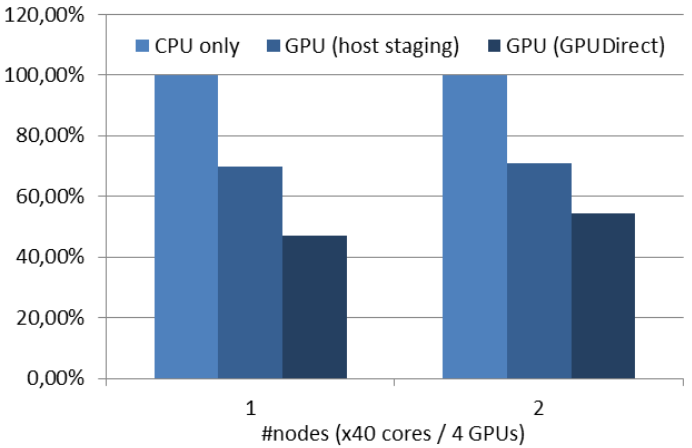
KonSENS
CODA and Spliss Development

DLR Aerodynamics and Flow Technology
DLR Propulsion Technology
DLR Software Methods for Product Virtualization
DLR Software Technology
Airbus
ONERA









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.

DLR