preCICE and Exascale

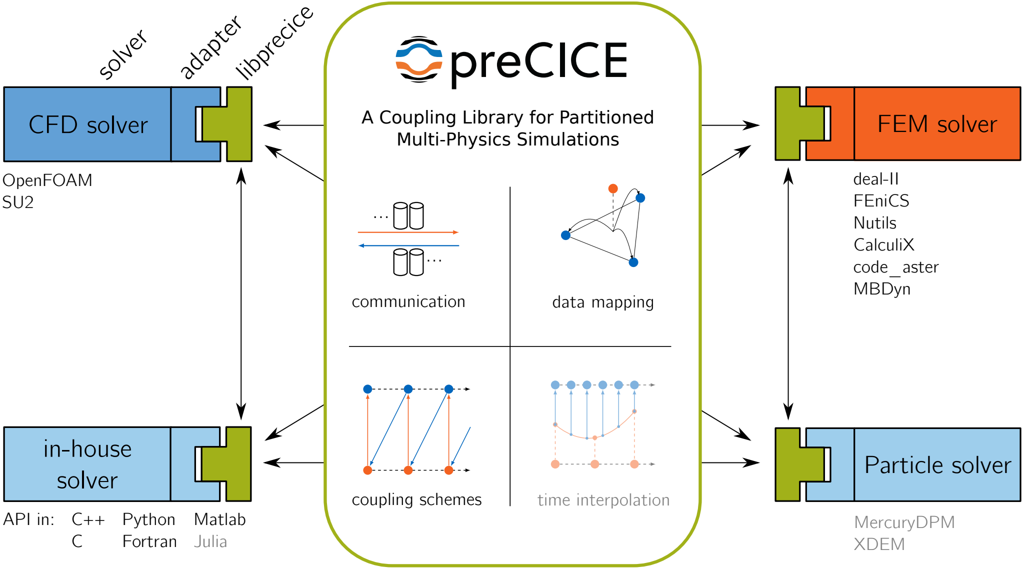
Benjamin Uekermann et al.

University of Stuttgart

Content

Core Developers

Overview



solver
adapter
libprecice
CFD solver
OpenFOAM
SU2
in-house solver
API in: C++ Python Matlab
C Fortran Julia
preCICE
A Coupling Library for Partitioned Multi-Physics Simulations
communication
data mapping
coupling schemes
time interpolation
FEM solver
deal-II
FEniCS
Nutils
CalculiX
code_aster
MBDyn
Particle solver
MercuryDPM
XDEM

Demo

Time to Brag

- Extensive user documentation (when converted to PDF, more than 250 pages)
- Ready-to-use adapters for OpenFOAM, FEniCS, deal.II, CalculiX, Nutils, SU2, code_aster
- API in C++, C, Fortran, Python, Matlab, Julia
- Vivid community exchange in chat and forum
- Used by more than 100 research groups (academia, research centers, industry)
- xSDK member, pre-installed on more and more supercomputers
- CI with tests on all levels (unit, integration, bindings, adapters, system)
- Details? `https://precice.org/` and
  Chourdakis et al. preCICE v2: A Sustainable and User-Friendly Coupling Library (2021)

SPPEXA

- German Priority Program *Software for Exascale Computing*
- 2013–2019
- 17 projects, in average 4-5 PIs and 1 PhD student per PI
- One of the 17 projects: ExaFSA – *Exascale Simulation of Fluid-Structure-Acoustics Interactions*
- Pushing preCICE towards exascale was a substantial part of this project
- Disclaimer: not every part of preCICE is ready for exascale yet
- Details? `http://www.sppexa.de/` and
  Bungartz et al. Partitioned FSA on Distributed Data (2016)

Meshes in preCICE

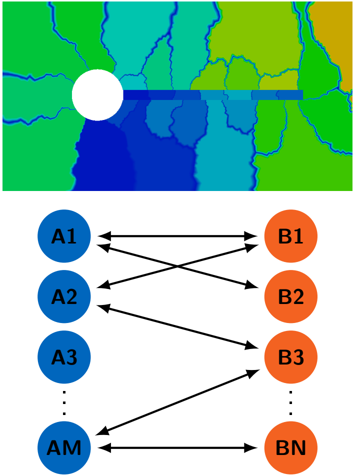
Static Coupling Meshes

- Coupling meshes are defined once (during initialization)
- No problem for many coupled problems: FSI with fluid solvers in ALE framework (*moving meshes*), particle methods with background meshes, ...
- ... but for some: immersed boundary methods, solvers with dynamically adaptive meshes, particle methods without background meshes, ...
- We are currently working on dynamic meshes in preCICE

Impact on Efficiency and Scalability

- Compute data mapping and communication pattern during initialization
- Initialization can be expensive, but should not be too expensive
- Per timestep/iteration negligible coupling effort and (thus) highly scalable (sometimes even embarrassingly parallel)

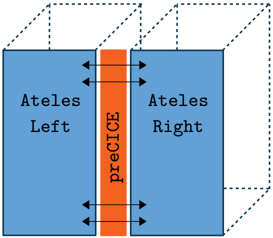
Peer-to-Peer Approach

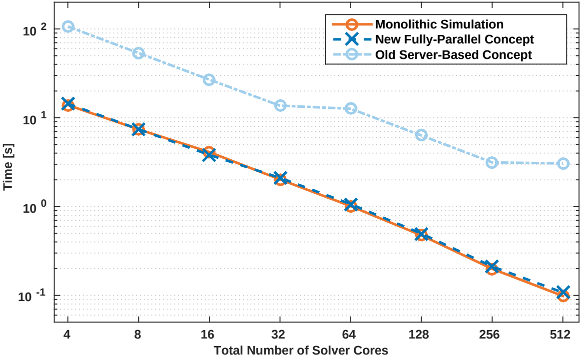


- No central server-like entity
- Only communication between ranks that share a part of the coupling mesh
- Coupling numerics computed directly on solver processors
- No scaling issues (per timestep)

Time per Timestep

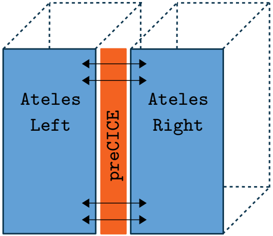
- Travelling density pulse (Euler equations) through artificial coupling interface
- DG solver Ateles (U Siegen), $7.1 \cdot 10^6$ dofs
- Nearest neighbor mapping and communication
- *Uekermann. FSI on Massively Parallel Systems (2016)*
- SuperMUC Thin Nodes

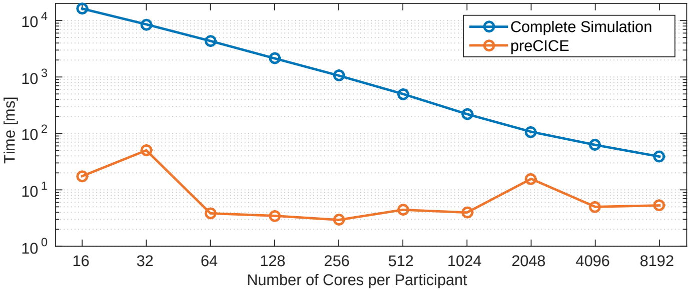




Time per Timestep

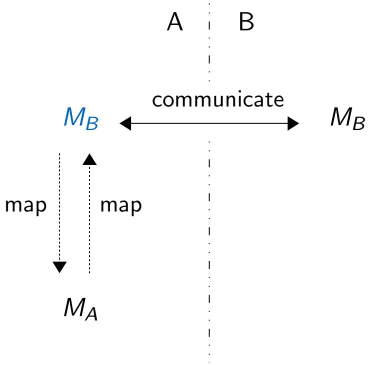
- Travelling density pulse (Euler equations) through artificial coupling interface
- DG solver Ateles (U Siegen), $5.7 \cdot 10^7$ dofs
- Nearest neighbor mapping and communication
- *Uekermann. FSI on Massively Parallel Systems (2016)*
- SuperMUC Thin Nodes



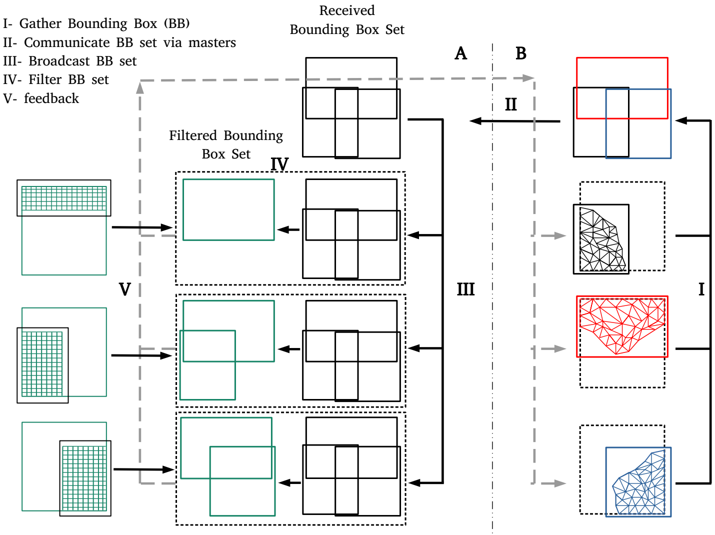


Initialization

- Example: two participants A and B
- Each one has a mesh: M_A and M_B
- Both meshes are decomposed, user implicitly defindes decomposition
- To compute data mapping and communication pattern during initialization, preCICE sends M_B from B to A and re-partitions it on A

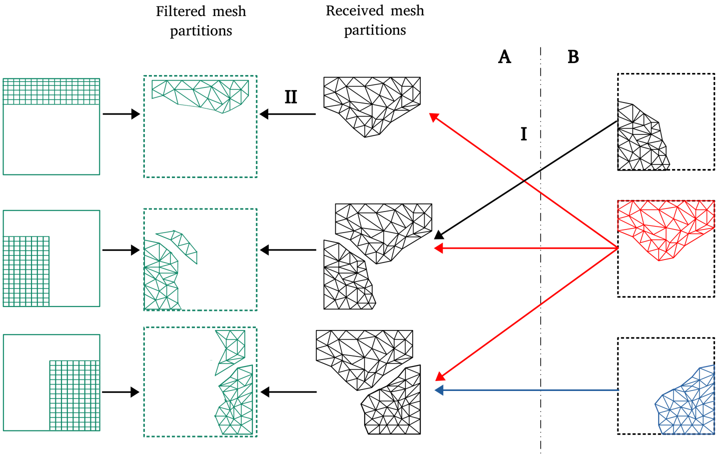


Two-Level Initialization – Level 1



Totounferoush et al. Efficient and Scalable Initialization . . . (2021)

Two-Level Initialization – Level 2

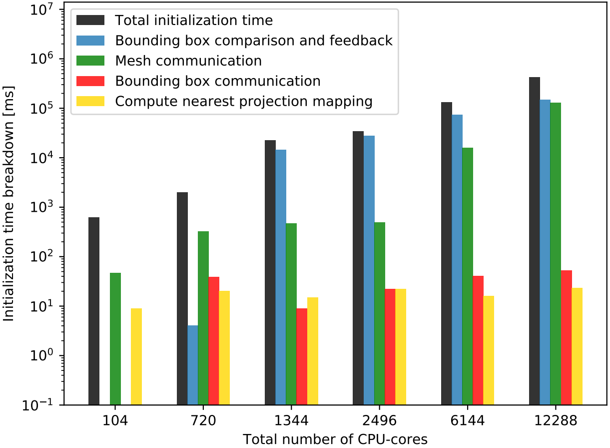


I- Communicate mesh to connected remote ranks
II- Filter the received mesh according to the mapping scheme

Totounferoush et al. Efficient and Scalable Initialization . . . (2021)

Two-Level Initialization – Weak Scalability

Turbine blade test geometry, approx. 1000 vertices per MPI rank of communicated mesh



Totounferoush et al. Efficient and Scalable Initialization . . . (2021)

Ongoing and Future Developments

Originally: low-order, mesh-based surface coupling, e.g. fluid-structure interaction. Now ...

- Data mapping for mixed-dimensional problems (geometric multi-scale)
- Solver-based (higher-order) data mapping
- Partition of Unity RBF data mapping
- Support for dynamic-adaptive meshes (re-initialization)
- Waveform relaxation for multi-rate coupling
- Volume coupling (mapping, communication & load balancing)
- Mesh-particle coupling
- Macro-micro coupling

Ongoing and Future Developments

Originally: low-order, mesh-based surface coupling, e.g. fluid-structure interaction. Now ...

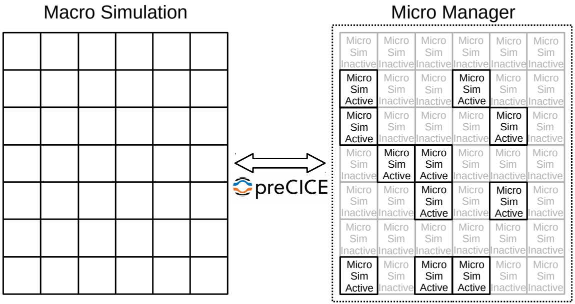
- Data mapping for mixed-dimensional problems (geometric multi-scale)
- Solver-based (higher-order) data mapping
- Partition of Unity RBF data mapping
- Support for dynamic-adaptive meshes (re-initialization)
- Waveform relaxation for multi-rate coupling
- Volume coupling (mapping, communication & load balancing)
- Mesh-particle coupling
- **Macro-micro coupling**

Macro-Micro Coupling

Within Cluster of Excellence EXC 2075 Simulation Technology (Stuttgart)

Target application

- Same time-scale
- Iterative (implicit) coupling per timestep
- Adaptive usage of micro simulations → load balancing



Examples in Project

- Heat conduction: FE^2 (Nutils or FEniCS)
- Porous media flow: REV / Darcy scale (OpenFOAM or DuMuX) + micro structure (Nutils or DuMuX)
- Muscle simulation: 3D CSM and EMG (LS-DYNA + OpenDiHu) + 1D fibers EMG (OpenDiHu)

Summary

- Library approach ⇒ minimally-invasive integration
- Peer-to-peer approach ⇒ scalability
- Black-box approach ⇒ minimally-invasive integration, flexibility
- High-level API ⇒ flexibility
- Ready-to-Use adapters ⇒ low entry barrier
- Community project ⇒ sustainability

Links

- 3rd Workshop: Feb 21-24, 2022 (online)
- `https://precice.org/`
- `https://github.com/precice/`
- `https://precice.discourse.group`
- `@preCICE_org`
- `https://www.youtube.com/c/preCICECoupling`
- `benjamin.uekermann@ipvs.uni-stuttgart.de`

Black-Box Data Mapping

- Only operate on clouds of vertices
- Nearest-neighbor mapping (1st order), nearest-projection mapping (2nd order)
- RBF interpolation mapping: $\sum_{i=1}^{n} \gamma_i \cdot \phi(\|x - x_i\|_2) + q(x)$ with RBF ϕ and global linear polynomial q
- ϕ either global (e.g., Thin Plate Splines, expensive, less tuning) or local (e.g., Gaussian, better choice for larger problems)

$$\left(\begin{array}{c|c} 0 & P^T \\ \hline P & M \end{array}\right) \left(\begin{array}{c} \beta \\ \hline \gamma \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline V \end{array}\right)$$

- Parallel computation: matrix decomposed row-wise, solved with GMRES (PETSc)

Coupled Fluid Solver in Python

```python
import precice
interface = precice.Interface("FluidSolver", "precice-config.xml")

mesh_id = interface.get_mesh_id("Fluid-Mesh")
force_id = interface.get_data_id("Force", mesh_id)
positions = ... #define interface mesh, 2D array with shape (n, dim)
vertex_ids = interface.set_mesh_vertices(mesh_id, positions)

interface.initialize()

while interface.is_coupling_ongoing(): # main time loop

    displacements = interface.read_block_vector_data(displ_id, vertex_ids)
    u = solve_time_step(displacements) # returns new solution
    forces = compute_forces(u) # returns 2D array with shape (n, dim)
    interface.write_block_vector_data(force_id, vertex_ids, forces)

    interface.advance()
```

Many details omitted. Follow preCICE Course to learn the API step-by-step.