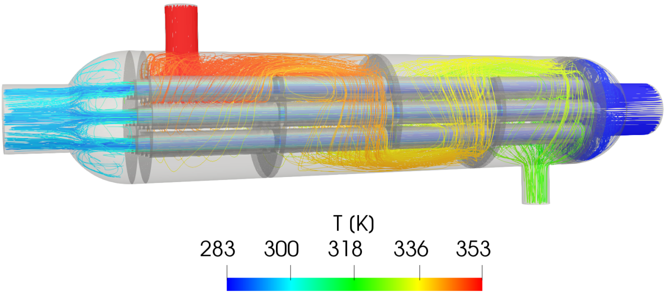
preCICE

A Coupling Library for Partitioned Multi-Physics Simulations

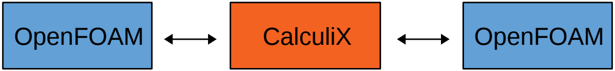
1. What is preCICE?
2. How to get started?
3. How can I couple my own code?

1. What is preCICE?

Example: Shell-And-Tube Heat Exchanger



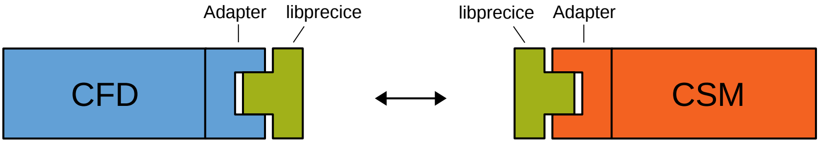
- Partitioned coupling: Usage of three independent solvers
- Reuse of existing solvers


OpenFOAM

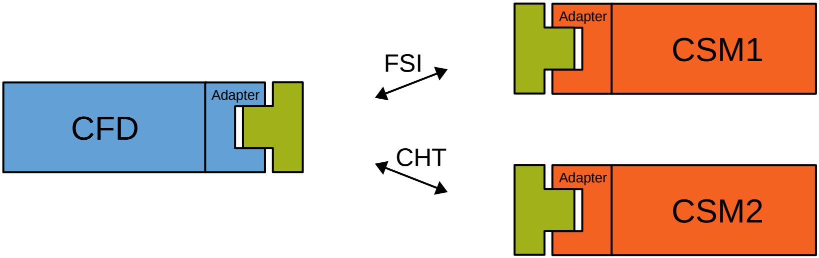
CalculiX

OpenFOAM

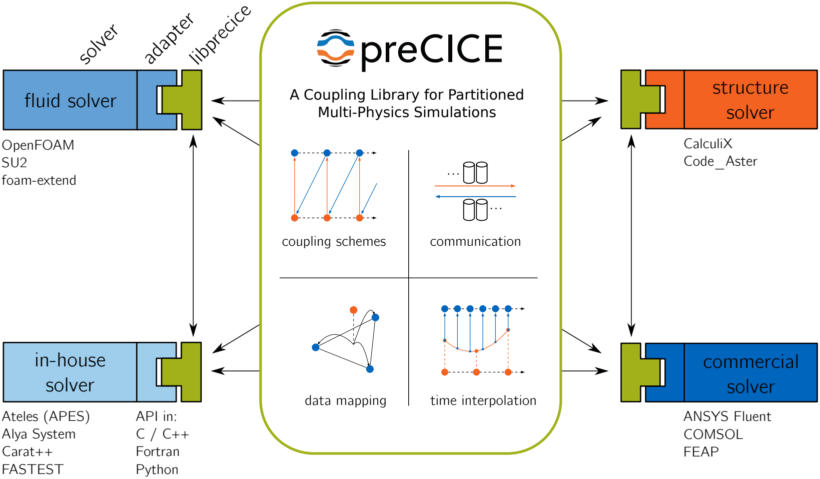
preCICE – A Plug-and-Play Coupling Library



preCICE – A Plug-and-Play Coupling Library



preCICE – A Plug-and-Play Coupling Library



solver

adapter

libprecice

fluid solver

OpenFOAM
SU2
foam-extend

in-house solver

Ateles (APES)
Alya System
Carat++
FASTEST

API in:
C / C++
Fortran
Python

preCICE

A Coupling Library for Partitioned Multi-Physics Simulations

coupling schemes

communication

data mapping

time interpolation

structure solver

CalculiX
Code_Aster

commercial solver

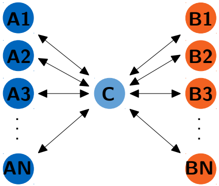
ANSYS Fluent
COMSOL
FEAP

USPs

1. Scalability
2. Robust quasi-Newton coupling
3. Coupling of arbitrary many components
   (*arbitrary many* = *more than two*)
4. Minimally-invasive coupling
5. Open-source, community

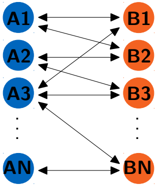
USP 1: Scalability

Server-Based Concept



- Complete communication through central server process
- Interface computations on server (in sequential)
- ⇒ Coupling becomes bottleneck for overall simulation already on moderate parallel systems

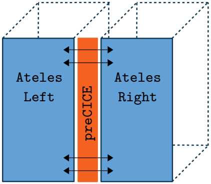
Our Peer-To-Peer Concept

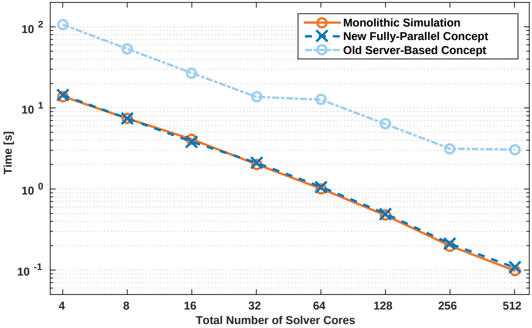


- No central entity
- ⇒ Easier to handle (user does not need to care about server)
- ⇒ No scaling issues

USP 1: Scalability

- 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

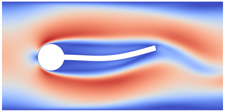




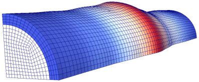
USP 2: Quasi-Newton Coupling

Coupled problem: $F : d \mapsto f,\ S : f \mapsto d \quad \rightsquigarrow \quad (S \circ F)(d) \overset{!}{=} d$

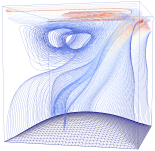
FSI3



3D-Tube



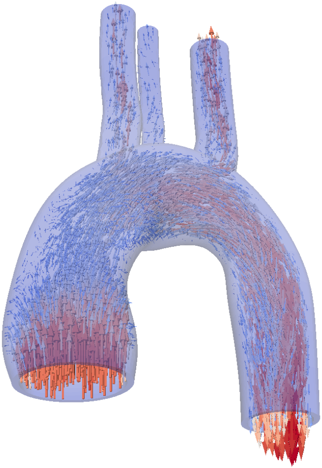
Driven Cavity



Mean Iterations	Aitken	Quasi-Newton
FSI3	17.0	3.3
3D-Tube	Div.	7.5
Driven Cavity	7.4	2.0

USP 2: Quasi-Newton Coupling

- Quasi-Newton can even handle biomedical applications, such as an Aortic bloodflow
- Stable coupling (no added-mass instabilities)
- Six times less iterations than Aitken



- Joint work with Juan-Carlos Cajas (Barcelona Supercomputing Center)
- Geometry by Jordi Martorell

Contributors

Miriam Mehl
U Stuttgart

Florian Lindner
U Stuttgart

Amin Totounferoush
U Stuttgart

Alexander Rusch
ETH Zürich

Hans Bungartz
TUM

Benjamin Rüth
TUM

Gerasimos Chourdakis
TUM

Frédéric Simonis
TUM

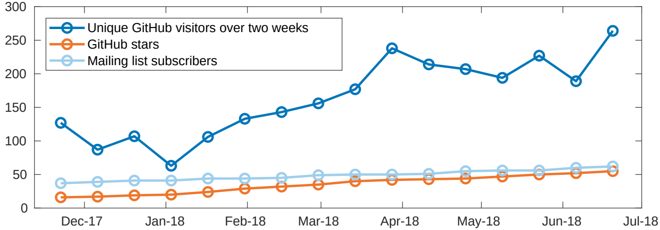
Benjamin Uekermann
TUM

Previous and minor contributors:

- Bernhard Gatzhammer, Klaudius Scheufele, Lucia Cheung, Alexander Shukaev, Peter Vollmer, Georg Abrams, Alex Trujillo, . . .

Users

- Dr.-Ing. Fettah Aldudak et al., LSM, University of Siegen, Germany
- Davide Cinquegrana, PhD et al., SCpA, CIRA, Italy
- Kyle Davis et al., Cardiothoracic Surgery, University of the Free State, South Africa
- Dr. Lee Margetts et al., Mechanical and Aeronautical Engineering, University of Manchester
- Vinh-Tan Nguyen, PhD et al., A*STAR, Singapore
- Dr. Thorsten Reimann et al., SC, TU Darmstadt, Germany
- Prof. Sabine Roller et al., STS, University of Siegen, Germany
- Prof. Michael Schäfer et al., FNB, TU Darmstadt, Germany
- Dr. Qing Xiao et al., CFD & FSI-RG, University of Strathclyde, UK
- Prof. Alexander van Zuijlen et al., Aerospace Engineering, TU Delft
- Global Research for Safety (GRS), Garching (upcoming)
- . . .

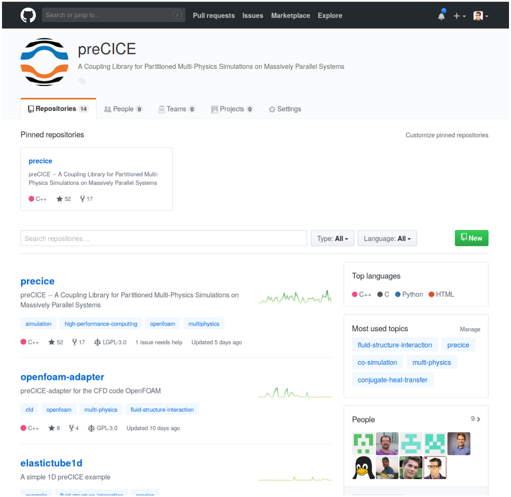


2. How to get started?

Infrastructure

We are on GitHub: `https://github.com/precice`

- LGPL3 license
- User documentation in the wiki



preCICE

A Coupling Library for Partitioned Multi-Physics Simulations on Massively Parallel Systems

Pinned repositories

precice
preCICE -- A Coupling Library for Partitioned Multi-Physics Simulations on Massively Parallel Systems
● C++ ★ 52 17

precice
preCICE -- A Coupling Library for Partitioned Multi-Physics Simulations on Massively Parallel Systems
simulation high-performance-computing openfoam multiphysics
● C++ ★ 52 17 LGPL-3.0 1 issue needs help Updated 5 days ago

openfoam-adapter
preCICE-adapter for the CFD code OpenFOAM
cfd openfoam multi-physics fluid-structure-interaction
● C++ ★ 8 4 GPL-3.0 Updated 10 days ago

elastictube1d
A simple 1D preCICE example
example fluid-structure-interaction precice

Top languages
● C++ ● C ● Python ● HTML

Most used topics
fluid-structure-interaction precice co-simulation multi-physics conjugate-heat-transfer

People 9

Building

Dependencies

- Eigen, Boost (version $\geq$ 1.60), libxml2
- Optional: PETSc, Python (incl. Numpy), MPI

The easy way

- Ubuntu 18.04: All dependencies available through distribution
- Ubuntu 16.04: All dependencies available except Boost

Still doable

- macOS
- Other Linux distributions

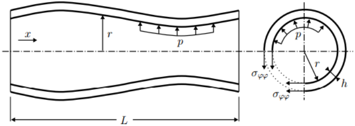
Experimental

- Conda, Docker, Debian package
- Windows

Tutorials

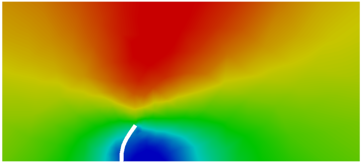
1D Elastic Tube

- Simple provided solvers
- Learn about API and configuration



Flexible beam

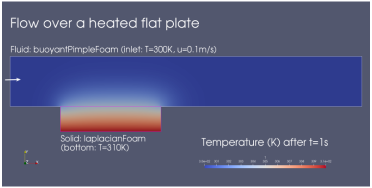
- Fluid-structure interaction
- Couple SU2 to CalculiX
- Learn about coupling schemes
- Also interactive version available in browser `http://run.coplon.de/`



Tutorials

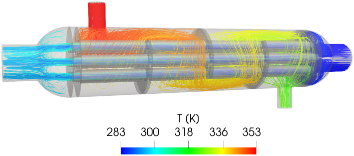
Flow over a Heated Plate

- Conjuagte-heat transfer
- Couple two OpenFOAM solvers
- Learn about OpenFOAM adapter

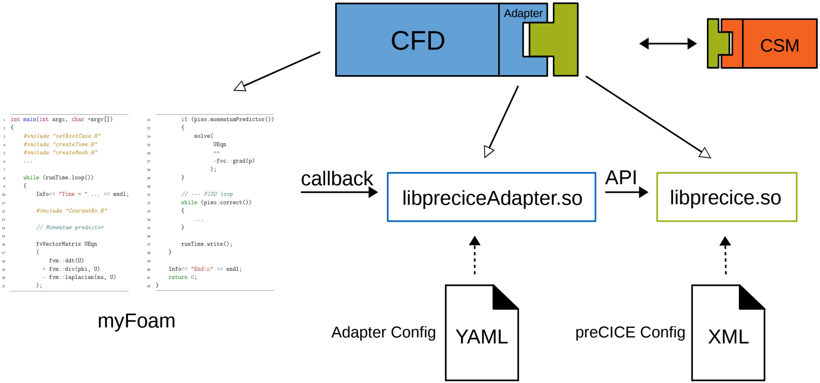

Flow over a heated flat plate
Fluid: buoyantPimpleFoam (inlet: T=300K, u=0.1m/s)
Solid: laplacianFoam
(bottom: T=310K)
Temperature (K) after t=1s
T
3.0e+02
301
302
303
304
305
306
307
308
309
3.1e+02
Y
Z
X

Heat exchanger

- Conjugate-heat transfer
- Couple two OpenFOAM instances with CalculiX
- Learn about multi coupling



The OpenFOAM Adapter



```
int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    ...

    while (runTime.loop())
    {
        Info<< "Time = " ... << endl;

        #include "CourantNo.H"

        // Momentum predictor

        fvVectorMatrix UEqn
        (
            fvm::ddt(U)
          + fvm::div(phi, U)
          - fvm::laplacian(nu, U)
        );
        if (piso.momentumPredictor())
        {
            solve(
                  UEqn
                  ==
                  -fvc::grad(p)
                 );
        }

        // --- PISO loop
        while (piso.correct())
        {
            ...
        }

        runTime.write();
    }

    Info<< "End\n" << endl;
    return 0;
}
```

Flow over a Heated Plate

Load adapter at runtime in `system/controlDict`:

```
functions
{
    preCICE_Adapter
    {
        type preciceAdapterFunctionObject;
        libs ("libpreciceAdapterFunctionObject.so");
    }
}
```

Define coupling boundary in `system/blockMeshDict`:

```
interface
{
    type wall;
    faces
    (
        (4 0 1 5)
    );
}
```

Flow over a Heated Plate

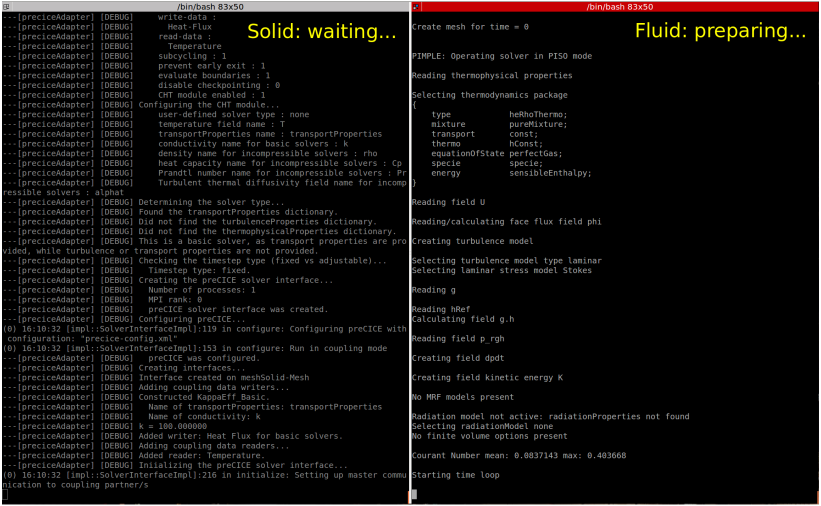
Configure adapter in `precice-adapter-config.yml`:

```yaml
participant: Fluid

precice-config-file: /path/to/precice-config.xml

interfaces:
- mesh: Fluid-Mesh
  patches: [interface]
  write-data: Temperature
  read-data: Heat-Flux
```

Flow over a Heated Plate



Solid: waiting...

```
---[preciceAdapter] [DEBUG]     write-data :
---[preciceAdapter] [DEBUG]       Heat-Flux
---[preciceAdapter] [DEBUG]     read-data :
---[preciceAdapter] [DEBUG]       Temperature
---[preciceAdapter] [DEBUG]     subcycling : 1
---[preciceAdapter] [DEBUG]     prevent early exit : 1
---[preciceAdapter] [DEBUG]     evaluate boundaries : 1
---[preciceAdapter] [DEBUG]     disable checkpointing : 0
---[preciceAdapter] [DEBUG]     CHT module enabled : 1
---[preciceAdapter] [DEBUG] Configuring the CHT module...
---[preciceAdapter] [DEBUG]     user-defined solver type : none
---[preciceAdapter] [DEBUG]     temperature field name : T
---[preciceAdapter] [DEBUG]     transportProperties name : transportProperties
---[preciceAdapter] [DEBUG]     conductivity name for basic solvers : k
---[preciceAdapter] [DEBUG]     density name for incompressible solvers : rho
---[preciceAdapter] [DEBUG]     heat capacity name for incompressible solvers : Cp
---[preciceAdapter] [DEBUG]     Prandtl number name for incompressible solvers : Pr
---[preciceAdapter] [DEBUG]     Turbulent thermal diffusivity field name for incomp
ressible solvers : alphat
---[preciceAdapter] [DEBUG] Determining the solver type...
---[preciceAdapter] [DEBUG] Found the transportProperties dictionary.
---[preciceAdapter] [DEBUG] Did not find the turbulenceProperties dictionary.
---[preciceAdapter] [DEBUG] Did not find the thermophysicalProperties dictionary.
---[preciceAdapter] [DEBUG] This is a basic solver, as transport properties are pro
vided, while turbulence or transport properties are not provided.
---[preciceAdapter] [DEBUG] Checking the timestep type (fixed vs adjustable)...
---[preciceAdapter] [DEBUG]   Timestep type: fixed.
---[preciceAdapter] [DEBUG] Creating the preCICE solver interface...
---[preciceAdapter] [DEBUG]   Number of processes: 1
---[preciceAdapter] [DEBUG]   MPI rank: 0
---[preciceAdapter] [DEBUG]   preCICE solver interface was created.
---[preciceAdapter] [DEBUG] Configuring preCICE...
(0) 16:10:32 [impl::SolverInterfaceImpl]:119 in configure: Configuring preCICE with
 configuration: "precice-config.xml"
(0) 16:10:32 [impl::SolverInterfaceImpl]:153 in configure: Run in coupling mode
---[preciceAdapter] [DEBUG]   preCICE was configured.
---[preciceAdapter] [DEBUG] Creating interfaces...
---[preciceAdapter] [DEBUG] Interface created on meshSolid-Mesh
---[preciceAdapter] [DEBUG] Adding coupling data writers...
---[preciceAdapter] [DEBUG] Constructed KappaEff_Basic.
---[preciceAdapter] [DEBUG]   Name of transportProperties: transportProperties
---[preciceAdapter] [DEBUG]   Name of conductivity: k
---[preciceAdapter] [DEBUG] k = 100.000000
---[preciceAdapter] [DEBUG] Added writer: Heat Flux for basic solvers.
---[preciceAdapter] [DEBUG] Adding coupling data readers...
---[preciceAdapter] [DEBUG] Added reader: Temperature.
---[preciceAdapter] [DEBUG] Iniializing the preCICE solver interface...
(0) 16:10:32 [impl::SolverInterfaceImpl]:216 in initialize: Setting up master commu
nication to coupling partner/s
```

Fluid: preparing...

```
Create mesh for time = 0


PIMPLE: Operating solver in PISO mode

Reading thermophysical properties

Selecting thermodynamics package
{
    type            heRhoThermo;
    mixture         pureMixture;
    transport       const;
    thermo          hConst;
    equationOfState perfectGas;
    specie          specie;
    energy          sensibleEnthalpy;
}

Reading field U

Reading/calculating face flux field phi

Creating turbulence model

Selecting turbulence model type laminar
Selecting laminar stress model Stokes

Reading g

Reading hRef
Calculating field g.h

Reading field p_rgh

Creating field dpdt

Creating field kinetic energy K

No MRF models present

Radiation model not active: radiationProperties not found
Selecting radiationModel none
No finite volume options present

Courant Number mean: 0.0837143 max: 0.403668

Starting time loop
```

3. How can I couple my own code?

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh



double dt; // solver timestep size




while (not simulationDone()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);


double dt; // solver timestep size




while (not simulationDone()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size




while (not simulationDone()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size



while (not simulationDone()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size

maxDt = precice.initialize()

while (not simulationDone()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size

maxDt = precice.initialize()

while (not simulationDone() && precice.isCouplingOngoing()){
  dt = beginTimeStep(); // e.g. compute adaptive dt

  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size

maxDt = precice.initialize()

while (not simulationDone() && precice.isCouplingOngoing()){
  dt = beginTimeStep(); // e.g. compute adaptive dt
  dt = min(maxDt, dt);
  computeTimeStep(dt);

  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size

maxDt = precice.initialize()

while (not simulationDone() && precice.isCouplingOngoing()){
  dt = beginTimeStep(); // e.g. compute adaptive dt
  dt = min(maxDt, dt);
  computeTimeStep(dt);
  maxDt = precice.advance(dt); // communication, data mapping, ...
  endTimeStep(); // e.g. update variables, increment time
}

turnOffSolver();
```

API 1: Steering

```
turnOnSolver(); //e.g. setup and partition mesh
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

double dt; // solver timestep size
double maxDt; // maximum precice timestep size

maxDt = precice.initialize()

while (not simulationDone() && precice.isCouplingOngoing()){
  dt = beginTimeStep(); // e.g. compute adaptive dt
  dt = min(maxDt, dt);
  computeTimeStep(dt);
  maxDt = precice.advance(dt); // communication, data mapping, ...
  endTimeStep(); // e.g. update variables, increment time
}
precice.finalize();
turnOffSolver();
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");









precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");







precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface






precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize





precice.initialize()





 [...]
```

API 2: Mesh and Data Access

```cpp
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)




precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates



precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates
int* vertexIDs = new int[vertexSize];


precice.initialize()





[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates
int* vertexIDs = new int[vertexSize];
precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);

precice.initialize()






[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates
int* vertexIDs = new int[vertexSize];
precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);

precice.initialize()

[...]

int forceID = precice.getDataID("Forces", meshID);

[...]
```

API 2: Mesh and Data Access

```
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates
int* vertexIDs = new int[vertexSize];
precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);

precice.initialize()

[...]

int forceID = precice.getDataID("Forces", meshID);
double* forces = new double[vertexSize*dim];
[...]
```

API 2: Mesh and Data Access

```cpp
precice::SolverInterface precice("FluidSolver",rank,size);
precice.configure("precice-config.xml");

int meshID = precice.getMeshID("FluidMesh");
int vertexSize; // number of vertices at coupling interface
// determine vertexSize
double* coords = new double[vertexSize*dim]; // coupling mesh (nodes)
// determine coordinates
int* vertexIDs = new int[vertexSize];
precice.setMeshVertices(meshID, vertexSize, coords, vertexIDs);

precice.initialize()

[...]

int forceID = precice.getDataID("Forces", meshID);
double* forces = new double[vertexSize*dim];
precice.writeBlockVectorData(forceID, vertexSize, vertexIDs, forces);
```

preCICE Configuration

```xml
<precice-configuration>

  <data:vector name="Forces" />
  <data:vector name="Displacements" />

  <mesh name="FluidMesh">
    <use-data name="Forces" />
    <use-data name="Displacements" />
  </mesh>

  <participant name="FluidSolver">
    <use-mesh name="FluidMesh" provide="yes" />
    <write-data name="Forces" mesh="FluidMesh" />

          [...]

  </participant>

  [...]
```

API 3: Implicit Coupling

```
while (not simulationDone() && precice.isCouplingOngoing()){





  dt = beginTimeStep();
  computeTimeStep(dt);
  precice.advance(dt); // communication, data mapping, ...






  endTimeStep(); // e.g. update variables, increment time

}
```

API 3: Implicit Coupling

```
while (not simulationDone() && precice.isCouplingOngoing()){
  if(precice.isActionRequired("WriteIterationCheckpoint")){
    saveCheckpoint(); // save internal state of solver
    precice.fulfilledAction("WriteIterationCheckpoint");
  }

  dt = beginTimeStep();
  computeTimeStep(dt);
  precice.advance(dt); // communication, data mapping, ...







  endTimeStep(); // e.g. update variables, increment time

}
```

API 3: Implicit Coupling

```
while (not simulationDone() && precice.isCouplingOngoing()){
  if(precice.isActionRequired("WriteIterationCheckpoint")){
    saveCheckpoint(); // save internal state of solver
    precice.fulfilledAction("WriteIterationCheckpoint");
  }

  dt = beginTimeStep();
  computeTimeStep(dt);
  precice.advance(dt); // communication, data mapping, ...

  if(precice.isActionRequired("ReadIterationCheckpoint")){
    reloadCheckpoint(); // set variables back to checkpoint
    precice.fulfilledAction("ReadIterationCheckpoint");
  }
  else{ // timestep converged
    endTimeStep(); // e.g. update variables, increment time
  }
}
```

Roadmap

Current Developments

- Debian package
- FSI module for the OpenFOAM adapter
- Fluid-fluid module for the OpenFOAM adapter
- Structure-structure coupling with CalculiX

Long-term Goals

- 3D-1D and 3D-2D data mapping
- Parallel initialization and support of dynamically changing coupling interfaces
- Consistent time interpolation

Funding

SPPEXA

DFG

More information

- Open-source project: `www.precice.org`
- Source code: `github.com/precice`

Contact us

- Mailing list
- Gitter chat room