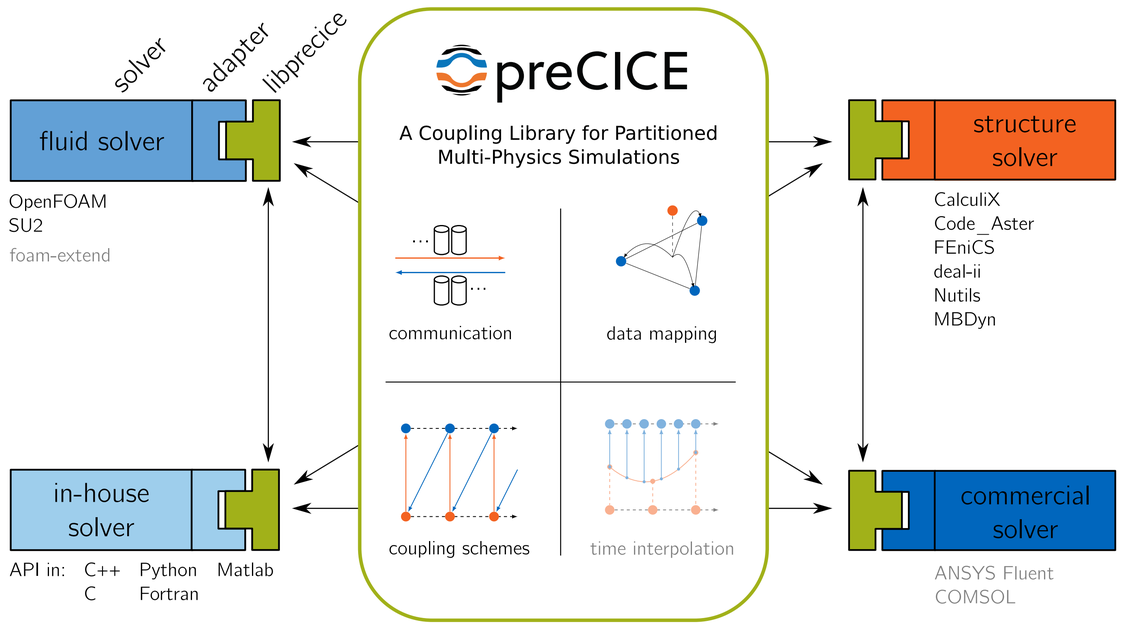
Chair of Scientific Computing in Computer Science
Faculty of Informatics
Technical University of Munich

preCICE - A Sustainable Foundation for Modern Multi-Physics Simulations

Hans-Joachim Bungartz[1], Gerasimos Chourdakis[1], Kyle Davis[2], Ishaan Desai[3], Konrad Eder[1], Oguz Ziya Koseomur[1], Miriam Mehl[2], Benjamin Rodenberg[1], David Schneider[3], Frédéric Simonis[1], Benjamin Uekermann[3]
[1]Chair of Scientific Computing, Technical University of Munich,
[2]Simulation of Large System, University of Stuttgart, [3]Usability and Sustainability of Simulation Software, University of Stuttgart

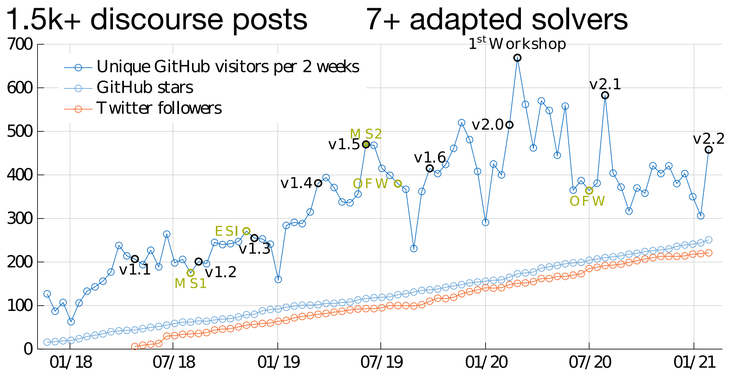
preCICE [1] is a coupling library for partitioned multi-physics simulations. Partitioned means that preCICE couples existing programs capable of simulating a subpart of the complete physics involved in a simulation. This allows for the high flexibility that is needed to keep a decent time-to-solution for complex multi-physics scenarios. preCICE runs efficiently on a wide spectrum of systems, from laptops up to 10000s of MPI Ranks.

This poster shows how preCICE has developed over the years and highlights three major challenges concerning sustainable software development we faced over the years including strategies used to tackle them.
Firstly, how to decide on a range of versions for dependencies to support and what type of package distribution to use.
Secondly, how to test a coupling library with logically partitioned tests.
Finally, how to profile a coupled simulation in contrast to a single program.



preCICE in numbers

70+ active user groups
2 workshops
1.5k+ discourse posts
30+ contributors
80k+ lines of code
7+ adapted solvers



Project workflow

PhD-driven development
Active collaboration
Active feedback
Coupling library
Separate adapters

Issue focused
Social contacts
Weekly meetings
Regular releases
ABI and API aware

API design

Low-entry barrier
Few basic functions
Minimally invasive
No dependency leak
C/Fortran-friendly

Rapid prototyping
Extensible
  Checkpointing
  Mesh connectivity

Python adapter example

```python
import precice
interface = precice.Interface("FluidSolver", "precice-config.xml", 0, 1)
set_mesh_vertices(positions)
dt = interface.initialize()
u = initialize_solution()

while interface.is_coupling_ongoing(): # main time loop
    # read coupling data
    displacement = interface.read_block_vector_data()
    # compute timestep
    u = solve_time_step(dt, u, displacement)
    force = compute_forces(u)
    # write coupling data
    write_block_vector_data(force)
    # continue to next time step
    dt = interface.advance(dt)
    t = t + dt

interface.finalize()
```

Most arguments, identifiers and less important methods omitted. Full guide at precice.org/couple-your-code-overview.html

Dependencies and packaging

Challenges

How to detect dependencies?
Which dependency versions to support?
What metadata and packages to provide?

Packaging strategy

Low entry-barrier and user focused
Ubuntu: 2 latest LTS and latest interim
Spack for HPC users [3,4]

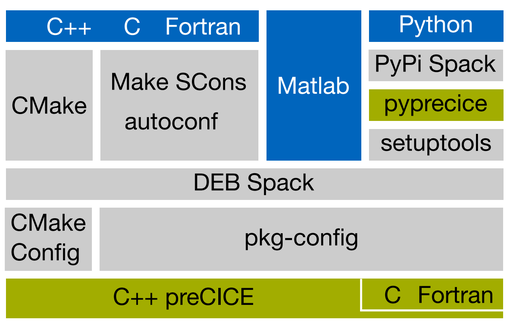
```
spack install precice@2.2.0
apt install ./libprecice2_2.2.0_focal.deb
```

Dependency strategy

Baseline-system: Ubuntu 18.04LTS
Support newest releases
Explicit dependencies, no auto-detection

Library metadata

pkg-config and CMake covers most build systems
Generated with CMake



How solvers (blue) use preCICE (green). Intermediate layers are build systems, packages and metadata.

Testing a coupling library

Challenges

Unit-testing

May require global state
non-communicating
Intra-participant comm
Inter-participant comm

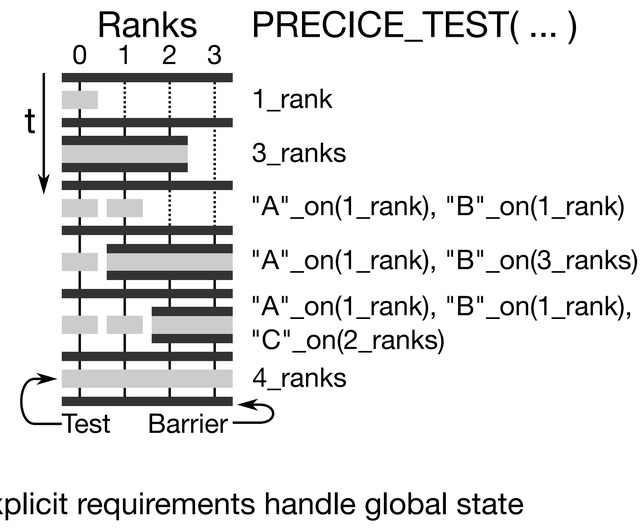
Integration-testing

Only calls the API
Handles global state
Communicating
Many setups required

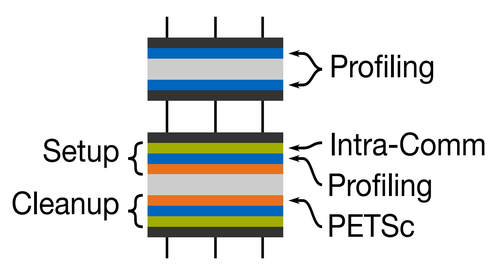
Testing strategy

Boost.Test running on 4 ranks
DSL to specify test setup and requirements

Custom test DSL



Explicit requirements handle global state



Requirements enable setup before and cleanup after tests. Top requires profiling, bottom requires PETSc & Intra-communication.

Performance analysis

Challenges

Context

One simulation
Multiple participants
Individual parallellism
Potentially multiple nodes

Technical

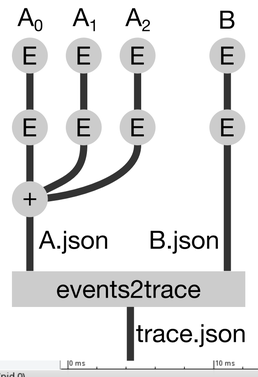
Clock jitter across nodes
Steady clock & system clock
Normalizing the timings

Goal

Visualize state across partitioned simulation
Locate bottlenecks in

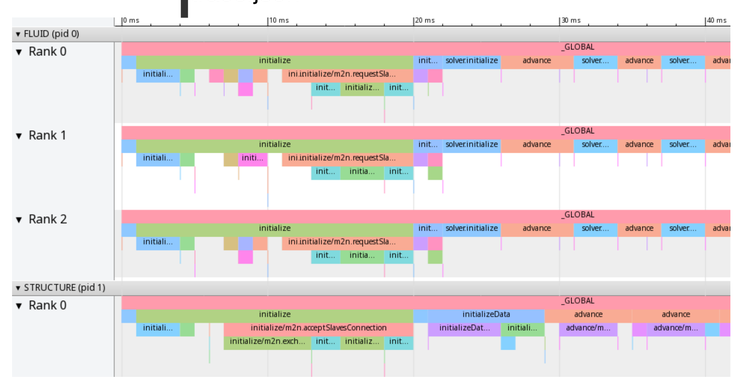
Our solution

Custom profiling library [2]
MPI sensitive



Usage

```
Event e{"Name"};
...
e.stop();
```



[1] preCICE -- A Fully Parallel Library for Multi-Physics Surface Coupling, Hans-Joachim Bungartz et al., 2016, In Computers and Fluids, Volume 141, p. 250—258. Elsevier.
[2] Data Transfer in Partitioned Multi-Physics Simulations: Interpolation & Communication, Florian Lindner, 2019, PhD thesis, IPVS, University of Stuttgart
[3] The Spack Package Manager: Bringing Order to HPC Software Chaos, Todd Gamblin, et al., November 15-20 2015, In Supercomputing 2015 (SC'15)
[4] XSDK Foundations: Toward an Extreme-scale Scientific Software Development Kit, Roscoe Bartlett, et al., 2017, In Supercomput. Front. Innov.: Int. J. 4, 1 (March 2017), p. 69–82.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754462

github.com/precice
precice.org