SPPEXA APM 2016

Partitioned Multi-Physics on Distributed Data via preCICE

Hans-Joachim Bungartz, Florian Lindner, Miriam Mehl,
Klaudius Scheufele, Alexander Shukaev, Benjamin Uekermann

Universität Stuttgart, Technische Universität München

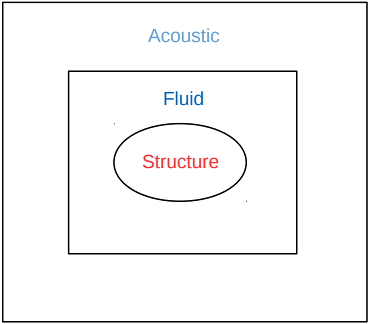
January 25, 2016

IAS TUM Institute for Advanced Study

Multi-Physics and Exa-Scale

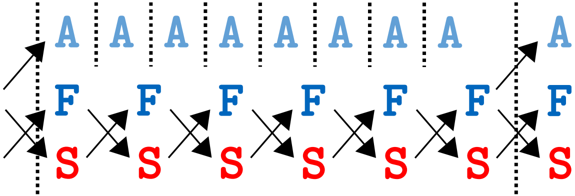
- many exciting applications need multi-physics
- more compute power $\Rightarrow$ more physics?
- many sophisticated, scalable,
  single-physics, legacy codes
- our goal: minimal invasive coupling,
  no deterioration of scalability

Our Example: Fluid-Structure-Acoustic Interaction



- FEAP - FASTEST - Ateles
- OpenFOAM - OpenFOAM - Ateles
- glue-software: preCICE

Our Example: Fluid-Structure-Acoustic Interaction



- implicit or explicit coupling
- subcycling

Agenda

This talk: glue-software preCICE
Next talk (Verena Krupp): application perspective

Agenda

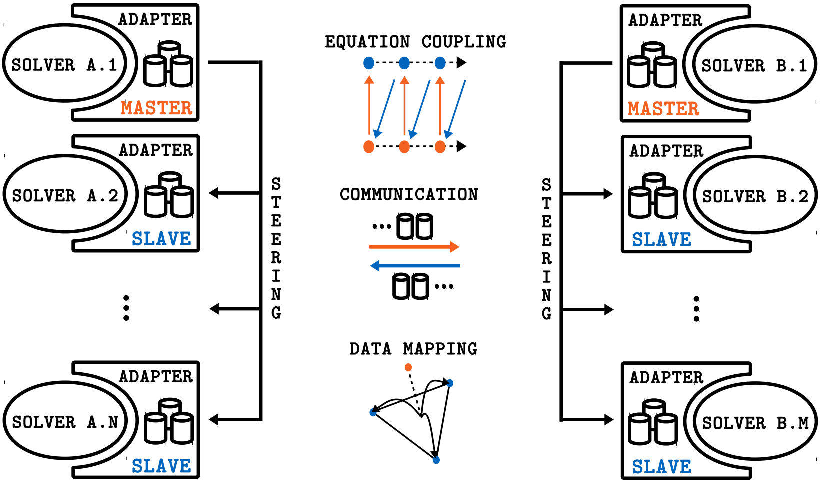
This talk: glue-software preCICE
Next talk (Verena Krupp): application perspective

1. Very brief introduction to preCICE
2. Realization on Distributed Data
3. Performance on Distributed Data

preCICE

- **pre**cise **C**ode **I**nteraction **C**oupling **E**nvironment
- developed in Munich and Stuttgart
- library approach, minimal invasive
- high-level API in `C++`, `C`, `Fortran77/90/95`, `Fortran2003`
- once adapter written $\Rightarrow$ plug and play
- `https://github.com/precice/precice`

Big Picture



ADAPTER
SOLVER A.1
MASTER
ADAPTER
SOLVER A.2
SLAVE
ADAPTER
SOLVER A.N
SLAVE
STEERING
EQUATION COUPLING
COMMUNICATION
DATA MAPPING
STEERING
ADAPTER
SOLVER B.1
MASTER
ADAPTER
SOLVER B.2
SLAVE
ADAPTER
SOLVER B.M
SLAVE

Coupled Solvers

Ateles (APES)	**CF**, **A**	in-house (U Siegen)
Alya System	**IF**, **S**	in-house (BSC)
Calculix	**S**	open-source (A*STAR)
CARAT	**S**	in-house (TUM STATIK)
COMSOL	**S**	commercial
EFD	**IF**	in-house (TUM SCCS)
FASTEST	**IF**	in-house (TU Darmstadt)
Fluent	**IF**	commercial
OpenFOAM	**CF**, **IF**, **S**	open-source (TU Delft)
Peano 1	**IF**	in-house (TUM SCCS)
SU2	**CF**	open-source

preCICE API

```
turn_on()

while time loop ≠ done do

   solve_timestep()

end while
turn_off()
```

preCICE API

```
turn_on()
precice_create("FLUID", "precice_config.xml", index, size)

precice_initialize()
while time loop ≠ done do

  solve_timestep()

end while
turn_off()
precice_finalize()
```

preCICE API

```
turn_on()
precice_create("FLUID", "precice_config.xml", index, size)

precice_initialize()
while time loop ≠ done do
  while precice_action_required(readCheckPoint) do

    solve_timestep()
    precice_advance()

  end while
end while
turn_off()
precice_finalize()
```

preCICE API

```
turn_on()
precice_create("FLUID", "precice_config.xml", index, size)
precice_set_vertices(meshID, N, pos(dim*N), vertIDs(N))
precice_initialize()
while time loop ≠ done do
  while precice_action_required(readCheckPoint) do

    solve_timestep()
    precice_advance()

  end while
end while
turn_off()
precice_finalize()
```

preCICE API

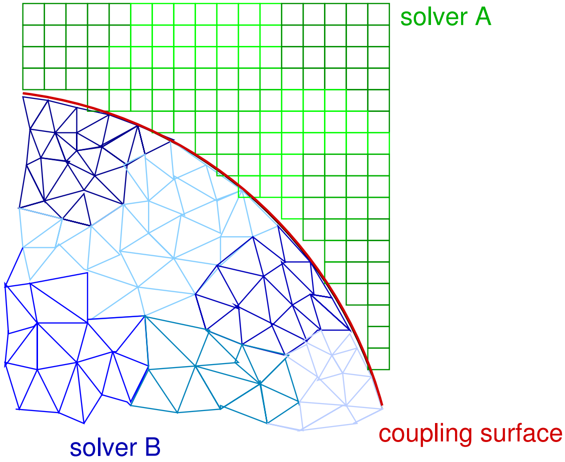
```
turn_on()
precice_create("FLUID", "precice_config.xml", index, size)
precice_set_vertices(meshID, N, pos(dim*N), vertIDs(N))
precice_initialize()
while time loop ≠ done do
  while precice_action_required(readCheckPoint) do
    precice_write_bvdata(stresID,N,vertIDs,stres(dim*N))
    solve_timestep()
    precice_advance()
    precice_read_bvdata(displID,N,vertIDs,displ(dim*N))
  end while
end while
turn_off()
precice_finalize()
```

preCICE Config

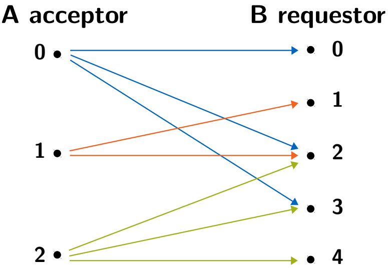
```
<coupling-scheme:parallel-explicit>
    <participants first="FLUID" second="ACOUSTIC" />
    <timestep-length value="1e-4" />
    <exchange data="Density"   mesh="AcousticSurface" from="FLUID" to="ACOUSTIC"/>
    <exchange data="Velocity"  mesh="AcousticSurface" from="FLUID" to="ACOUSTIC"/>
</coupling-scheme:parallel-explicit>

<coupling-scheme:parallel-implicit>
    <participants first="FLUID" second="STRUCTURE"/>
    <timestep-length value="1e-3"/>
    <exchange data="Forces"        mesh="WetSurface" from="FLUID" to="STRUCTURE"/>
    <exchange data="Displacements" mesh="WetSurface" from="STRUCTURE" to="FLUID"/>
    <rel-conv-measure data="Displacements" mesh="WetSurface" limit="1e-3"/>
    <post-processing:IQN-ILS>
        <data name="Forces" mesh="WetSurface">
        <data name="Displacements" mesh="WetSurface">
    </post-processing:IQN-ILS>
</coupling-scheme:parallel-implicit>
```

Communication on Distributed Data



Communication on Distributed Data

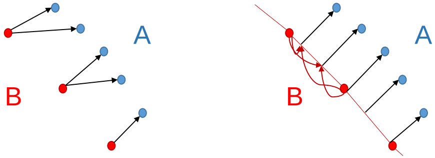


- kernel: 1:N communication based on either TCP/IP or MPI Ports
  ⇒ no deadlocks at initialization (independent of order at B)
- asynchronous communication (preferred over threads)
  ⇒ no deadlocks at communication

Interpolation on Distributed Data

Projection-based Interpolation

- first or second order
- example: consistent mapping from B to A
- parallelization: almost trivial

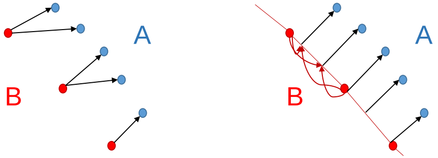


nearest neighbour mapping   nearest projection mapping

Interpolation on Distributed Data

Projection-based Interpolation

- first or second order
- example: consistent mapping from B to A
- parallelization: almost trivial



nearest neighbour mapping nearest projection mapping

Radial Basis Function Interpolation

- higher order
- parallelization: far from trivial (realized with `PETSc`)

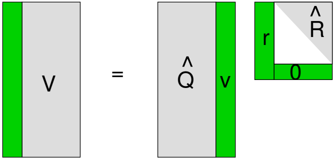
Fixed-Point Acceleration on Distributed Data

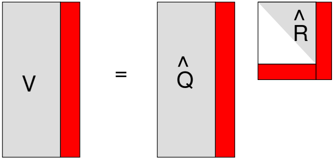
Anderson Acceleration (IQN-ILS)

Find $x \in D \subset \mathbb{R}^n : \; H(x) = x, \; H : D \to \mathbb{R}^n$

initial value x^0
$\tilde{x}^0 = H(x^0)$ and $R^0 = \tilde{x}^0 - x^0$
$x^1 = x^0 + 0.1 \cdot R^0$
for $k = 1 \ldots$ **do**
   $\tilde{x}^k = H(x^k)$ and $R^k = \tilde{x}^k - x^k$
   $V^k = [\Delta R_0^k, \ldots, \Delta R_{k-1}^k]$ with $\Delta R_i^k = R^i - R^k$
   $W_k = [\Delta \tilde{x}_0^k, \ldots, \Delta \tilde{x}_{k-1}^k]$ with $\Delta \tilde{x}_i^k = \tilde{x}^i - \tilde{x}^k$
   decompose $V^k = Q^k U^k$
   solve the first k lines of $U^k \alpha = -{Q^k}^T R^k$
   $\Delta \tilde{x} = W \alpha$
   $x^{k+1} = \tilde{x}^k + \Delta \tilde{x}^k$
end for

Fixed-Point Acceleration on Distributed Data





Insert Column

In: $\hat{Q} \in \mathbb{R}^{n \times m}$, $\hat{R} \in \mathbb{R}^{m \times m}$, $v \in \mathbb{R}^n$, Out: $Q \in \mathbb{R}^{n \times (m+1)}$, $R \in \mathbb{R}^{(m+1) \times (m+1)}$

for $j = 1 \dots m$ **do**

$r(j) = \hat{Q}(:, j)^T v$

$v = v - r(j) \cdot \hat{Q}(:, j)$

end for

$r(m+1) = \|v\|_2$

$Q(:, m+1) = r(m+1)^{-1} \cdot v$

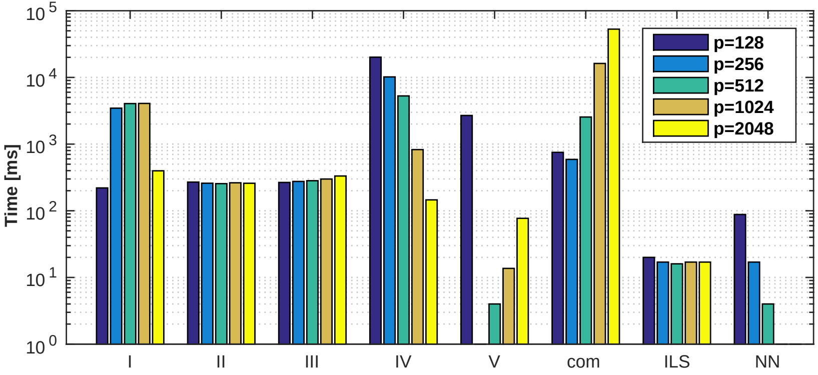
$R = \left[r, \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix} \right]$

Given's rotations $G_{i,j}$ s.t. $R = G_{1,2} \dots G_{m,m+1} R$ upper triangle

$Q = Q G_{m,m+1} \dots G_{1,2}$

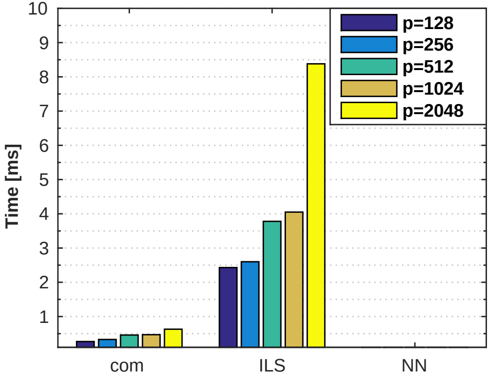
Performance Tests, Initialization

#DOFs at interface: $2.6 \cdot 10^5$, strong scaling



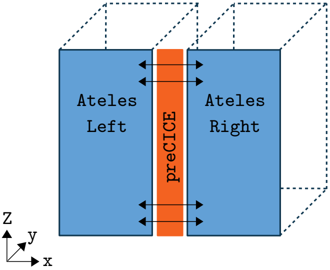
Performance Tests, Work per Timestep

#DOFs at interface: $2.6 \cdot 10^5$, strong scaling



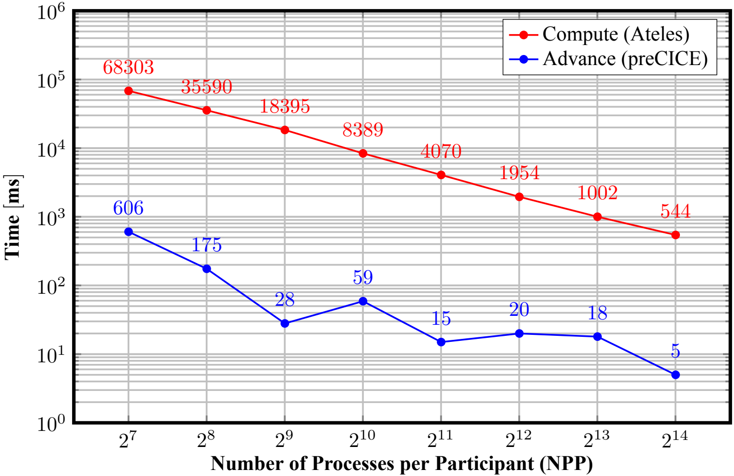
Performance Tests, Traveling Pulse

DG solver Ateles, Euler equations,
#DOFs: total: $5.9 \cdot 10^9$, at interface: $1.1 \cdot 10^7$,
NN mapping and communication
strong scaling from 128 to 16384 processors per participant.



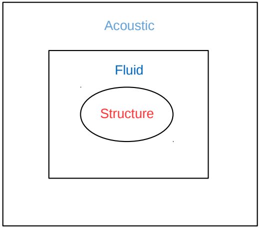
Joint work with V. Krupp et al. (Universität Siegen)

Performance Tests, Work per Timestep

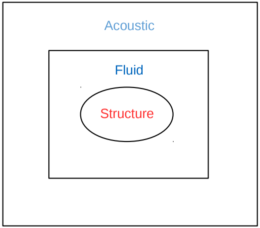


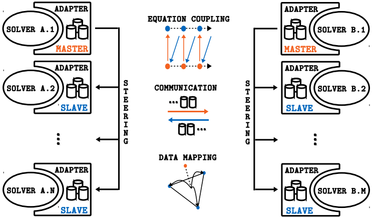
Summary

Summary

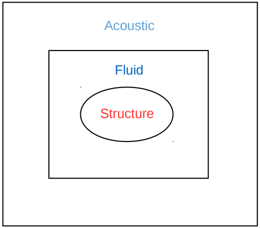


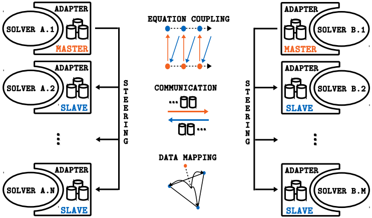
Summary


Acoustic
Fluid
Structure


ADAPTER
SOLVER A.1
MASTER
ADAPTER
SOLVER A.2
SLAVE
ADAPTER
SOLVER A.N
SLAVE
STEERING
EQUATION COUPLING
COMMUNICATION
DATA MAPPING
STEERING
ADAPTER
SOLVER B.1
MASTER
ADAPTER
SOLVER B.2
SLAVE
ADAPTER
SOLVER B.M
SLAVE

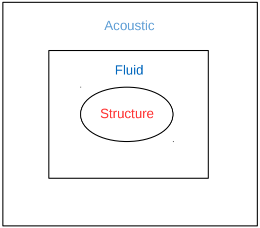
Summary

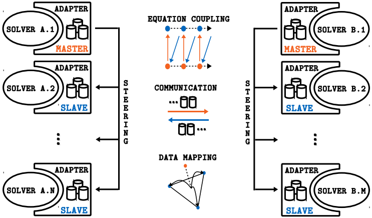





```
turn_on()
precice_create("FLUID", "precice_config.xml", index, size)
precice_set_vertices(meshID, N, pos(dim*N), vertIDs(N))
precice_initialize()
while time loop ≠ done do
    while precice_action_required(readCheckPoint) do
        pre_write_bvdata(stresID,N,vertIDs,stres(dim*N))
        solve_timestep()
        precice_advance()
        pre_read_bvdata(displID,N,vertIDs,displ(dim*N))
    end while
end while
turn_off()
precice_finalize()
```

Summary





turn_on()
precice_create("FLUID", "precice_config.xml", index, size)
precice_set_vertices(meshID, N, pos(dim*N), vertIDs(N))
precice_initialize()
while time loop $\neq$ done **do**
  while precice_action_required(readCheckPoint) **do**
    pre_write_bvdata(stresID,N,vertIDs,stres(dim*N))
    solve_timestep()
    precice_advance()
    pre_read_bvdata(displID,N,vertIDs,displ(dim*N))
  end while
end while
turn_off()
precice_finalize()