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
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
precise Code Interaction Coupling Environment
developed in Munich and Stuttgart
library approach, minimal invasive
high-level API in C++, C, Fortran77/90/95, Fortran2003
once adapter written ⇒ plug and play
https://github.com/precice/precice
Big Picture
## Coupled Solvers

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ateles (APES)</td>
<td>CF, A</td>
<td>in-house (U Siegen)</td>
</tr>
<tr>
<td>Alya System</td>
<td>IF, S</td>
<td>in-house (BSC)</td>
</tr>
<tr>
<td>Calculix</td>
<td>S</td>
<td>open-source (A*STAR)</td>
</tr>
<tr>
<td>CARAT</td>
<td>S</td>
<td>in-house (TUM STATIK)</td>
</tr>
<tr>
<td>COMSOL</td>
<td>S</td>
<td>commercial</td>
</tr>
<tr>
<td>EFD</td>
<td>IF</td>
<td>in-house (TUM SCCS)</td>
</tr>
<tr>
<td>FASTEST</td>
<td>IF</td>
<td>in-house (TU Darmstadt)</td>
</tr>
<tr>
<td>Fluent</td>
<td>IF</td>
<td>commercial</td>
</tr>
<tr>
<td>OpenFOAM</td>
<td>CF, IF, S</td>
<td>open-source (TU Delft)</td>
</tr>
<tr>
<td>Peano 1</td>
<td>IF</td>
<td>in-house (TUM SCCS)</td>
</tr>
<tr>
<td>SU2</td>
<td>CF</td>
<td>open-source</td>
</tr>
</tbody>
</table>
preCICE API

```
  turn_on()

  while time loop \neq 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 \(\neq\) done do

    solve_timestep()

end while

turn_off()
precice_finalize()
preCICE API

```c

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

  precice_initialize()
  while time loop \neq done do
    while precice_action_required(readCheckPoint) do

      solve_timestep()
      precice_advance()

    end while
  end while

  turn_off()
  precice_finalize()

```
preCICE API

```c

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
    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 \neq 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

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

9 <coupling-scheme: parallel-implicit>
10  <participants first="FLUID" second="STRUCTURE"/>
11  <timestep-length value="1e-3"/>
12  <exchange data="Forces" mesh="WetSurface" from="FLUID" to="STRUCTURE"/>
13  <exchange data="Displacements" mesh="WetSurface" from="STRUCTURE" to="FLUID"/>
14  <rel-conv-measure data="Displacements" mesh="WetSurface" limit="1e-3"/>
15 <post-processing: IQN-ILS>
16   <data name="Forces" mesh="WetSurface">
17   <data name="Displacements" mesh="WetSurface">
18 </post-processing: IQN-ILS>
19 </coupling-scheme: parallel-implicit>
20
Communication on Distributed Data

solver A

solver B

coupling surface
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

\[ \text{nearest neighbour mapping} \quad \text{nearest projection mapping} \]
Interpolation on Distributed Data

Projection-based Interpolation

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

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 \( U^k \alpha = -Q^k R^k \)
\( \Delta \tilde{x} = W \alpha \)
\( x^{k+1} = \tilde{x}^k + \Delta \tilde{x}^k \)
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: \( Q \in \mathbb{R}^{n \times (m+1)}, R \in \mathbb{R}^{(m+1) \times (m+1)} \)

\[
\text{for } j = 1 \ldots m \text{ do}
\]
\[
r(j) = \hat{Q}(\cdot, j)^T v
\]
\[
v = v - r(j) \cdot \hat{Q}(\cdot, j)
\]
\[\text{end for}\]
\[
r(m + 1) = \|v\|_2
\]
\[
Q(\cdot, m + 1) = r(m + 1)^{-1} \cdot v
\]
\[
R = \begin{bmatrix} r, & \left(\hat{R}\right) \\
0 & 0
\end{bmatrix}
\]

Given’s rotations \( G_{i,j} \) s.t. \( R = G_{1,2} \ldots G_{m,m+1} R \) upper triangle
\[
Q = QG_{m,m+1} \ldots G_{1,2}
\]
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

Number of Processes per Participant (NPP)

Time [ms]

- Compute (Ateles)
- Advance (preCICE)
Summary
Summary

Structure
  Fluid
    Structure
  Acoustic
**Summary**

Structure
Fluid
Acoustic

**Equation Coupling**

**Communication**

**Data Mapping**

**Solver A.1**
**Adapter**
**Master**

**Solver A.2**
**Adapter**
**Slave**

**Solver A.N**
**Adapter**
**Slave**

**Solver B.1**
**Adapter**
**Master**

**Solver B.2**
**Adapter**
**Slave**

**Solver B.M**
**Adapter**
**Slave**

```
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()
```
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()
Summary

```
turn_on()
pPrecice::create("FLUID", "precice_config.xml", index, size)
pPrecice::set_vertices(meshID, N, pos(dim*N), vertIDs(N))
pPrecice::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()
pPrecice::finalize()
```