Coupled Multi-Physics Simulation Framework on Octree Data Structures

Kannan Masilamani, Verena Krupp, Sabine Roller
Email: kannan.masilamani@uni-siegen.de
Outline

• Introduction to Multiphysics Applications
• Coupling Requirements
• APES Simulation Framework
• APESmate: Integrated Coupling Environment
• Results:
  – Load Balancing
  – Strong scaling
  – Coupled Simulations
• Conclusion and Current Work
Introduction: Application with Multi-physics

3-field surface coupling
largest domain - acoustic far field
→ large grid cells, higher order

2-field volume coupling

(a) Clotting process inside an aneurysm
Introduction: Application with Multi-physics

Electrodialysis:
Multi-physics heterogeneous system
3-field surface and volume coupling
Coupling Requirements

• Exchange of data from simulations in the space time domain between domains/solvers
• Coordinated execution of involved solvers
• Data exchange in the volume or the surface at common points in time or not coinciding points in time
  – Interpolation in space
  – For not coinciding time intervals, interpolate also in time
• Mesh independent
• Exploiting advantages of numerical solver to interpolate spatial values
• Individual numerical solver must be efficient and scalable in supercomputers
• Coupling should be scalable too
  – Load balancing of domains
  – Distribution of domains over nProcs_tot
APES framework

- Simulation framework for large scale parallel computations
- Based on Octree meshes
- End-to-end distributed memory parallel (MPI)
- Allows coupling of solvers
- Musubi:
  - Lattice Boltzmann Method
  - Multi-component LBM
  - Passive scalar
- Ateles:
  - High order discontinuous Galerkin solver
  - Euler equations
  - Linear-euler equations
  - Navier-Stokes equations
  - Maxwell equations
Octree data structure

- 8 Byte Integer: TreeID
- 8 Byte Integer: Property
- Distributed IO straight forward

Simulated Multi-Physics Simulation Framework on Octree Data Structures - 24th WSSP 2016
APESmate: Integrated Coupling Environment

• Software to couple APES solvers via TreElm
• Single executable
• Communication between domains is via global MPI communicator
• Communication within a domain is via its local MPI sub-communicator
• Configurable domain distribution with different process counts per domain
• Minimal access to solver routines enables to create dummy modules to build APESmate only with required solvers
APESmate: Integrated Coupling Environment

• Exchange point values -> Tree independent
• Coupling via space-time function - point data needed to be stored directly in ST-fun for Apesmate to access
  – Surface as boundary condition
  – Volume as source terms
• Easy to configure
• Evaluate point values using derive interface from variable system
• Uses solver data structure to evaluate variable requested by remote domain

• Limitations:
  – Supports only APES solvers
  – Time stepping - domains with different time steps works but without time interpolation
APESmate Algorithm

1. Load Apesmate config
2. Domain distribution
3. Load domains config
4. Initialize domains
5. Solve domains
6. Finalize domains
7. Sync. domains
8. Determine max dt between domains
9. Solve each domain until max dt
10. i < tmax

Initialize coupling
- Create list of coupling ST-fun
- Exchange points via Round robin
- Identify points
- Exchange point ranks
- Initialize comm buffer
**APESmate Algorithm**

1. Load Apesmate config
2. Domain distribution
3. Load domains config
4. Initialize domains
5. Solve domains
6. Finalize domains
7. Sync. domains
8. Determine max dt between domains
9. Solve each domain until max dt
10. Create list of coupling ST-fun
11. Exchange points via Round robin
12. Identify points
13. Exchange point ranks
14. Build comm buffer
15. i < tmax

**Steps:**
- Initialize coupling
- Load Apesmate config
- Domain distribution
- Load domains config
- Initialize domains
- Solve domains
- Finalize domains
- Sync. domains
- Determine max dt between domains
- Solve each domain until max dt
- Create list of coupling ST-fun
- Exchange points via Round robin
- Identify points
- Exchange point ranks
- Build comm buffer
- i < tmax
## Domain Distribution

**Apesmate**

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>D1</td>
</tr>
<tr>
<td>P1</td>
<td>D1</td>
</tr>
<tr>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>D2</td>
</tr>
<tr>
<td>P4</td>
<td>D2</td>
</tr>
<tr>
<td>P1</td>
<td>D3</td>
</tr>
</tbody>
</table>

- nProcs\(_{tot}\) = 5
- nDomains = 3

**Nr. Ranks fraction:**
- D1 = $\frac{2}{5}$
- D2 = $\frac{2}{5}$
- D3 = $\frac{1}{5}$
Domain Distribution

<table>
<thead>
<tr>
<th>Ranks</th>
<th>Domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>D1</td>
</tr>
<tr>
<td>P1</td>
<td>D1</td>
</tr>
<tr>
<td>P2</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td></td>
</tr>
</tbody>
</table>

nPrcs_tot = 5, nDomains=3

Nr. Ranks fraction: D1=2/5, D2=3/5, D3=2/5
Initialize coupling

- Exchange points and varNames between domains using round robin communication
  - Store requested points and varNames in linked list since we do not know how many request before hand
- Identify points
  - Requires offset_bit for surface coupling
- Check if requested varNames exist in local domain varSys
- Exchange point ranks
- Build point data communication buffer to send point data to correct process
- Exchange point data and varNames by point-point communication
  - Store requested point data and varNames in linked list
Surface Coupling

Domain left

G6, L6
G4, L4
G2, L2
G0, L0

G7, L7
G5, L5
G3, L3
G1, L1

Domain right

G14, L6
G12, L4
G10, L2
G8, L0

G15, L7
G13, L5
G11, L3
G9, L1
Offset bit?

Link based

Face normal
Offset bit

• Required to identify point in local domain
• Points are shifted along offset direction
• Offset direction is converted to character for communication

• Transforming offset direction into character:
  – offset_bit = achar((coord(1)+1) + (coord(2)+1)*4 + (coord(3)+1)*16)

• Backward transformation from character to direction
  – coordX = mod(ichar(offset_bit),4) - 1
  – coordY = mod(ichar(offset_bit),16)/4 - 1
  – coordZ = ichar(offset_bit)/16 - 1
Initialize coupling: 3-Domains

<table>
<thead>
<tr>
<th>Dom 1</th>
<th>Dom 2</th>
<th>Dom 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1, L1</td>
<td>G5, L3</td>
<td>G6, L0</td>
</tr>
<tr>
<td>G0, L0</td>
<td>G4, L2</td>
<td>G3, L1</td>
</tr>
<tr>
<td>E7, E8</td>
<td>E7, E8</td>
<td>E4</td>
</tr>
</tbody>
</table>
Initialize coupling: Round Robin Communication

Dom 1, G0, L1
Dom 1, G1, L2
Dom 2, G2, L0
Dom 2, G2, L0
Dom 2, G3, L1
Dom 2, G4, L2
Dom 2, G5, L3
Dom 2, G6, L0

E1, E2, E3, E4
E2, E4
E6, E8

E7
E8
E1
E5
E3
E4
E6
G1, L1
G2, L0
G3, L1
G4, L2
G5, L3
G6, L0
G0, L0
Initialize coupling: Point-Point Communication

Dom 1, G0, L1

Dom 1, G1, L2

Dom 2, G2, L0

Dom 2, G2, L0

Dom 2, G3, L1

Dom 2, G4, L2

Dom 2, G5, L3

Dom 2, G2, L0

Dom 2, G2, L0

Apesmate

E1

E2

E3

E4

E5

E6

E7

E8
Synchronize Domains

- At every synchronize time steps:
  - Evaluate variables on requested points with solver data structure
  - Exchange evaluated values to corresponding domains via global MPI-communicator using point-point communication
Ideal Load balancing

both participants should spend the same time in computation
→ no idling time
→ only mapping /evaluation + communication

+ Ideal load balancing when both domain use appropriate number of processes
+ Static load balancing is based on heuristics (but highly dependent on order)
Performance of Academic Testcase Coupling Euler - Linearized Euler

- monolithic (euler) \(1.014\)s on 512 MPI-ranks
- matching flow:
  - flow: \(dx=1,8000, O(6)\)
  - acoustic: \(dx=1,208.000, O(6)\)
  - save 20% compute time
- non-matching flow:
  - flow: \(dx=1,8000, O(6)\)
  - acoustic: \(dx=5, 1664, O(12)\)

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of MPI-ranks</th>
<th>Total computation time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>flow domain</td>
<td>acoustic domain</td>
</tr>
<tr>
<td>matching coupling</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>384</td>
</tr>
<tr>
<td></td>
<td>112</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>96</td>
<td>416</td>
</tr>
<tr>
<td></td>
<td>80</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>448</td>
</tr>
<tr>
<td></td>
<td>48</td>
<td>464</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>480</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>496</td>
</tr>
</tbody>
</table>

increase of work per proc decrease of work per proc
Strong Scaling: SuperMUC

Euler-Euler coupling
Order: O(20)
nElems: 4096 / domain
Subsonic Jet in 2D

- Inflow condition: expanded jet
- \( \text{Re} = 23 \) (\( \mu = 1 \times 10^{-6}, d=1 \times 10^{-4}, v=0.23 \))
- \( \text{Ma} = 0.2 \)
- Ramping of initial shock from over first 10s
- Sponge zone in Navier and Euler domain
Coupled Simulation for 2D Jet
Pipe flow in 3D

- Incompressible flow (D3Q19) + Passive scalar species transport (D3Q7)
- 1-way coupling - Species driven by velocity from flow field
Coupled Simulation for 3D Species Transport
Conclusion and Current Work

• APESmate implementation is working
• Tested with simple test cases for both surface and volume coupling
• Currently work:
  – Production test cases like electrodialysis process, 3-field acoustic and 2D jet are under testing phase
  – Scaling analysis
• Future work: Run production test cases
Thanks for your attention 😊

Questions?