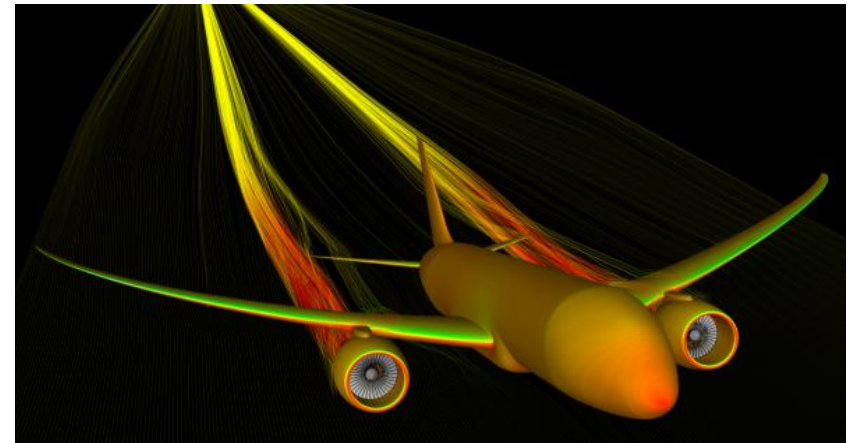
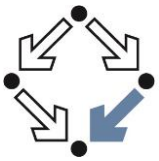


PFAU 9.0: „Fluid-Structure Simulations with OpenFOAM for Aircraft Designs“

9th OpenFOAM User Meeting
3.11.2014, JKU Linz

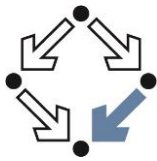
DI Thomas Ponweiser
RISC Software GmbH
Softwarepark 35, 4232 Hagenberg, Austria







Outline

- Introduction
 - RISC Software GmbH / Unit ISA
- Project description
 - Context: EU Project PRACE
 - Objectives and Scientific Case
- Accomplished Work
 - Geometric Modelling
 - FSI Solver implementation (based on OF 2.2)
- Performance and Scalability Analysis – Results
 - OF's Pstream lib:
 - Identifying a scalability bottleneck
 - Implementing a simple workaround
- Conclusion




About RISC Software GmbH

| RISC Institute | RISC Software GmbH | Ownership Structure | Business Units |
|---|---|---|---|
| <ul style="list-style-type: none"> • <i>Basic Research in Symbolic Computation</i> • Chair: Prof. Peter Paule • Founder(1987): Prof. Bruno Buchberger • 60 Members (including PhD Students) | <ul style="list-style-type: none"> • Software Development • Applied Research (<i>Algorithmic Mathematics</i>) • Transfer of Technology • approx. 50 Employees | <ul style="list-style-type: none"> • 80% Johannes Kepler University Linz • 20% State Upper Austria (UAR GmbH)  |  |



working group for industrial applications at RISC Institute

Foundation
RISC Software GmbH (Prof. Bruno Buchberger)

 JOHANNES KEPLER UNIVERSITY LINZ | JKU
RISC Software GmbH
100%-subsidiary of JKU

 RISC Software GmbH
Headcount 52
approx. 4,5 Mio. EUR



1990

Foundation of **Softwarepark Hagenberg** under the management of RISC



1995

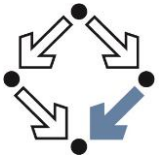
RISC SW specializes in software for logistics and production planning



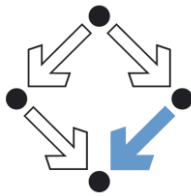
2008

Incorporation of the department of **Medical Informatics** and equity stake of State Upper Austria with 20%





RISC Software GmbH - Units



RISC Software GmbH

JOHANNES KEPLER
UNIVERSITY LINZ | JKU

**LOGISTICS
INFORMATICS**

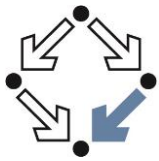


**INDUSTRIAL
SOFTWARE
APPLICATIONS**

**MEDICAL
INFORMATICS**



**ADVANCED
COMPUTING
TECHNOLOGIES**



Unit Industrial Software Applications (ISA)



Computational Engineering (CE)

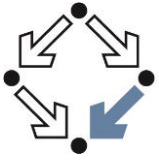
- Design Analysis and Optimization
- Virtual product development
- Engineering workflow management



Manufacturing Processes and Control Systems (MP)

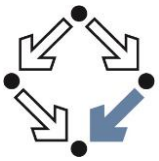
- Simulation of machining processes
- Geometric modeling and visualization for CAM





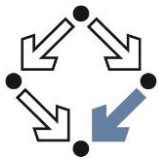
Project Context:

PRACE – The European HPC Infrastructure



What is PRACE?

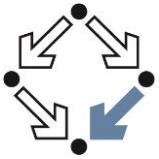
- PRACE is ...
 - a Pan-European Research Infrastructure (RI) for High Performance Computing (HPC)
 - an international non-profit association with seat in Brussels (PRACE AISBL)
- PRACE mission:
 - Enhancing European HPC competitiveness
 - Strengthen the European HPC user community
 - Improving HPC systems (energy efficiency / environmental footprint)



PRACE Members

- PRACE Funding
 - 68+ MEUR from EC FP7
 - ≈ 50 MEUR from PRACE Members
- Currently 25 Members
 - Austria: JKU

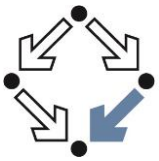




PRACE Services

- Education and training:
 - PRACE Advanced Training Centers (PATC)
 - PRACE Seasonal Schools
 - Partner Trainings
 - Summer of HPC
 - Best practice guides / Whitepapers (available online)

- Access to HPC Systems
 - For academia (world-wide) and (European) industry
 - Proposals for projects can be submitted periodically
 - Selection: Peer-review process
 - Condition for free access: All results must be published

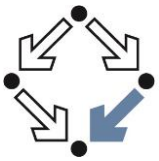


Access to HPC Systems – Resources

- Tier-0
 - European Centers with Multi-Petaflop performance
 - Currently 6 Systems:
 - CURIE (FR), FERMI (IT), Hornet (DE),
 - JUQUEEN (DE), MareNostrum (ES), SuperMUC (DE)

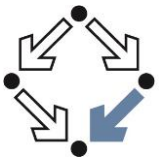
- Tier-1
 - National Centers (> 20 Systems)

- Tier-2
 - Regional / University Centers



Access to HPC Systems

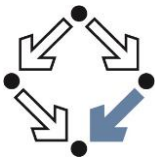
- Tier-0:
 - Project access
 - For large-scale computationally intensive projects
 - Every 6 months
 - 1 year duration; 500.000 – 100.000.000 CPU hours
 - Preparatory access
 - For preparing proposals for regular project access
 - Type A: Code scalability testing
 - Type B: Code optimization by applicant
 - Type C: Code optimization with PRACE expert support
 - Every 3 months
 - 6 months duration; 50.000 – 250.000 CPU hours



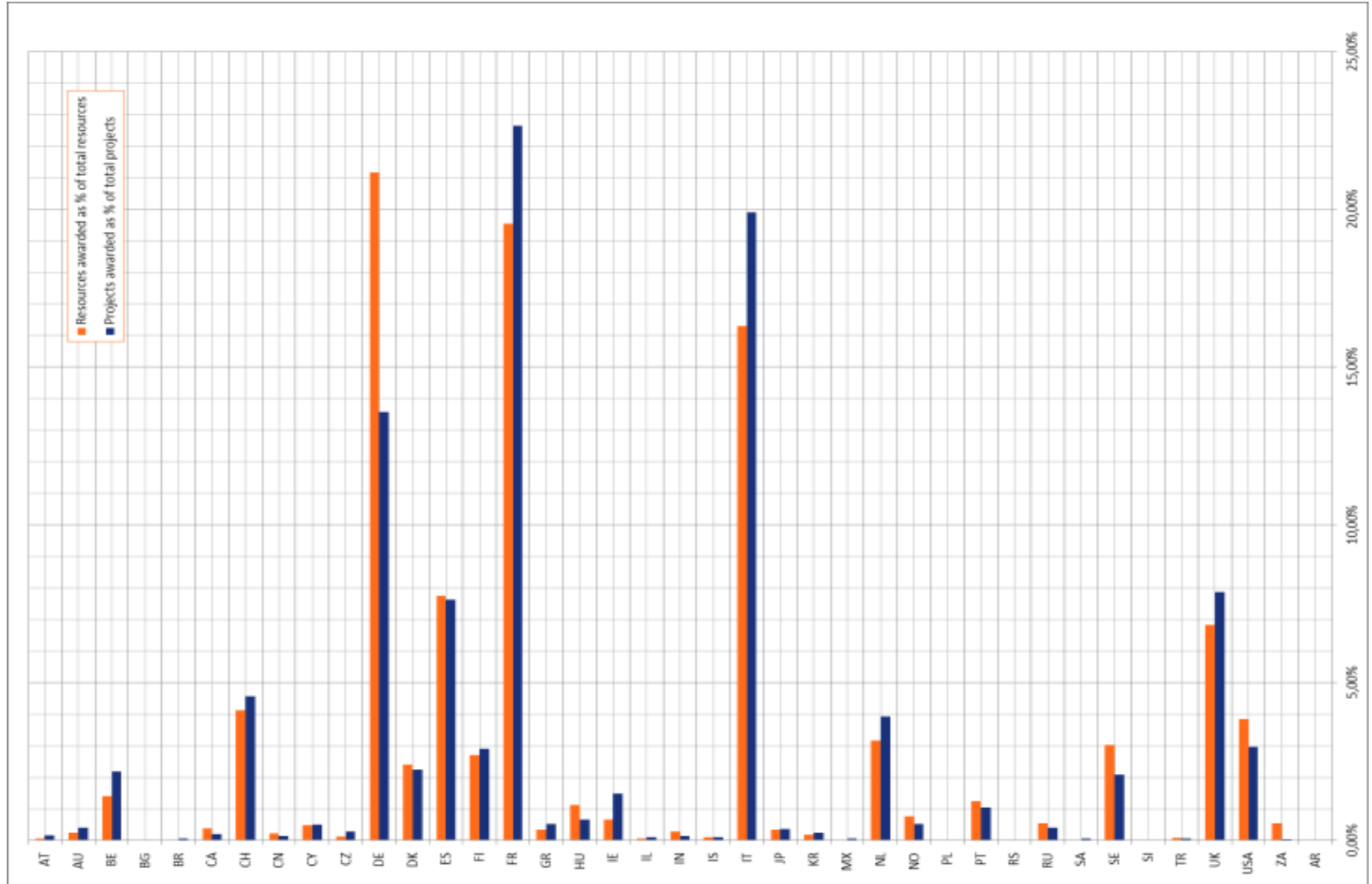
Access to HPC Systems

- Tier-1:
 - DECI (Distributed European Computing Initiative)
 - 12 national centers provide around 100 million CPU-hours per call
 - Calls every 6 months
 - 1 year duration; 1.000.000 – 10.000.000 CPU-hours
 - Multi-year access

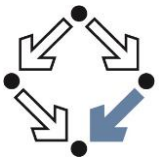
- More information online:
 - <http://www.prace-ri.eu/>



PRACE Resources awarded by Country



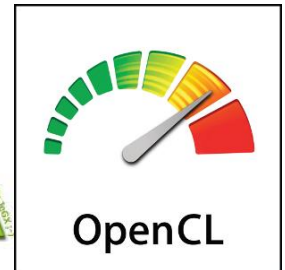
Source: <http://www.prace-ri.eu/statistics/>



Further PRACE activities

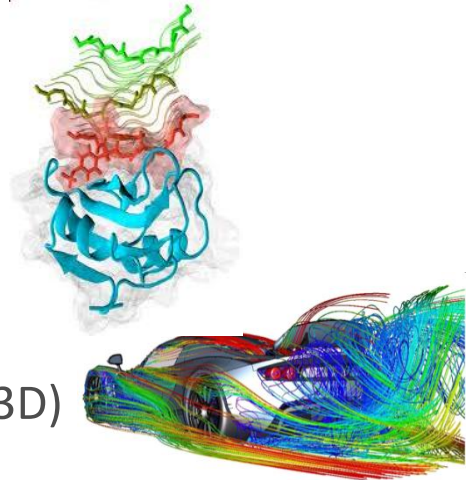
- New Programming Models, Libraries and Tools

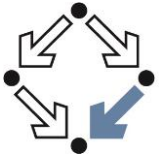
- Reviews of tool suites
- Creation of synthetic benchmarks
- Programming paradigms (MPI, OpenMP, PGAS, ...)
- Accelerator Languages (CUDA, RapidMind, openCL, ...)
- Petascale libraries
- Porting key kernels to wide range of platforms



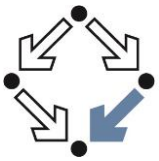
- Petascaling Key Community applications

- Material & Life Sciences (GPAW, GROMACS, Quantum_Espresso, CP2K)
- Computational Chemistry (DL_POLY, DALTON)
- Astrophysics (EUTERPE)
- Engineering/CFD (OpenFOAM, Code_Saturne)
- Meteorology/Climatology (EC Earth 3, SPECFEM3D)





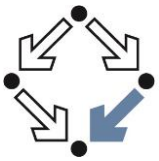
Fluid-Structure Simulations with OpenFOAM for Aircraft Designs



Project Motivation

- PRACE “Socio-Economic Challenges”
 - Projects proposed by PRACE partners having *significant impact / high benefit* for society and economy

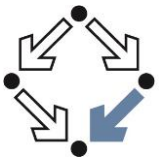
- Aircraft design is such a challenging task – goals:
 - Reducing weight
 - Improving energy-efficiency / environmental footprint
 - Reducing noise
 - Increasing safety



Project Motivation

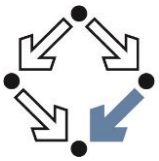
- State of the art
 - Highly specialized simulation codes for CFD/CSM
 - CFD calculations:
 - Rigid aircraft geometry
 - Simplified aerodynamic models (potential flow, etc.) are common
 - CSM calculations
 - Predefined aerodynamic forces

- Problem
 - Iterative, cost-intensive design process



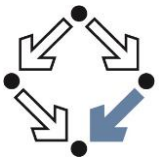
Project Goals

- Implementing an FSI Solver
 - Strongly coupled
 - Independent Meshes for Fluid/Solid domain
 - Optional:
Coupling different simulation codes for CFD and CSM
- Feasibility demonstration
 - Transient high-fidelity simulations of Aircrafts in high-lift configurations
 - Targeted scale:
CFD Mesh with > 50 Mio. Cells



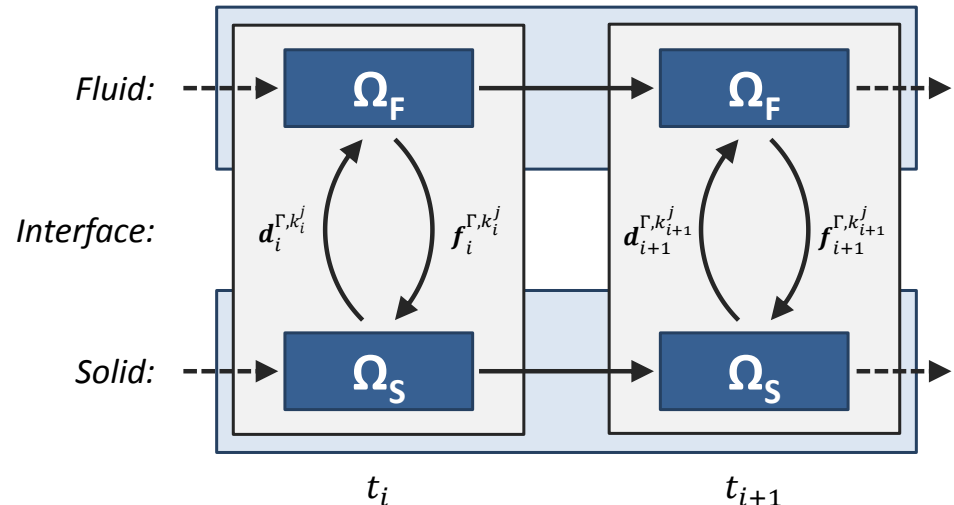
Decisions for implementation

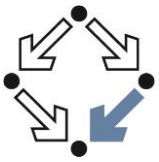
- Optional goal of using different simulation codes for CSM/CFD has been dropped.
- Implemented FSI Solver is entirely based on OpenFOAM 2.2
 - Independent meshes for Fluid and Solid domain realized with “mesh regions”.
 - Coupling with “arbitrary mesh interface” (AMI).
 - Inspired by:
 - *icoFSIElasticNonLinULSolidFoam* (OpenFOAM Extend Project)
 - *chtMultiRegionFoam* (OF 2.2.1)



Strong coupling scheme

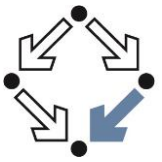
- For each time step:
 1. Set interface displacement for fluid mesh (Aitken relaxation is used)
 2. Compute mesh deformation (mesh motion solver)
 3. Solve Fluid (PISO)
 4. Set interface pressure for solid mesh
 5. Solve Solid (Linear elastic material model)





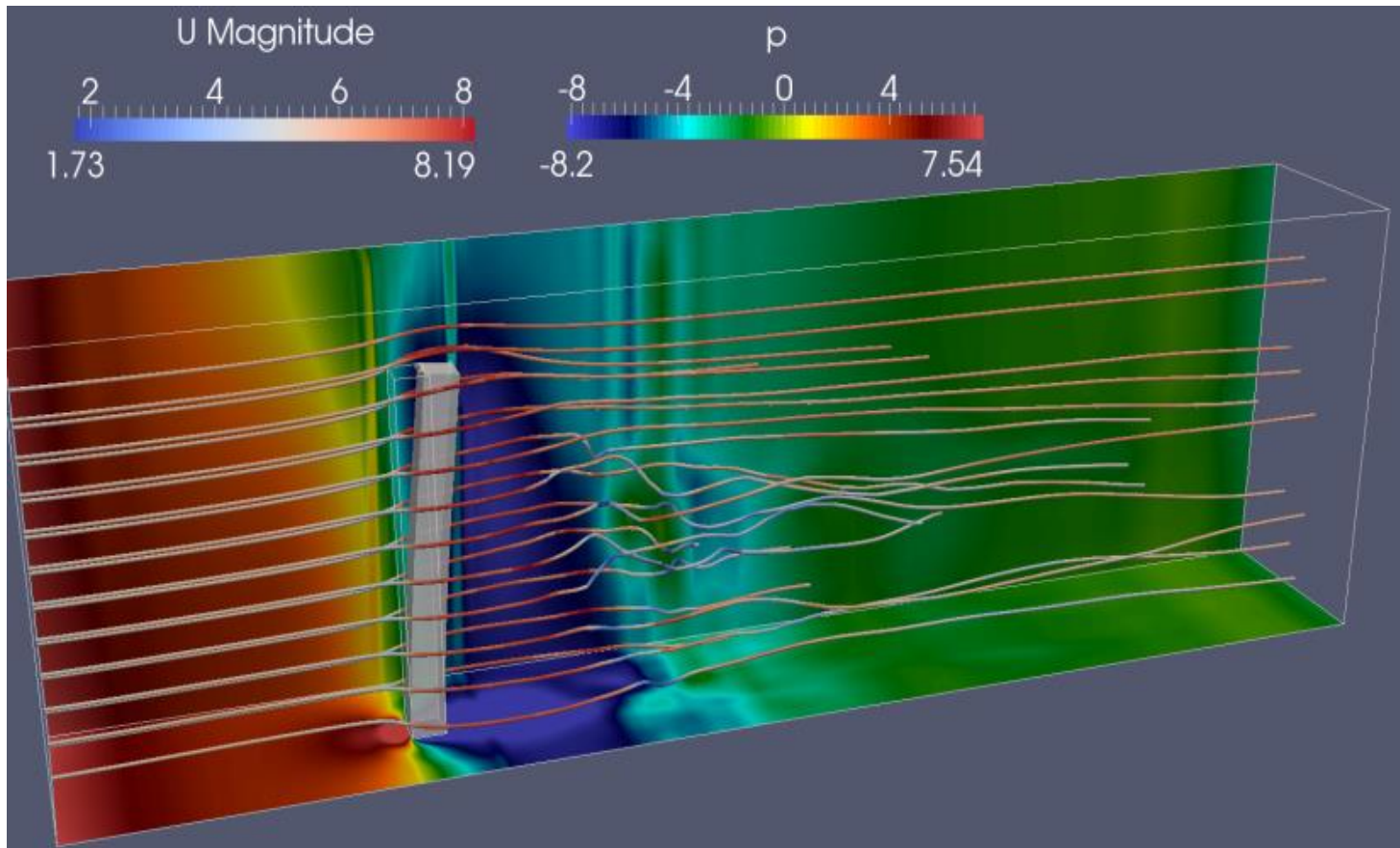
Modelling and Problems

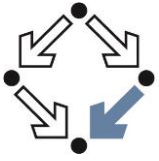
- Problems with Mesh Import
 - Available (sequential) Tools for converting CGNS to OF had serious problems with large meshes (> 1 Mio. Cells)
 - Self-written importer worked, but OF did not accept the generated mesh (OF has very strict quality criteria)



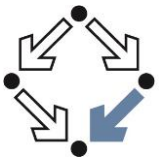
Mesh model for testing

- For testing and scalability analyses, a very simplified geometry has been used:



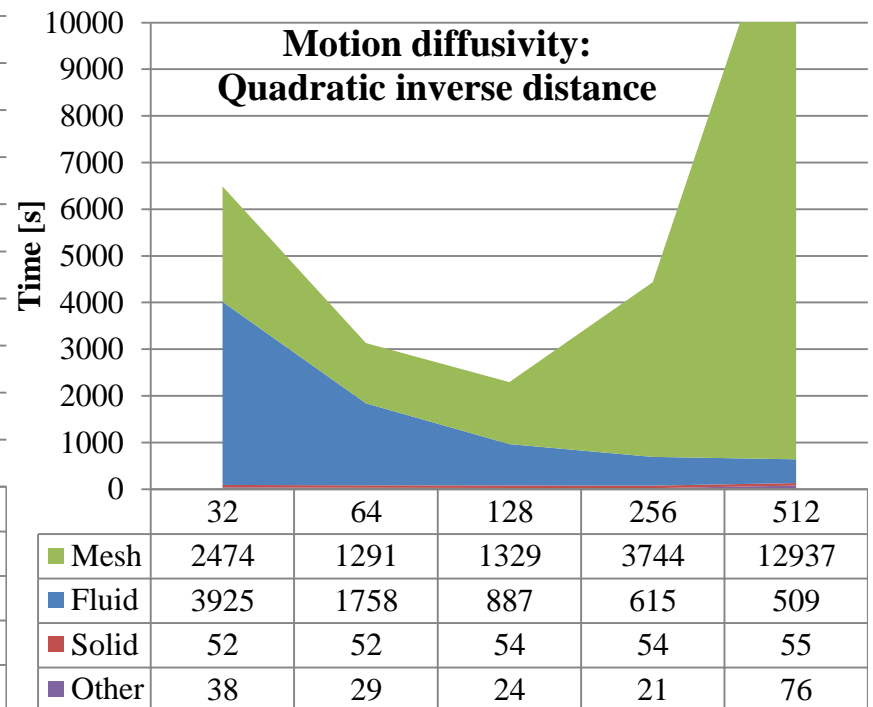
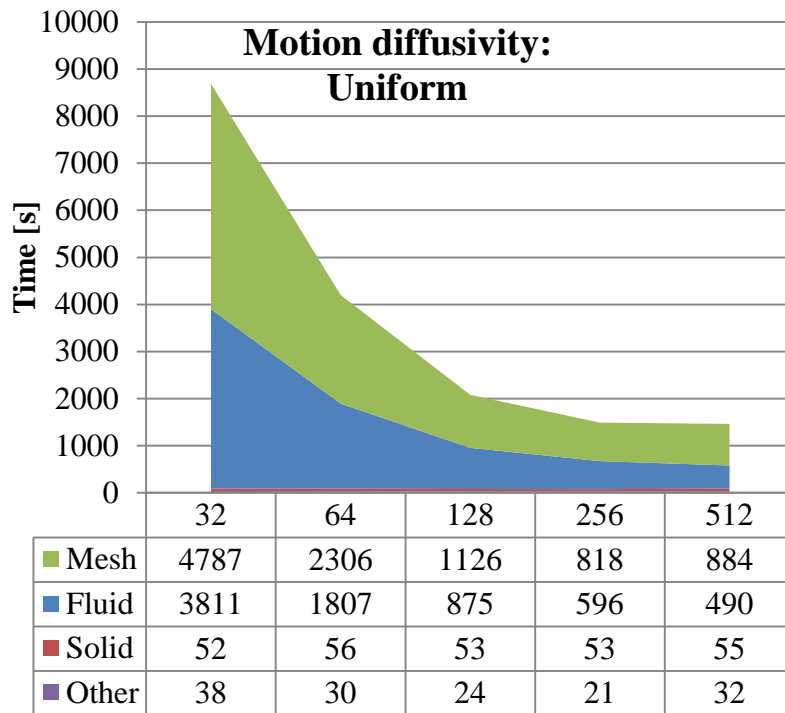


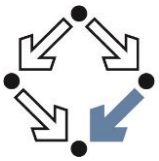
Performance Analyses and Results



Performance and Scalability

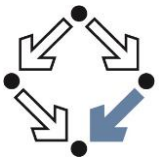
- Solution time is dominated by motion solver
 - Scalability strongly depends on setting for “Motion diffusivity”:





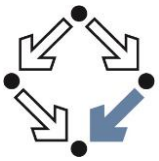
Scalability and Motion diffusivity

- What is the difference?
 - The difference is how the coefficients in the Laplace equation for mesh motion are determined.
 - Uniform:
 - Constant coefficient
 - Quadratic inverse distance:
 - Each cell's coefficient depends on the distance to the coupling interface.
 - Recalculated in every inner iteration of the strong-coupling scheme.



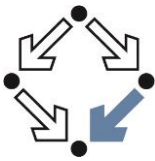
Identifying the Bottleneck (1/2)

- **Distance to wall** (coupling interface) is calculated using a “Moving-front algorithm” (**MeshWave**).
- In MeshWave, **collective data exchange** takes place in every iteration.
- In OpenFOAM, the first step for all collective data exchanges is **transferring message sizes**.
 - The complete matrix of message sizes is replicated to all processes (although only one row of this matrix is needed)
 - Implemented using MPI-Point-to-Point communications (strange hand-written tree-topology behind the scenes)



Identifying the Bottleneck (2/2)

- Summary:
 - The Bottleneck is **transfer of message sizes** (not transfer of actual messages).
 - Contributes to **≈60% (!) of the total runtime** for large core-counts
 - MPI-Point-to-Point communications are used, where MPI Collectives (MPI_Alltoall) would just be natural.
 - The Bottleneck affects **OpenFOAM's basic functionality** (Pstream::exchange()), used practically everywhere)



Eliminating the bottleneck

- Using MPI_Alltoall for message size transfer

Original code:

```
66         << " does not equal the number of processors:"
67         << UPstream::nProcs()
68         << Foam::abort(FatalError);
69     }
70
71     sizes.setSize(UPstream::nProcs());
72     labelList& nsTransPs = sizes[UPstream::myProcNo()];
73     nsTransPs.setSize(UPstream::nProcs());
74
75     forAll(sendBufs, procI)
76     {
77         nsTransPs[procI] = sendBufs[procI].size();
78     }
79
80     // Send sizes across. Note: blocks.
81     combineReduce(sizes, UPstream::listEq(), tag);
82
83     if (Pstream::parRun())
84     {
85         label startOfRequests = Pstream::nRequests();
86     }
```

Modified code:

```
181     /*sizes.setSize(UPstream::nProcs());
182     labelList& nsTransPs = sizes[UPstream::myProcNo()];
183     nsTransPs.setSize(UPstream::nProcs());*/
184     int nProcs = UPstream::nProcs();
185     int *sendCounts = new int[nProcs];
186     int *recvCounts = new int[nProcs];
187
188     forAll(sendBufs, procI)
189     {
190         //nsTransPs[procI] = sendBufs[procI].size();
191         sendCounts[procI] = sendBufs[procI].size();
192     }
193
194     // Send sizes across. Note: blocks.
195     //combineReduce(sizes, UPstream::listEq(), tag);
196     UPstream::alltoall(sendCounts, recvCounts);
197
198     if (Pstream::parRun())
199     {
200         label startOfRequests = Pstream::nRequests();
201     }
```

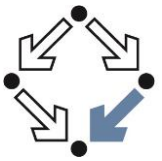
1-Calling Context Vie... 1-Callers View(icoFsi... 1-Flat View(icoFsiFo... ⌵

| Scope | PAPI_TOT_CYC:Sum (I) | PAPI_TOT_CYC:Sum |
|---------------------------------------|----------------------|------------------|
| exchange.C | 6.16e+14 60.0% | |
| void Foam::Pstream::exchange<Foam::Dy | 6.16e+14 60.0% | 1.88e+10 |
| 81: void Foam::combineReduce<Foam::Dy | 5.18e+14 90.5% | 1.28e+08 |
| 144: Foam::UPstream::waitRequests(ir | 9.76e+13 9.5% | 4.49e+08 |

2-Calling Context View(i... 2-Callers View(icoFsiFoa... 2-Flat View(icoFsiFoam) ⌵

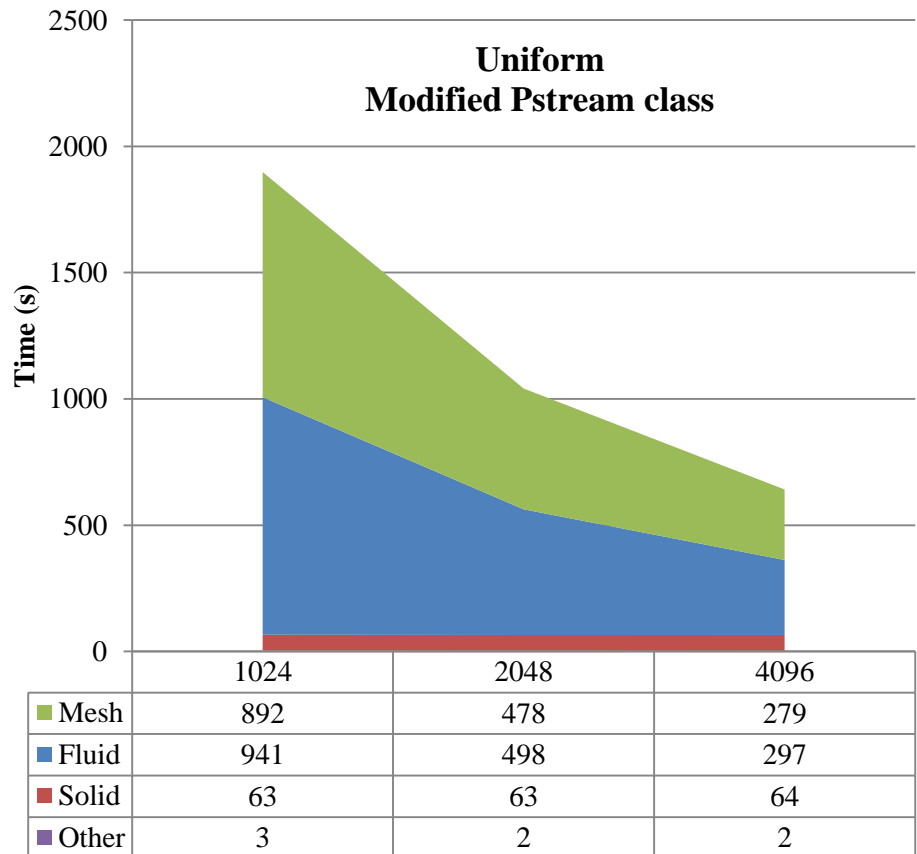
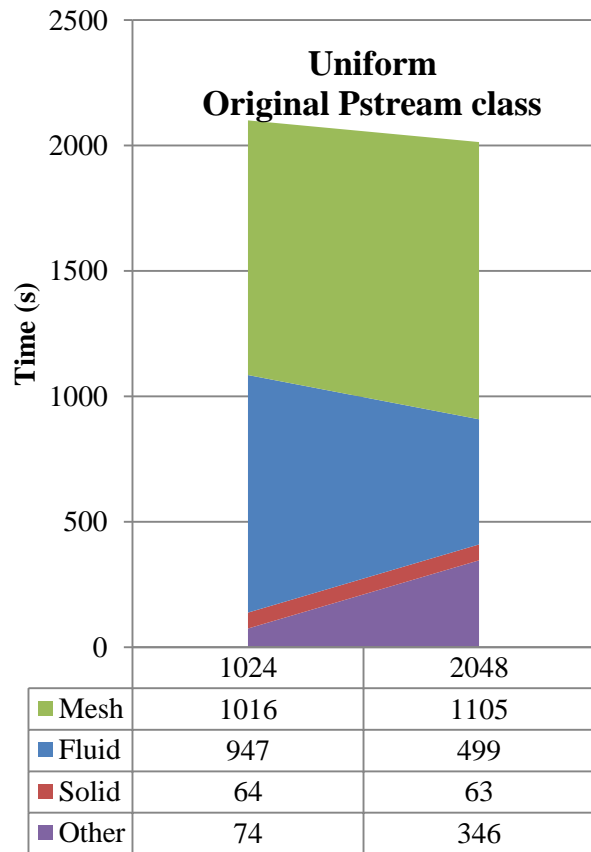
| Scope | PAPI_TOT_CYC:Sum (I) | PAPI_TOT_CYC:Sum (E) |
|--|----------------------|----------------------|
| exchange.C | 7.41e+12 9.3% | 8.89e+09 0.0% |
| void Foam::Pstream::exchange<Foam::Dy | 7.41e+12 9.3% | 1.08e+10 0.0% |
| 196: Foam::UPstream::alltoall(int*, int* | 7.34e+12 9.2% | 1.28e+08 0.0% |
| 262: Foam::UPstream::waitRequests(ir | 2.55e+10 0.0% | 2.25e+08 0.0% |

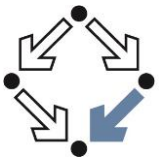
26M of 97M



Scalability comparison

- Effect for Uniform Motion diffusivity:



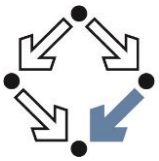


Conclusion

- Developing with OpenFOAM
 - Tutorials and examples are very helpful
 - Overall good software design (extensibility, flexibility)
 - Object-Orientedness has probably been pushed too far in some places

- Profiling Tools and OpenFOAM
 - Problems TAU or SCALASCA
 - Assumed reason:
Code instrumentation parsers cannot cope with OpenFOAM's unconventional programming style.
(using source files containing raw code blocks and #include statements instead of proper function calls)

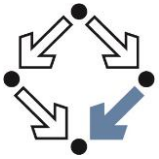
 - HPCToolkit works nicely
 - Call paths in OF are very deep / highly complex.
 - When compiler inlining is not explicitly disabled, understanding the generated profiles is nearly impossible.



Conclusion

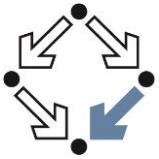
- OpenFOAM and mesh import:
 - For real industrial applications
 - Better import tools are needed and
 - OpenFOAM's high mesh quality requirements may be a further obstacle.

- Scalability of OpenFOAM:
 - A severe scalability bottleneck has been identified
 - Using MPI_Alltoall for message size exchange
 - Significantly improves scalability (up to 4k processes)
 - Reduces Code complexity (no strange tree-topology stuff)



Further Reading / Interesting Links

- Official PRACE Site:
 - <http://www.prace-ri.eu>
- Whitepaper
“Fluid Structure Interactions using OpenFOAM for Aircraft Designs”:
 - <http://www.prace-ri.eu/IMG/pdf/wp172.pdf>
- Reported scalability issue + patch for workaround:
 - <http://www.openfoam.org/mantisbt/view.php?id=1330>
- HPCToolkit:
 - <http://hpctoolkit.org/>

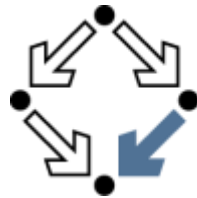


Castor, 4228m

Pollux, 4092m

between
Monte-Rosa-Massiv
and Matterhorn

Wallis, Schweiz



RISC
Software GmbH

www.risc-software.at