28th Workshop on Sustained Simulation Performance (WSSP)
October 9-10, 2018, Stuttgart, Germany

An Object Oriented Multiphysics Simulation Concept for HPC

Matthias Meinke
L. Schneiders, M. Schlottke-Lakemper, W. Schröder

Institute of Aerodynamics
RWTH Aachen University
Aachen, Germany

Outline

- Motivation
- Cartesian Mesh Multiphysics Concept
- Object Oriented Implementation
- Application Computational Aeroacoustics
- Dynamic Load Balancing
- Application Particulate Flow
- Conclusions

Motivation Particulate Flow

Coal & biomass combustion

SFB/TRR 129 (*source: Hitachi*)

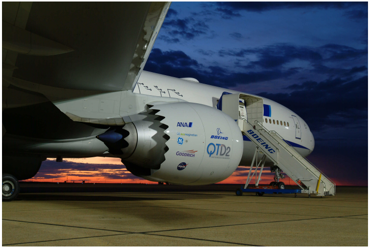
Electrical discharge machining

SFB/TRR 136 (*source: RGF*)

Improved **two-way-coupled** particle models required for:

- $d_p \gtrsim \eta \quad (d_p \sim 50 \ldots 200 \mu m)$
- non-spherical, inertial particles
- high-temperature environments

Motivation Noise Reduction



Source: J. Banke. NASA Helps Create a More Silent Night. `https://www.nasa.gov`

- Noise reduction up to 3 dB with thrust loss of 0.5% [?]
- Design variables: number of chevrons, penetration angle, chevron length, etc.

  → Optimization problem

Numerical analysis and optimization

- CFD: LES of the turbulent flow field
- CAA: solve acoustic pertubation equations for the acoustic field
- Long term goal: Perform (chevron) shape optimization to reduce noise using an adjoint CFD-CAA solver

[?] N. Saiyed, K. Mikkelsen, and J. Bridges, AIAA Journal 41(3) 372–378 (2003).

Object Oriented Implementation

Controller
- adaptation
- load balancing

→ **Cartesian grid**
- hierarchical, tree structure
- unified grid for all solvers

↓

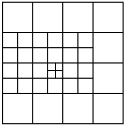
Solvers

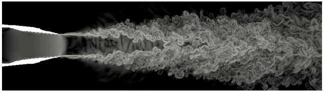
- Finite Volume Method
- Levelset
- Discontinuous Galerkin
- Lattice Boltzmann

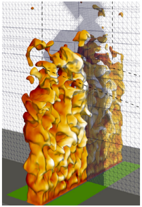
Coupler #1 → Finite Volume Method, Levelset

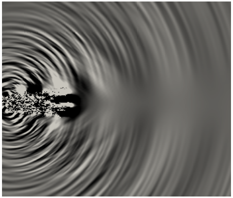
Coupler #2 → Finite Volume Method, Discontinuous Galerkin

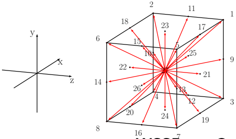
Coupler #3 ⇢ Levelset, Discontinuous Galerkin, Lattice Boltzmann



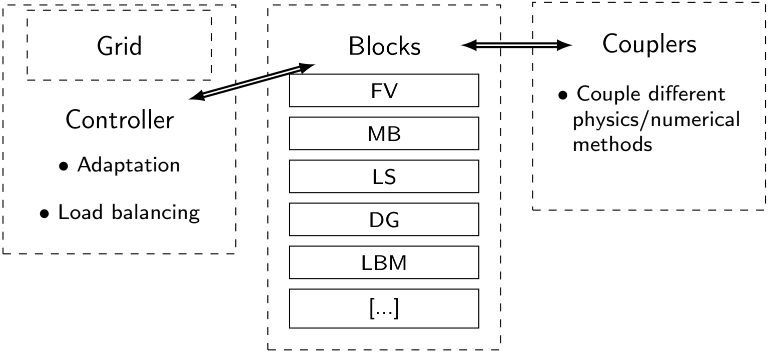








Multiphysics Concept



- Unstructured hierarchical Cartesian grid used for all solvers
- Each solver can utilize different cells of the joint Cartesian mesh
- Multiphysics coupling via separate coupling classes
- Unified control of grid adaptation and load balancing for all solvers

Multiphysics Concept

Listing 1: Initialization of multiphysics application

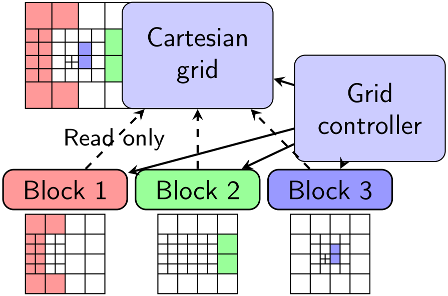
```
// Create grid and grid controller
const ZFSId noBlocks = ZFSContext::property("noBlocks").asInt();
ZFSCartesianGrid<nDim> grid(noBlocks);
typename zfs::grid::Controller<nDim> controller(grid, noBlocks);

// Create blocks
vector<shared_ptr<ZFSBlock>> blocks(noBlocks);
for (ZFSId i = 0; i < noBlocks; i++) {
  const ZFSString blockType = ZFSContext::blockProperty("blockType", i).asString();
  blocks[i] = ZFSApplication::makeBlock(blockType, grid);
  controller.addBlock(blocks.back());
}

// Create couplers
const ZFSId noCouplers = ZFSContext::property("noCouplers").asInt();
vector<shared_ptr<ZFSCoupler>> couplers(noCouplers);
for (ZFSId i = 0; i < noCouplers; i++) {
  const ZFSString couplerType = ZFSContext::property("couplerType", i).asString();
  couplers[i] = ZFSApplication::makeCoupler(couplerType, grid);
  controller.addCoupler(couplers.back());
}

// Initialize blocks and couplers
for (auto&& block : blocks) {
  block->init();
}
for (auto&& coupler : couplers) {
  coupler->init();
}
```

- High level abstraction allows flexible coupling of various numerical methods for multiphysics applications

Multiphysics Mesh Adaptation

Listing 2: ZFS multiblock mesh adaptation

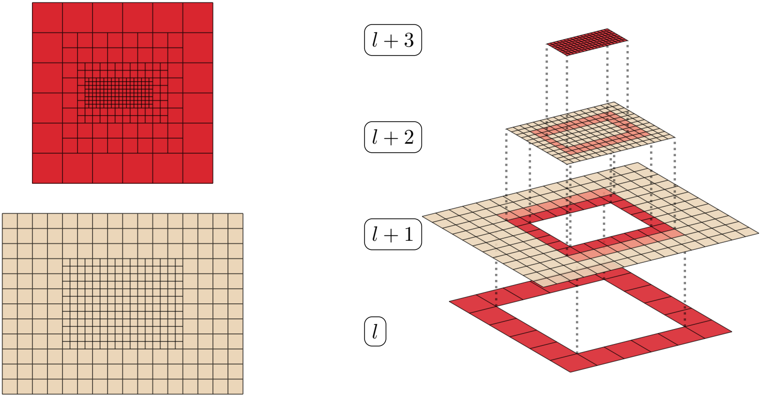
```
for (ZFSId i = 0; i < noBlocks(); i++) {
  block(i).prepareAdaptation(...); // Prepare block/solver for adaptation and collect
        refinement sensors
}

grid().meshAdaptation(...);

for (ZFSId i = 0; i < noBlocks(); i++) {
  block(i).reinitAfterAdaptation();
}
```



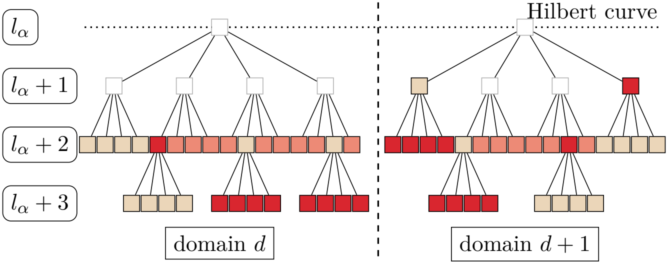
Block cells and refinement sensors

Joint hierarchical Cartesian mesh

- Hierarchical grid: parent-child relation between cells leads to tree structure
- Multiphysics method uses same tree structure for all physics
- Individual cells may be used for either physics1, physics 2, or both



Domain decomposition using cell weights

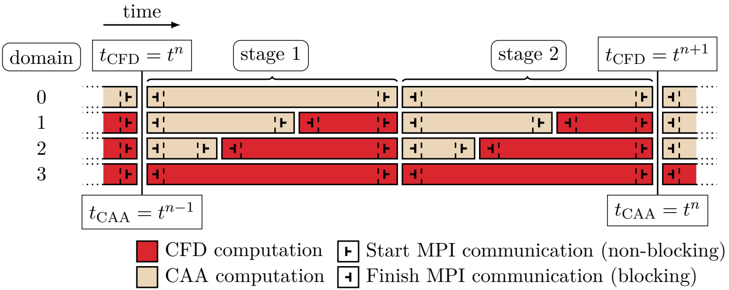


- Different cell weights ω_x for physics 1 cells, physics 2 cells, or cells used for both
- Domain decomposition based on Hilbert curve
- Partitioning takes place at coarse level
- Complete subtrees distributed among ranks

No MPI communication needed between all solvers

Parallel coupling algorithm

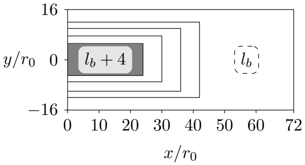
- Challenges for an efficient coupling algorithm
  - Computational load composition varies between domains
  - Solvers regularly need to exchange data internally



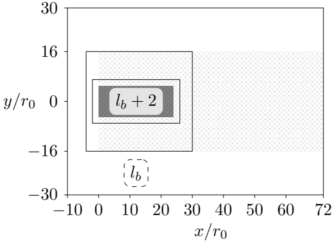
- Key components of algorithm
  - Both solvers composed of same number of "stages"
  - Identical effort for each stage, only one communication step per stage
  - Stages are interleaved for maximum efficiency

Schlottke-Lakemper et al., Comput. Fluids, 2017

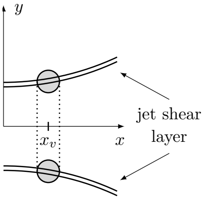
CFD grid



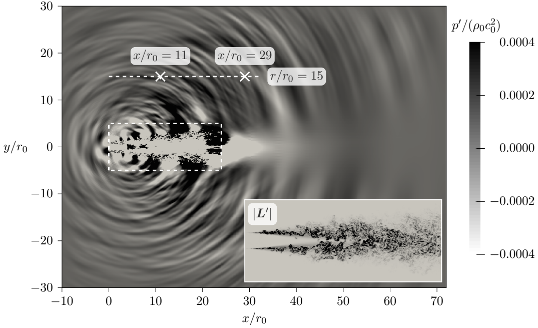
CAA grid

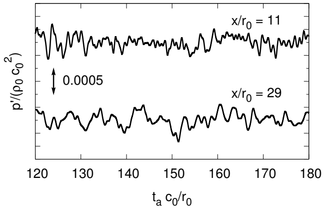


- Isothermal jet with $M = 0.9$ and $\mathrm{Re}_D = 400{,}000$
- Partial overlap between CFD and CAA domains (with interpolation)
- Vortex ring forcing near inlet to accelerate transition to turbulence (right)



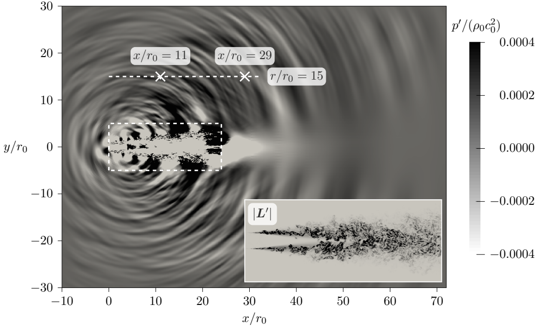
Direct-hybrid simulation of turbulent jet

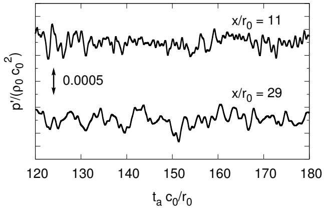


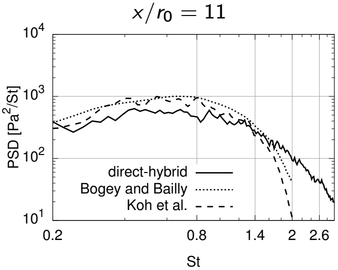


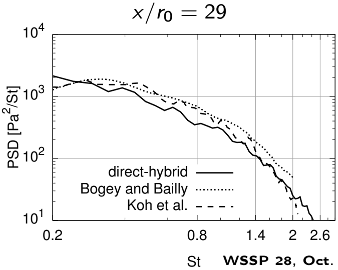
- Isothermal jet with $M = 0.9$ and $\mathrm{Re}_D = 400{,}000$
- Vortex ring forcing near inlet to accelerate transition to turbulence
- Anisotropic sound propagation: broadband noise in lateral direction, lower-frequency waves in downstream direction
- Acoustic sources strongest in shear layer around potential core

Direct-hybrid simulation of turbulent jet



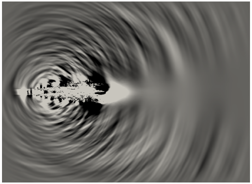






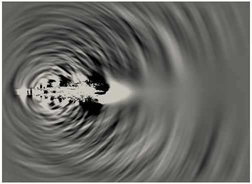
Direct-hybrid simulation of a turbulent jet

- FV-LES: 57.3 million cells
- DG-CAA: 237.4 million degrees of freedom (DOF)
- Partitioning:
  - Relate measured run times for standalone CFD/CAA simulations without coupling
  - Estimated computational weight ratio between CAA and CFD cells on 3072 cores: $w_{CAA} = 8.5$



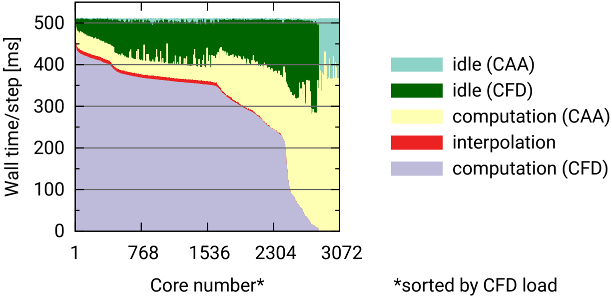
Direct-hybrid simulation of a turbulent jet

- FV-LES: 57.3 million cells
- DG-CAA: 237.4 million degrees of freedom (DOF)
- Partitioning:
  - Relate measured run times for standalone CFD/CAA simulations without coupling
  - Estimated computational weight ratio between CAA and CFD cells on 3072 cores: $w_{CAA} = 8.5$



Coupled performance

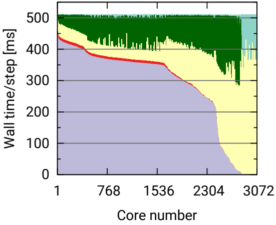
- Compute load:
  - CFD: 72%
  - CAA: 27%
  - Interpolation: 1%
- Idle time: 24%



Dynamic load balancing required to attain high parallel efficiency!

DLB algorithm

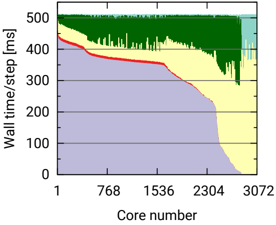
- Check for load imbalance
  - Measure local compute time t
  - Imbalance percentage: $I_{\%} = \frac{t_{max} - t_{avg}}{t_{max}} \cdot \frac{N}{N-1}$
- Determine new domain decomposition
  - Computational weights for CFD/CAA cells?
  - New space-filling curve (SFC) partitioning?
- Redistribute computational grid and solution data
  - Possible change of processes used for CFD/CAA



Dynamic load balancing (DLB)

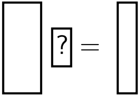
DLB algorithm

- Check for load imbalance
  - Measure local compute time t
  - Imbalance percentage: $I_{\%} = \frac{t_{max} - t_{avg}}{t_{max}} \cdot \frac{N}{N-1}$
- Determine new domain decomposition
  - Computational weights for CFD/CAA cells?
  - New space-filling curve (SFC) partitioning?
- Redistribute computational grid and solution data
  - Possible change of processes used for CFD/CAA



Approach

1. Weight estimation based on:
   - Measured compute times on each parallel process
   - Current distribution of cells of both solvers
2. Partitioning:
   - Individually vary each domain offset
   - Refine based on measured loads

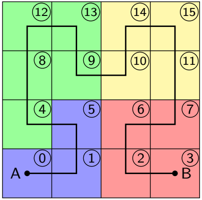


Linear least-squares problem

DLB: Partitioning approach (naive)

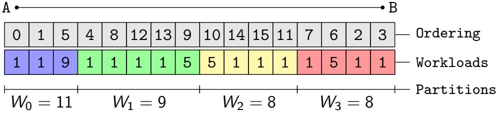
Standard SFC based partitioning

- 1D ordering of cells on coarse level by Hilbert SFC
- **Computational weights** ⇒ workload distribution
- Split into partitions of similar total workload
  ⇒ **Chains-on-chains partitioning** (CCP) problem



CCP algorithms

- Extensively studied in literature
- Goal: optimal partition quality
- Assume accurate knowledge of workload distribution



Partitioning approach: Weight estimation + CCP

- Linear workload model ⇒ local workload variations not captured
- Suboptimal load balanced partitioning for complex applications

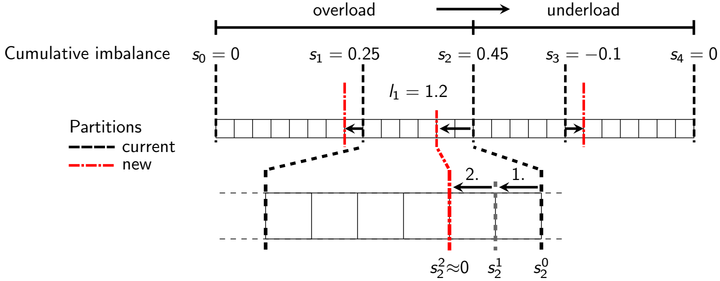
DLB: Partitioning approach (enhanced)

- Compute times r_i $\Rightarrow$ loads $l_i = r_i/\overline{r}$
- Cumulative imbalances: $s_j = \sum_{i=0}^{j-1} l_i - 1.0$

  $\Rightarrow$ Load imbalance of domains left and right of each splitting position

Approach:

- Iteratively shift offsets along SFC to minimize s_j
- Assess necessary displacements using estimated workload distribution

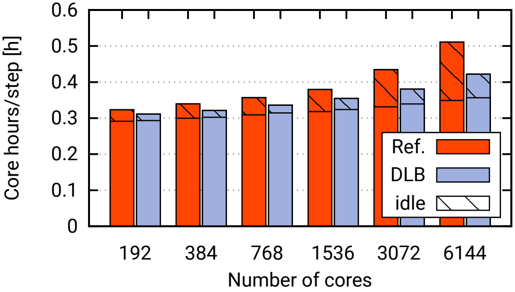


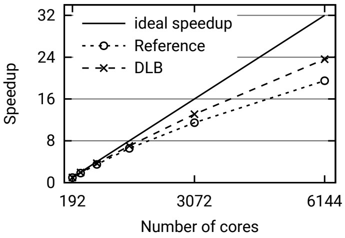
Assumption: optimize individually $\Rightarrow$ global load balance attainable

DLB: strong scaling from 192 to 6144 cores

- Low #cores: huge compute loads ⇒ small imbalances
- Decreasing local problem sizes ⇒ required comp. resources +20%
  - 6144 cores: Ø 9300 CFD cells and 38,500 CAA DOF

- Reference: $I_{\%,6144} = 31.2\%$
- DLB: $I_{\%,6144} = 14.6\%$





DLB benefit grows with degree of parallelism

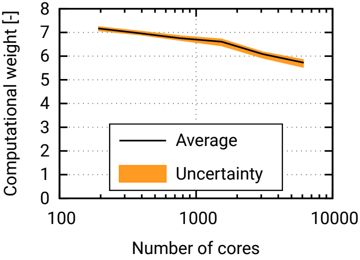
Resource savings of 17.5% for 6144 cores

DLB: Heterogeneous Systems

Variation of estimated weights

- CFD efficiency decline due to decreasing local problem sizes
- Computational weights depending on degree of parallelism

  ⇒ Automatically handled by DLB!

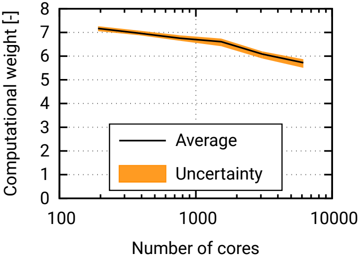


DLB: Heterogeneous Systems

Variation of estimated weights

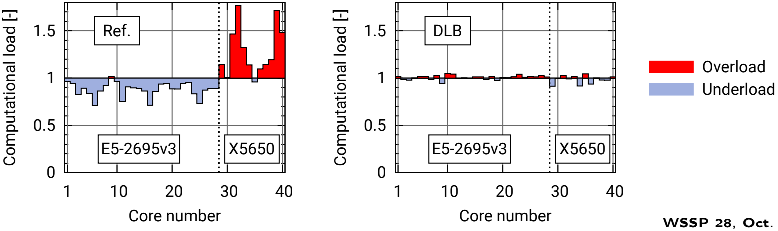
- CFD efficiency decline due to decreasing local problem sizes
- Computational weights depending on degree of parallelism

$\Rightarrow$ Automatically handled by DLB!

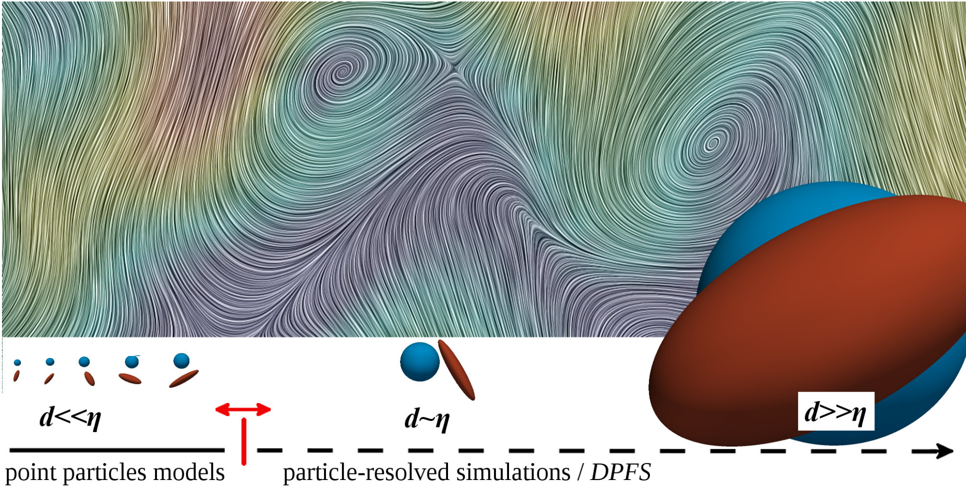


Heterogeneous systems: Estimate computing power of each process

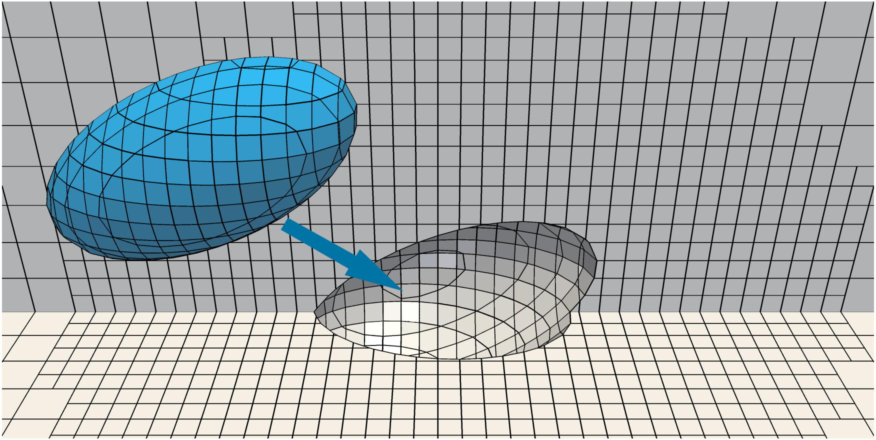
- Example: 40 cores, 2 * Intel Xeon E5-2695v3 + 2 * Intel Xeon X5650 $\Rightarrow$ Imbalance: Reference $I_{\%} = 45\%$, DLB $I_{\%} < 5\%$



Particulate Flows

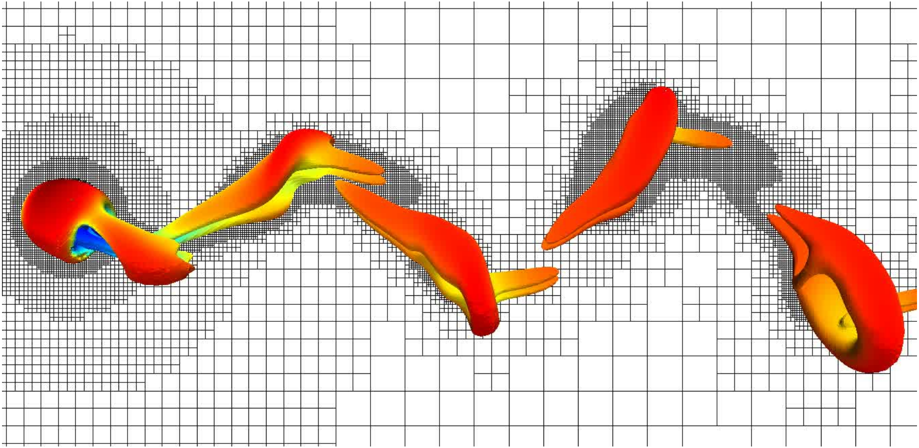


Cut-cell discretization



- Strict local conservation of fluid mass, momentum, energy
- High accuracy and stability in fluid-structure coupling

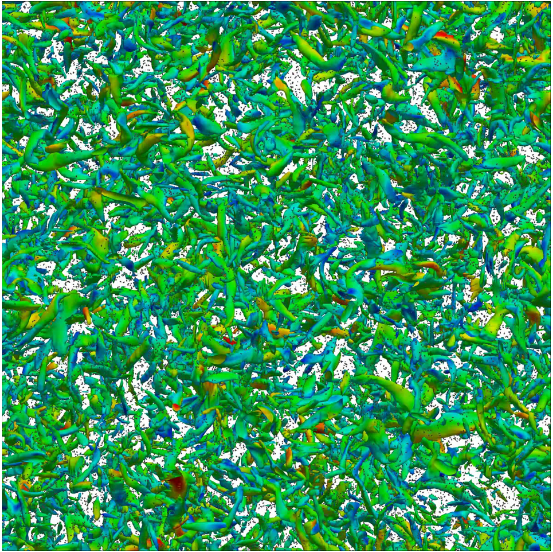
Dynamic mesh refinement



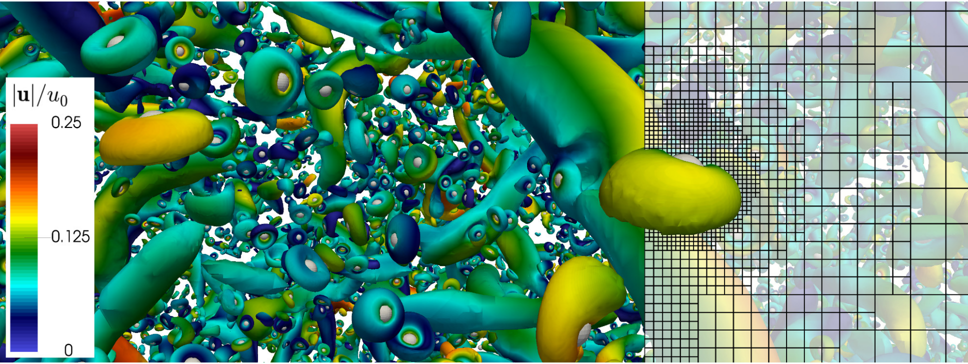
- $\boldsymbol{Re} = \mathbf{300}$, adaptation based on vorticity sensor

Schneiders, Günther, Meinke, Schröder; J. Comput. Phys. 311 (2016); J. Comput. Phys. 235 (2013)

Flow configuration



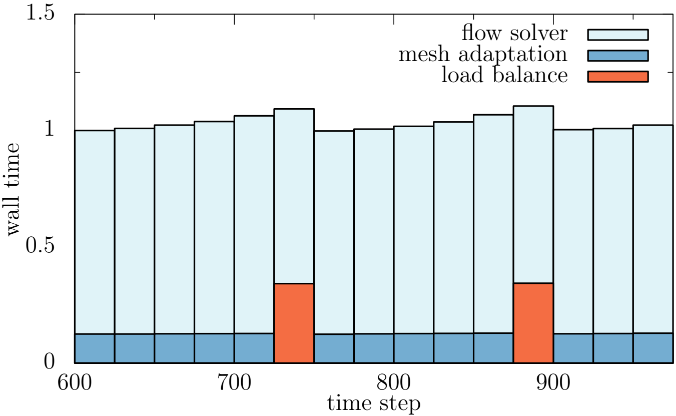
Flow configuration



- Decaying isotropic turbulence, initial $Re_\lambda(t_0) = 79$
- $E(k) = (3/2)u_0^2(k/k_p^2)exp(-k/k_p)$ (Ferrante & Elghobashi 2003, Lucci et al. 2010, etc.)
- 45,000 spheres at $d_p \sim \eta$: Schneiders, Meinke, Schröder J. Fluid Mech. 819 (2017)

Schneiders, Meinke, Schröder; J. Fluid Mech. 819 (2017); Fuel 201 (2017)

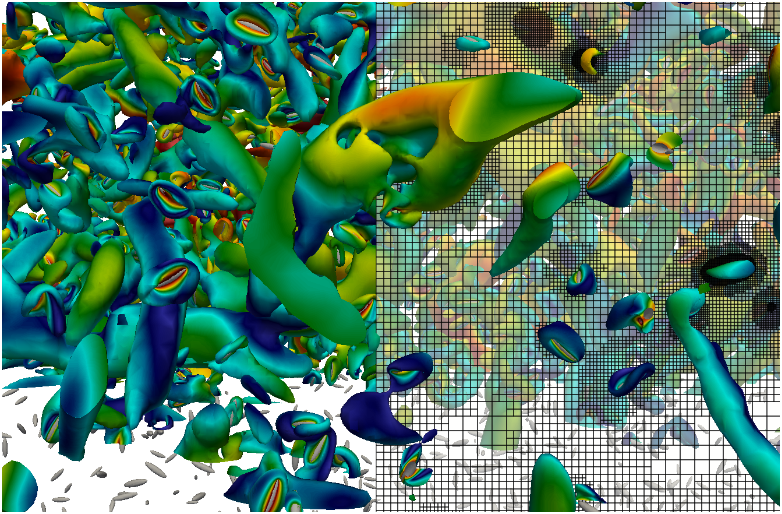
Dynamic load balancing



- **400 000 fully resolved particles (**$d_p \sim 0.5\eta$**)** in decaying isotropic turbulence
- 17.1 billion cells, 33 120 cores, HLRS Stuttgart (Hazelhen)

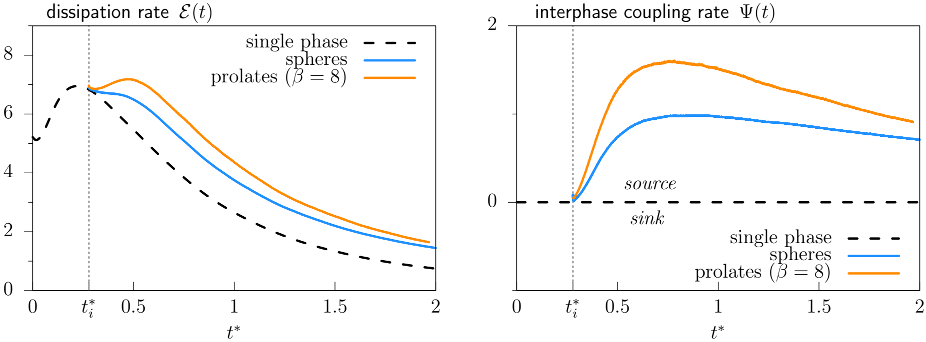
Schneiders, Grimmen, Meinke, Schröder, Proc. Appl. Math. Mech. Vol. 15. (2015)

Flow configuration



- **12 000 prolate particles:** $\beta = 8$, $\underline{d_p^{equiv} \sim 2\eta}$, $\rho_p/\rho = 1000$

Kinetic energy budget ($d_p^{equiv} \sim 2\eta$)



Fluid TKE decay: $\frac{\partial E_k}{\partial t} = \Psi(t) - \mathcal{E}(t)$

- TKE reduction at $t^* = 1$: **18% spheres vs. 25% prolates**

Summary & Conclusions

- Multiphysics applications based on hierarchical meshes have been presented
- Object oriented programming allows a flexible coupling for various physics
- Coupling is based on a joint Cartesian mesh allowing solution adaptive meshes
- Dynamic load balancing reduces computational effort and allows to use heterogeneous platforms
- Large scale applications demonstrate excellent performance

Summary & Conclusions

- Multiphysics applications based on hierarchical meshes have been presented
- Object oriented programming allows a flexible coupling for various physics
- Coupling is based on a joint Cartesian mesh allowing solution adaptive meshes
- Dynamic load balancing reduces computational effort and allows to use heterogeneous platforms
- Large scale applications demonstrate excellent performance

Thanks for your attention!

Acknowledgements:

DFG Deutsche Forschungsgemeinschaft

HLRS Stuttgart