Philipp Suffa, Markus Holzer, Ulrich Rüde

Chair for System Simulation, Friedrich Alexander-Universität Erlangen-Nürnberg (FAU)

Generated domain-specific sparse data kernels for high-performance Lattice Boltzmann Methods

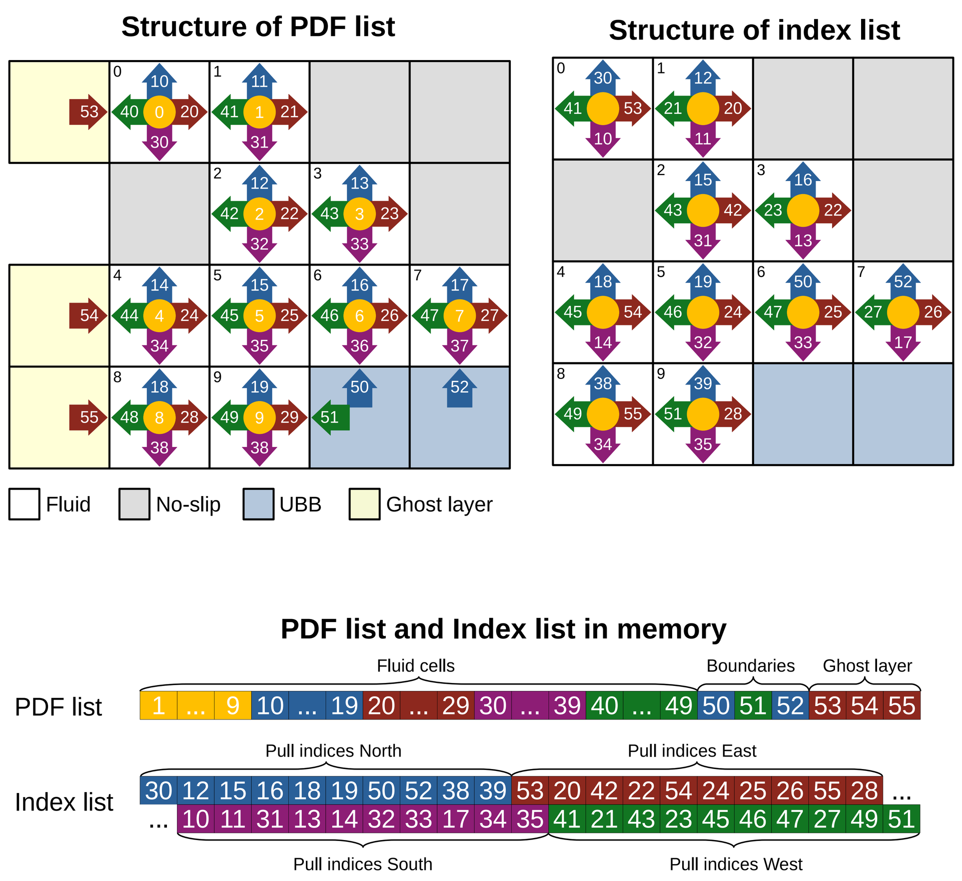
Sparse Lattice Boltzmann Method in *waLBerla*

Sparse Data Structure:

- **Indirect addressing**: Store only fluid cells of domain
  - ➔ ***Save memory and computation time***
- No-slip and periodic boundaries handled implicitly
  - ➔ ***No extra boundary kernel needed***

Code Generation:

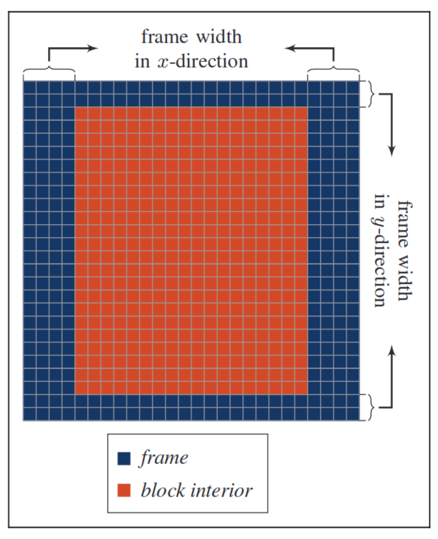
- Integrated in the code generation pipeline of `lbmpy/pystencils`
  - ➔ ***Generate sparse kernels for CPU and GPU architectures***
  - ➔ ***Flexible stencils and collision (SRT, TRT, MRT, Cumulants, ...)***

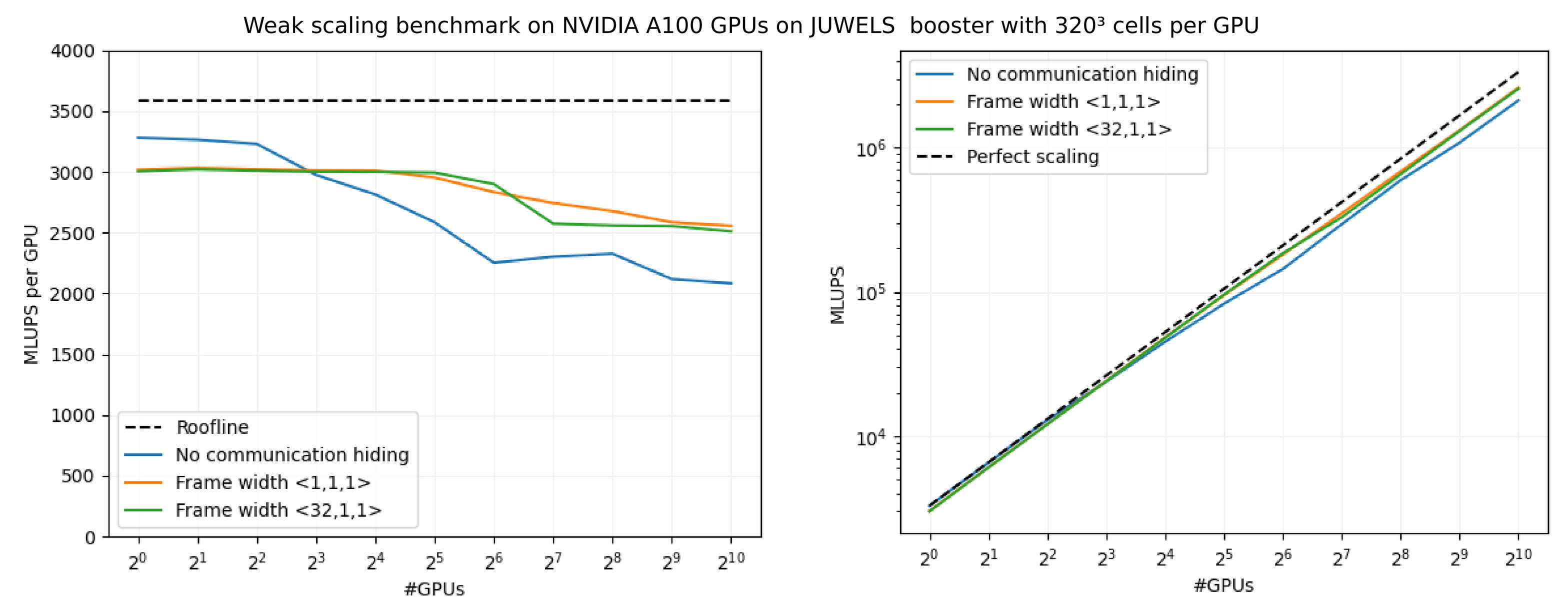


Optimization: Communication Hiding and Scalability

Communication Hiding for Sparse Data Structure:

- Divide block into interior and frame cells
- Algorithm:
  1) Start communication of ghost layers
  2) Run kernels on interior cells
  3) Wait for communication to finish
  4) Run kernels on frame cells
- ➔ ***Hide communication behind inner kernel runs***

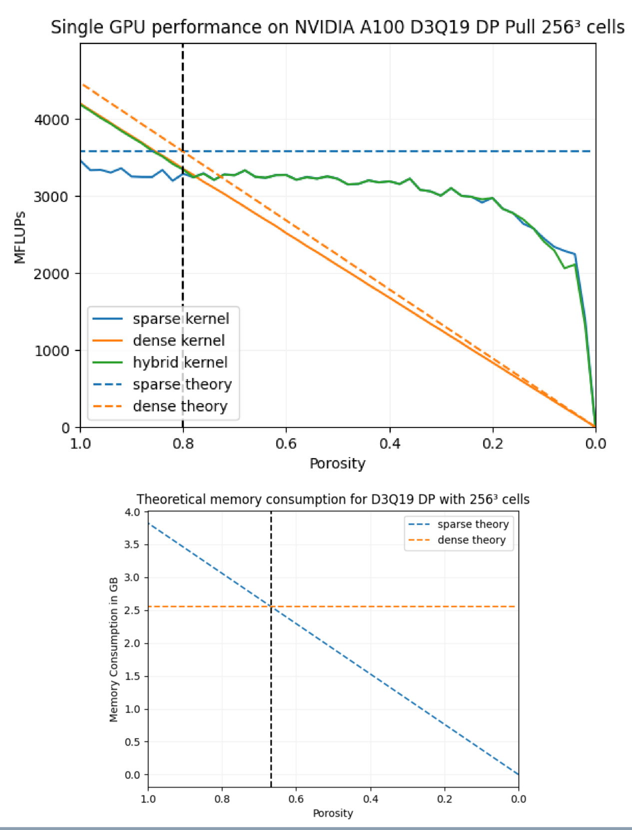




Optimization: Hybrid Data Structure

Domain Decomposition into blocks:

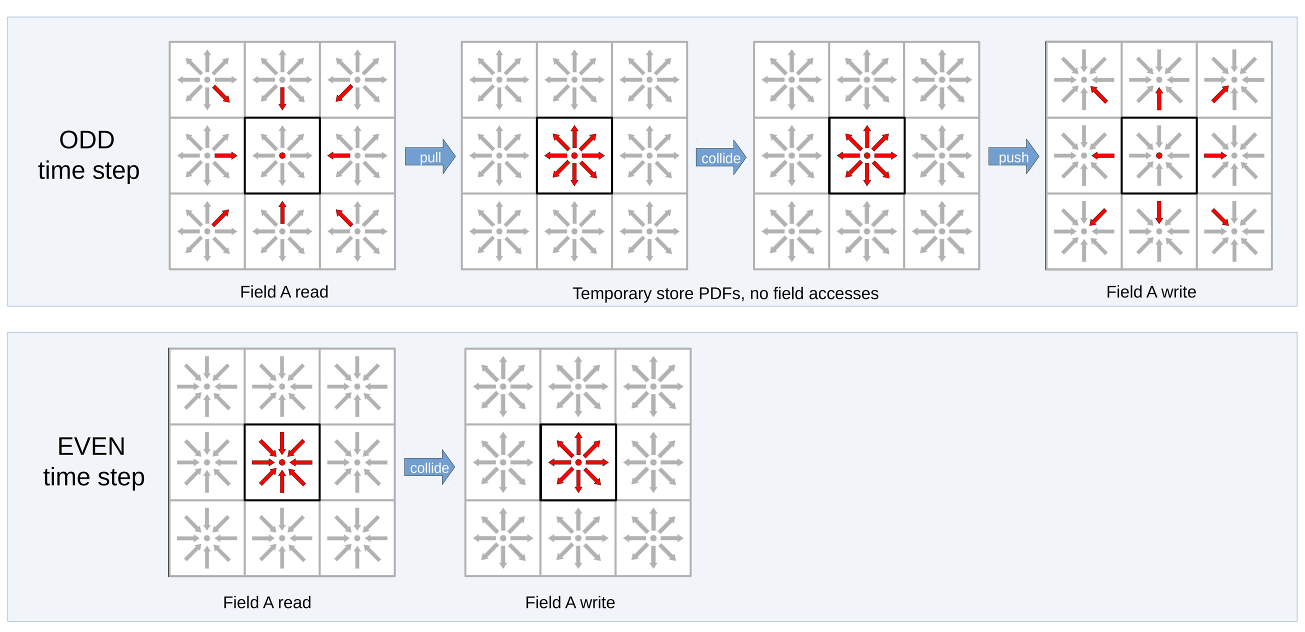
- **Indirect addressing (sparse)** ↔ **Direct addressing (dense)**
- Decision about data structure per block
  - ➔ **Generate sparse and dense kernels based on block porosity**
- Porosity = fluid cells / total cells
  - ➔ ***Performance superiority of sparse data structure at porosity < 0.8***
  - ➔ Theoretical memory consumption superiority at porosity < 0.66
- ***Best of both worlds: Hybrid data structure***

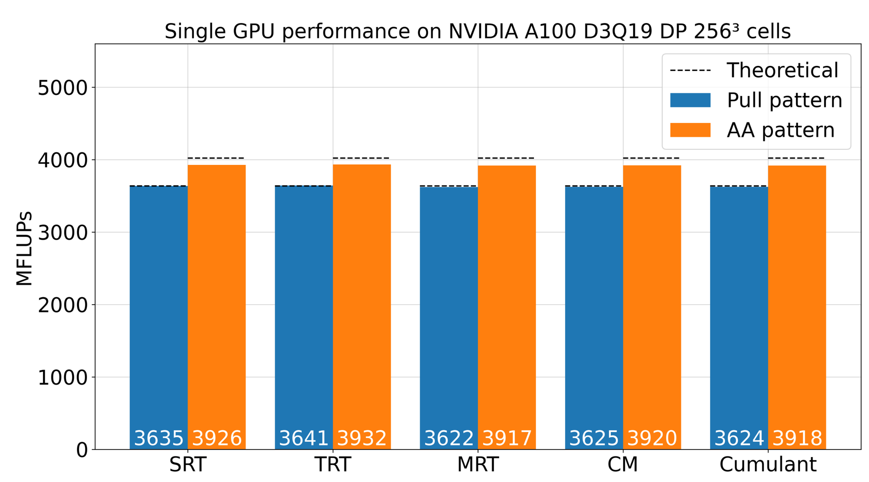


Optimization: AA Streaming Pattern

Theory of in-place Streaming - AA Pattern:

- Two alternating time steps
- **ODD time step**: PDFs are loaded and stored at the same position
  - ➔ For parallel updating no temporary PDF field is needed ➔ **~50% reduced memory consumption**
  - ➔ Avoid "write allocate" memory access on CPU ➔ ***Save 33% of memory accesses***
- **EVEN time step**:
  - ➔ No streaming, therefore no index list access needed ➔ ***Save ~10% of memory accesses***

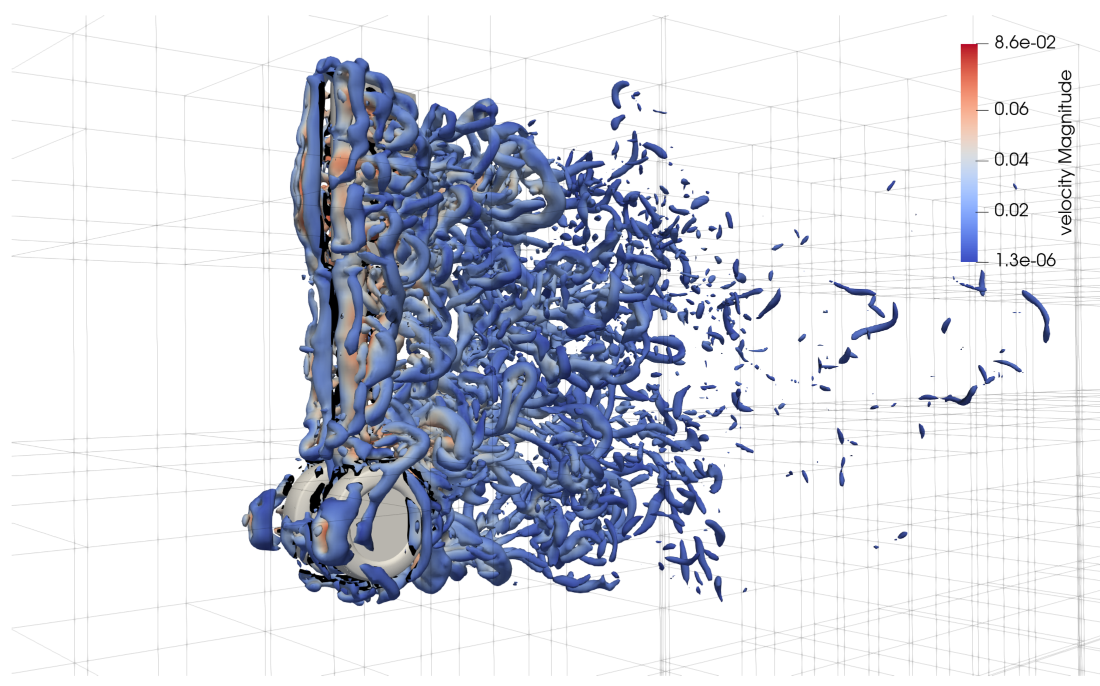




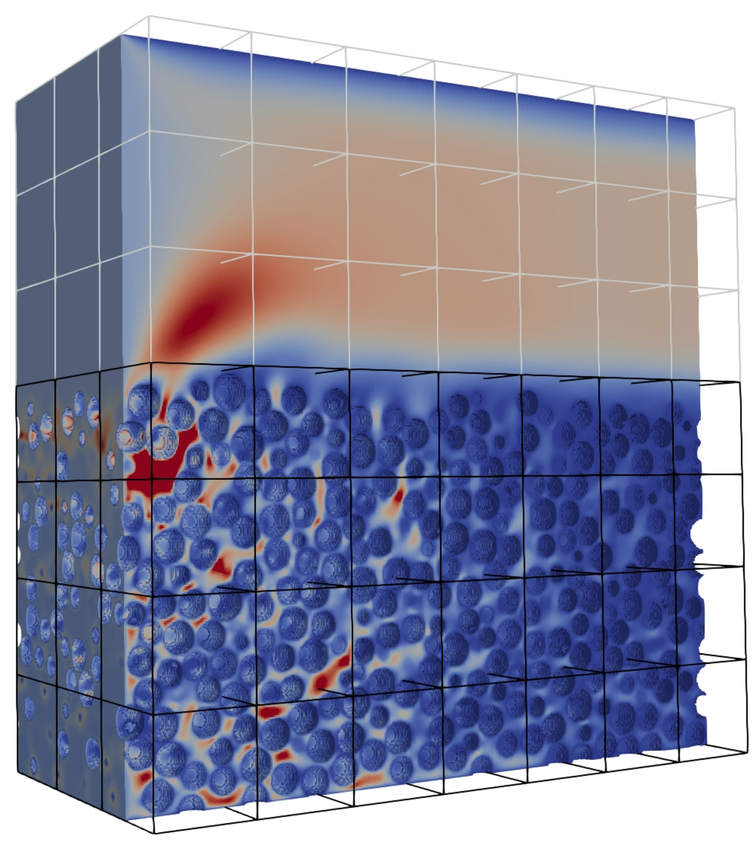
Sparse LBM Applications

LAGOON Test Case:

- SCALABLE application: **Aircraft landing gear**
- 720 Block (720 cores on JUWELS Cluser)
  - ➔ 188,743,680 cells
- 2 h run time for 3s simulation time
- Reynolds number: 1.59 * 10^6
- Sparse data structure only
- But: Low number of boundary cells
  - ➔ **Test case not suited for sparse LBM**



LAGOON test case visualized by Q-criterion. Grey lines are indicating domain decomposition into blocks



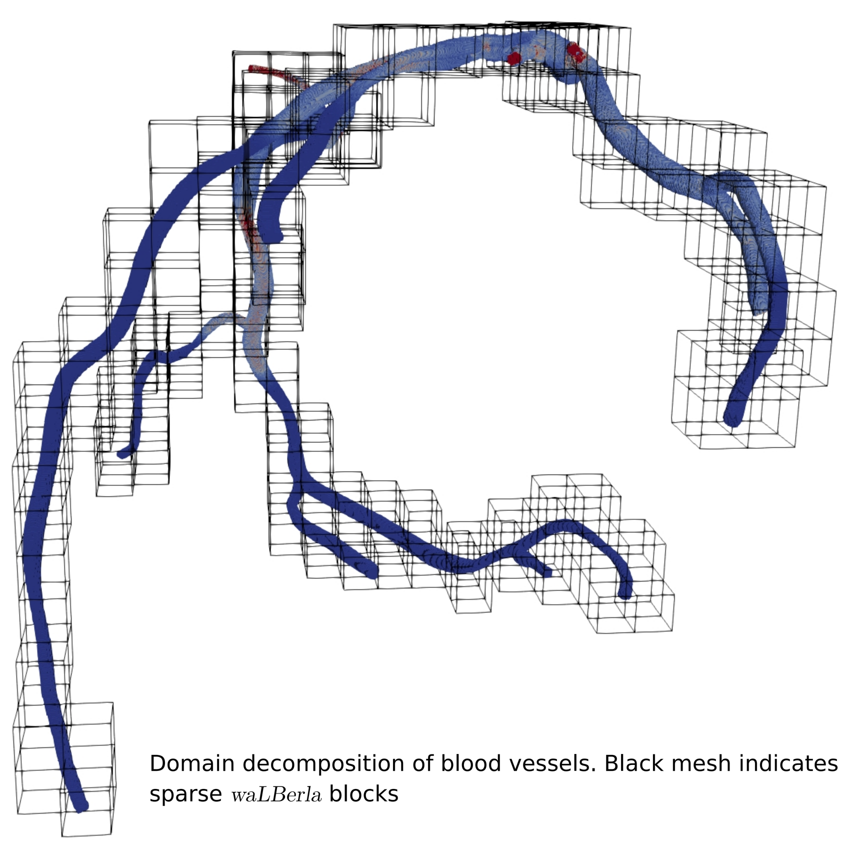
Riverbed velocity profile. Grey mesh indicates dense blocks, black mesh indicates sparse blocks

Artificial Riverbed Example:

- Porous media and free flow interaction
- Sediment porosity: 0.4 – 0.6
- Simulation automatically decides per block where to use sparse or dense data structure (porosity threshold < 0.8)
- Well suited case for hybrid data structure
  - ➔ ***Good performance and memory consumption for every part of the domain***

Blood Vessels:

- Complex geometry case with high number of small blocks
  - ➔ ***Exclude all blocks without fluid cells from domain***
- Remaining blocks have porosity between 0.01 and 1.0 (Ø 0.15)
  - ➔ ***Sparse Lattice Boltzmann Method very worth***
- ***Outlook: Balancing workload over processors based on block porosity***



Domain decomposition of blood vessels. Black mesh indicates sparse *waLBerla* blocks