WALBERLA, an ultra-scalable multi-physics simulation framework for piecewise regular grids

ParCo 2015, Edinburgh

September 3rd, 2015

Christian Godenschwager, Florian Schornbaum, Martin Bauer, Harald Köstler and Ulrich Rüde

Chair for System Simulation
Friedrich-Alexander Universität Erlangen-Nürnberg, Erlangen, Germany

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Outline

- Introduction
  - the *waLBerla* Simulation Framework
- Lattice Boltzmann method
  - domain decomposition & parallelization
  - uniform grids: performance / benchmarks
  - grid refinement
  - refined grids: performance / benchmarks
- Phase-field Methods
  - application & performance
- Outlook / Conclusion

The waLBerla framework

- originally designed for **CFD**
- using the **lattice Boltzmann method** (LBM)
- large-scale particulate flows: coupling with in-house rigid body physics engine *pe*
- Free surface module (VOF)
- designed as an **HPC** software framework:
  - scales from laptops to current petascale supercomputers
  - largest simulation: 1,835,008 processes (IBM Blue Gene/Q @ Jülich)
  - **hybrid parallelization**: MPI + OpenMP
  - **hand-crafted** compute kernels for optimal performance

The waLBerla framework

- written in **C++(11)**, optional **Python** interface
- support for different platforms (Linux, Windows, MacOS) and compilers (GCC, Intel XE, Visual Studio, llvm/clang, IBM XL)
- Large number of algorithms & supported platforms lead to high maintenance cost
- CI-workflow with automated testing on available target platforms

LBM on Uniform Grids

- Domain Decomposition & Parallelization
- Performance / Benchmarks

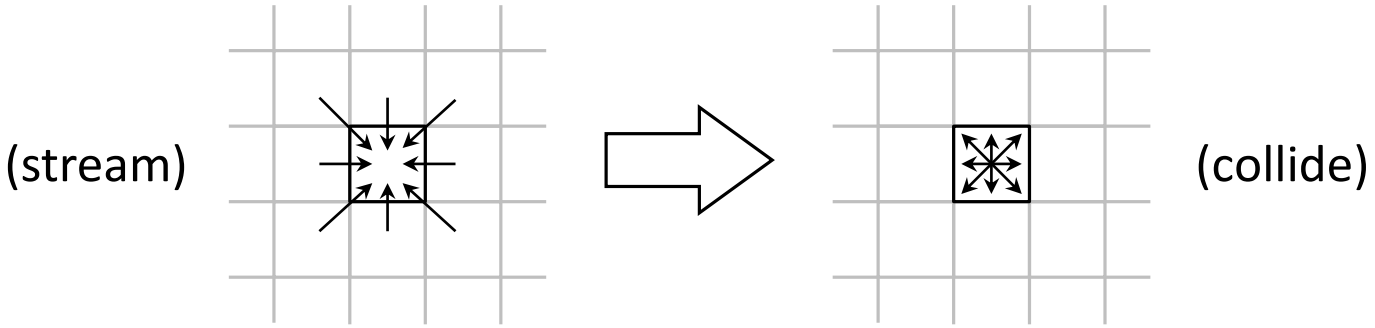
C. Godenschwager, F. Schornbaum, M. Bauer, H. Köstler, and U. Rüde, *A Framework for Hybrid Parallel Flow Simulations with a Trillion Cells in Complex Geometries*, SC13, Denver

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

The lattice Boltzmann method

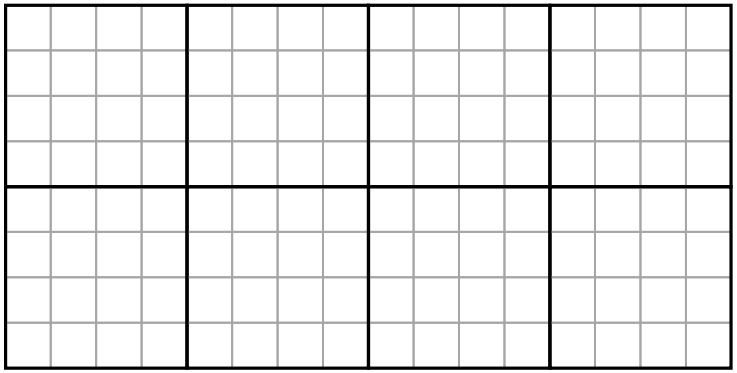
- regular grid with multiple particle distribution functions (= scalar values) per cell [D2Q9, D3Q19, D3Q27, …]
- explicit method → time stepping
- two steps: stream (neighbors) & collide (cell-local)



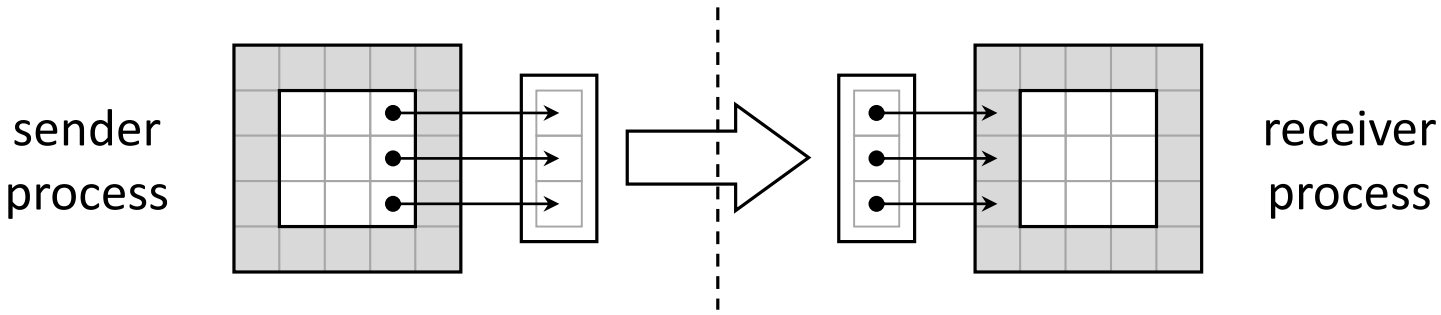
- For the collision, different operators exist: SRT, TRT, MRT.
- Macroscopic quantities (velocity, density, …) can be calculated from the particle distribution functions.

Uniform Grids

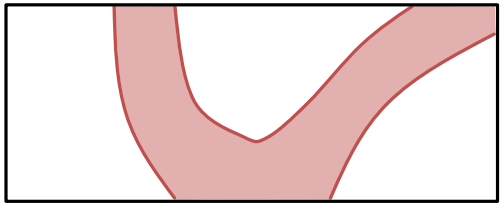
- Domain Decomposition:
  - regular decomposition into blocks containing uniform grids



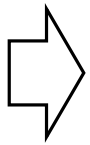
- Parallelization:
  - data exchange on borders between blocks via ghost layers

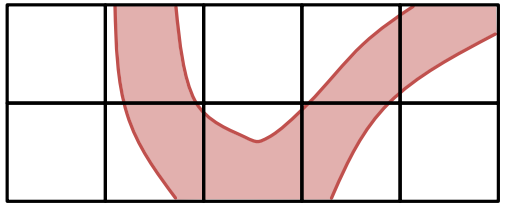


Uniform Grids



geometry given by surface mesh





domain decomposition into blocks

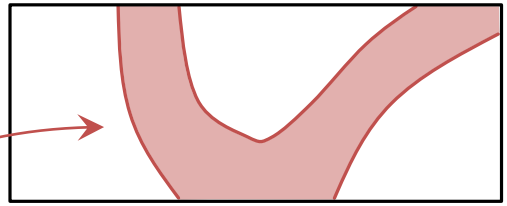
load balancing

empty blocks are discarded

Load balancing can be based on either space-filling curves (Z-order/Morton order, Hilbert curve) using the underlying forest of octrees or graph partitioning (METIS, …).

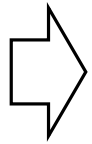
Whatever fits best the needs of the simulation.

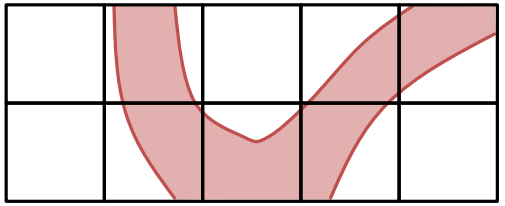
Uniform Grids



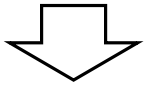
geometry given by surface mesh

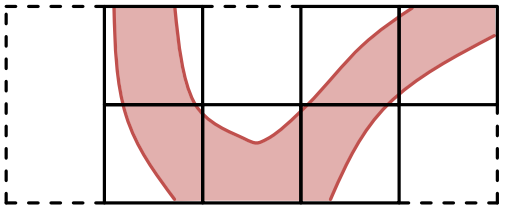
flow simulation only in here
(example: complex geometry
of an artery)





domain decomposition into blocks





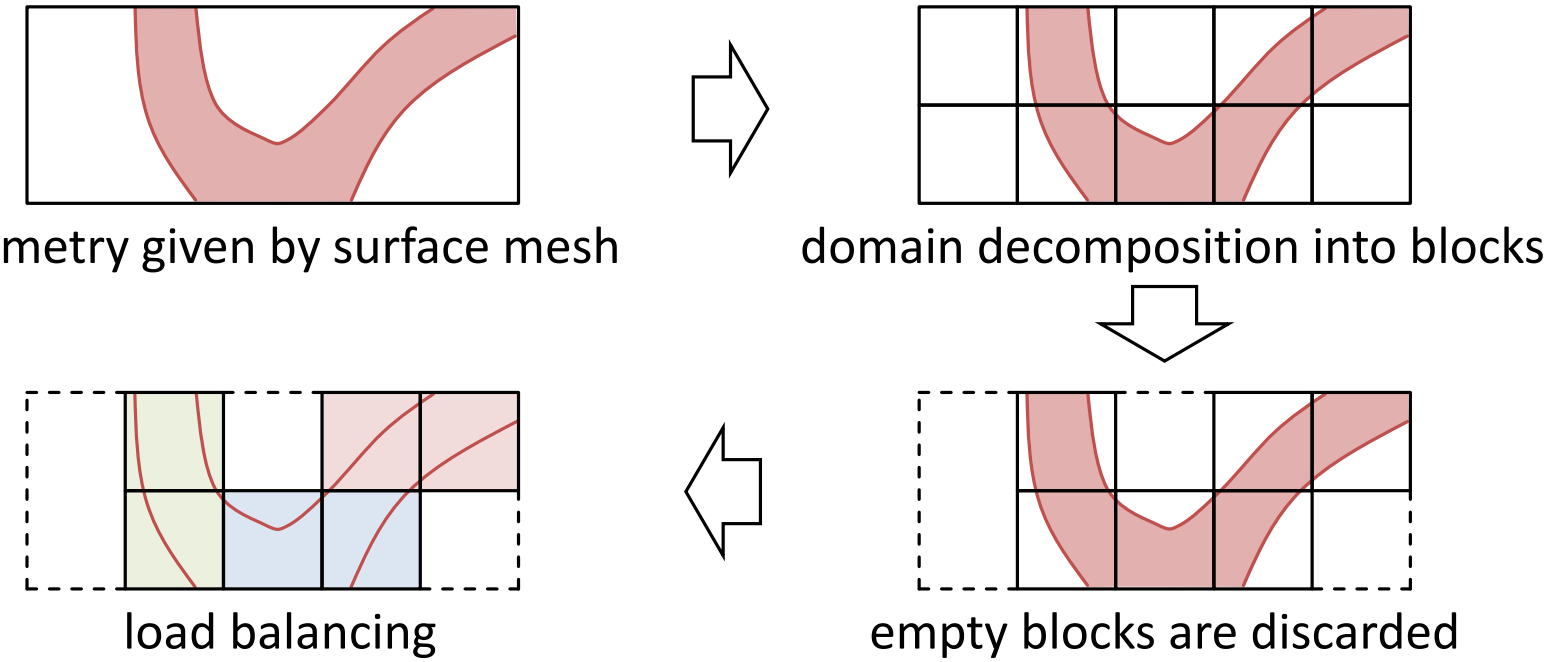
empty blocks are discarded

load balancing

Load balancing can be based on either space-filling curves (Z-order/Morton order, Hilbert curve) using the underlying forest of octrees or graph partitioning (METIS, …).

Whatever fits best the needs of the simulation.

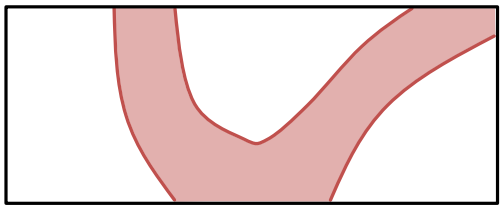
Uniform Grids



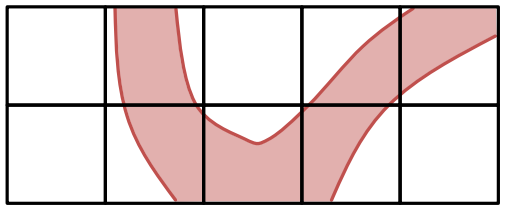
Load balancing by space-filling curves
(Z-order/Morton order, Hilbert curve)
or graph partitioning (METIS, …).

Whatever fits best the needs of the simulation.

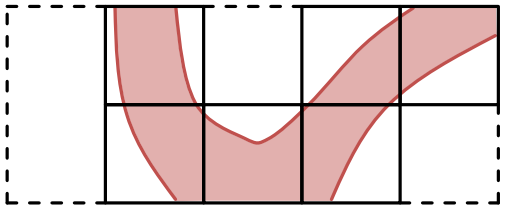
Uniform Grids



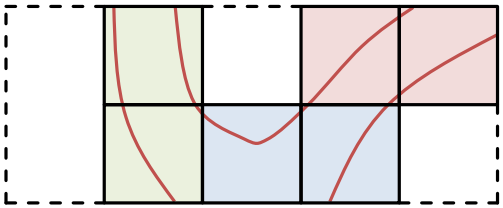
geometry given by surface mesh



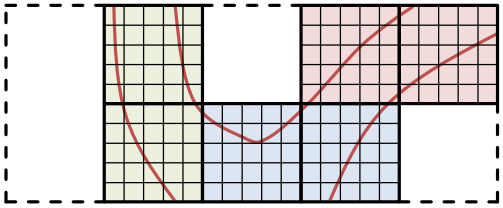
domain decomposition into blocks



empty blocks are discarded



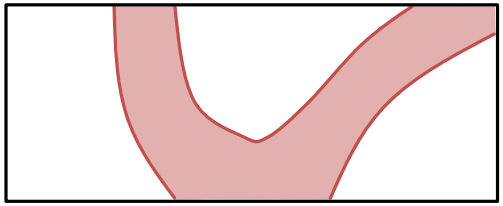
load balancing



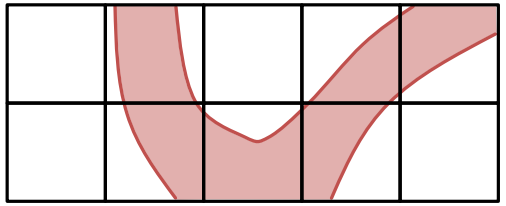
allocation of block data (→ grids)

The domain decomposition and load balancing can be performed during the actual simulation … OR …

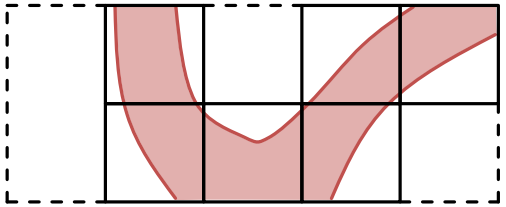
Uniform Grids



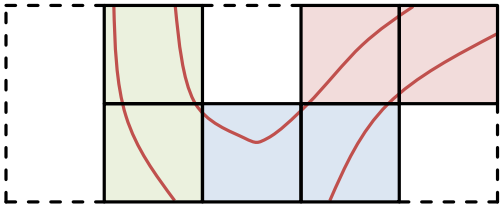
geometry given by surface mesh



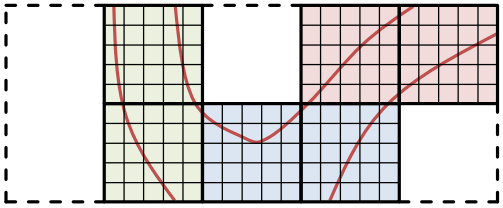
domain decomposition into blocks



empty blocks are discarded



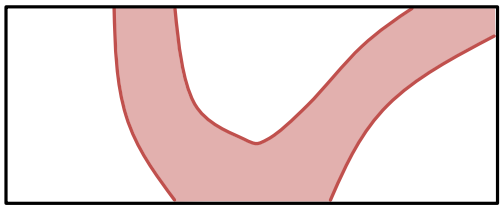
load balancing



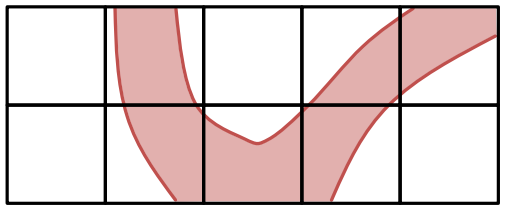
allocation of block data (→ grids)

The domain decomposition and load balancing can be performed during the actual simulation … OR …

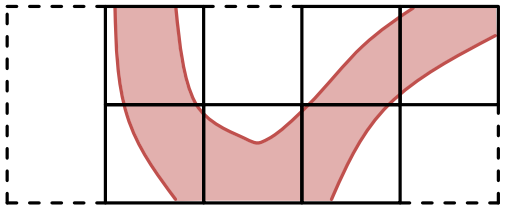
Uniform Grids



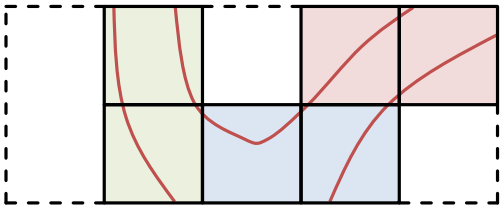
geometry given by surface mesh



domain decomposition into blocks

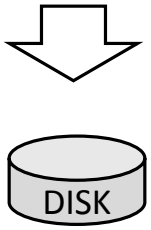


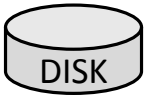
empty blocks are discarded

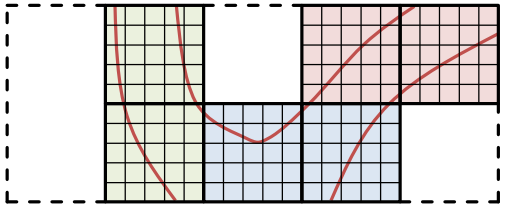


load balancing

separation of domain partitioning from simulation







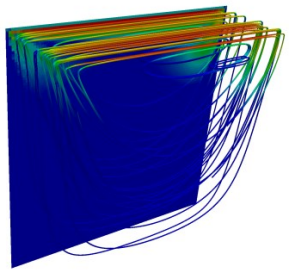
allocation of block data (→ grids)

file size: kilobytes to few megabytes

Uniform Grids - Performance

- Benchmark Environments:
  - JUQUEEN
    - Blue Gene/Q, 459K cores, 1 GB/core
    - compiler: IBM XL / IBM MPI
  - SuperMUC (Sandy Bridge)
    - Intel Xeon, 147K cores, 2 GB/core
    - compiler: Intel XE / IBM MPI

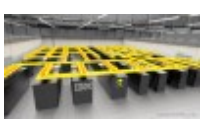
- Benchmark (LBM D3Q19):

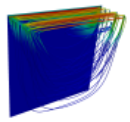


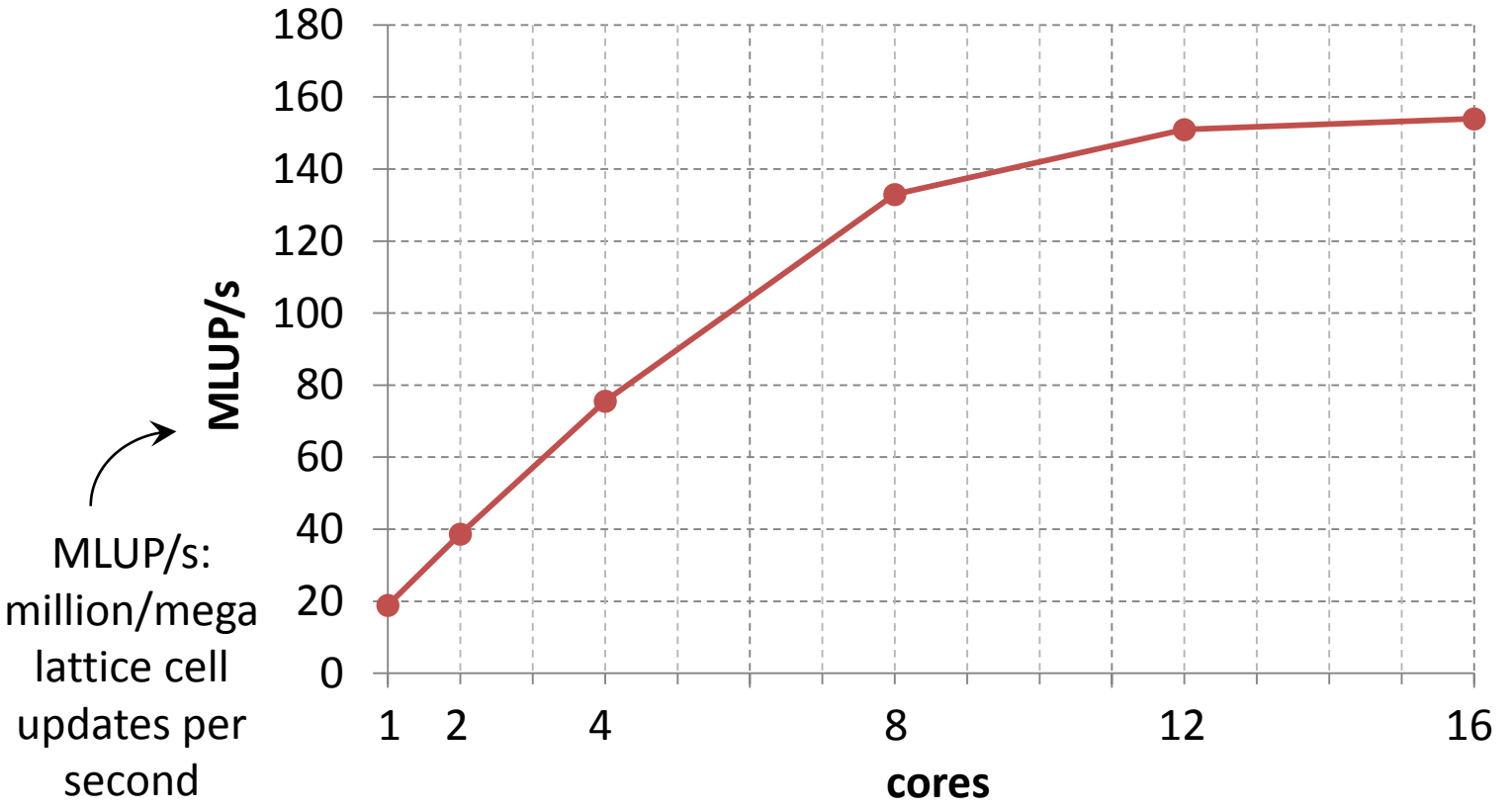
lid-driven cavity
weak scaling
(= const. number
of cells per core)

Uniform Grids - Performance

- SuperMUC – single node (3.34 million cells per core)



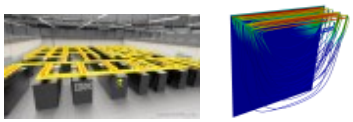


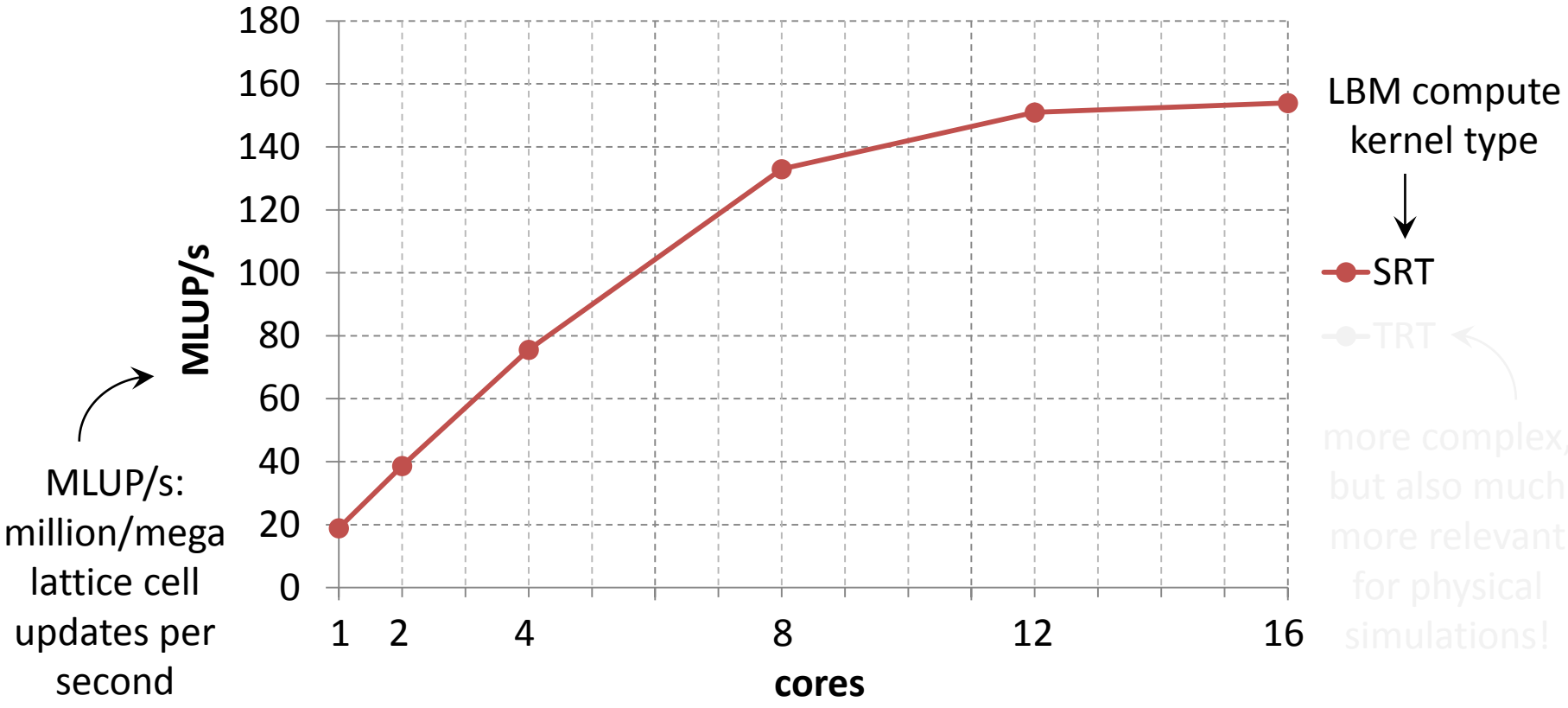


⇒ LBM: low FLOPs to bytes ratio → memory intensive

Uniform Grids - Performance

- SuperMUC – single node (3.34 million cells per core)

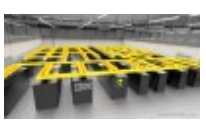


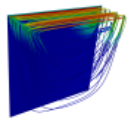


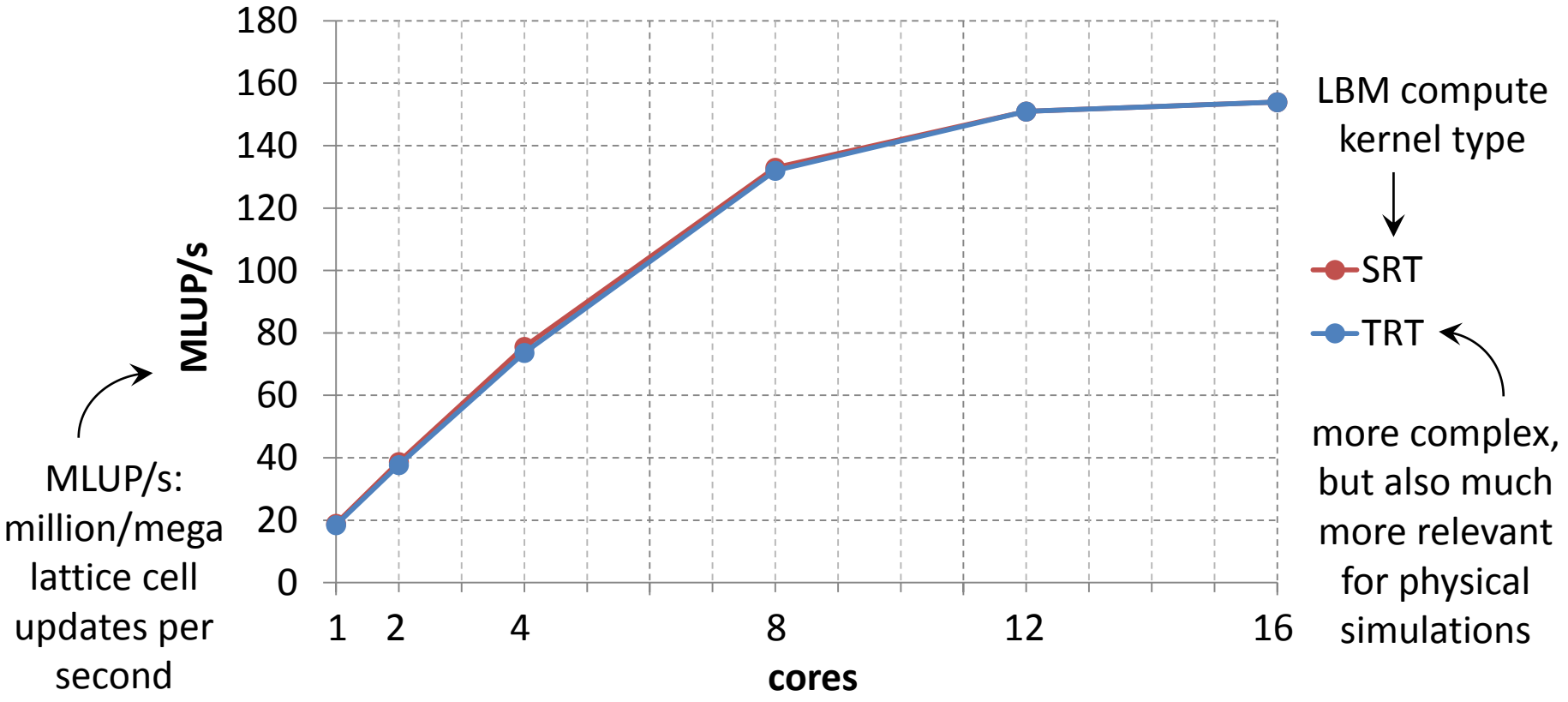
⇒ LBM: low FLOPs to bytes ratio → memory intensive

Uniform Grids - Performance

- SuperMUC – single node (3.34 million cells per core)



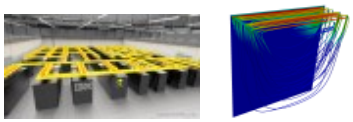


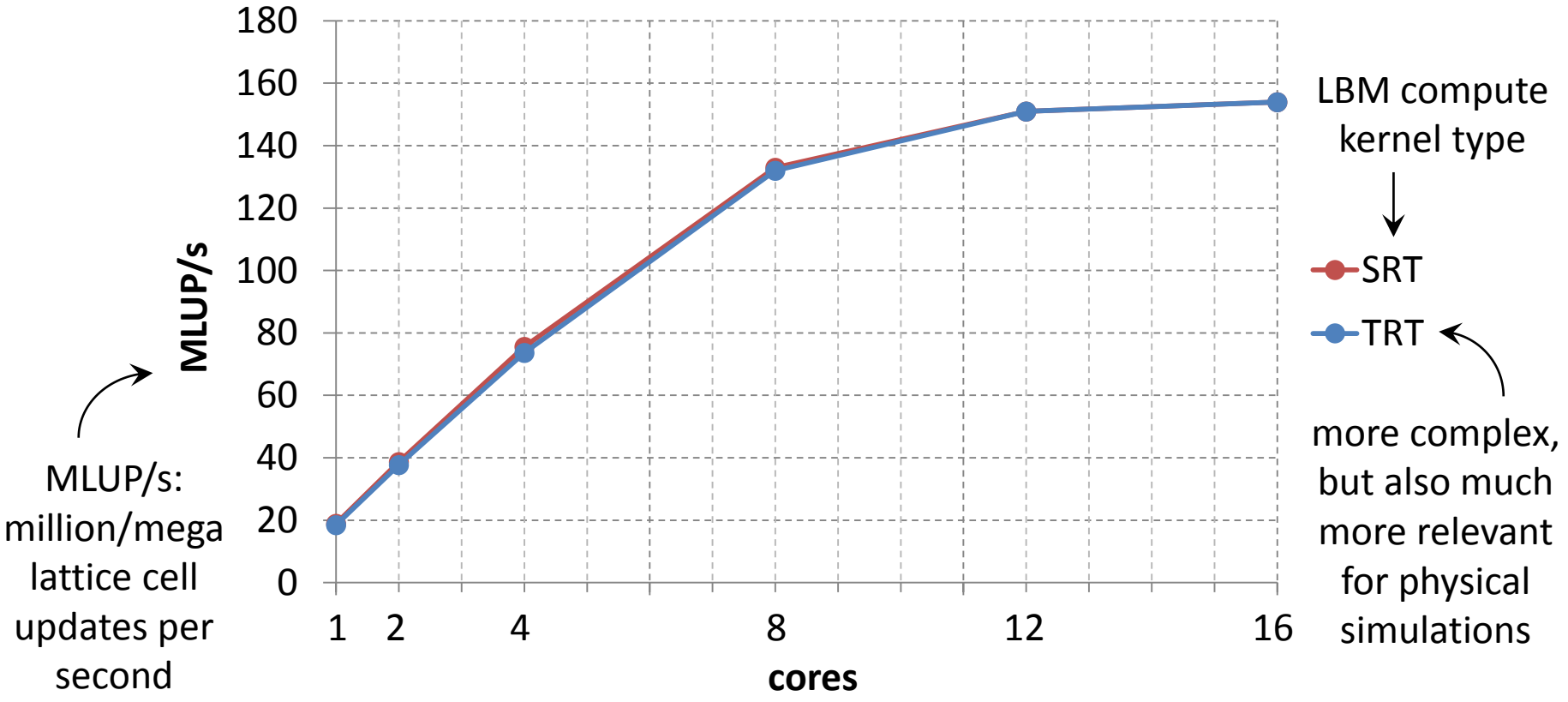


⇒ LBM: low FLOPs to bytes ratio → memory intensive

Uniform Grids - Performance

- SuperMUC – single node (3.34 million cells per core)

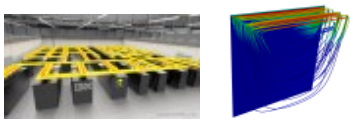


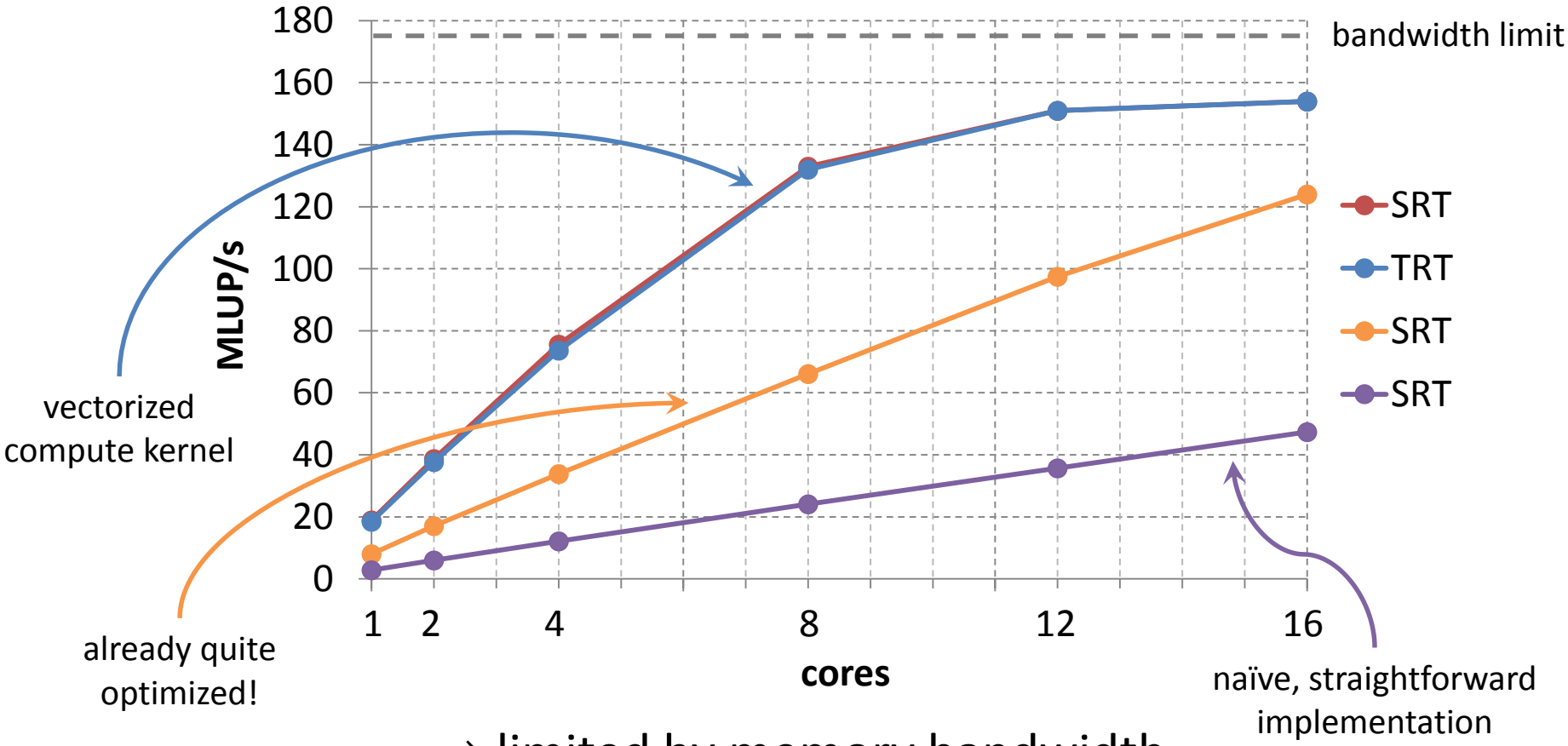


⇒ LBM: low FLOPs to bytes ratio → memory intensive

Uniform Grids - Performance

- SuperMUC – single node (3.34 million cells per core)

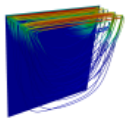


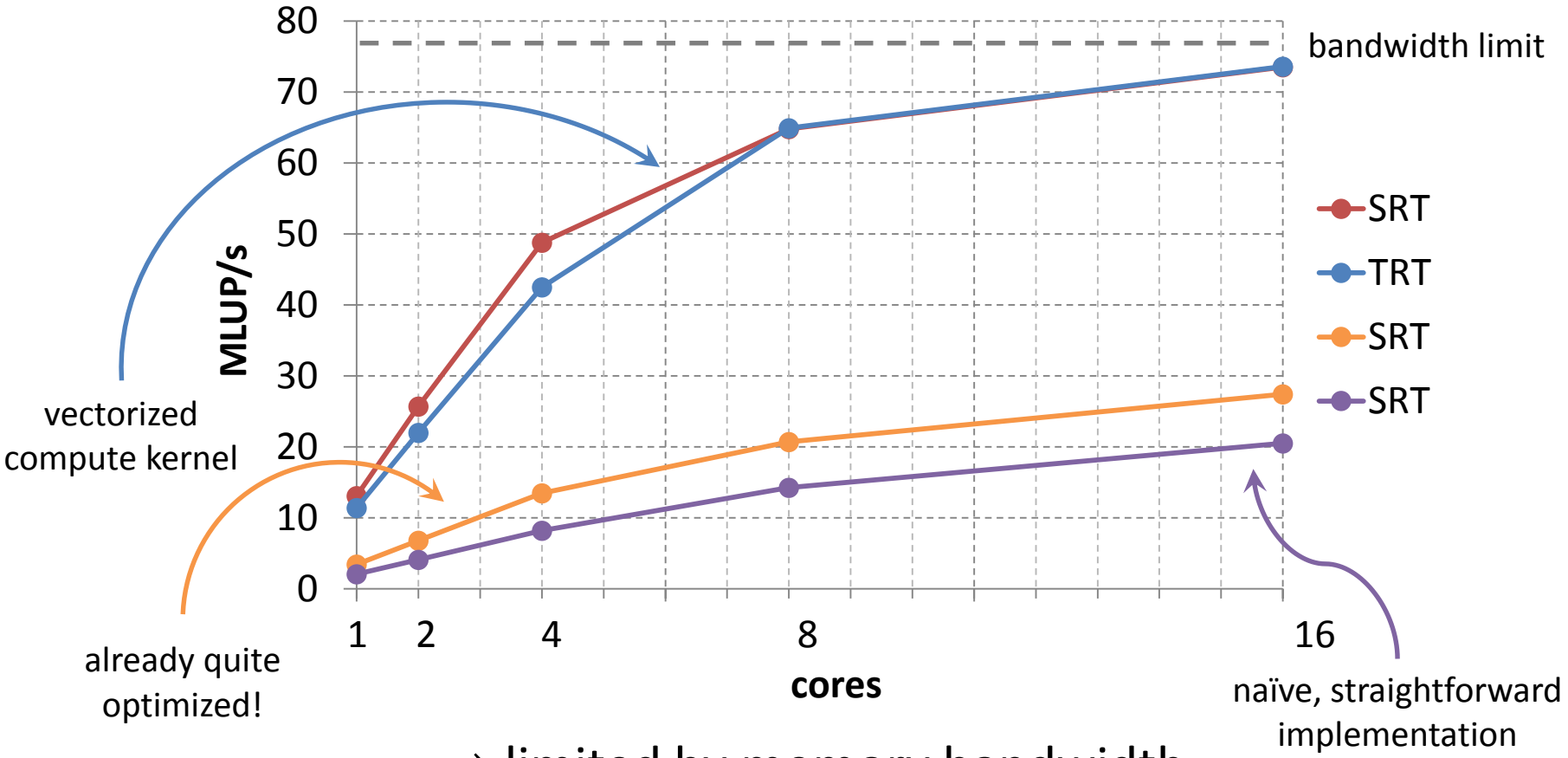


⇒ limited by memory bandwidth

Uniform Grids - Performance

- JUQUEEN – single node (1.73 million cells per core)

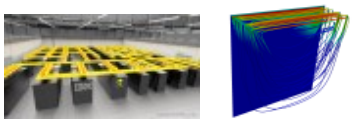


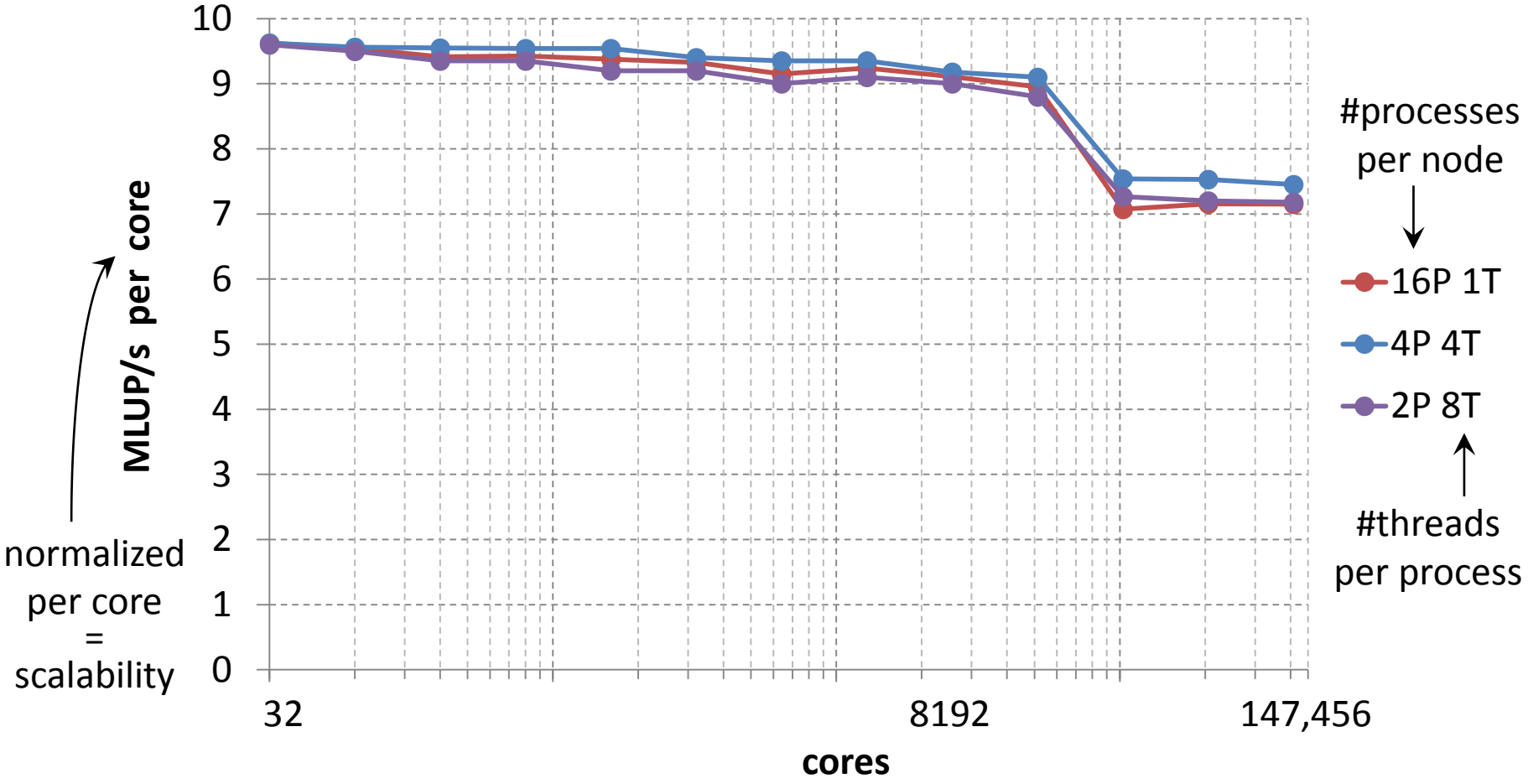


⇒ limited by memory bandwidth

Uniform Grids - Performance

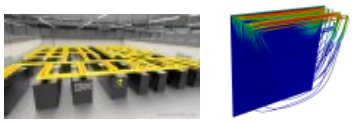
- SuperMUC – TRT kernel (3.34 million cells per core)

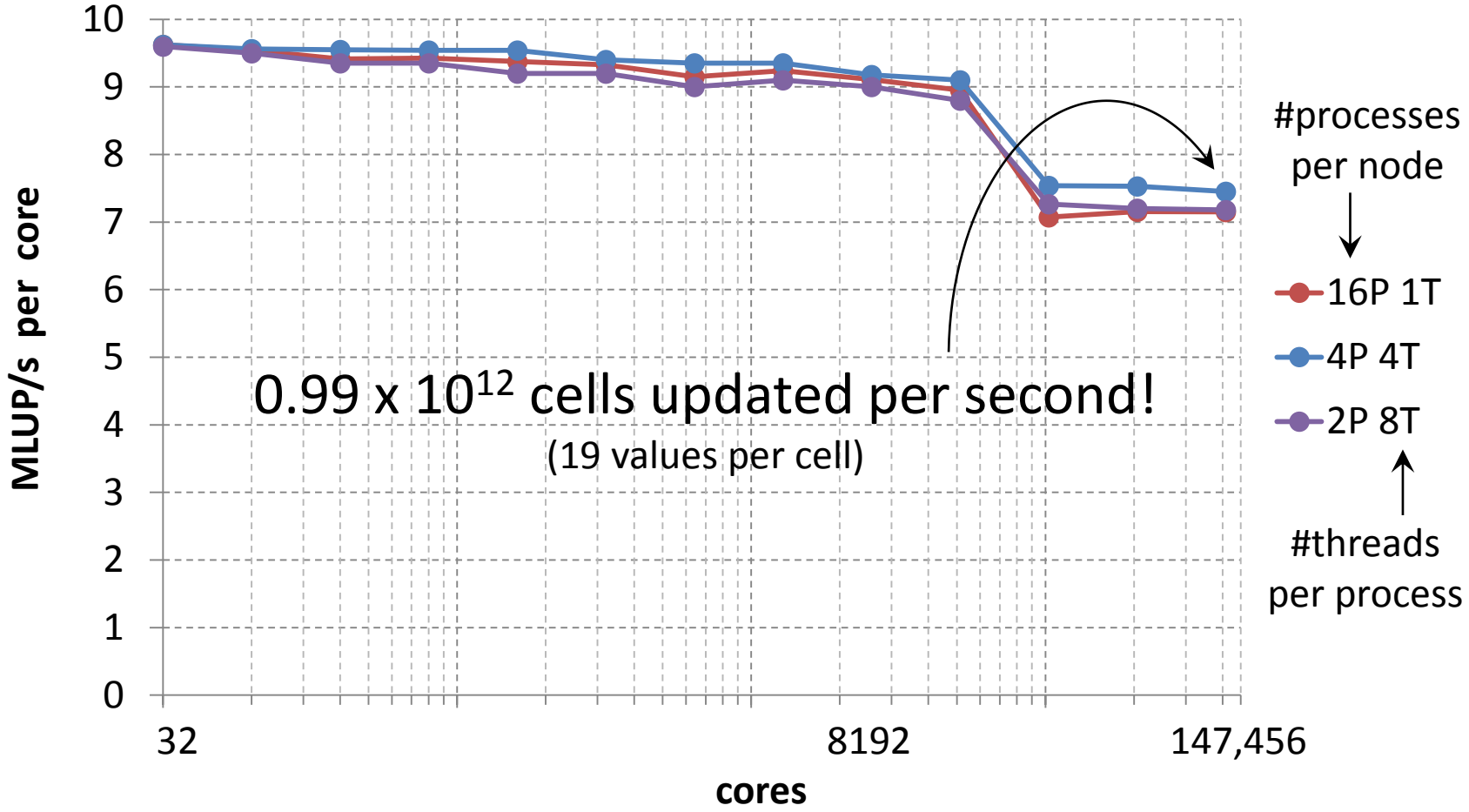




Uniform Grids - Performance

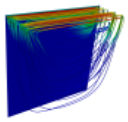
- SuperMUC – TRT kernel (3.34 million cells per core)

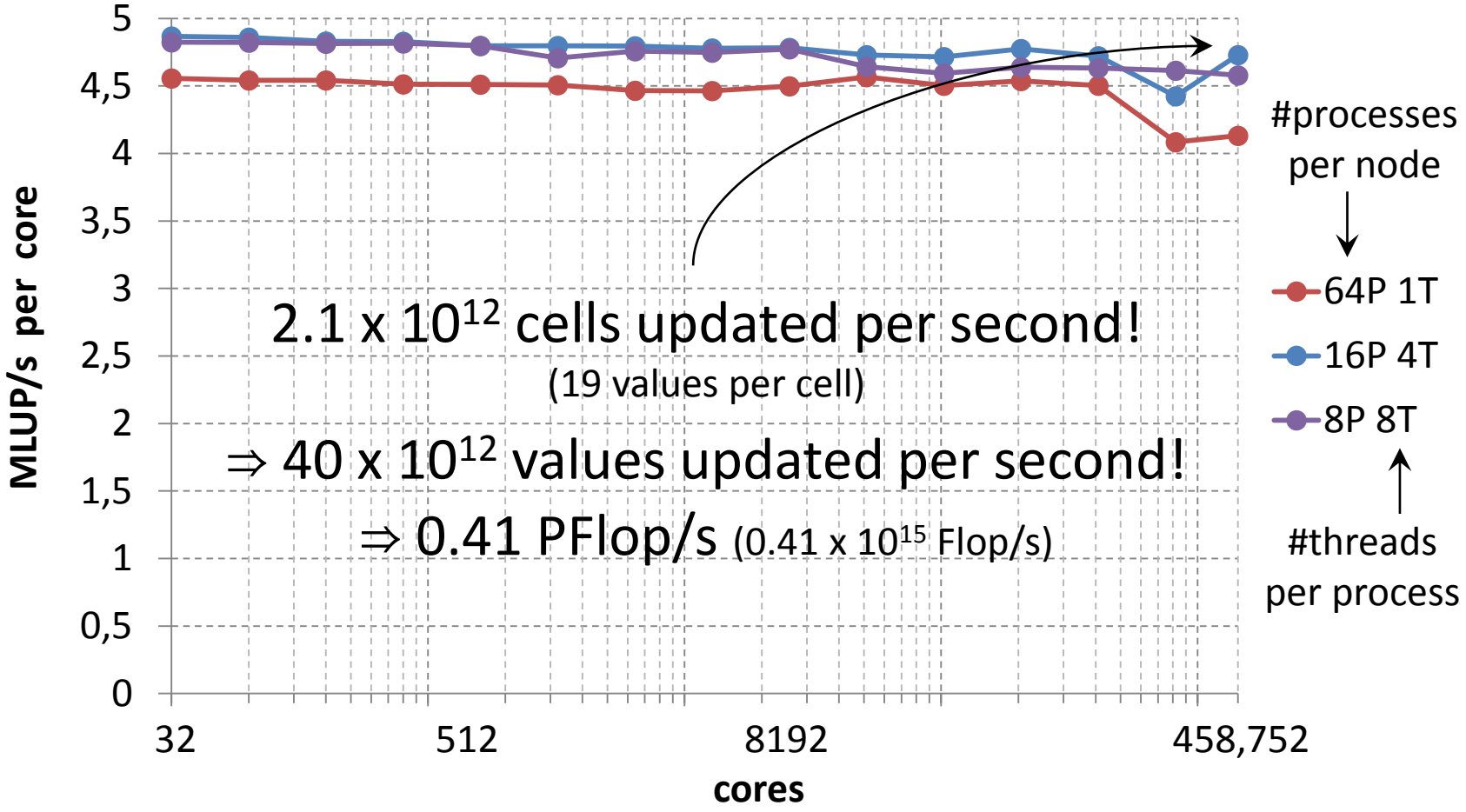




Uniform Grids - Performance

- JUQUEEN – TRT kernel (1.73 million cells per core)





Statically Refined Grids

- Lattice Boltzmann & Grid Refinement
- Domain Decomposition & Load Balancing
- Performance / Benchmarks

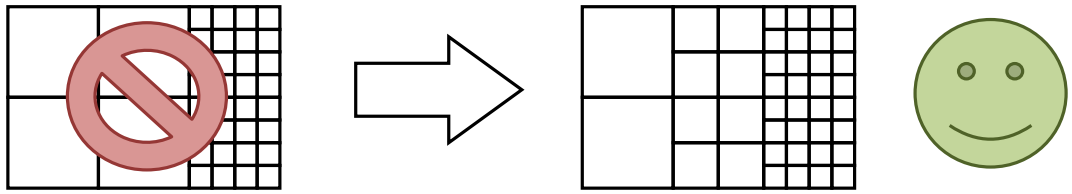
F. Schornbaum and U. Rüde, 2015, *Massively Parallel Algorithms for the Lattice Boltzmann Method on Non-uniform Grids*, arXiv:1508.07982, submitted to SISC

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Lattice Boltzmann & Grid Refinement

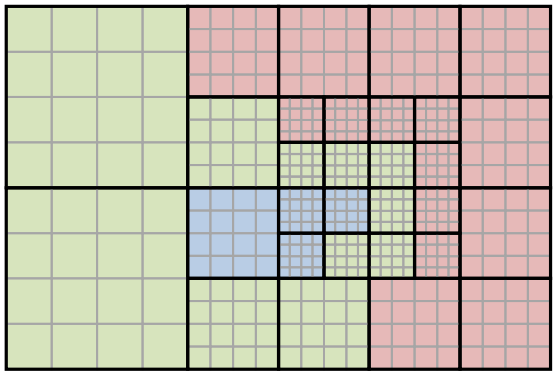
- Almost all grid refinement schemes for the lattice Boltzmann method rely on a **2:1 balance** between neighboring cells:



- memory requirement increases by a factor of 8 and the generated workload by a factor of 16
- **twice as many time steps on the fine grid** as on the next coarser grid

Statically Refined Grids

- Domain Decomposition:
  - "blocks containing regular grids distributed among all processes"
  - ⇒ **distributed forest of octrees** (→ 2:1 balance)
  - Each process only knows about its **own blocks** and their **neighbors**, but has no information about the **rest of the domain**.
  - ⇒ perfectly distributed data structure that **scales to huge numbers of processes** without any runtime overhead



- Property of this Distributed Data Structure:
  - can be viewed as a graph: each block is connected to all of its neighboring blocks → enables all kinds of graph algorithms

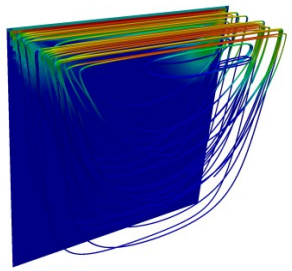
Statically Refined Grids

- Comparison with Uniform LBM:
  - The **setup phase remains unchanged**:
    The decoupling (via a file) of the domain decomposition & initial load balancing from the actual simulation is still possible.
  - All the refinement “magic” happens during communication (fine and coarse blocks share ghost layer regions → refinement requires **multiple ghost layers** per block).
  - **Time stepping becomes more complicated**
  - **All the compute kernels remain unchanged**

Statically Refined Grids

- Benchmark Environments:
  - JUQUEEN
    - Blue Gene/Q, 459K cores, 1 GB/core
    - compiler: IBM XL / IBM MPI
  - SuperMUC
    - Intel Xeon, 147K cores, 2 GB/core
    - compiler: Intel XE / IBM MPI

- Benchmark (LBM D3Q19 TRT):

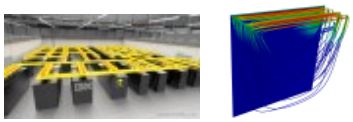


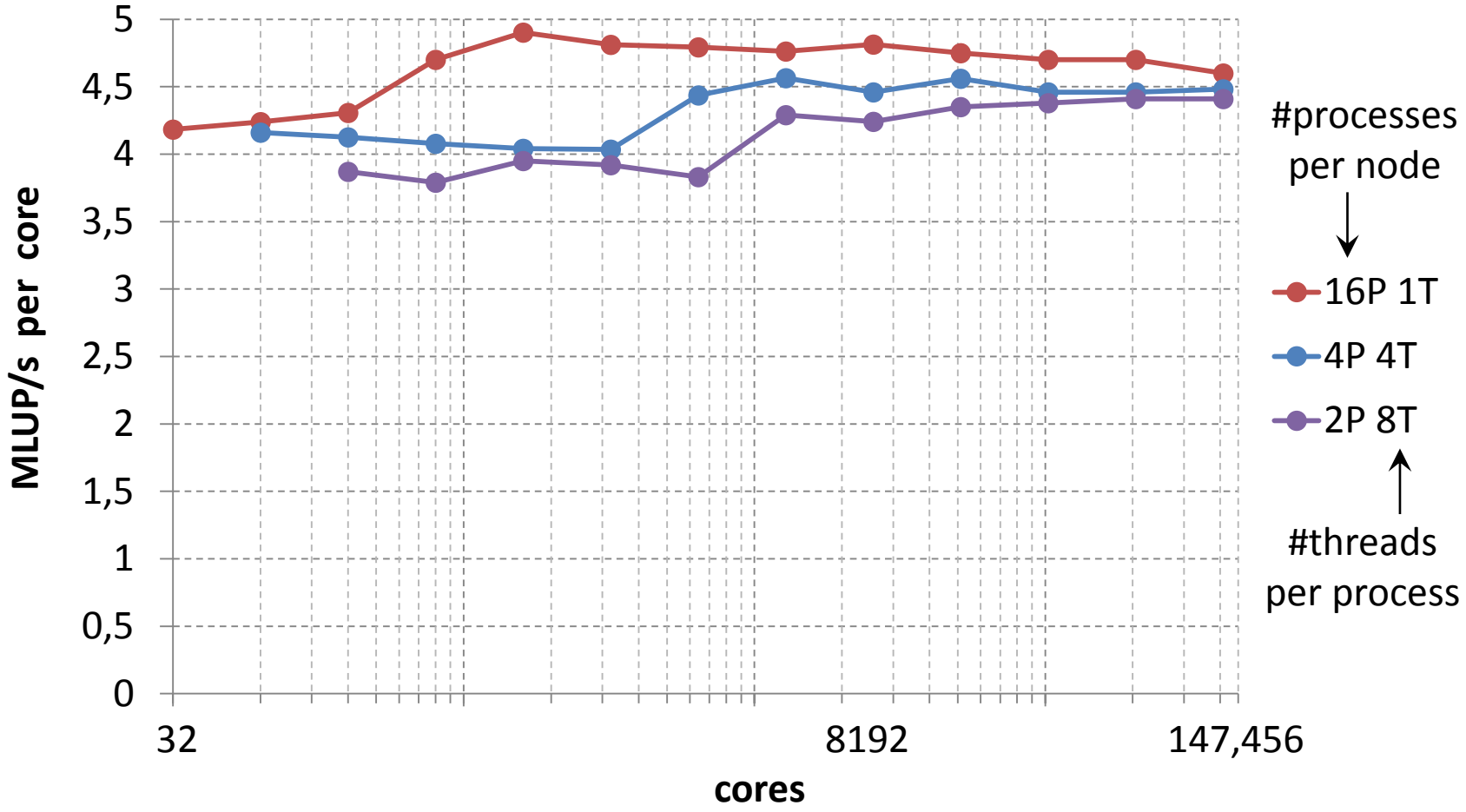
- lid-driven cavity
- **weak scaling**
- 6.69 blocks per process
- **4 grid levels**

- finest level …
  - … covers 1.4% of space
  - … generates 80% of workload
- coarsest level …
  - … covers 78% of space
  - … generates 1.1% of workload

Statically Refined Grids

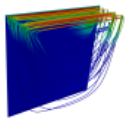
- SuperMUC – TRT kernel (3.34 million cells per core)

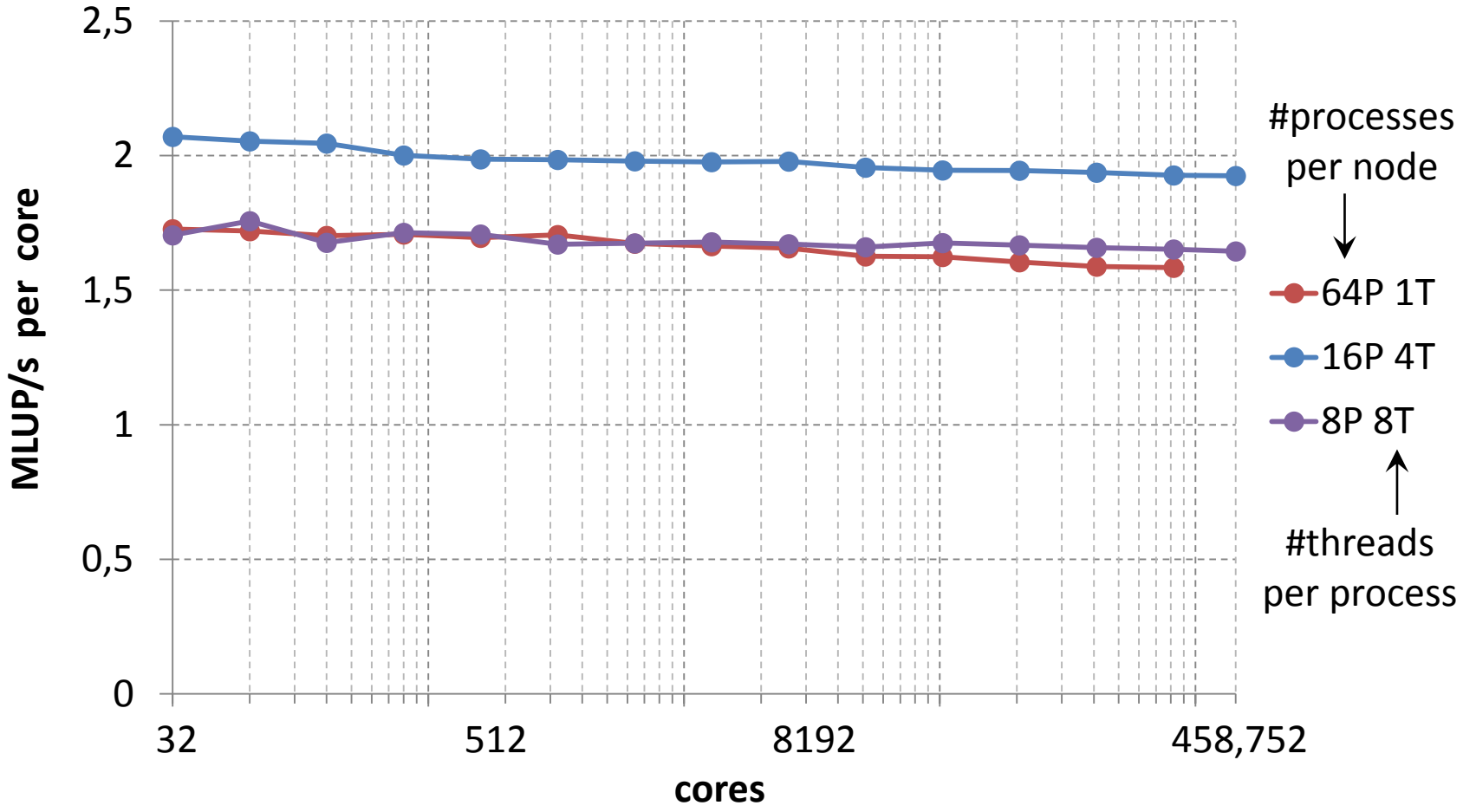




Statically Refined Grids

- JUQUEEN – TRT kernel (1.81 million cells per core)



Statically Refined Grids

- LBM with refinement only achieves half the number of cell updates per second as regular LBM without refinement.
- Refinement Overhead:
  - much more complex communication patterns which additionally involve interpolation
  - more complex time stepping scheme
  - multiple blocks per process ⇔ smaller blocks

BUT

LBM with grid refinement requires

much **less memory** and far **fewer cell updates**

Phase-field Method

- Application
- Performance

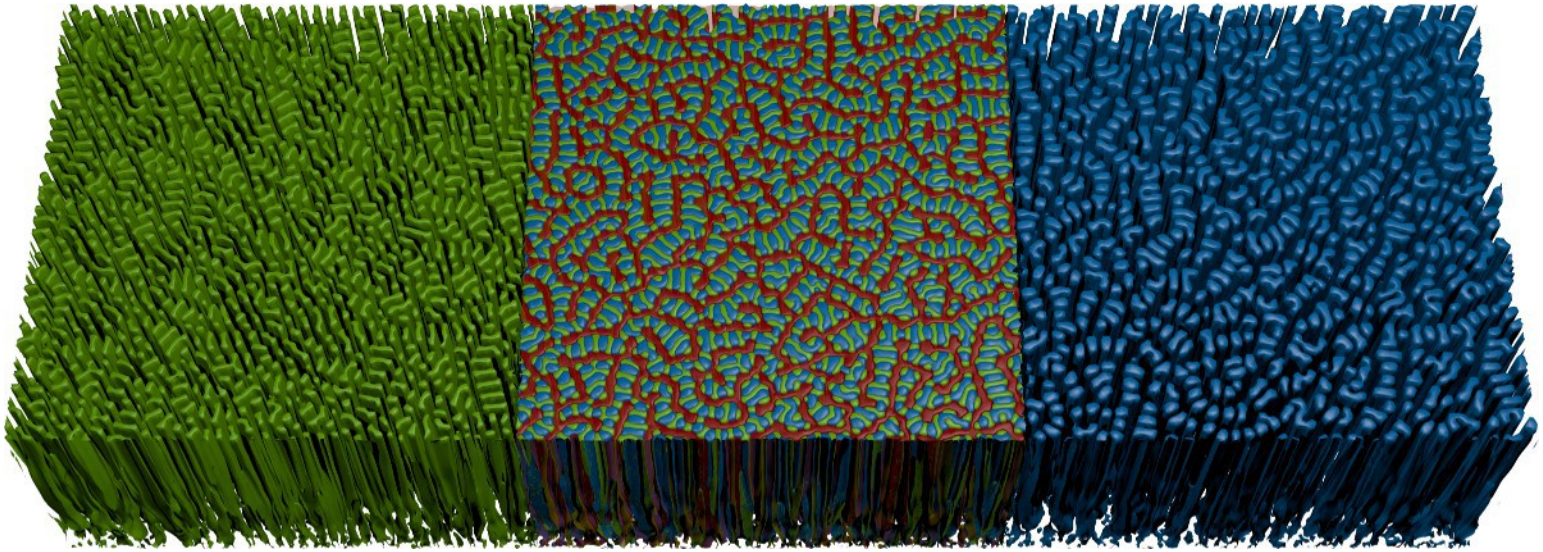
M. Bauer et al., *Massively Parallel Phase-Field Simulations for Ternary Eutectic Directional Solidification,* arXiv:1506.01684, accepted for SC15

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

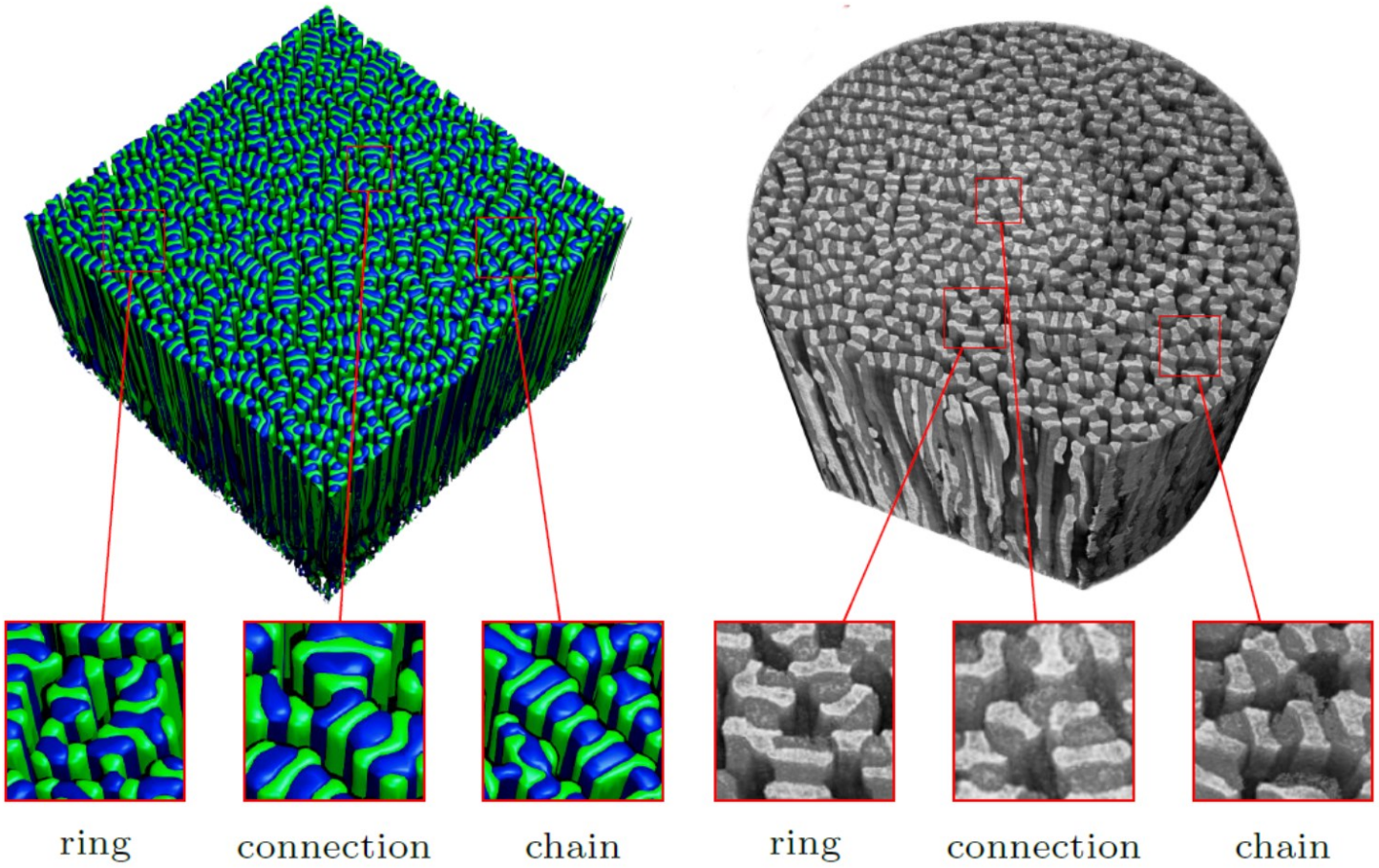
Phase-field Method

- Simulation of solidification of alloys (ternary eutectics)
- Simulates microstructure evolution
- Collaboration with material sciences group of Prof. Nestler, KIT



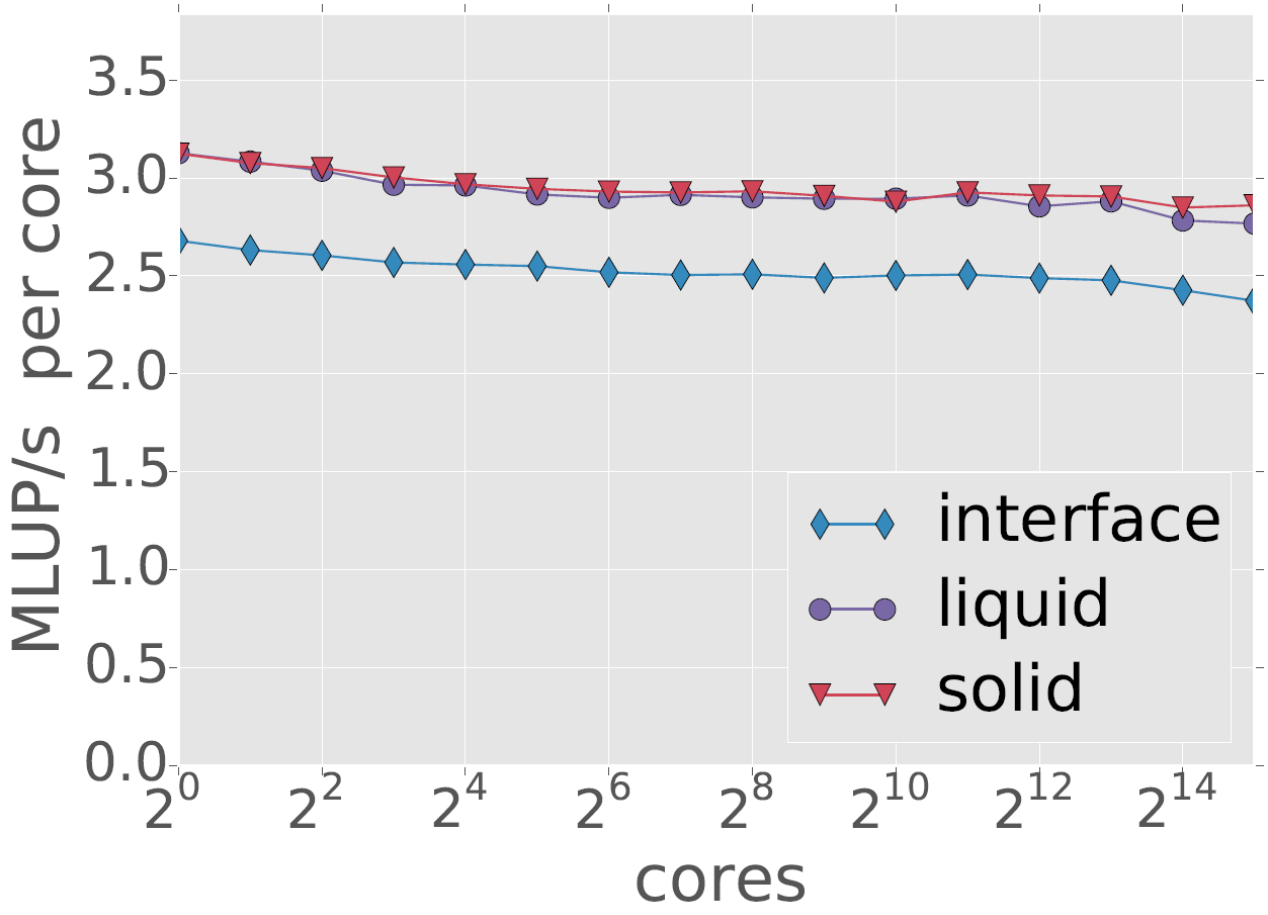
- Complex, **compute intensive** stencil code
- Ghost layer-based communication pattern (same as LBM)

Phase-field Method



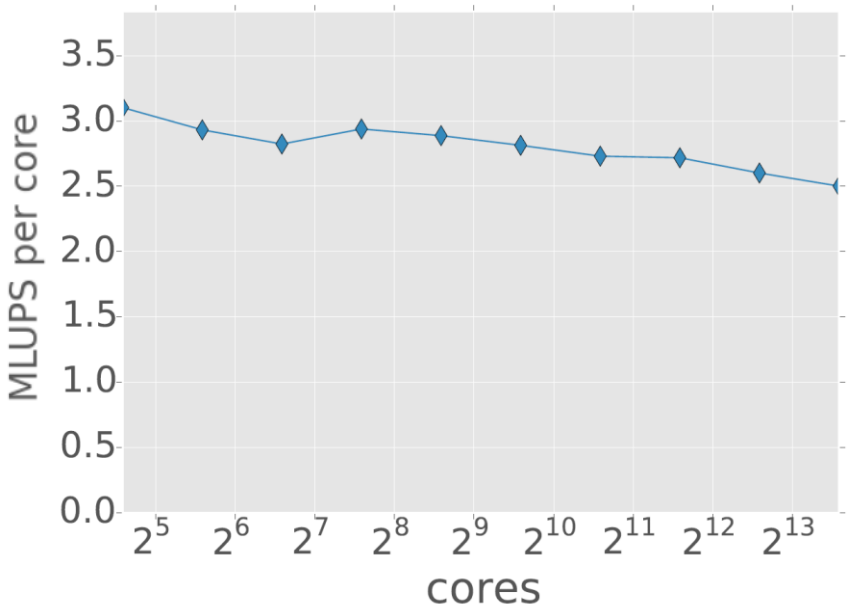
simulation <-> experiment

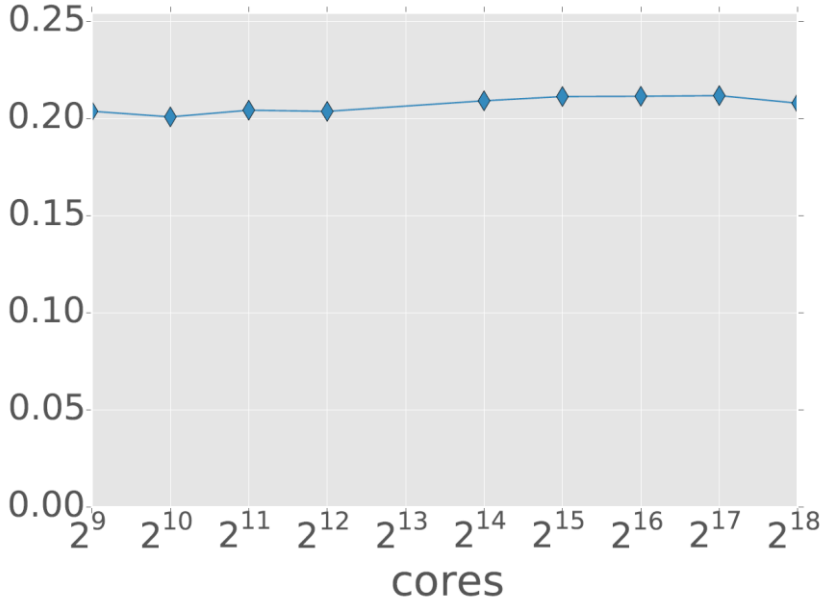
Phase-field Method



Weak scaling SuperMUC (Sandy Bridge)

Phase-field Method



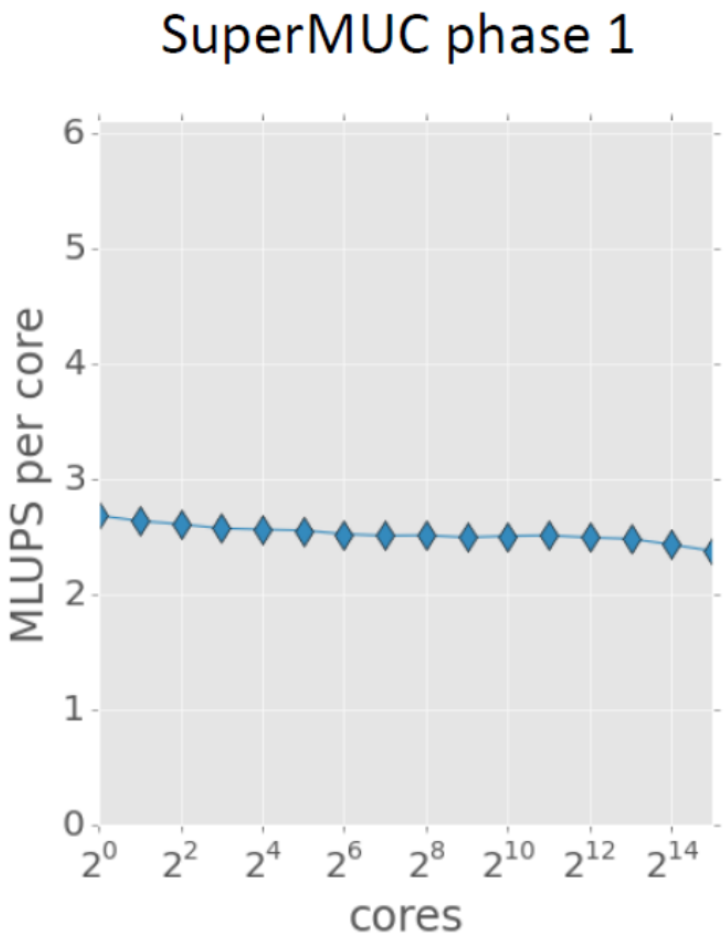


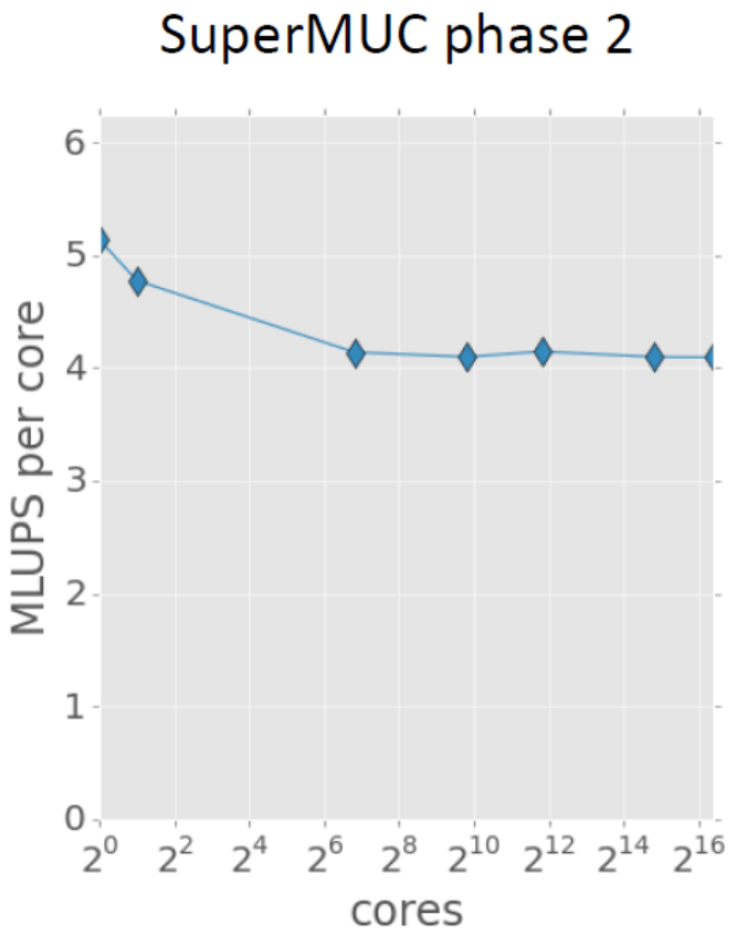
Hornet @ HLRS, Stuttgart

3944 nodes, 2x Intel Xeon E5-2680
24 cores / node (Haswell)
128 GiB RAM / node

JUQUEEN

Phase-field Method





Comparison SuperMUC

Sandy Bridge <-> Haswell (3072 nodes, 2 x Intel Xeon E5-2697 w. 14 cores)

Outlook/Conclusion

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT

Outlook

What we are currently working on:

- LBM:
  - Efficient data structures and compute kernels for sparse domains (indirect addressing scheme)
  - GPU/MIC support
- Refinement:
  - Dynamic, adaptive load balancing & refinement
- Phase-field Methods:
  - More models
  - GPU/MIC compute kernels

Conclusion

- waLBerla can achieve strong **single node performance** *and* excellent **scalability**
- It has an efficient implementation of **grid refinement** for LBM
- It is evolving from a pure LBM to a **multi-physics** framework

Thank you for your Attention!

Questions ?

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

TECHNISCHE FAKULTÄT