ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/328175584

Performance Engineering for Computational Science

Presentation · October 2018

CITATIONS
0

READS
475

1 author:

Ulrich Rüde
Friedrich-Alexander-University of Erlangen-Nürnberg

465 PUBLICATIONS **4,255** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project GESOP View project

Lattice Boltzmann Methods View project

All content following this page was uploaded by Ulrich Rüde on 09 October 2018.

The user has requested enhancement of the downloaded file.

Gauß-Allianz

Performance Engineering for Computational Science

U. Ruede

CERFACS
CENTRE EUROPÉEN DE RECHERCHE ET DE FORMATION AVANCÉE EN CALCUL SCIENTIFIQUE

Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique
www.cerfacs.fr

Lehrstuhl für Simulation
Universität Erlangen-Nürnberg
www10.informatik.uni-erlangen.de

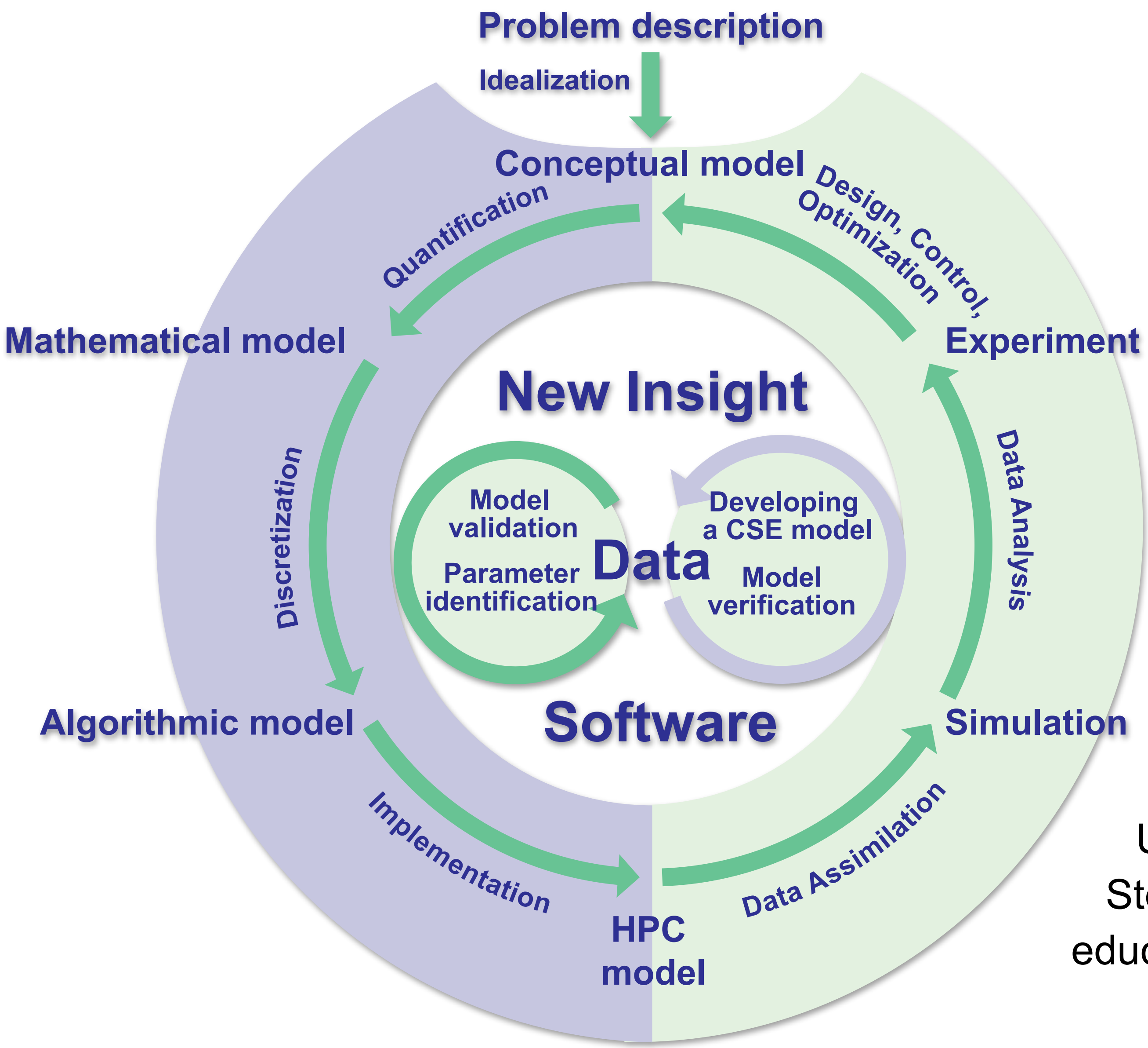
MetaXa
Physics Engine
waLBerla
TERRA NEO
E:C:E
TOTAL
HPC2SE
Hardware- und Leistungsorientierte Codegenerierung
ExaStencils
eDF
SKAMPY
Ultra-Skalierbare Multiphysik-Simulationen
KONWIHR
SPPEXA
PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR
ANR
EUROPÄISCHE UNION
Bundesministerium für Bildung und Forschung
DFG Deutsche Forschungsgemeinschaft

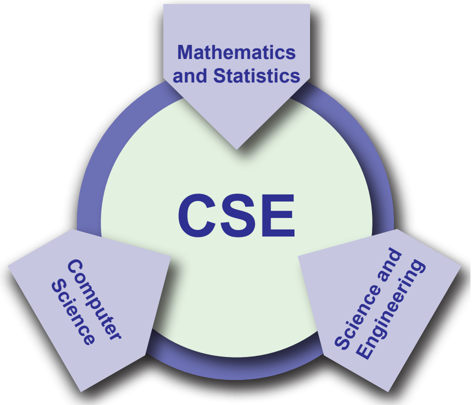
Outline

- ExaScale computing
  - node efficiency, scalability, and algorithmic complexity
- TerraNeo
  - HHG
  - HyTeG
- waLBerla
  - Lattice Boltzmann
  - Rigid Body Dynamics
- Conclusions

Part I: Extreme Scale Computing

What is it about

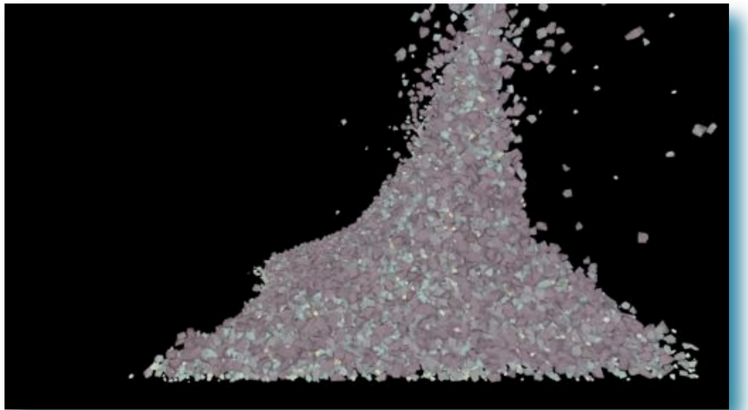




UR, Willcox, K., McInnes, L. C., & Sterck, H. D. (2018). Research and education in **computational science and engineering**. *Siam Review*, *60*(3), 707-754.

What is ExaScale possibly good for?

- ExaScale: 10^{18} FLOPS (floating point operations per second)
- When we have
  - 1000
  - x 1000
  - x 1000 particles (or pores)
  - each resolved by 1000 cells
- then we still can still execute 1 Mflop per each cell
  - 1 MFLPOS = 10^6 FLOPS = the performance of PC in 1990



Simulation performed with in-house multi-physics framework waLBerla/PE

Preclik, T. & UR (2015). **Ultrascale** simulations of non-smooth granular dynamics; Computational Particle Mechanics, DOI: 10.1007/s40571-015-0047-6

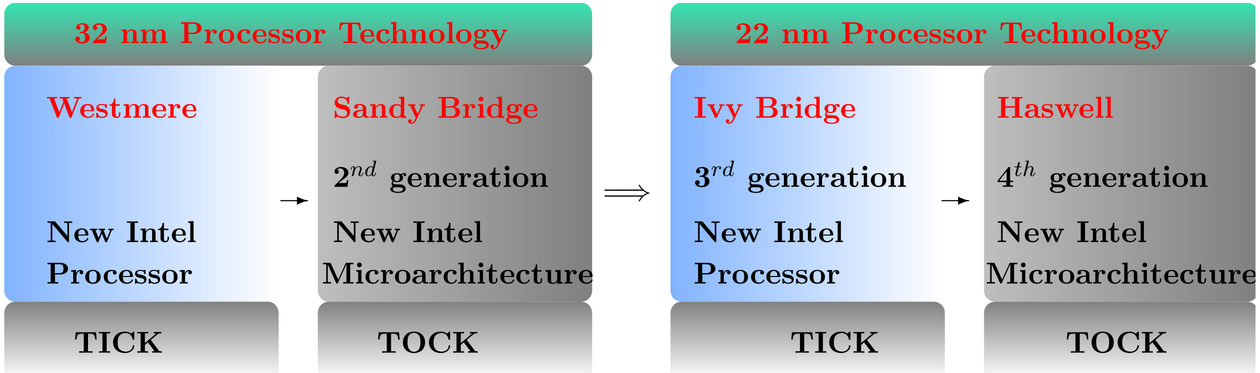
10^{18} FLOPS …

- At clock rates of 1 Ghz a single stream of operands will produce results at 10^9/sec
  - degree of concurrency= 10^9
- This will be achieved hierarchically
  - 10^4 nodes
  - 10^3 cores/node
  - 10^2 instructions on the fly/core (vectorization, pipelining, …)
- We must use them all!
- Energy: 1nJ/Flop
  - 1 GW for Exascale
- Resilience: MTBF of 100 years per cell phone (= 10 Gflop)
  - 10^8 cell phones: MTBF for Exascale = 30 sec

Energy consumption of floating point operations

based on: Ayesha Afzal: *The Cost of Computation: Metrics and Models for Modern Multicore-based Systems in Scientific Computing*, Master Thesis, FAU Erlangen, 2015

Clock rates of 2-3 GHz correspond to cycle times of 0.3-0.5 nsec.



Metrics	Energy cost				
	ε_{flop}	$\varepsilon_{byte}^{L1-REG}$	$\varepsilon_{byte}^{L2-L1}$	$\varepsilon_{byte}^{L3-L2}$	$\varepsilon_{byte}^{MEM-L3}$
Flop only	830 pJ/F	0	0	0	0
Load only	0	227 pJ/B	314 pJ/B	256 pJ/B	1880 pJ/B
Store only	0	377 pJ/B	300 pJ/B	340 pJ/B	2977 pJ/B

8 core Sandy Bridge System - measured through systematic benchmarking
see also Georg Hager's talk at PACO 2015

Best values on Green 500 currently convert to 0.1 nJ/Flop:
equivalent to 100 MW for ExaFlops performance

The first performance question:

What is the minimal cost of solving a PDE?

(such as Poisson's or Stokes' equation in 3D)

- asymptotic results of the form

  $$\text{Cost} \leqq C\, N^p \qquad \text{(Flops?)}$$

  with unspecified constant C are inadequate to predict performance

The key goal of numerical analysis:

relate accuracy and cost!

How do we quantify true cost?

(i.e. resources needed)

- Number of flops?
- Memory consumption?
- Memory bandwidth? (aggregated?)
- Communication bandwidth?
- Communication latency?
- Power consumption?

How do we predict cost?

- What is the cost of solving the discrete Poisson equation?
  - C'mon: it's the mother of all PDE!
  - Yep, there are O(N) algorithms:
    wavelet, fast multipole, multigrid, …
  - but what is the **best constant** published?
- for Poisson 2D, second order:

  #Flops = **30 N** (Stüben, 1982)

- Intel Haswell CPU: 1030.4 GFlops single precision performance
  - $N=10^6$
  - expected time to solution: **$3*10^{-5}$ sec** (microseconds!)
- standard computational practice in 2017 misses this by **many orders of magnitude**!
- why do we have this enormous **gap** between theory and practice?
- **no**, it cannot be explained by „cache effects" alone.
- This talk tries to contribute to the failure analysis.

What is the largest system that we can solve today?

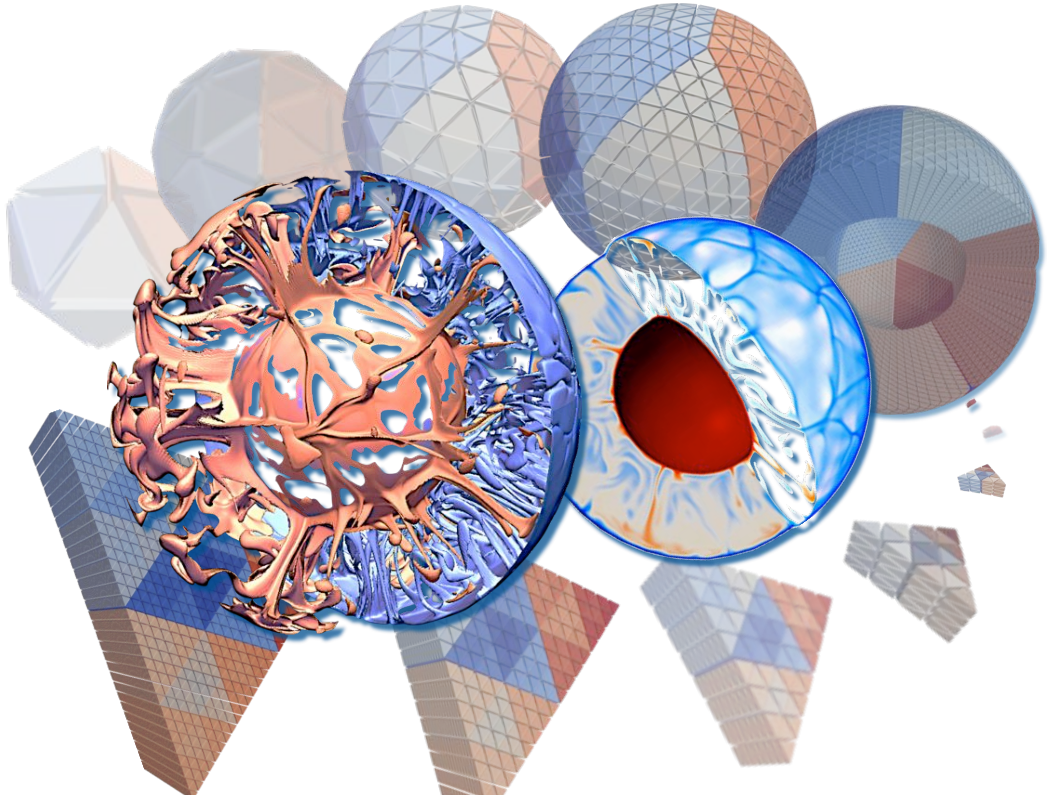
Bergen, B, Hülsemann, F, UR (2005): Is $1.7 \cdot 10^{10}$ unknowns the largest finite element system that can be solved today? Proceedings of SC'05.

- and now, 13 years later?
- we have 400 TByte main memory = $4 \cdot 10^{14}$ Bytes = 5 vectors each with **$N=10^{13}$** double precision elements
- matrix-free implementation necessary
  - even with a sparse matrix format, storing a matrix of dimension **$N=10^{13}$** is not possible
- Which algorithm?
  - multigrid
  - Cost = C*N
  - C „moderate“, e.g. C=100.
- does it parallelize well on that scale?
- should we worry since $\kappa = O(N^{2/3})$?

Algorithms Matter!

- Solution of Laplace equation in 3D wit $N=n^3$ unknowns
- Direct methods:
  - banded: $\sim n^7 = N^{2.33}$
  - nested dissection: $\sim n^6 = N^2$
- Iterative Methods:
  - Jacobi: $\sim 50\ n^5 = 50\ N^{1.66}$
  - CG: $\sim 100\ n^4 = 100\ N^{1.33}$
  - Full Multigrid: $\sim 200\ n^3 = 200\ N$

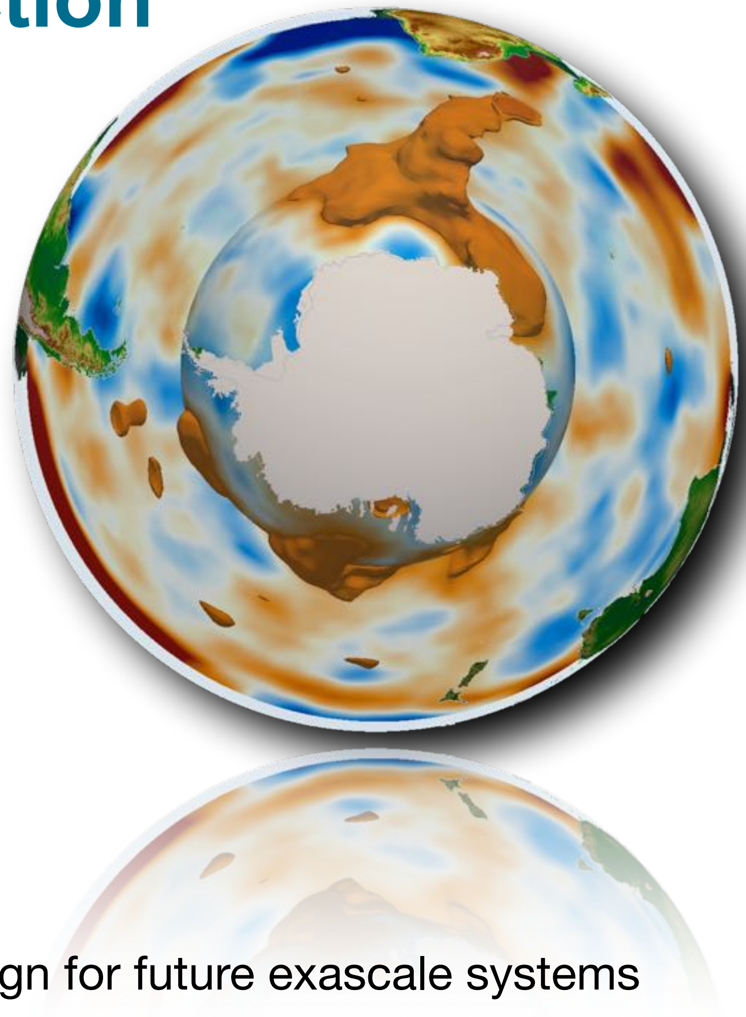
Energy per FLOP: 1nJ				
Computer Generation	gigascale: 10^9	terascale: 10^{12}	petascale: 10^{15}	exascale: 10^{18}
problem size: DoF=N	10^6	10^9	10^{12}	10^{15}
Direct method: $1*N^2$	0.278 Wh	278 kWh	278 GWh	278 PWh
Krylov method: $100*N^{1.33}$	10 Ws	28 Wh	278 kWh	2.77 GWh
Full Multigrid: 200 N	0.2 Ws	0.056 Wh	56 Wh	56 kWh
TerraNeo prototype (est. for Juqueen)	0.13 Wh	30 Wh	27 kWh	?



Part II: Hierarchical Hybrid Grids

TERRANEO

Example: Earth Mantle Convection



Why Mantle Convection?

- driving force for plate tectonics
- mountain building and earthquakes

Why Exascale?

- mantle has 10^{12} km^3
- inversion and UQ blow up cost

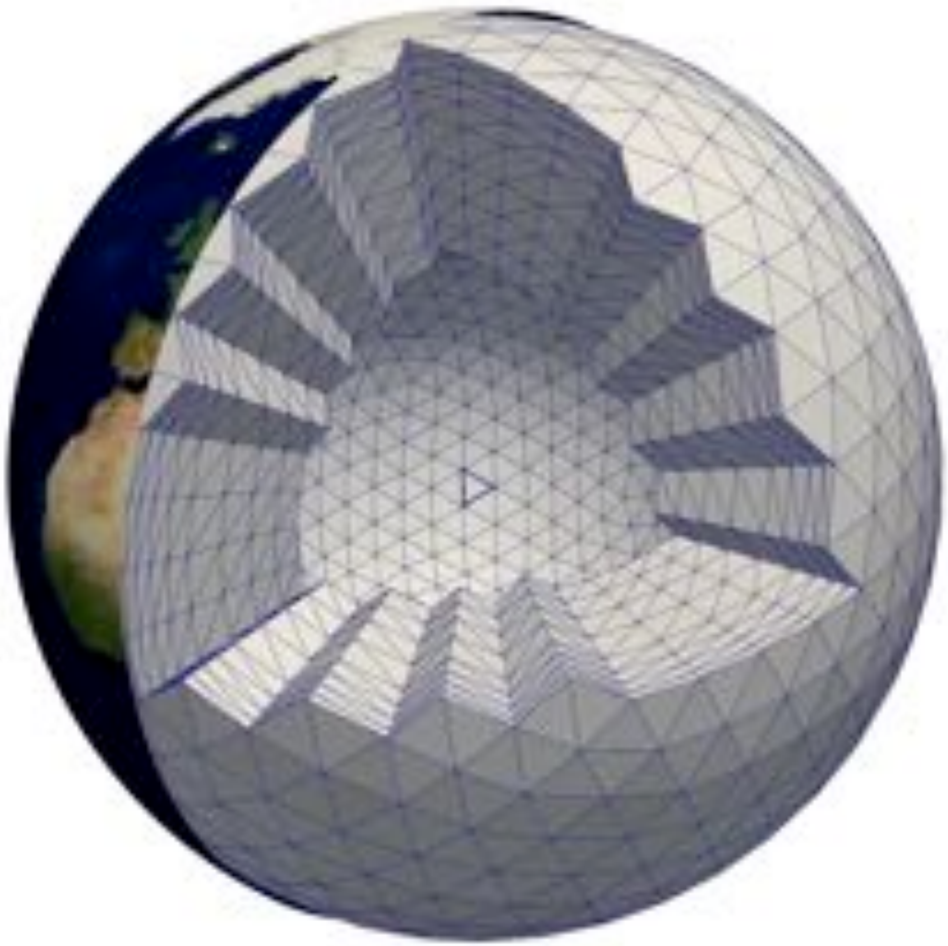
Why TerraNeo?

- ultra-scalable and fast
- sustainable framework

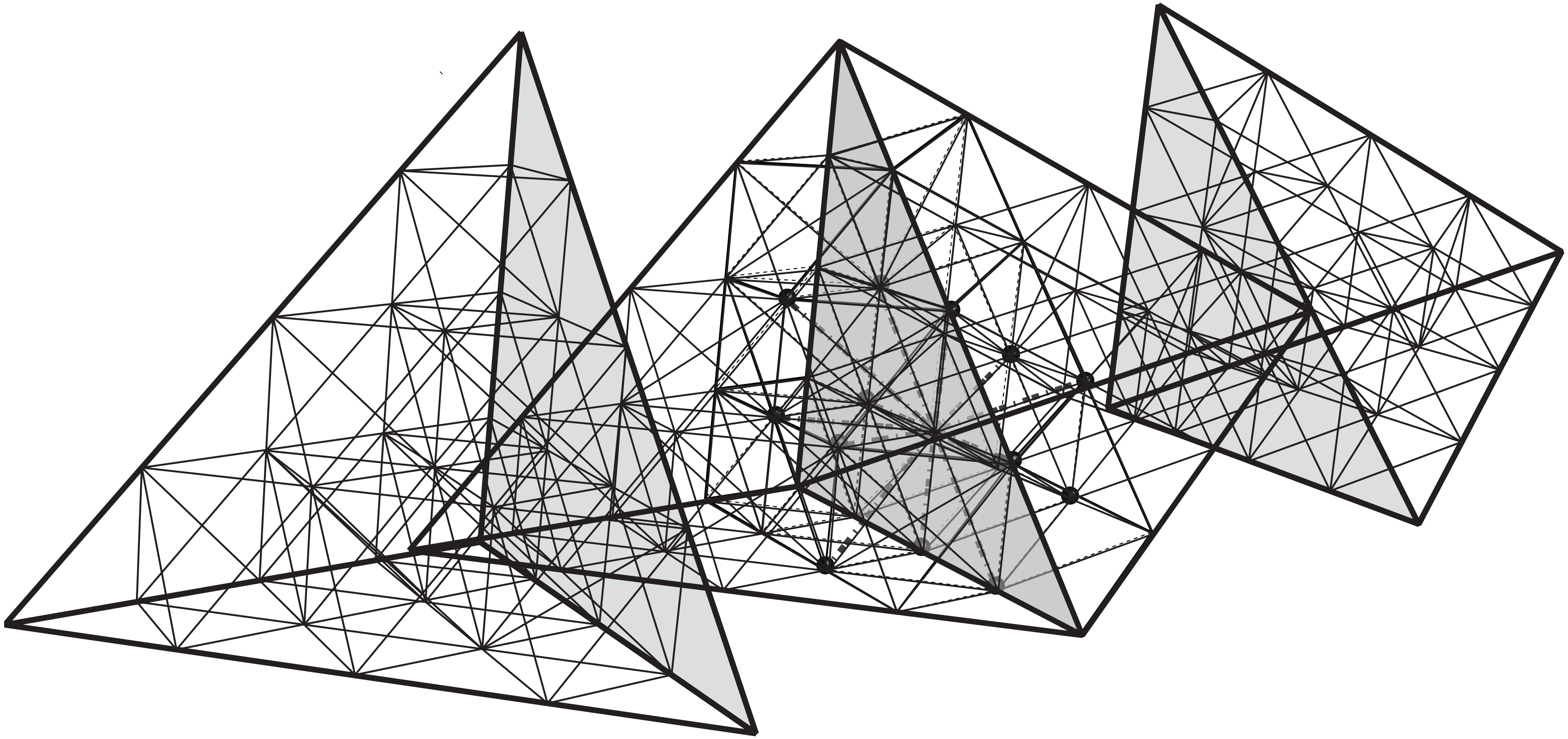
Challenges

- **computer sciences:** software design for future exascale systems
- **mathematics:** HPC performance oriented metrics
- **geophysics:** model complexity and uncertainty
- **bridging disciplines:** integrated co-design

Meshing of the Mantle with Tets



HHG: A modern mesh-free architecture for FE

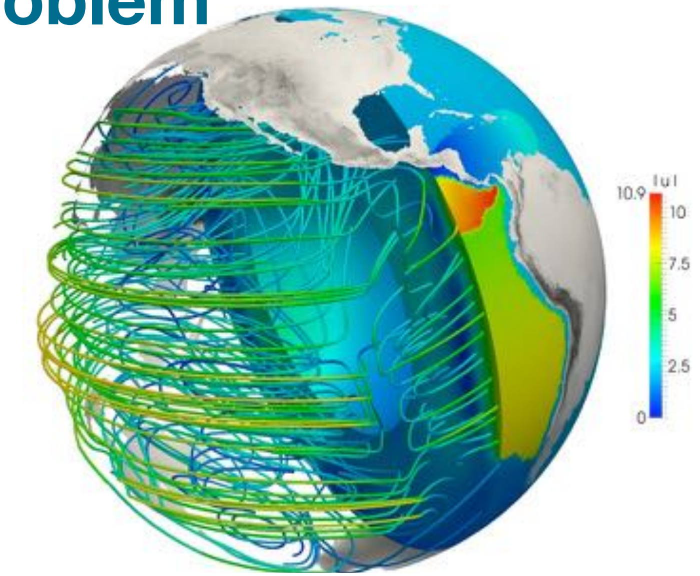


Structured refinement of an unstructured base mesh

Geometrical Hierarchy: Volume, Face, Edge, Vertex

HHG Solver for Stokes System
Motivated by Earth Mantle convection problem

Gmeiner, Waluga, Stengel, Wohlmuth, UR: Performance and Scalability of Hierarchical Hybrid Multigrid Solvers for Stokes Systems, SISC, 2015.



$$
\begin{aligned}
-\nabla \cdot (2\eta\epsilon(\mathbf{u})) + \nabla p &= \rho(T)\mathbf{g}, \\
\nabla \cdot \mathbf{u} &= 0, \\
\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) &= \gamma.
\end{aligned}
$$

$\mathbf{u}$	velocity
p	dynamic pressure
T	temperature
ν	viscosity of the material
$\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$	strain rate tensor
ρ	density
$\kappa, \gamma, \mathbf{g}$	thermal conductivity, heat sources, gravity vector

Scale up to ~10^{13} nodes/ DOFs

⇒ resolve the whole Earth Mantle globally with 1km resolution

Stokes equation: $-\mathrm{div}(\nabla \mathbf{u} - p\mathbf{I}) = \mathbf{f}, \quad \mathrm{div}\,\mathbf{u} = 0$

FEM Discretization:

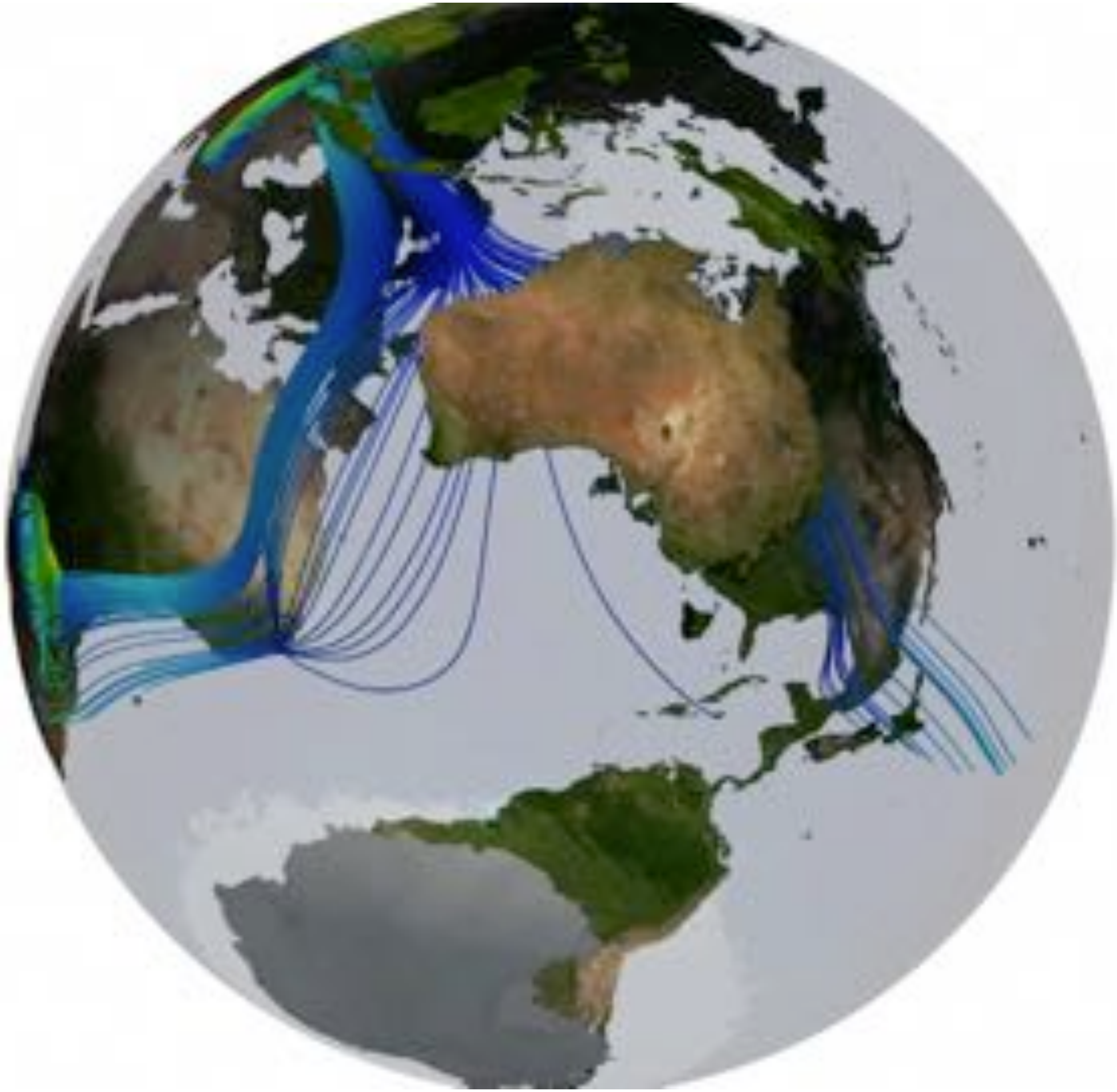
$$
\begin{aligned}
\mathbf{a}(\mathbf{u}_l, \mathbf{v}_l) + \mathbf{b}(\mathbf{v}_l, p_l) &= \mathbf{L}(\mathbf{v}_l) && \forall\, \mathbf{v}_l \in \mathbf{V}_l, \\
\mathbf{b}(\mathbf{u}_l, q_l) - \mathbf{c}(p_l, q_l) &= 0 && \forall\, q_l \in Q_l,
\end{aligned}
$$

with: $\mathbf{a}(\mathbf{u}, \mathbf{v}) := \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v}\, dx, \quad \mathbf{b}(\mathbf{u}, q) := -\int_\Omega \mathrm{div}\,\mathbf{u} \cdot q\, dx$

Schur-complement formulation:

$$
\begin{bmatrix} \mathbf{A}_l & \mathbf{B}_l^\top \\ 0 & \mathbf{C}_l + \mathbf{B}_l \mathbf{A}_l^{-1} \mathbf{B}_l^\top \end{bmatrix} \begin{bmatrix} \underline{\mathbf{u}}_l \\ \underline{p}_l \end{bmatrix} = \begin{bmatrix} \underline{\mathbf{f}}_l \\ \mathbf{B}_l \mathbf{A}_l^{-1} \underline{\mathbf{f}}_l \end{bmatrix}
$$

Stationary Flow Field



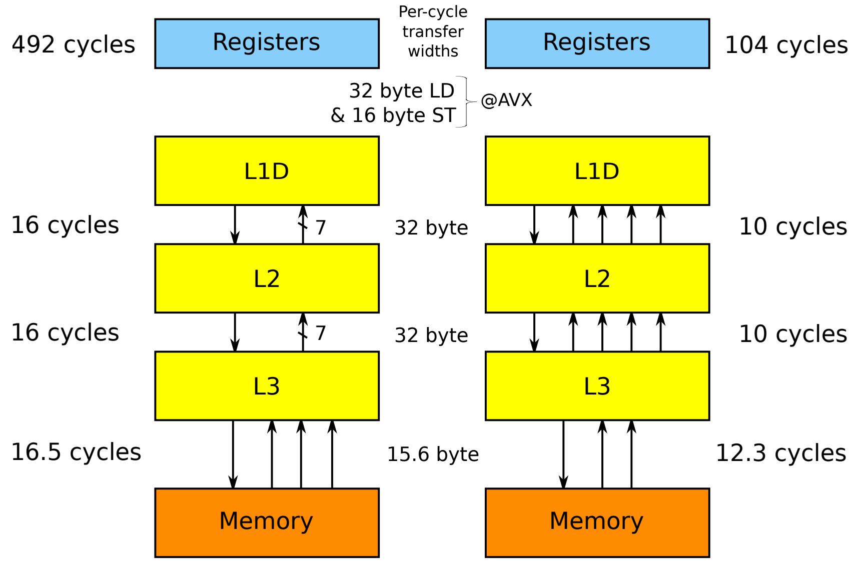
Exploring the Limits …

Gmeiner B., Huber M, John L, UR, Wohlmuth, B: A quantitative performance study for Stokes solvers at the extreme scale, Journal of Computational Science, 2016.

- Multigrid with Uzawa smoother
- Optimized for minimal memory consumption
  - 10^{13} Unknowns correspond to 80 TByte for the solution vector
  - Juqueen has 450 TByte Memory
  - matrix free implementation essential

is this the largest FE system solved to date?

nodes	threads	DoFs	iter	time	time w.c.g.	time c.g. in %
5	80	$2.7 \cdot 10^9$	10	685.88	678.77	1.04
40	640	$2.1 \cdot 10^{10}$	10	703.69	686.24	2.48
320	5 120	$1.2 \cdot 10^{11}$	10	741.86	709.88	4.31
2 560	40 960	$1.7 \cdot 10^{12}$	9	720.24	671.63	6.75
20 480	327 680	$1.1 \cdot 10^{13}$	9	776.09	681.91	12.14



ECM Analysis of multigrid smoother for variable/constant coefficient performance

Towards systematic performance engineering:

Parallel Textbook Efficiency
as guiding design principle

Brandt, A. (1998). Barriers to achieving **textbook multigrid efficiency** (TME) in CFD.

Thomas, J. L., Diskin, B., & Brandt, A. (2003). **Textbook Multigrid Efficiency** for Fluid Simulations*. Annual review of fluid mechanics, 35(1), 317-340.

Gmeiner, UR, Stengel, Waluga, Wohlmuth: Towards **Textbook Efficiency for Parallel Multigrid**, Journal of Numerical Mathematics: Theory, Methods and Applications, 2015

Gmeiner, Huber, John, UR, Wohlmuth, A quantitative **performance study for Stokes solvers** at the extreme scale, Journal of Computational Science, Volume 17, 2016, Pages 509-521

Textbook Multigrid Efficiency (TME)

„Textbook multigrid efficiency means solving a discrete PDE problem with a computational effort that is only a small (less than 10) multiple of the operation count associated with the discretized equations itself." [Brandt, 98]

- work unit (**WU**) = single elementary relaxation
  classical ***algorithmic*** TME-factor:
  ops for solution/ ops for work unit
- Here we introduce a new
  parallel TME-factor:
  - algorithmic efficiency
  - scalability & node efficiency

ParTME paradigm for parallel performance analysis

- Analyse **cost of an elementary** relaxation to define cost of work unit (WU) depending on idealized HW
  - micro-kernel benchmarks
  - measure aggregate performance
- Measure **parallel solver performance**
- Compute ParTME factor as quotient

Parallel TME

μ_{sm} # of elementary relaxation steps on single core/sec

U # cores

$U\mu_{\mathrm{sm}}$ aggregate peak relaxation performance

$T_{\mathrm{WU}}(N,U) = \frac{N}{U\mu_{\mathrm{sm}}}$ **idealized** time for a work unit

$T(N,U)$ time to solution for N unknowns on U cores

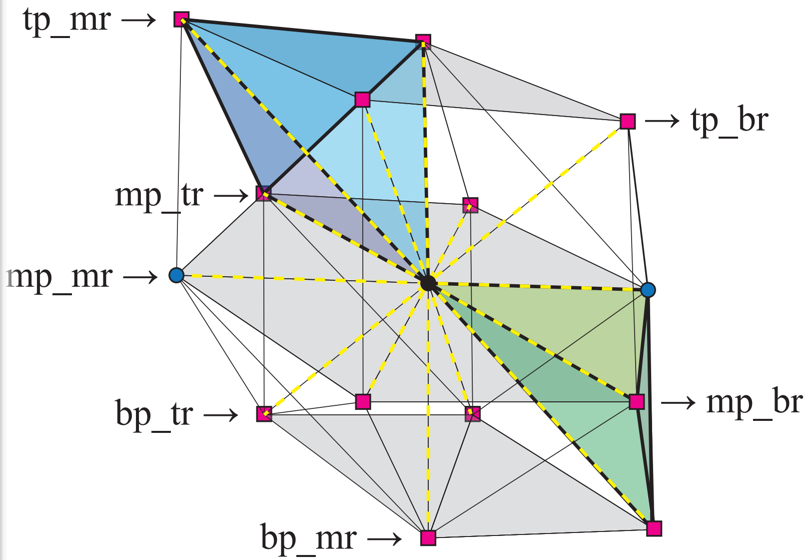
Parallel textbook efficiency factor

$$\boxed{E_{\mathrm{ParTME}}(N,U) = \frac{T(N,U)}{T_{\mathrm{WU}}(N,U)} = T(N,U)\frac{U\mu_{\mathrm{sm}}}{N}}$$

combines **algorithmic** and **implementation** efficiency.

TME Efficiency Analysis: RB-GS Smoother

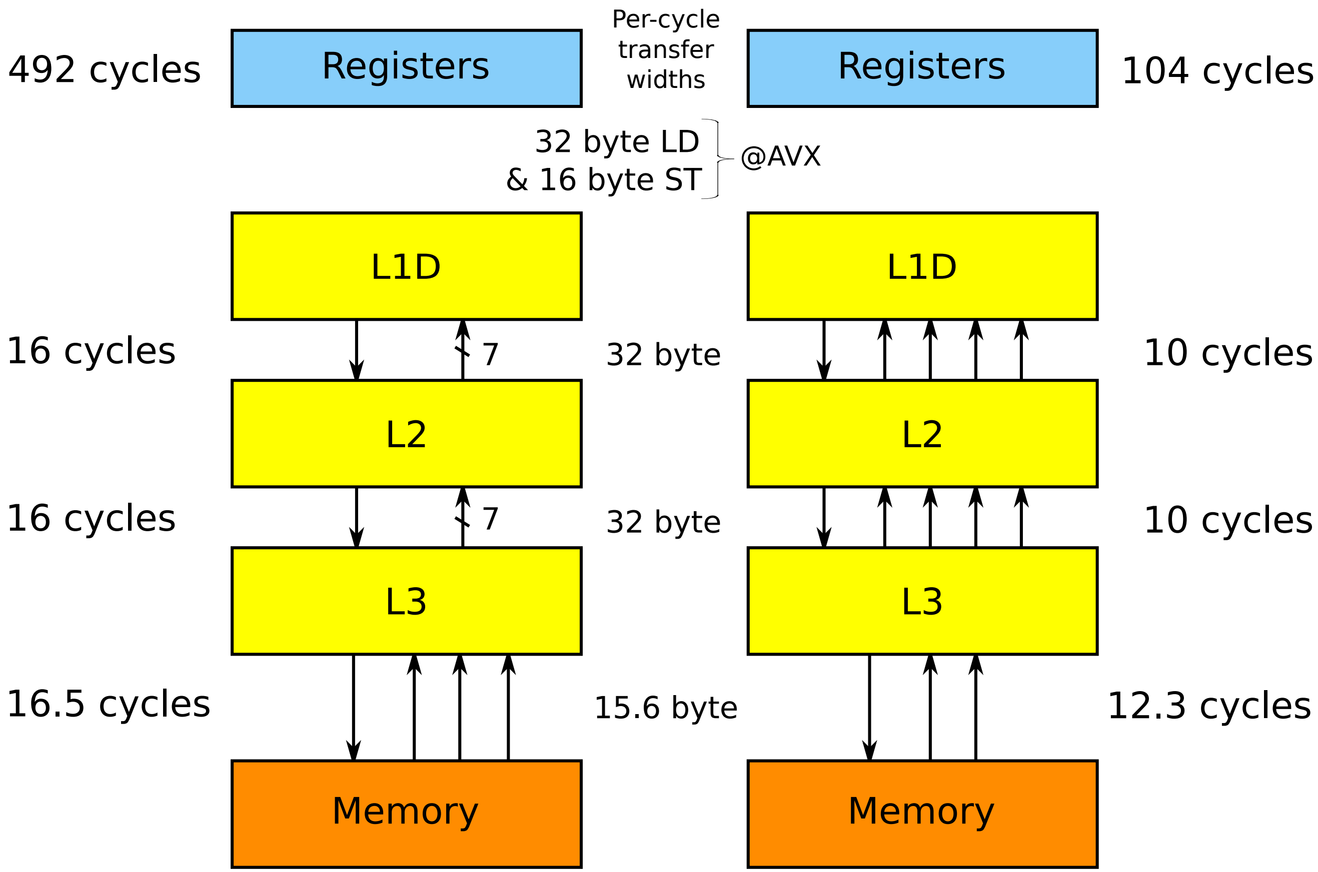
```
for (int i=1; i < (tsize-j-k-1); i=i+2) {
   u[mp_mr+i] = c[0] * (
    -c[1] *u[mp_mr+i+1] - c[2] *u[mp_tr+i-1] -
     c[3] *u[mp_tr+i]   - c[4] *u[tp_br+i]   -
     c[5] *u[tp_br+i+1] - c[6] *u[tp_mr+i-1] -
     c[7] *u[tp_mr+i]   - c[8] *u[bp_mr+i]   -
     c[9] *u[bp_mr+i+1] - c[10]*u[bp_tr+i-1] -
     c[11]*u[bp_tr+i]   - c[12]*u[mp_br+i]   -
     c[13]*u[mp_br+i+1] - c[14]*u[mp_mr+i-1] +
     f[mp_mr+i] );
```



- This loop should be executed on **single SuperMuc core** at
  - 720 M updates/sec (*in theory* - peak performance)
  - μ_{sm} = **176 M** updates/sec (*in practice* - memory access bottleneck; RB-ordering prohibits vector loads)
- Thus **whole SuperMuc** should perform
  - $U\mu_{\mathrm{sm}}$ = 147456*176M ≈ **26 T** (updates/sec)

Execution-Cache-Memory Model (ECM)

Hager et al. Exploring performance and power properties of modern multi−core chips via simple machine models. Concurrency and Computation: Practice and Experience, 2013.



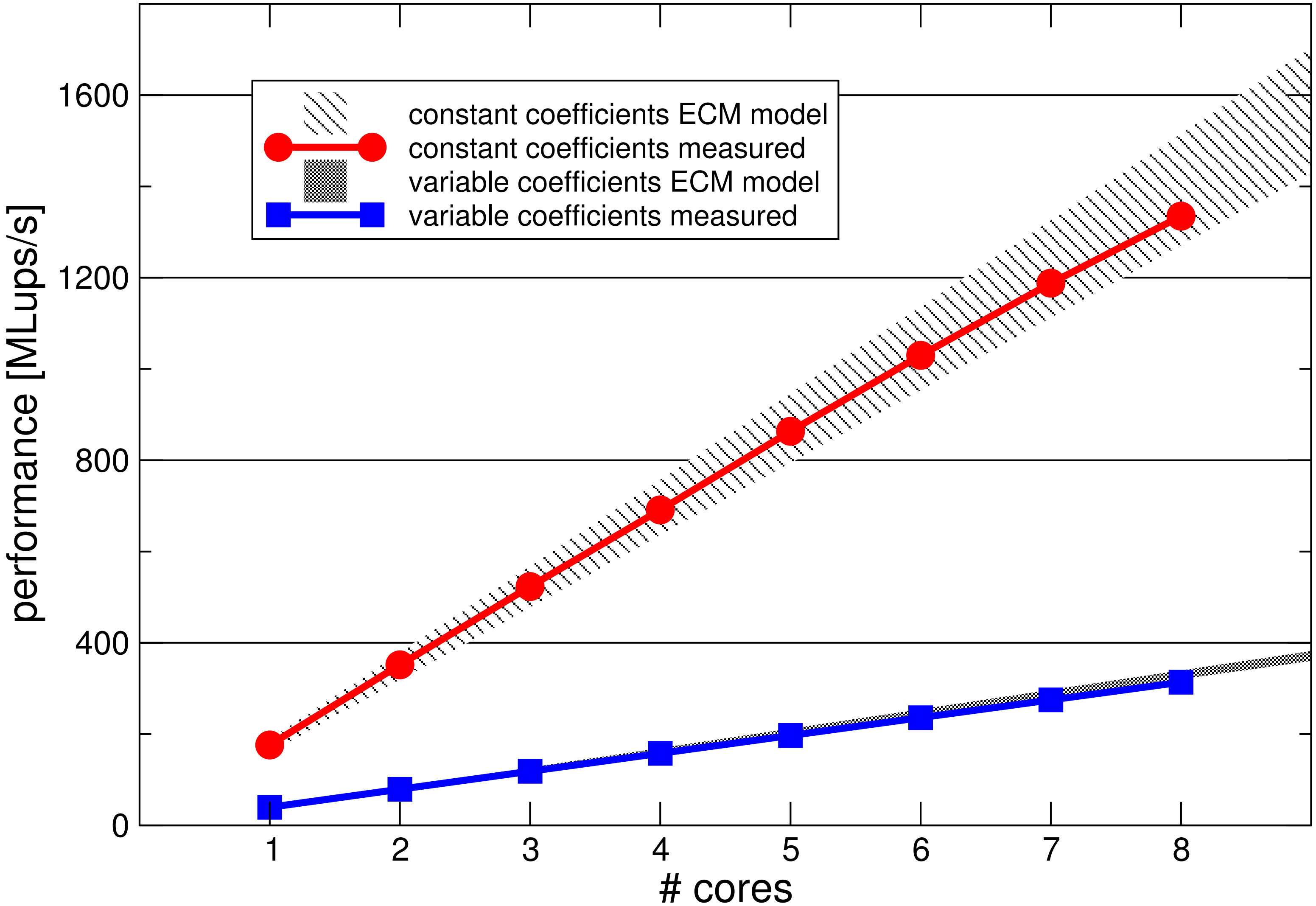
Variable coefficients   Constant coefficients

ECM model for the 15-point stencil on SNB core.

Arrow indicates a 64 Byte cache line transfer.

Run-times represent 8 elementary updates.

ECM: single-chip prediction vs. measurement



Sandy Bridge single-chip performance scaling of the stencil smoothers on 256^3 grid points. Measured data and ECM prediction ranges shown.

ParTME results for Stokes system

Gmeiner, Huber, John, UR, Wohlmuth, A quantitative performance study for Stokes solvers at the extreme scale, Journal of Computational Science, vol.17, 2016, pp. 509-521

Drzisga, D., John, L., UR, Wohlmuth, B., & Zulehner, W. (2018). On the Analysis of Block Smoothers for Saddle Point Problems. *SIAM Journal on Matrix Analysis and Applications*, *39*(2), 932-960.

	TME	ParTME				
DoFs		8.2×10^6	5.3×10^8	4.3×10^9	3.4×10^{10}	2.7×10^{11}
cores		1	192	1536	12288	98304
FMG-1V(2,2)	9.07	43.43	150	200	278	500

- variable V-cycle, Uzawa-type smoother
- TME factor <10
- ParTME factor >200
- coarse grid solver becomes a bottleneck

Nevertheless we are reaching **10^{13}** unknowns.

Redesigning the HHG prototype
Hybrid Tetrahedral Grids - HyTeG

- Geometric and algebraic multigrid solvers with scalable complexity
  - Scalable solvers
  - Robust higher order discretization (Fast4HHO, EoCoE)
  - Scalable & Robust Coarse Grid solvers, MUMPS, …
  - …
- Advanced algorithms with potential for exascale computing
  - Matrix-free Methods (EoCoE)
  - Algorithm Based Fault Tolerance, ABFT
  - …
- Software for ExaScale Computing
  - TerraNeo Project
  - HyTeG Software (HHG)
  - …

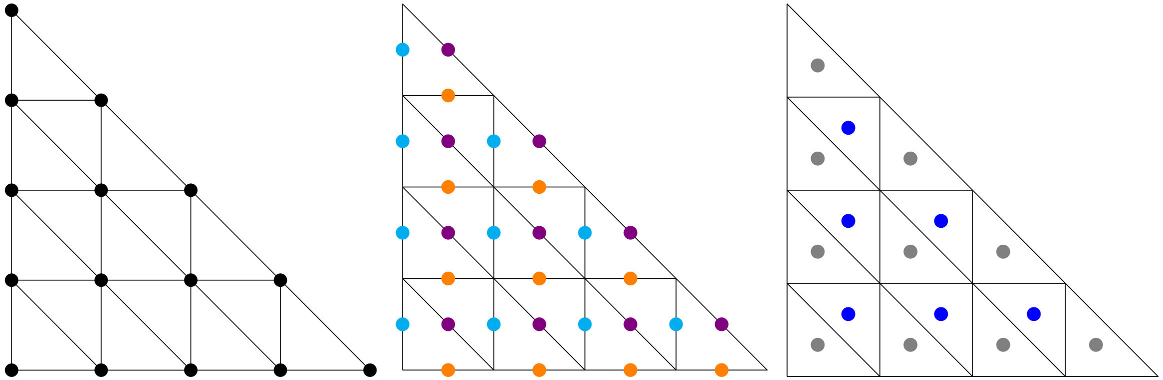
Bauer, S., Mohr, M., UR, Weismüller, J., Wittmann, M., & Wohlmuth, B. (2017). A two-scale approach for efficient **on-the-fly operator assembly** in massively parallel high performance multigrid codes. *Applied Numerical Mathematics*, *122*, 14-38.

Bauer, S., Drzisga, D., Mohr, M., UR, Waluga, C., & Wohlmuth, B. (2017). A stencil scaling approach for accelerating **matrix-free** finite element implementations. *arXiv preprint arXiv:1709.06793 (submitted to SISC)*

Kohl, N., Thönnes, D., Drzisga, D., Bartuschat, D., UR (2018). The **HyTeG** finite-element software framework for scalable multigrid solvers. *International Journal of Parallel, Emergent and Distributed Systems*, 1-20.

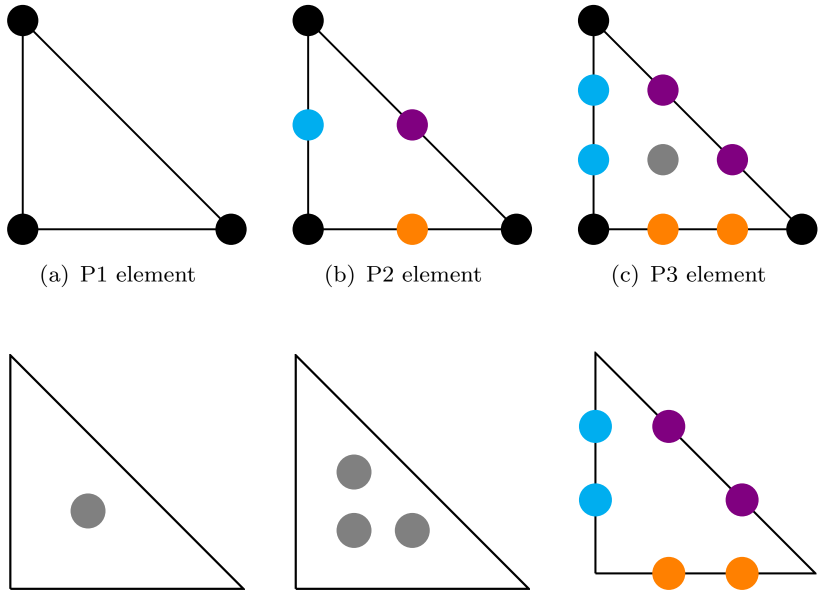
HyTeG: Hybrid tetrahedral grids

- Generalizes HHG mesh structures for arbitrary discretizations
- Structured-within-unstructured mesh



(a) vertex DoFs (b) edge DoFs (c) face DoFs

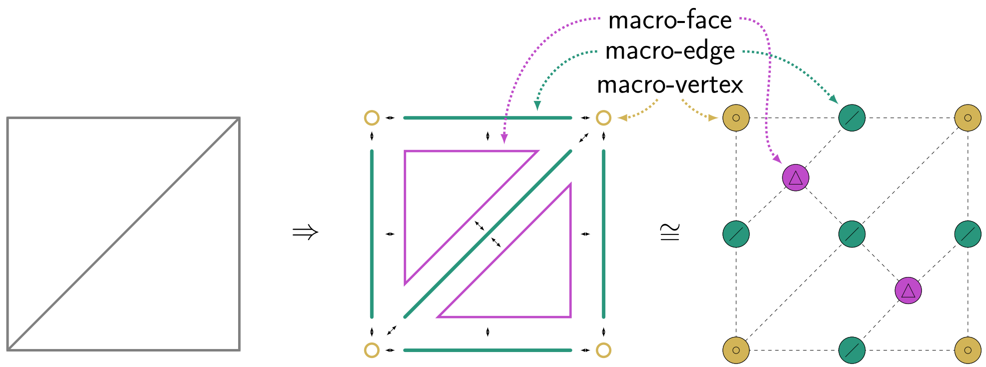
Geometric structure



(a) P1 element (b) P2 element (c) P3 element

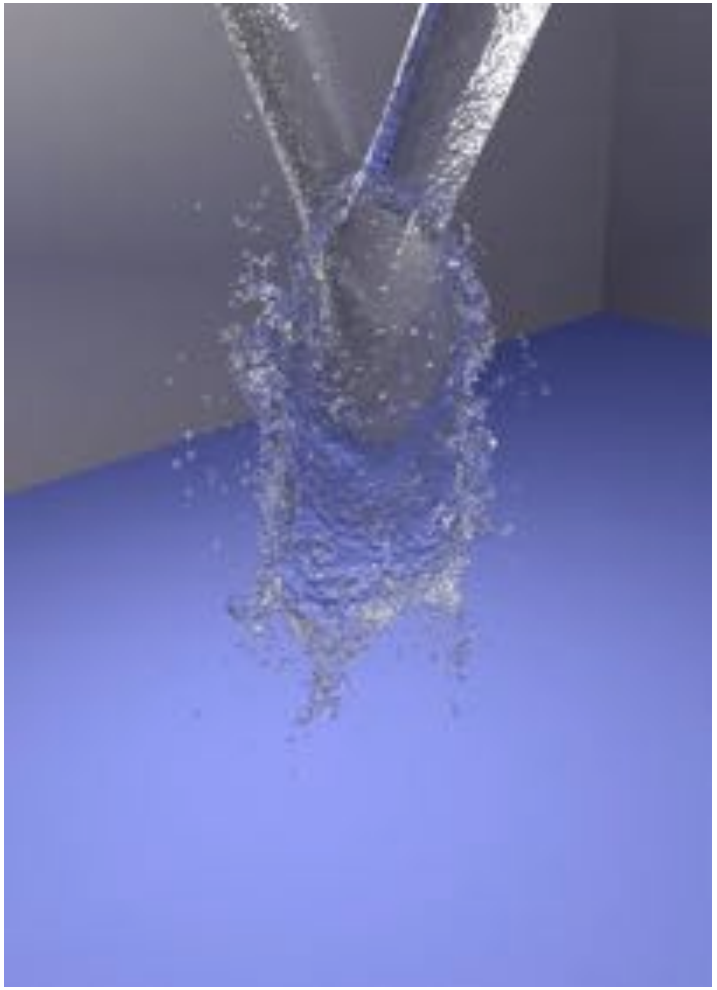
(d) FV/DG0 element (e) DG1 (C) element (f) DG1 (E) element

Example element types



unstructured mesh macro-primitives graph representation

Topological structure



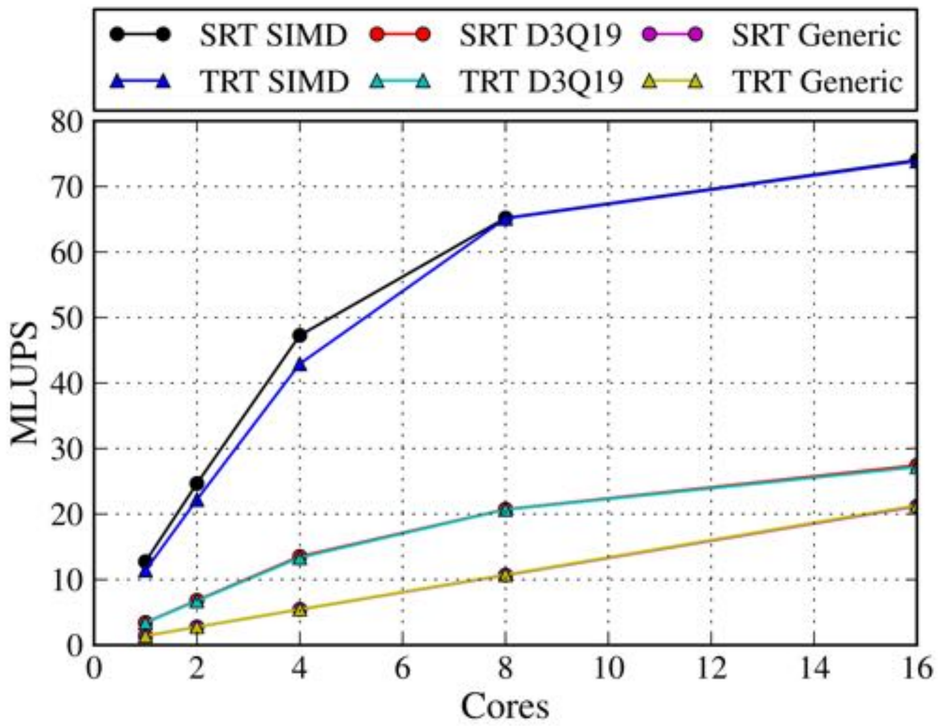
Part III: Lattice Boltzmann

Succi, S. (2001). *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press.

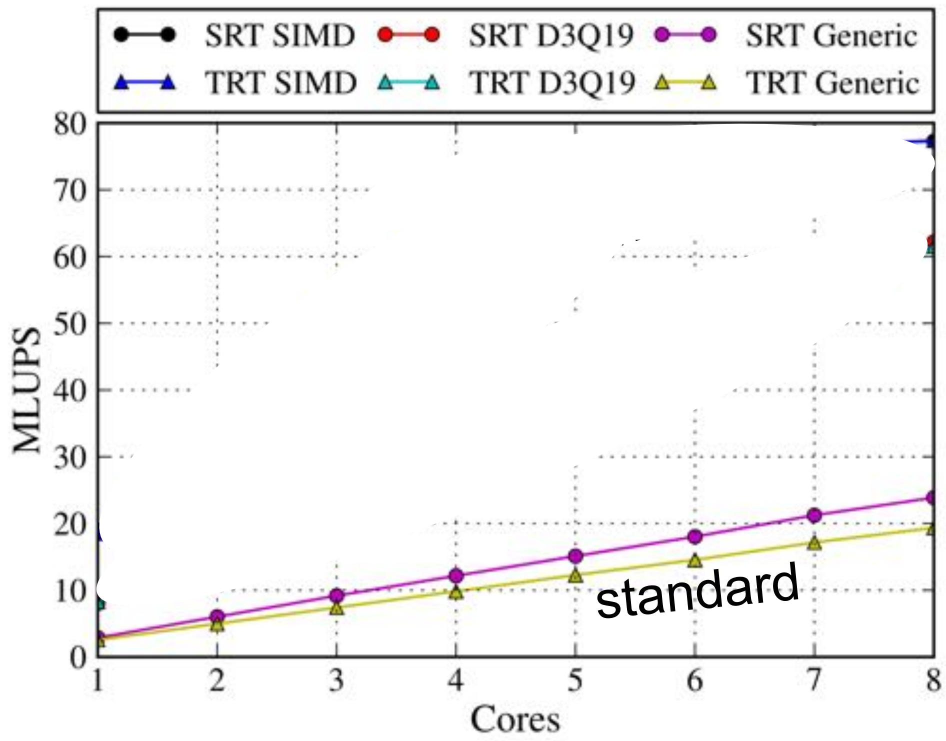
Feichtinger, C., Donath, S., Köstler, H., Götz, J., & Rüde, U. (2011). WaLBerla: HPC software design for computational engineering simulations. *Journal of Computational Science*, *2*(2), 105-112.

Single Node Performance

JUQUEEN



SuperMUC

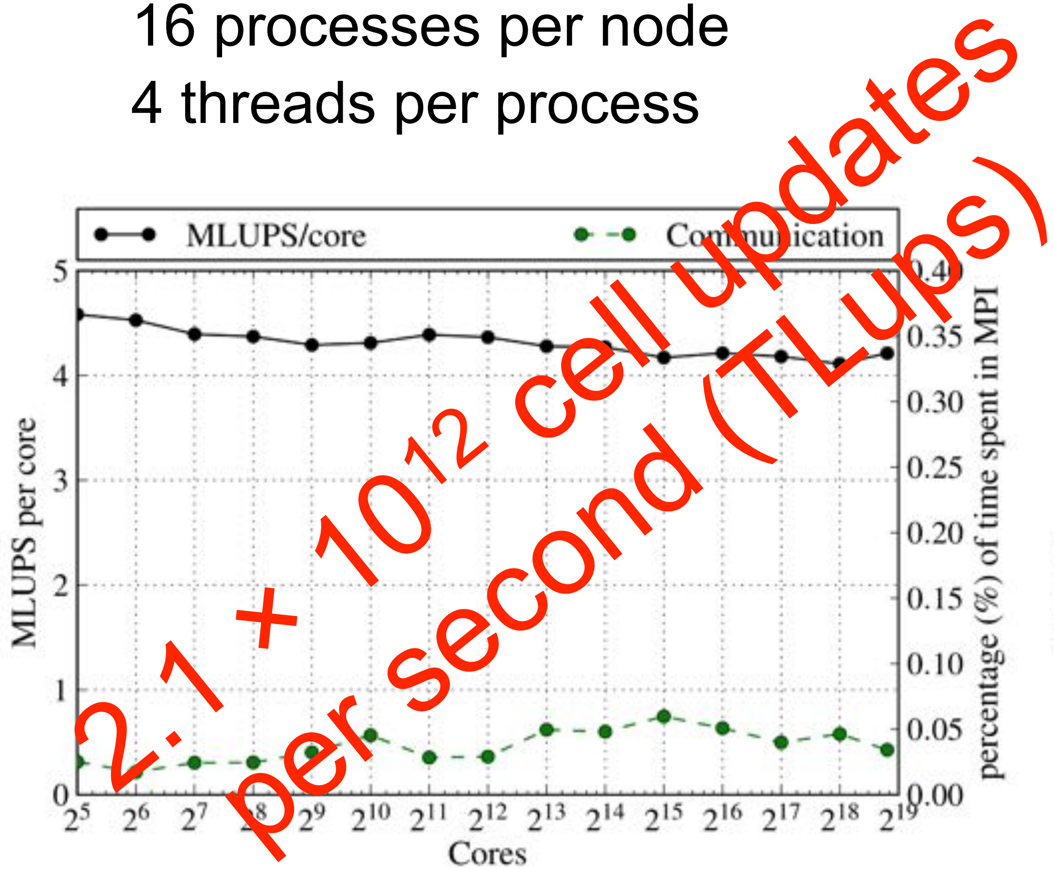


Pohl, T., Deserno, F., Thürey, N., UR, Lammers, P., Wellein, G., & Zeiser, T. (2004). Performance evaluation of parallel large-scale lattice Boltzmann applications on three supercomputing architectures. *Proceedings of the 2004 ACM/IEEE conference on Supercomputing* (p. 21). IEEE Computer Society.

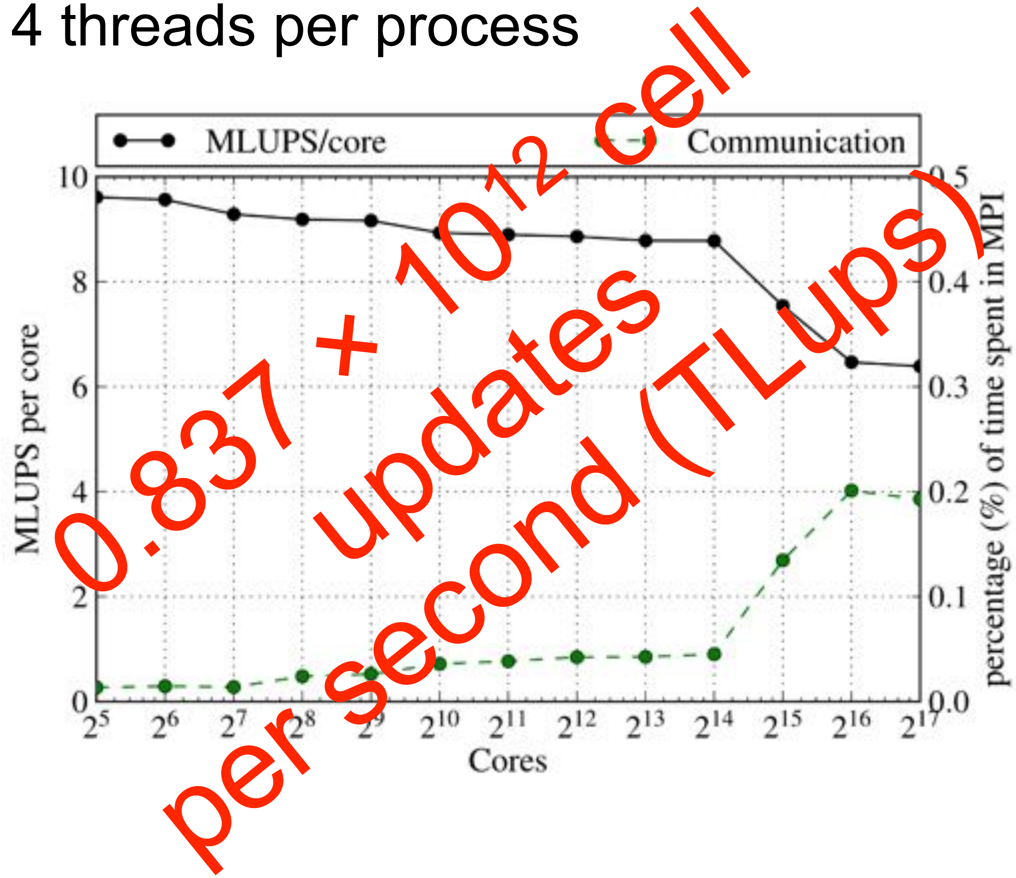
Donath, S., Iglberger, K., Wellein, G., Zeiser, T., Nitsure, A., & UR (2008). Performance comparison of different parallel lattice Boltzmann implementations on multi-core multi-socket systems. *International Journal of Computational Science and Engineering*, *4*(1), 3-11.

Weak scaling for TRT lid driven cavity - uniform grids

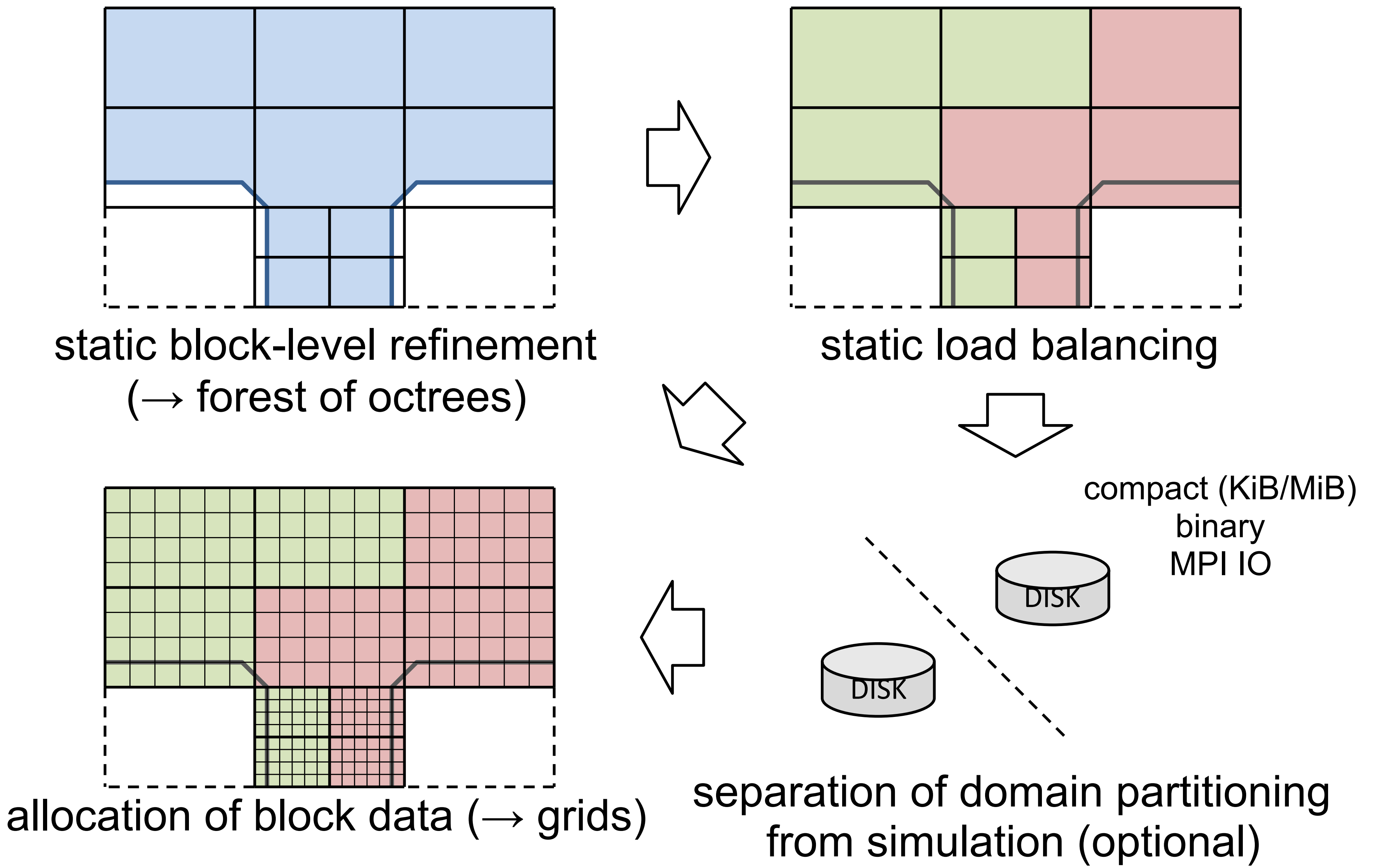
JUQUEEN
16 processes per node
4 threads per process



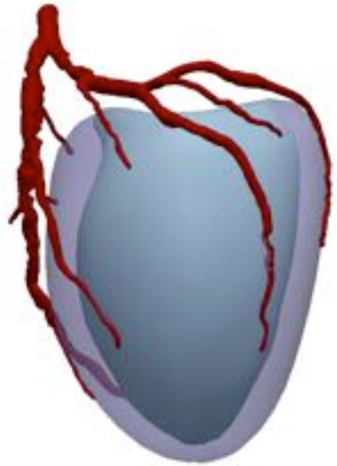
SuperMUC
4 processes per node
4 threads per process



Domain Partitioning and Parallelization



Performance on Coronary Arteries Geometry

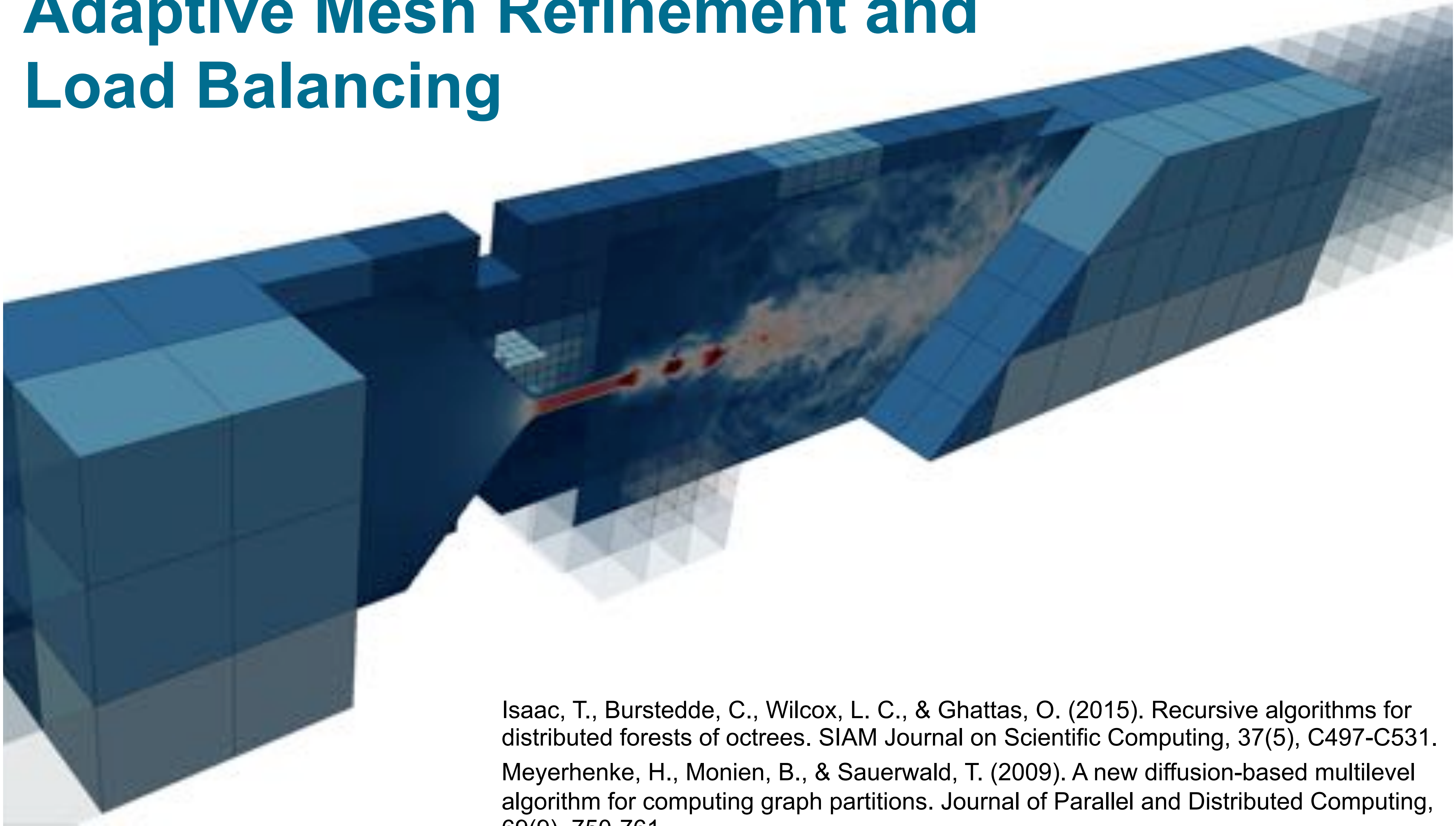


Color coded proc assignment

Godenschwager, C., Schornbaum, F., Bauer, M., Köstler, H., & UR (2013). A framework for hybrid parallel flow simulations with a trillion cells in complex geometries. In *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis* (p. 35). ACM.

- **Weak scaling**
  458,752 cores of JUQUEEN
  over a trillion (10^{12}) fluid lattice cells
- **Strong scaling**
  32,768 cores of SuperMUC
  - cell sizes of 0.1 mm
  - 2.1 million fluid cells
  - 6000 time steps per second

Adaptive Mesh Refinement and Load Balancing



Isaac, T., Burstedde, C., Wilcox, L. C., & Ghattas, O. (2015). Recursive algorithms for distributed forests of octrees. SIAM Journal on Scientific Computing, 37(5), C497-C531.

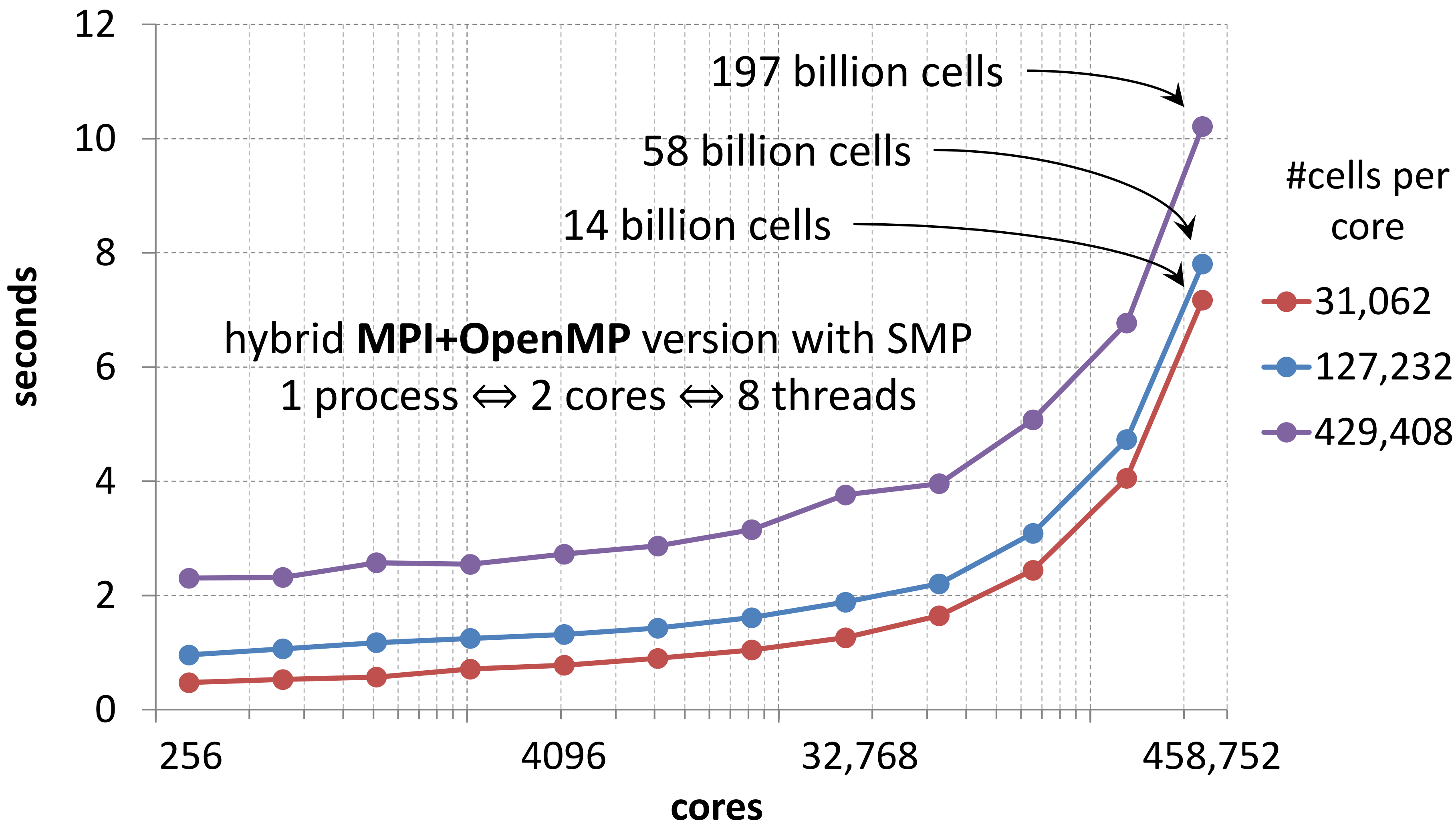
Meyerhenke, H., Monien, B., & Sauerwald, T. (2009). A new diffusion-based multilevel algorithm for computing graph partitions. Journal of Parallel and Distributed Computing, 69(9), 750-761.

Schornbaum, F., UR (2016). Massively Parallel Algorithms for the Lattice Boltzmann Method on **Nonuniform Grids**. SIAM Journal on Scientific Computing, 38(2), C96-C126.

Schornbaum, F., UR (2018). Extreme-scale block-structured **adaptive mesh refinement**. *SIAM Journal on Scientific Computing*, *40*(3), C358-C387.

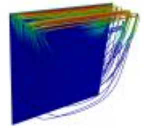
AMR Performance

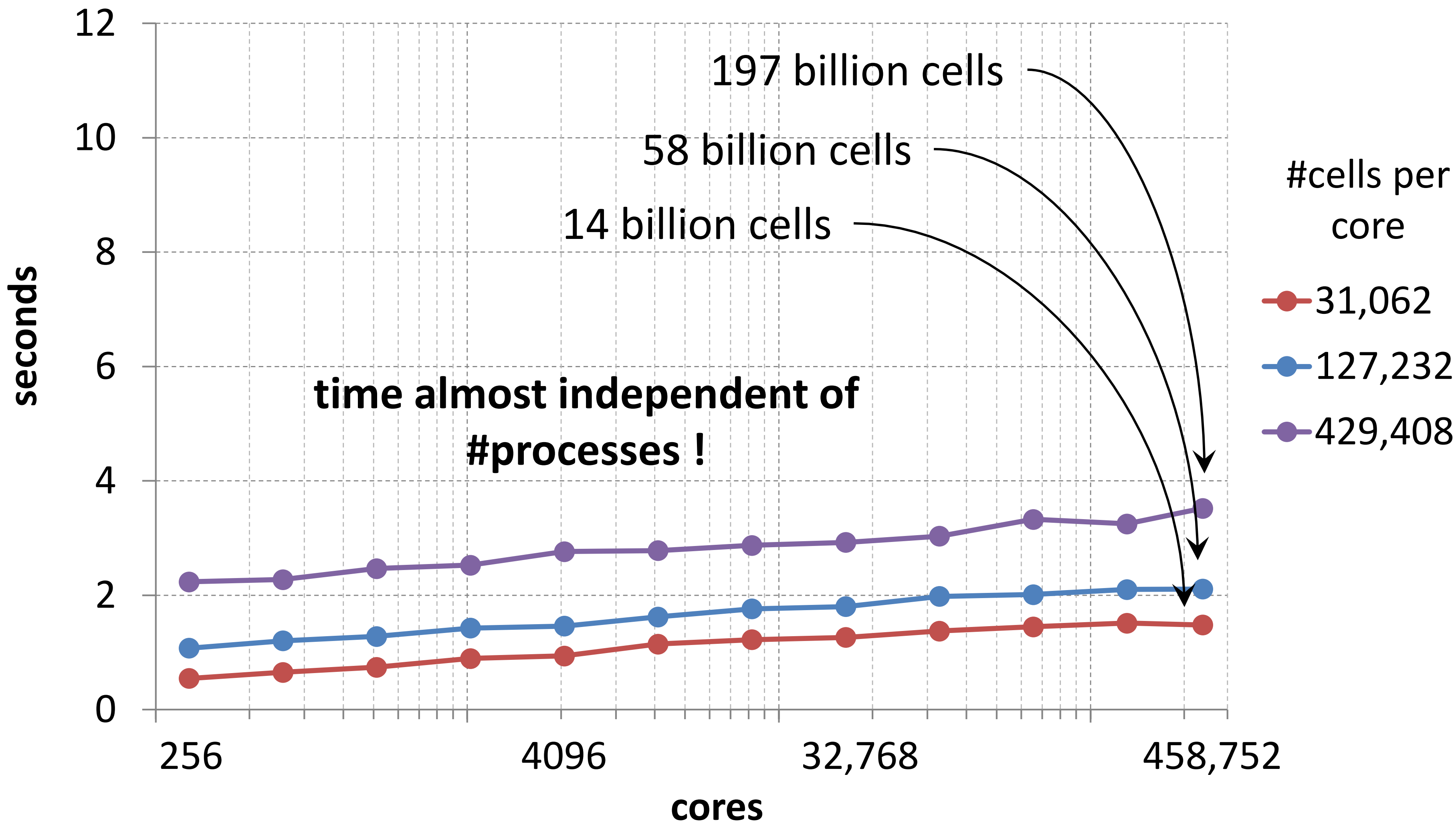
- JUQUEEN – space filling curve: Morton



AMR Performance

- JUQUEEN – diffusion load balancing

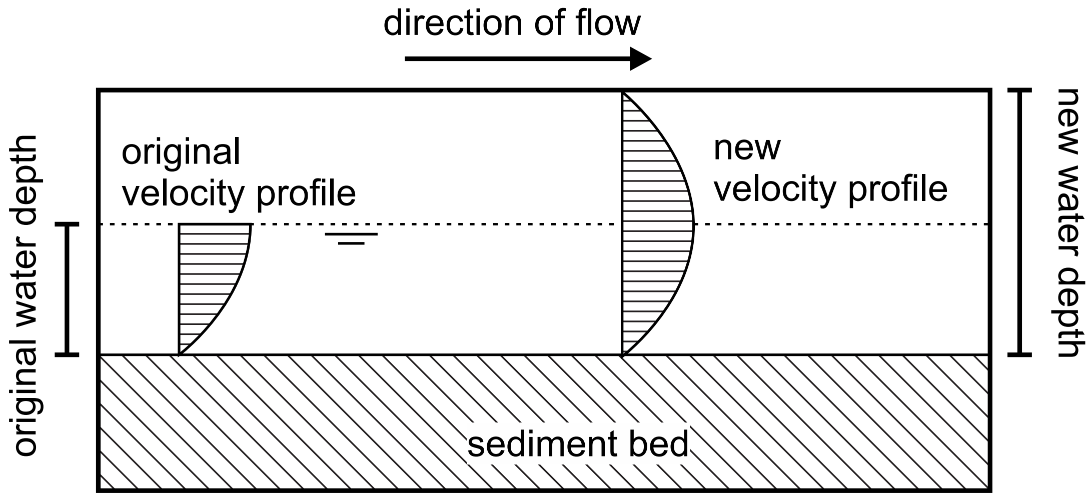


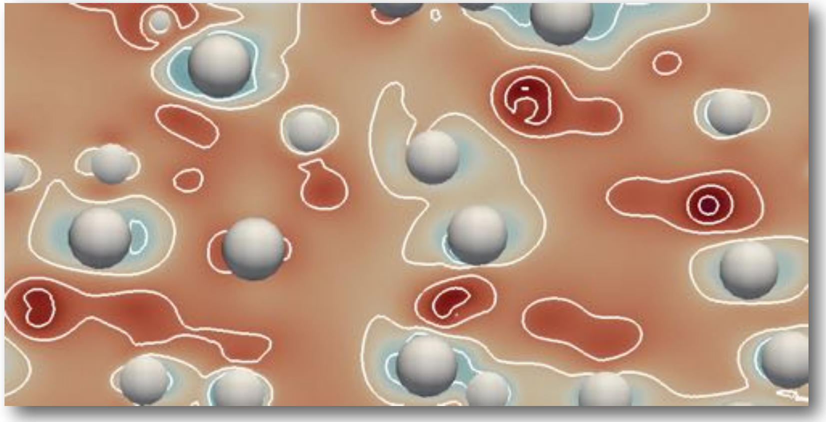


Application Example:

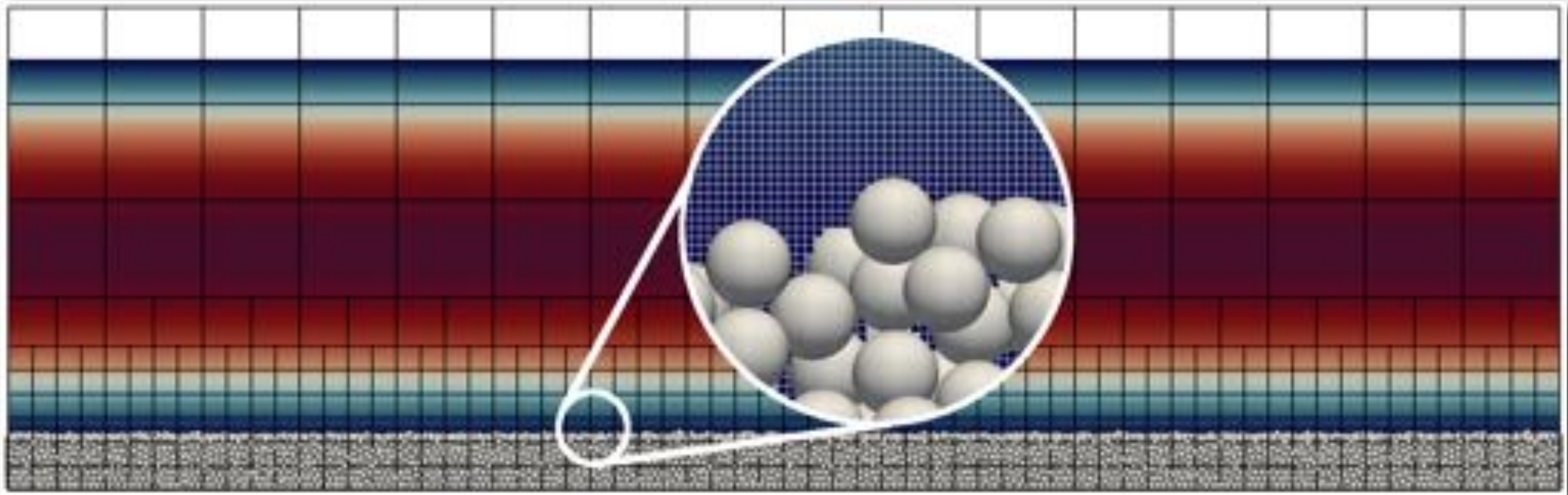
Direct simulation of complex fluids

Simulation of suspended particle transport



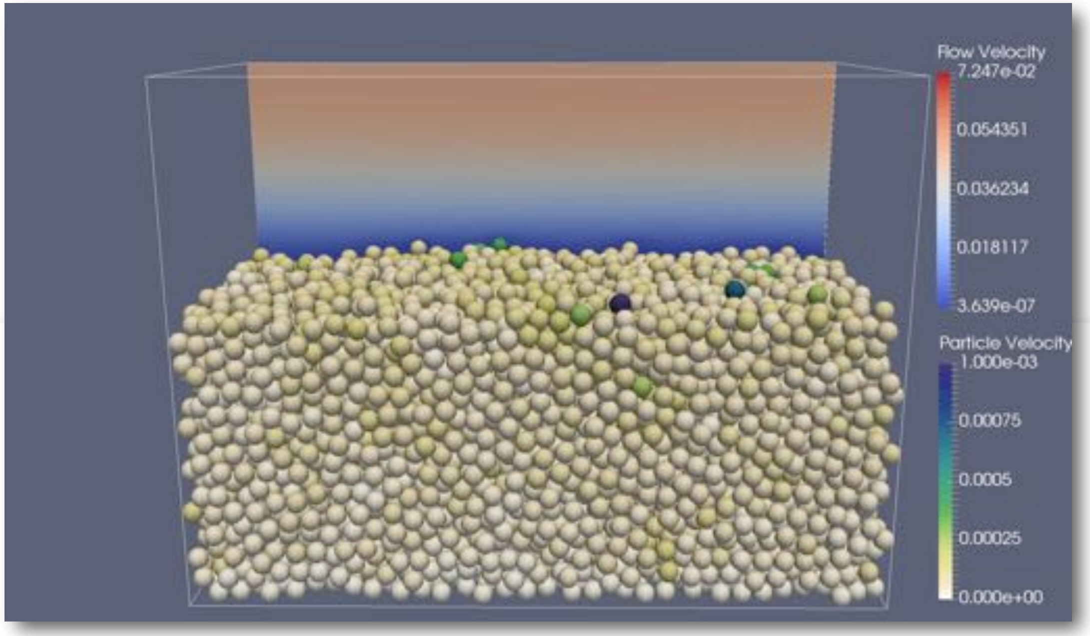


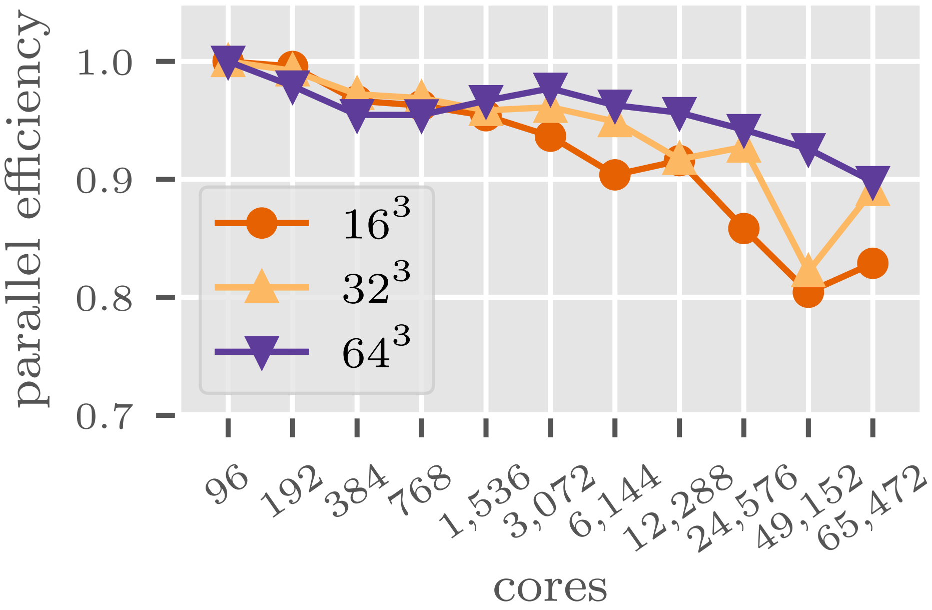
shear stress at bed surface



Preclik, T., Schruff, T., Frings, R., & Rüde, U. (2017, August). Fully Resolved Simulations of **Dune Formation in Riverbeds**. In *High Performance Computing: 32nd International Conference, ISC High Performance 2017, Frankfurt, Germany, June 18-22, 2017, Proceedings* (Vol. 10266, p. 3). Springer.

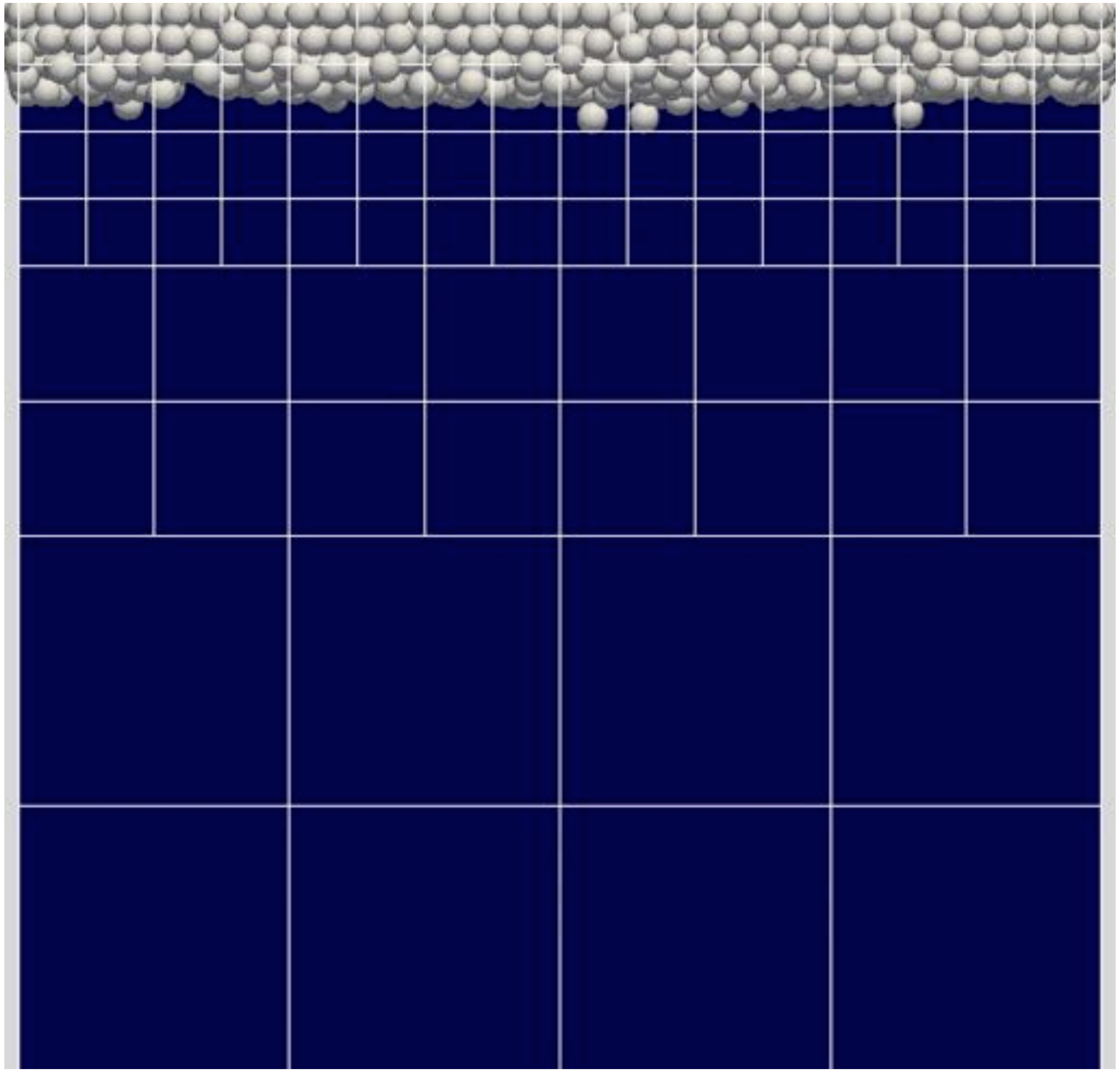
Simulation of suspended particle transport





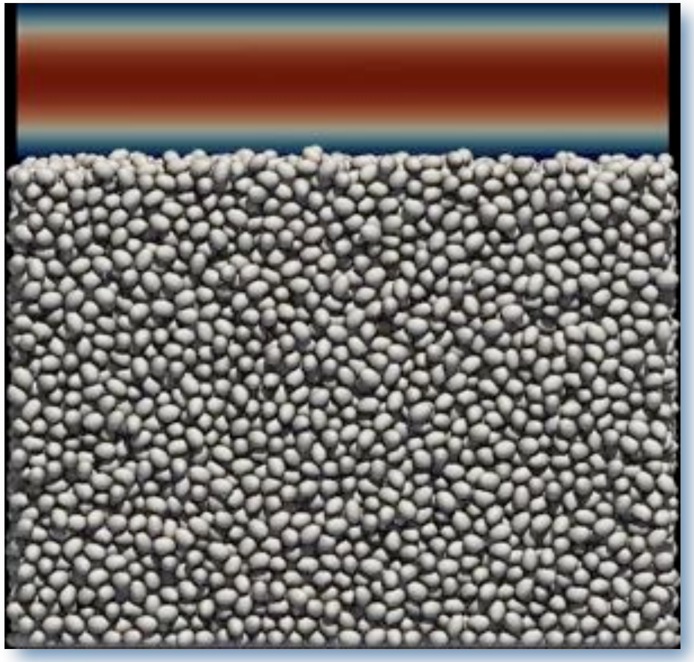
$0.864*10^9$ LBM cells; 350 000 spherical particles

Sedimentation and fluidized beds

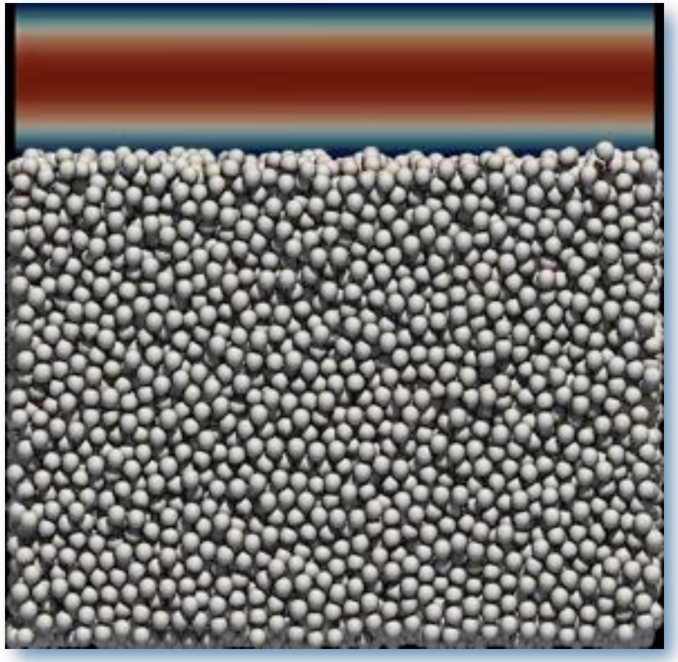


- 3 levels mesh refinement
- 3800 spherical particles
- Galileo number 50
- 128 processes
- 1024-4000 blocks
- Block size 32^3

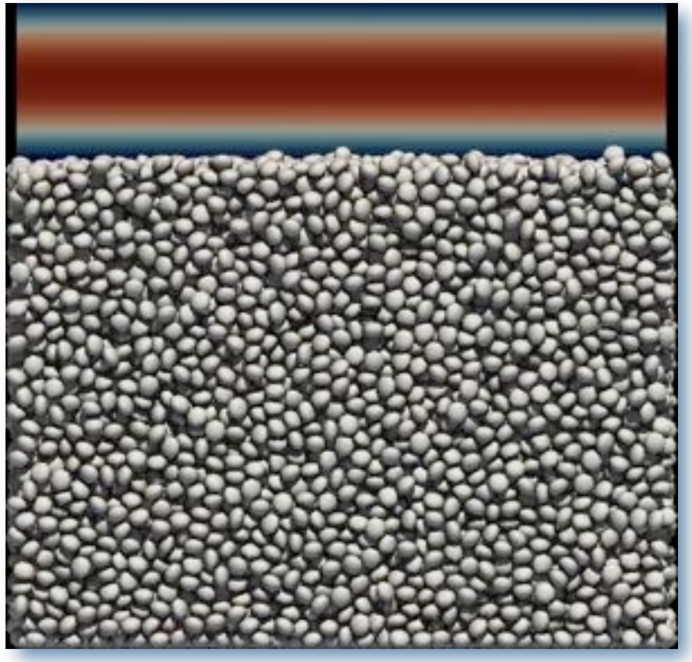
Study of particle shape effects



prolate



spherical



oblate

Rettinger, C., UR (2017). A comparative study of **fluid-particle coupling** methods for fully resolved lattice Boltzmann simulations. *Computers & Fluids, 154, 74-89.*

Rettinger, C., UR (2018). A coupled lattice Boltzmann method and discrete element method for discrete particle simulations of **particulate flows**. *Computers & Fluids*.

Preclik, T., UR (2015). Ultrascale simulations of non-smooth **granular dynamics**. *Computational Particle Mechanics*, *2*(2), 173-196.

Eibl, S., UR (2018). A Systematic Comparison of Dynamic **Load Balancing** Algorithms for Massively Parallel Rigid Particle Dynamics. *arXiv preprint arXiv:1808.00829*.

Why are these codes so fast?

~~Optimizing~~ Pessimizing the Performance

with greetings from D. Bailey's: *Twelve Ways to Fool the Masses...*

Pessimize ↓

- traverse memory in unfavorable order, ignore caches, use many small MPI messages
  - **$10^{13} \rightarrow 10^{12}$ unknowns**
- Do not use a matrix-free implementation: a single multiplication with the mass and stiffness matrix can easily cost 50 memory accesses
  - **$10^{12} \rightarrow 10^{11}$ unknowns**
- Write „generic code“: implement unstructured grids, pointers, using extensively indirect memory access
  - **$10^{11} \rightarrow 10^{10}$ unknowns**
- algorithmic overhead: check convergence redundantly, use an expensive error estimator, etc.
  - **$10^{10} \rightarrow 10^{9}$ unknowns**
    (... still a large system ...)

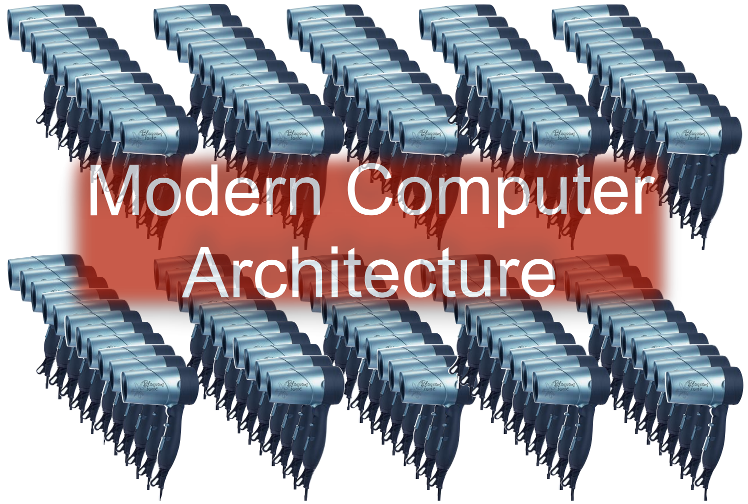
Optimize ↑

Why is optimization hard?

Would you want to propel a Superjumbo

with four strong jet engines

or with 1,000,000 blow dryer fans?



The End of Moore's Law

We are used to 1000x improvement per decade through:

- Transistor Scaling - this will come to a halt

We must change strategy:

- Architectural „improvements“ (GPU, manycore)
- Algorithms
- Implementation

Moore's *deflation of computational cost* has dictated the economics of computing - and now this will gradually change.

In the coming decade …

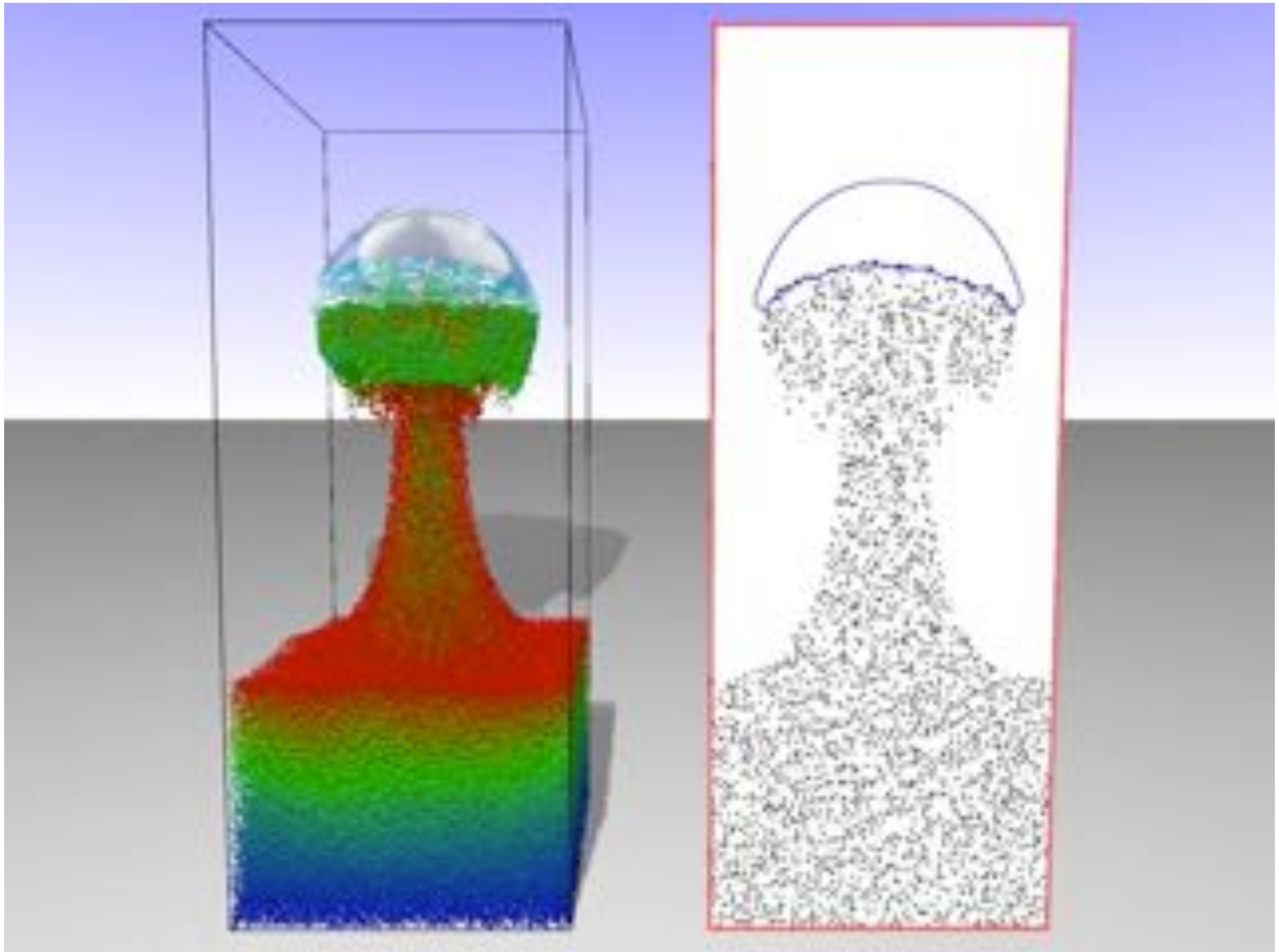
- absolute „performance“ instead of „scalability“
- We will gradually refactor/redesign many of today’s application codes
  - it is not a question of cost but of development time
- Gradual shift of investments from HW to SW
  - slowdown of HW investment cycles
  - performance improvements through „performance engineering“

Computational Science is done in Teams

- Dr.-Ing. Dominik Bartuschat
- Martin Bauer, M.Sc. (hons)
- Dr. Regina Degenhardt
- Sebastian Eibl, M. Sc.
- Dipl. Inf. Christian Godenschwager
- Marco Heisig, M.Sc.(hons)
- PD Dr.-Ing. Harald Köstler
- Nils Kohl, M. Sc.
- Sebastian Kuckuk, M. Sc.
- Christoph Rettinger, M.Sc.(hons)
- Jonas Schmitt, M. Sc.
- Dipl.-Inf. Florian Schornbaum
- Dominik Schuster, M. Sc.
- Dominik Thönnes, M. Sc.

- Dr.-Ing. Benjamin Bergen
- Dr.-Ing. Simon Bogner
- Dr.-Ing. Stefan Donath
- Dr.-Ing. Jan Eitzinger
- Dr.-Ing. Uwe Fabricius
- Dr. rer. nat. Ehsan Fattahi
- Dr.-Ing. Christian Feichtinger
- Dr.-Ing. Björn Gmeiner
- Dr.-Ing. Jan Götz
- Dr.-Ing. Tobias Gradl
- Dr.-Ing. Klaus Iglberger
- Dr.-Ing. Markus Kowarschik
- Dr.-Ing. Christian Kuschel
- Dr.-Ing. Marcus Mohr
- Dr.-Ing. Kristina Pickl
- Dr.-Ing. Tobias Preclik
- Dr.-Ing. Thomas Pohl
- Dr.-Ing. Daniel Ritter
- Dr.-Ing. Markus Stürmer
- Dr.-Ing. Nils Thürey

Thank you for your attention!



Bogner, S., & UR. (2013). Simulation of floating bodies with the lattice Boltzmann method. *Computers & Mathematics with Applications*, *65*(6), 901-913.

Anderl, D., Bogner, S., Rauh, C., UR, & Delgado, A. (2014). Free surface lattice Boltzmann with enhanced bubble model. *Computers & Mathematics with Applications*, *67*(2), 331-339.

Bogner, S. Harting, J., & UR (2017). Simulation of liquid-gas-solid flow with a free surface lattice Boltzmann method. Submitted.