VSB TECHNICAL UNIVERSITY OF OSTRAVA

HPCSE 2026
High Performance Computing in Science and Engineering

NHR FAU

Parallel Solution of Large Scale Finite Element Problems

in honor of **Zdenek Dostal** on his 80th birthday

Ulrich Rüde (ulrich.ruede@fau.de)

May 19, 2026

LSS
Lehrstuhl für Simulation
Universität Erlangen-Nürnberg
https://www.cs10.tf.fau.de

VSB
Technical University of Ostrava

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

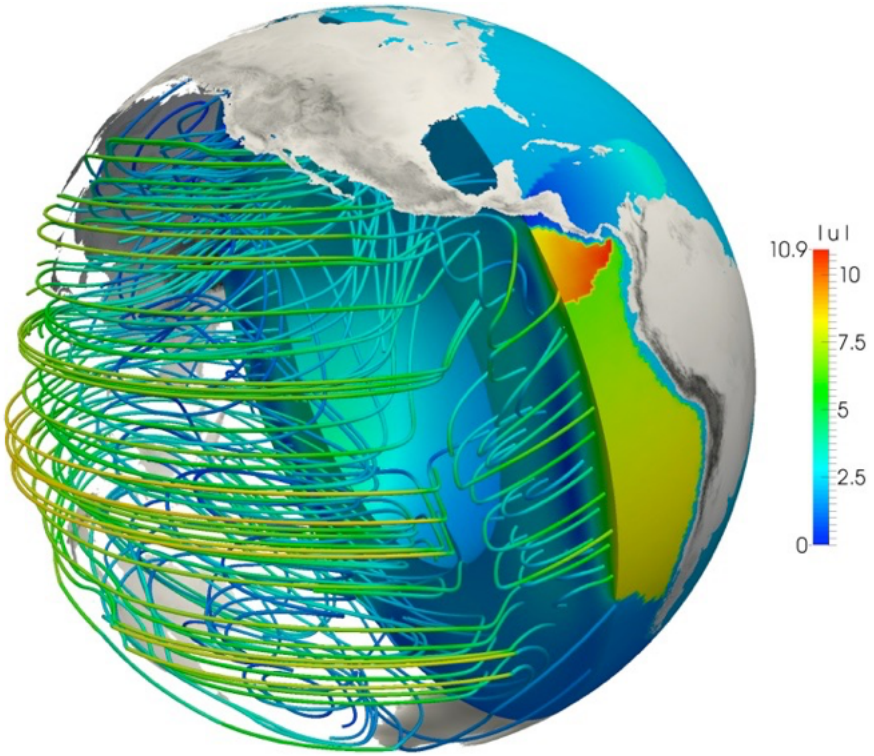
Overview

- Starting from an example: Earth mantle convection

- Scalable Solvers, Multigrid
- Automatic Program Generation
- Achieving Exa-Scale Performance
- „Textbook“ Efficiency

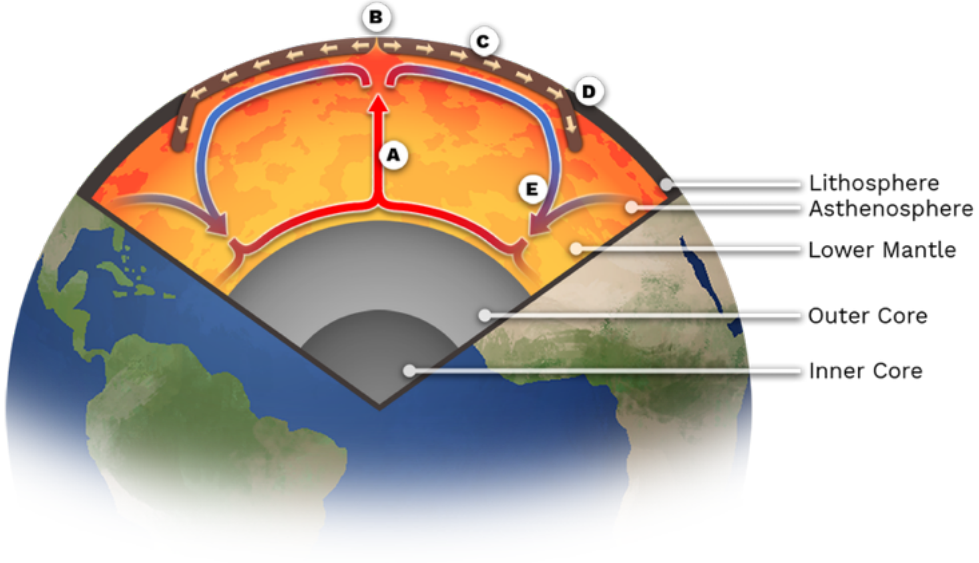
- Supercomputers
- Lattice Boltzmann for Complex Flows (time permitting)

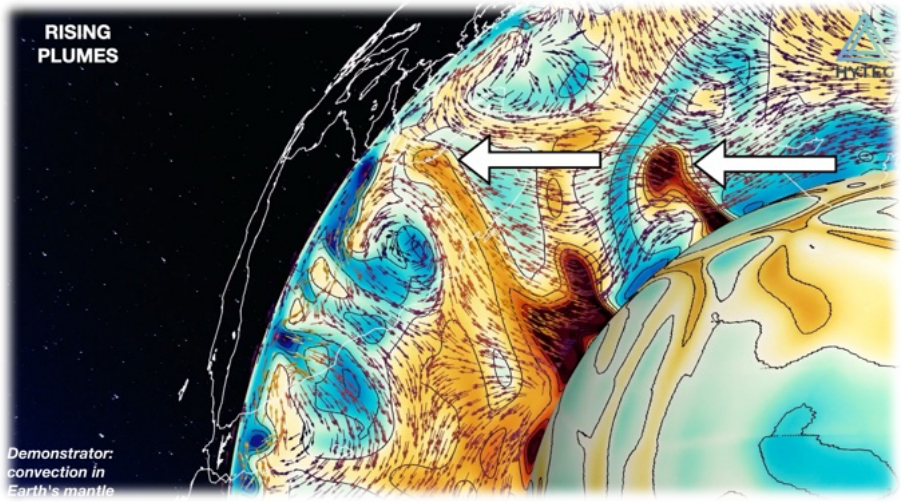
My talk presents results of my teams
and the outcome of many years of collaboration with colleagues



An introductory excursion to Earth Mantle Convection

Earth Mantle convection models:
Stokes equation coupled with energy transport



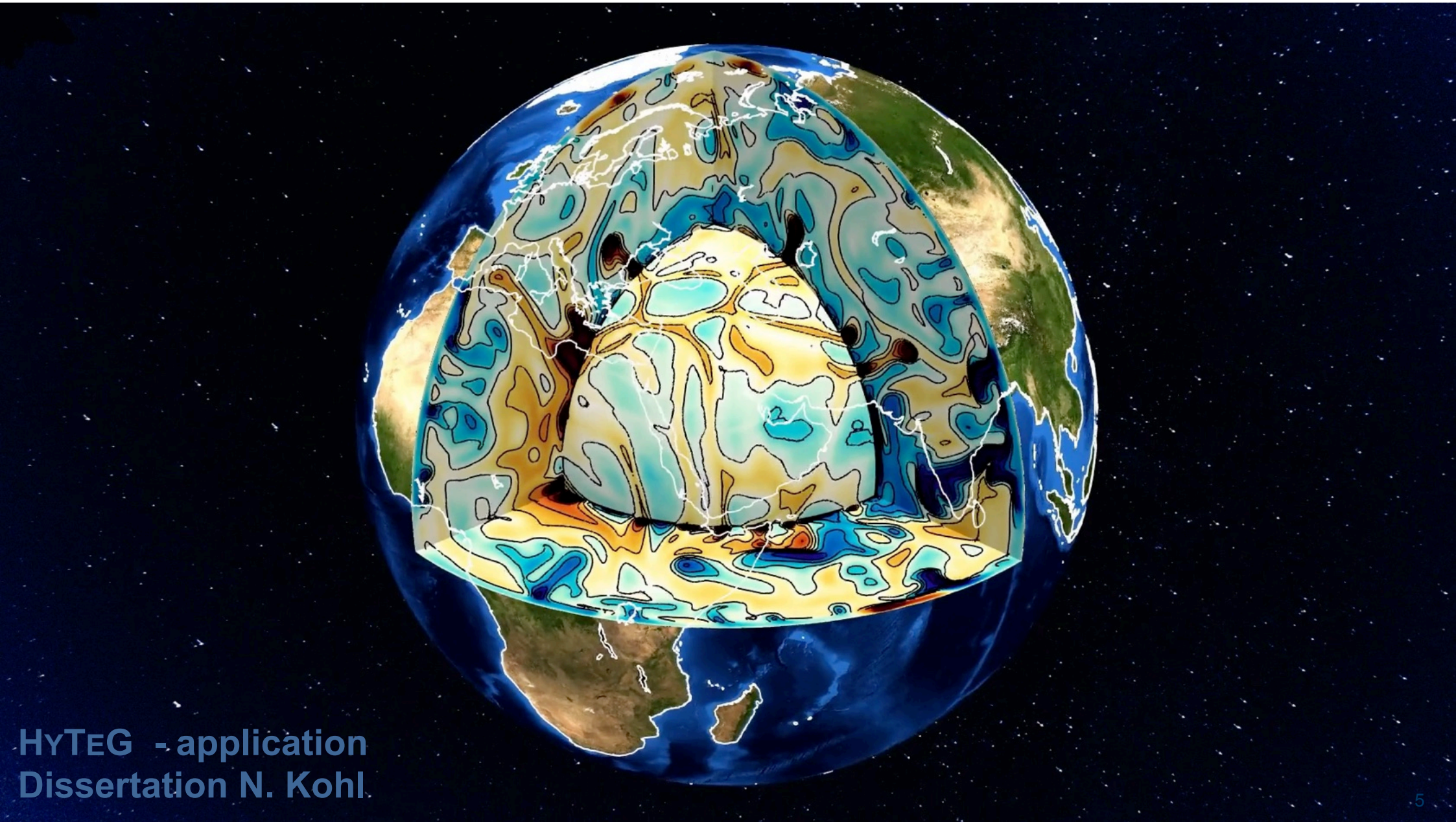


Earth Mantle:

- 2900 km thick spherical shell
- Volume ~ 10^{12} km^3
- Viscuous fluid on geological time scales

Convection:

- Driven by buoyancy
- Hot plumes rising from the core
- Cold lithosphere slabs sink (subduction)



HyTeG - application
Dissertation N. Kohl

Mathematical Model in Weak Form (1)

Momentum: Variable-coefficient, "full" Stokes equation

$$\int_\Omega \rho \, \frac{\partial \boldsymbol{u}}{\partial t} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} + \int_\Omega 2\,\mu\,\boldsymbol{\varepsilon}(\boldsymbol{v}) : \boldsymbol{\varepsilon}(\boldsymbol{u}) - \tfrac{2}{3}\,\mu \,\mathrm{div}(\boldsymbol{v})\,\mathrm{div}(\boldsymbol{u}) \, \mathrm{d}\boldsymbol{x} - \int_\Omega p \, \nabla \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \, \mathrm{d}\boldsymbol{x}$$

Mass: incompressibility constraint

$$\int_\Omega q \, \nabla \cdot \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = 0$$

Energy: advection–diffusion of temperature

$$\int_\Omega \frac{\partial T}{\partial t} \, w \, \mathrm{d}\boldsymbol{x} + \int_\Omega (\boldsymbol{u} \cdot \nabla T) \, w \, \mathrm{d}\boldsymbol{x} + \int_\Omega \kappa \, \nabla T \cdot \nabla w \, \mathrm{d}\boldsymbol{x} = \int_\Omega f_T \, w \, \mathrm{d}\boldsymbol{x}$$

Mathematical Model (2)

Governing equations: Stokes

$$-\nabla \cdot \left(2\,\mu\,\varepsilon(\boldsymbol{u}) - \tfrac{2}{3}\,\mu\,(\nabla\cdot\boldsymbol{u})\,\boldsymbol{I}\right) + \nabla p = \boldsymbol{f}, \qquad \nabla\cdot\boldsymbol{u} = 0$$

$\Downarrow$ (test, integrate by parts)

Weak Formulation — find $(\boldsymbol{u}, p) \in V \times Q$ s.t. $\forall (\boldsymbol{v}, q) \in V \times Q$:

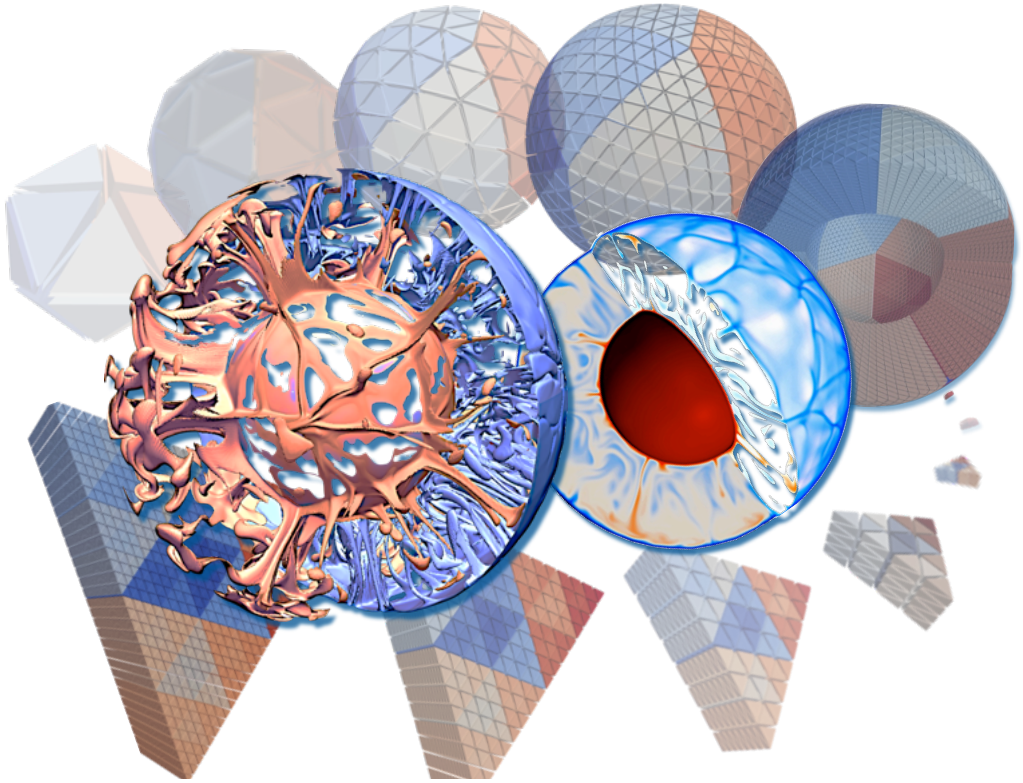
$$\int_\Omega 2\,\mu\,\varepsilon(\boldsymbol{v}) : \varepsilon(\boldsymbol{u}) - \tfrac{2}{3}\,\mu\,\mathrm{div}(\boldsymbol{v})\,\mathrm{div}(\boldsymbol{u})\,\mathrm{d}\boldsymbol{x} - \int_\Omega p\,\nabla\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}\boldsymbol{x}$$

$$\int_\Omega q\,\nabla\cdot\boldsymbol{u}\,\mathrm{d}\boldsymbol{x} = 0$$

$\Downarrow$ (FE discretization)

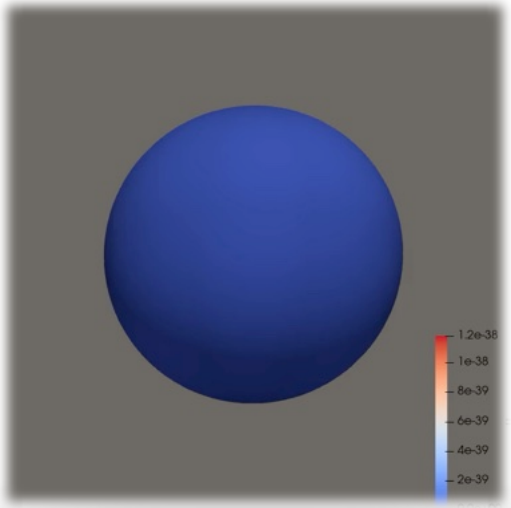
Saddle-point system

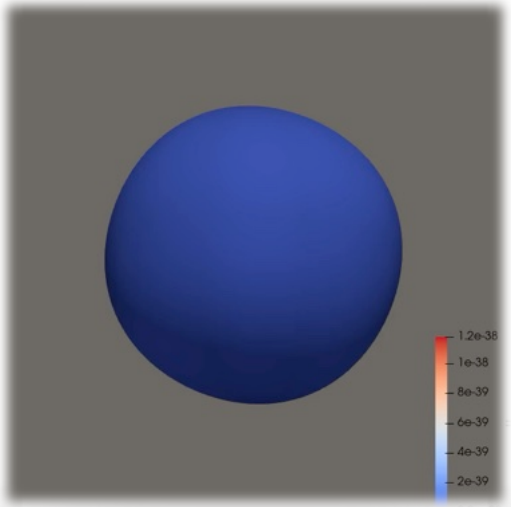
$$\begin{pmatrix} A & B^\top \\ B & 0 \end{pmatrix} \begin{pmatrix} U \\ P \end{pmatrix} = \begin{pmatrix} F \\ 0 \end{pmatrix}, \quad \text{solved by multigrid + Krylov}$$



TerraNeo Project: History & Perspectives

Year	Milestone	
1983	First version of **TERRA** (Baumgardner), sequential, constant viscosity	Fortran 77, MPI
1996	First parallel version using PVM (Bunge)	
1997	Variable viscosity, operator-dependent MG transfers (Yang)	
1999	Ported to Cray T3D/T3E, >1 TFLOP/s (NASA/SGI)	
2008	Local grid refinement in boundary layers (Davies)	
2013–19	**TerraNeo**/SPPEXA: HyTeG framework (Rüde, Wohlmuth, Bunge)	C++, MPI
2017/2022	Surrogate polynomials (Driszga, Wagner, Mann)	
2020	Scalability beyond 10^{12} DoFs for CCVL (S. Bauer)	
2022	Textbook multigrid efficiency for Stokes (Kohl)	
2022	Lagrangian-Eulerian advection / particles (Kohl, Mohr)	
2025	Adaptive mesh refinement (Mann)	
2025	Code generation (Böhm, D. Bauer)	
2025	EMC on Full SNG Phase 1, >1 PFLOP/s, 10^{11} DoFs (P. Ilangovan, Mohr)	
2026-X	GPU porting, new proposal by LMU/TUM, Aim: JUPITER (1 EFLOP/s)	+ Kokkos





Hülsemann, F., Bergen, B., Rüde, U. (2003, August). Hierarchical hybrid grids as basis for parallel numerical solution of PDE. In *European Conference on Parallel Processing* (pp. 840-843). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bergen, B., Gradl, T., Hulsemann, F., Rüde, U. (2006). A massively parallel multigrid method for finite elements. *Computing in Science & Engineering*, *8*(6), 56-62.

The problem: Solving systems with #dof ~ 10^{12}

- Each double precision vector will be 8 TByte large
- We cannot use global 32 bit indices
- If the stiffness matrix has 100 nonzeros per row, we will need >1.6 PByte to store the matrix in a typical sparse matrix format
  - Even for an exascale machine this is a lot
- The inverse matrix is dense and thus requires 10^{24} double precision entries.
  - If computing each of them costs a nJ energy, the total energy is 277 GWh
- The solution algorithm and no part of it must have quadratic or worse complexity. Only linear (or log-lin?) complexity is acceptable
- We must go matrix free to save memory and memory traffic
- We should use reduced precision when feasible

Parallel Algorithms for saddle point systems

Benzi, M., Golub, G. H., Liesen, J. (2005). Numerical solution of saddle point problems. *Acta numerica*, *14*, 1-137.

Rozložník, M. (2018). *Saddle-point problems and their iterative solution*. Basel: Birkhäuser.

- Monolithic multigrid or block approaches

Gmeiner, B., Rüde, U., Stengel, H., Waluga, C., Wohlmuth, B. (2015). Towards textbook efficiency for parallel multigrid. *Numerical Mathematics: Theory, Methods and Applications*, *8*(1), 22-46.

Darrigrand, V., Dumitrasc, A., Kruse, C., & Rüde, U. (2023). Inexact inner–outer Golub–Kahan bidiagonalization method: A relaxation strategy. *Numerical Linear Algebra with Applications*, *30*(5), e2484.

Lukáš, D., & Dostál, Z. (2007). Optimal multigrid preconditioned semi-monotonic augmented Lagrangians applied to the Stokes problem. *Numerical linear algebra with applications*, *14*(9), 741-750.

- Parallel Methods, Domain Decomposition

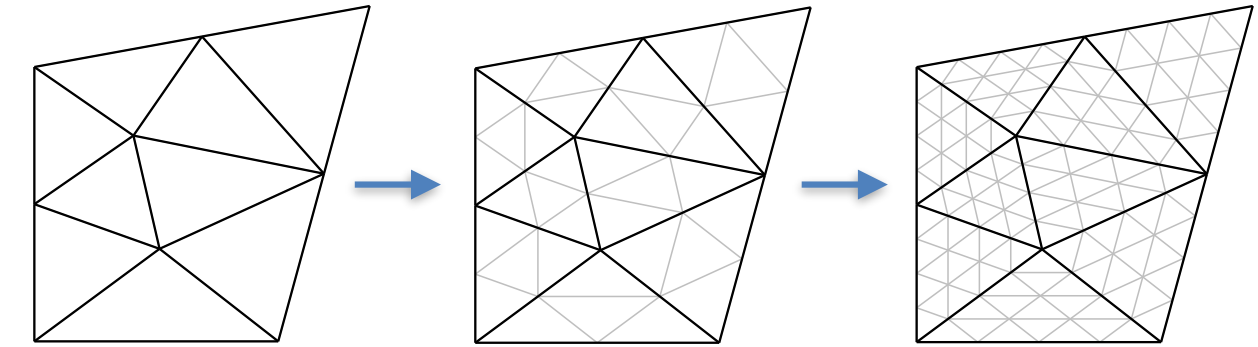
Kohl, N., & Rüde, U. (2022). Textbook efficiency: massively parallel matrix-free multigrid for the Stokes system. *SIAM Journal on Scientific Computing*, *44*(2), C124-C155.

Dostál, Z., & Horák, D. (2003). Scalability and FETI based algorithm for large discretized variational inequalities. *Mathematics and Computers in Simulation*, *61*(3-6), 347-357.

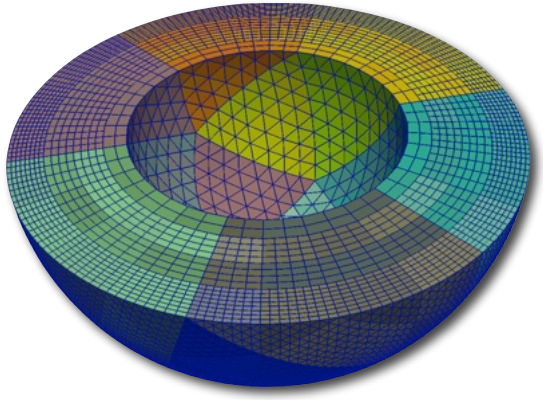
Blaheta, R., Luber, T., & Kružík, J. (2018). Schur Complement-Schwarz DD Preconditioners for Non-stationary Darcy Flow Problems. In High Performance Computing in Science and Engineering: Third International Conference, HPCSE 2017, Karolinka, Czech Republic, May 22–25, 2017, Revised Selected Papers 3 (pp. 59-72). Springer.

Dostál, Z., Horák, D., & Kučera, R. (2006). Total FETI—an easier implementable variant of the FETI method for numerical solution of elliptic PDE. *Communications in Numerical Methods in Engineering*, *22*(12), 1155-1162.

Dostál, Z., Horák, D. (2025). The least squares solution of inconsistent discretized elliptic problems using the FETI method. *Applications of Mathematics*, *70*(6), 929-939.



Hierarchical Hybrid Tetrahedral Grids for Finite Elements

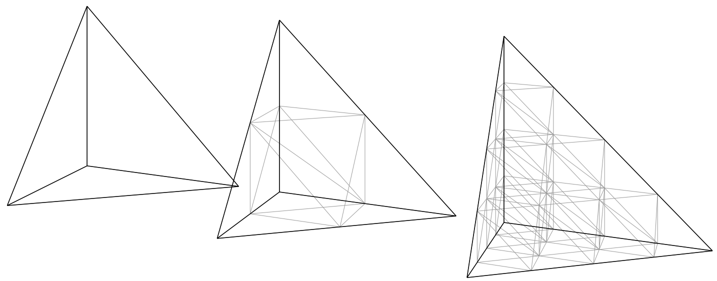


Bergen, B. K., & Hülsemann, F. (2004). **Hierarchical hybrid grids**: data structures and core algorithms for multigrid. *Numerical linear algebra with applications*, *11*(2-3), 279-291.

Bergen, B., Gradl, T., Hulsemann, F., & UR (2006). A massively **parallel multigrid** method for finite elements. *Computing in science & engineering*, *8*(6), 56-62.

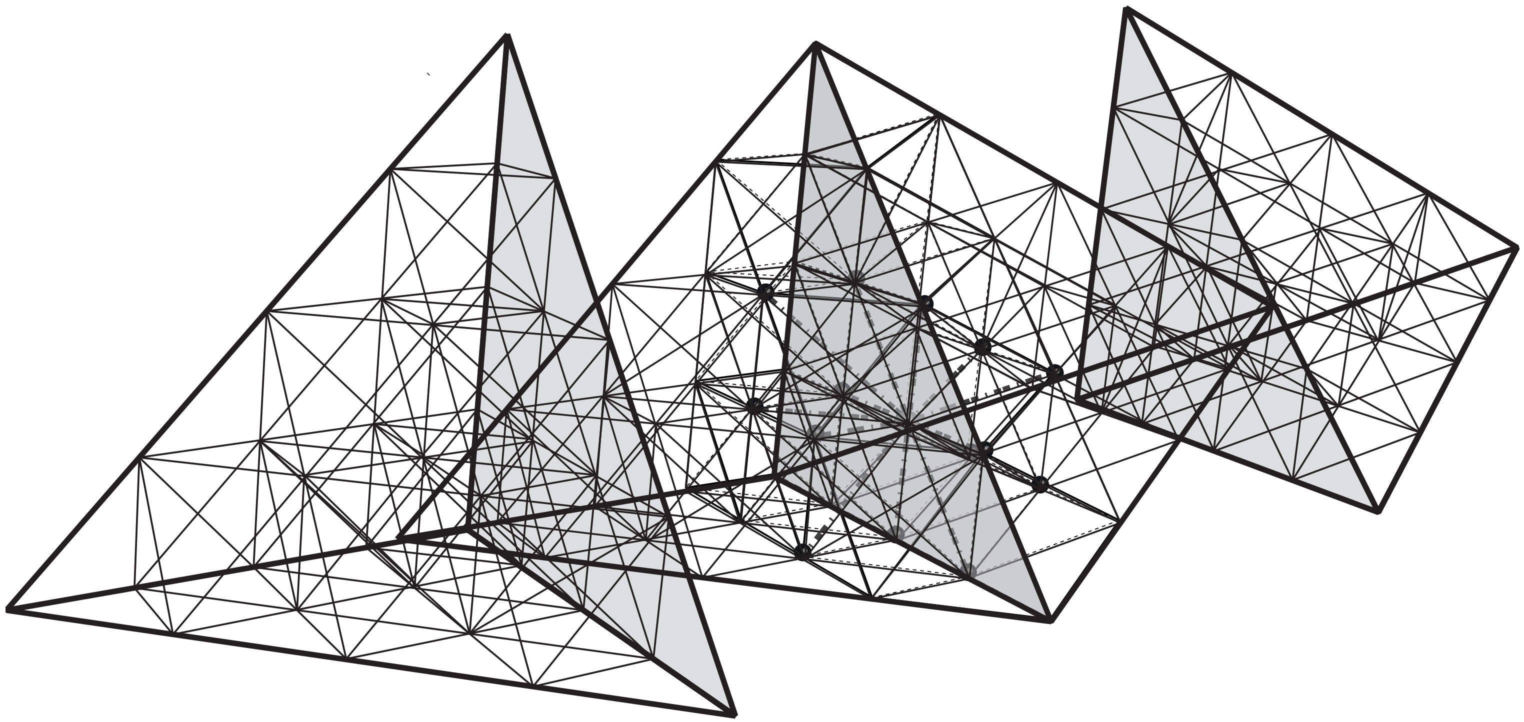
Kohl, N., Thönnes, D., Drzisga, D., Bartuschat, D., UR (2019). The **HyTeG** finite-element software framework for scalable multigrid solvers. *International Journal of Parallel, Emergent and Distributed Systems*, *34*(5), 477-496.

HyTeG: A matrix-free architecture for FE



Structured refinement of an unstructured base mesh

Geometrical Hierarchy: Volume, Face, Edge, Vertex



10 years ago …

Gmeiner et al. 2016, A quantitative **performance study for Stokes** solvers at the extreme scale, Journal of Computational Science.

- Stokes system
- matrix-free multigrid with Uzawa smoother
- optimized for minimal memory consumption

- 10^{13} Unknowns correspond to 80 TByte for the solution vector
- Juqueen had ~450 TByte memory
- matrix free implementation essential

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

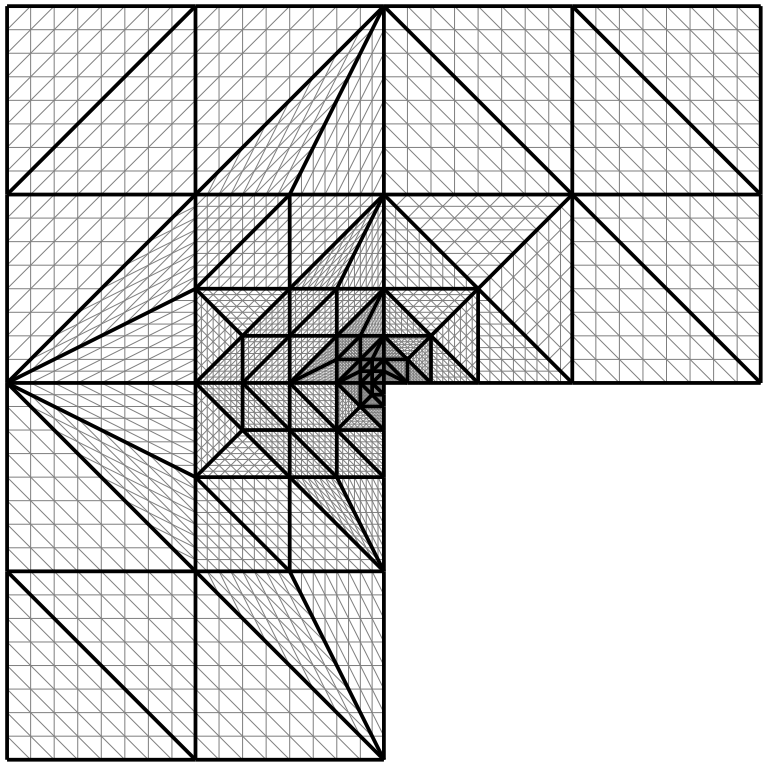
Current Development Lines

- Sustainable Scientific Software Development with User Focus
  - making the software better usable for the application community
- Model Evolution (in collaboration with LMU and TUM)
  - ever more complex models (viscosity, compressibility, …)
  - inverse problems, UQ
- Other Applications - Fusion Plasma Physics
  - In collaboration with CEA and CERFACS (Aakash Dasari, C. Kruse, P. Mycek)
- Performance Portability
  - Automatic Code Generation (F. Böhm, D. Bauer, N. Kohl)
  - GPU Port (N. Kohl, F. Böhm)
- Adaptive Meshes (B. Mann)
- Mixed Precision (D. Bauer and collab. with P. Vacek/Erin Carson)

HYTEG

Hybrid Tetrahedral Grids

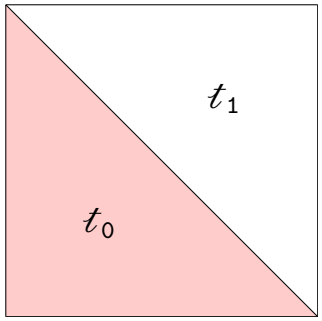
Adaptive Mesh Refinement



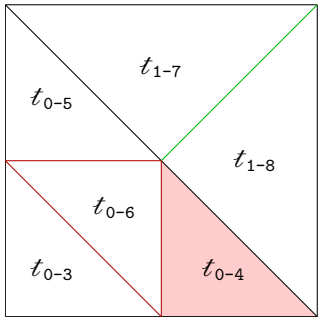
Mann, B., & Rüde, U. (2025). An adaptive mesh refinement scheme for hierarchical hybrid grids. *Applications of Mathematics*, *70*(6), 875-905.

Parallel Adaptive Mesh Refinement (1) (B. Mann)

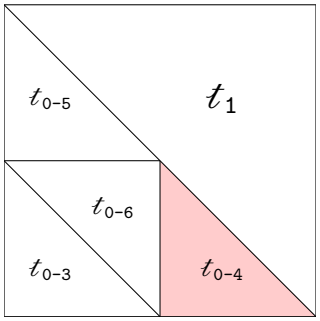
- Red-Green-Refinement revisited



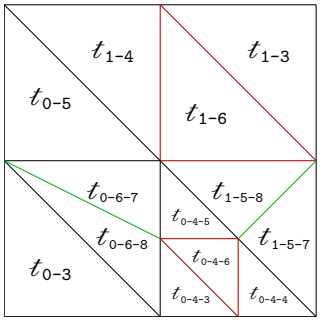
one triangle
marked for refinement



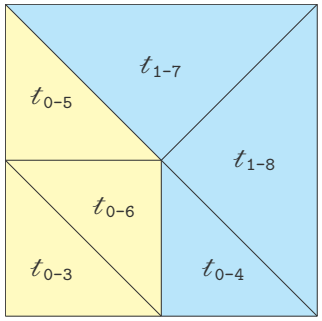
refinement lev. 1
with „green closure“



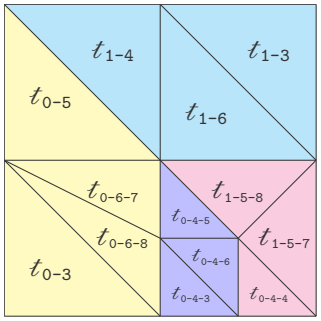
„green closure“
removed before
next level refinement



refinement lev. 2
with „green closures“



load balancing +
processor assigment



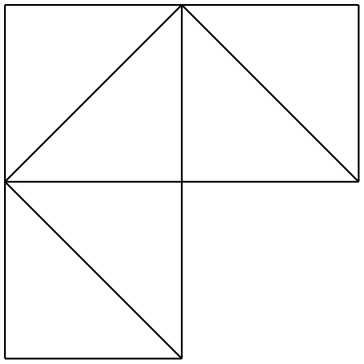
load balancing +
processor assigment

Does this process make sense?

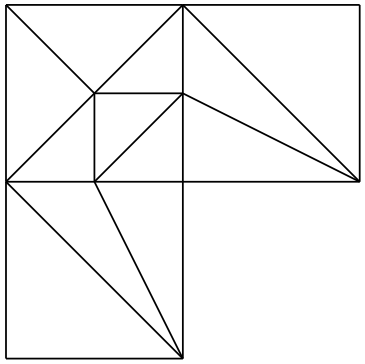
When there are 10^{12} elements in 3D distributed across thousands of cores?

Parallel Adaptive Mesh Refinement (2) (B. Mann)

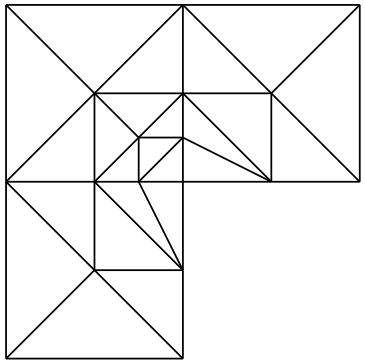
- KL-Refinement (red-green-refinement adapted to HyTeG)



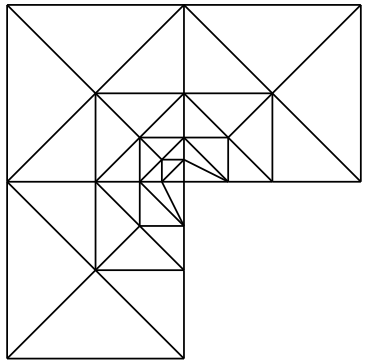
k = 0



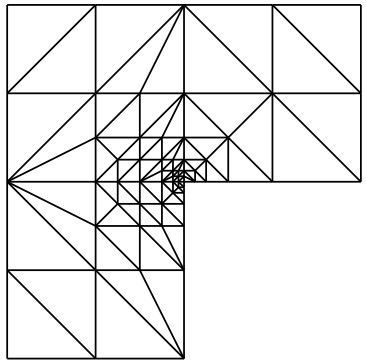
k = 1



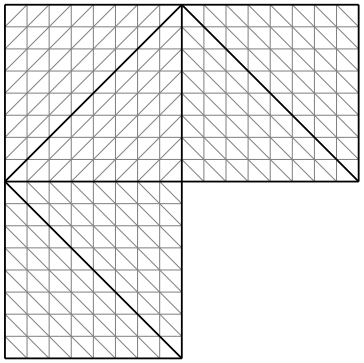
k = 2



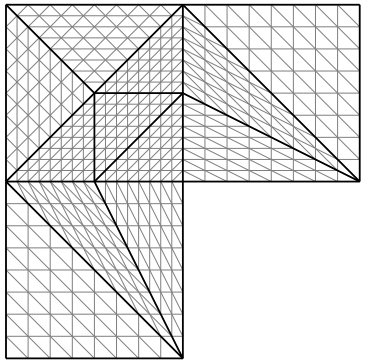
k = 3



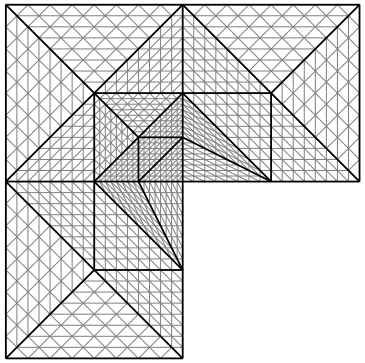
k = 4



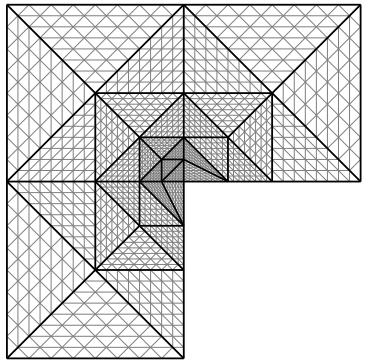
k=0, l=3



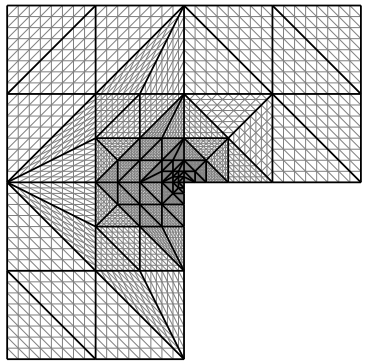
k=1, l=3



k=2, l=3

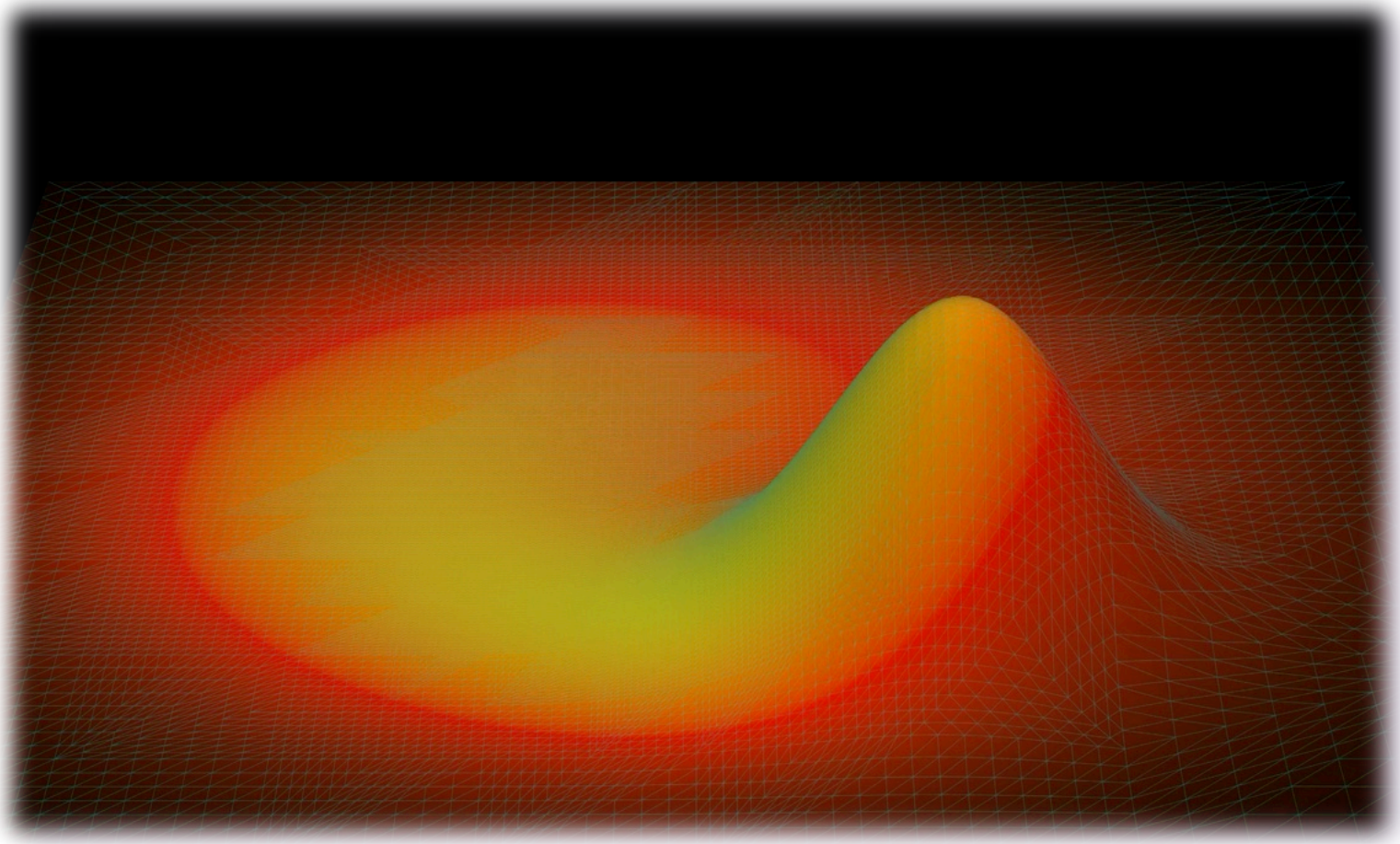


k=3, l=3

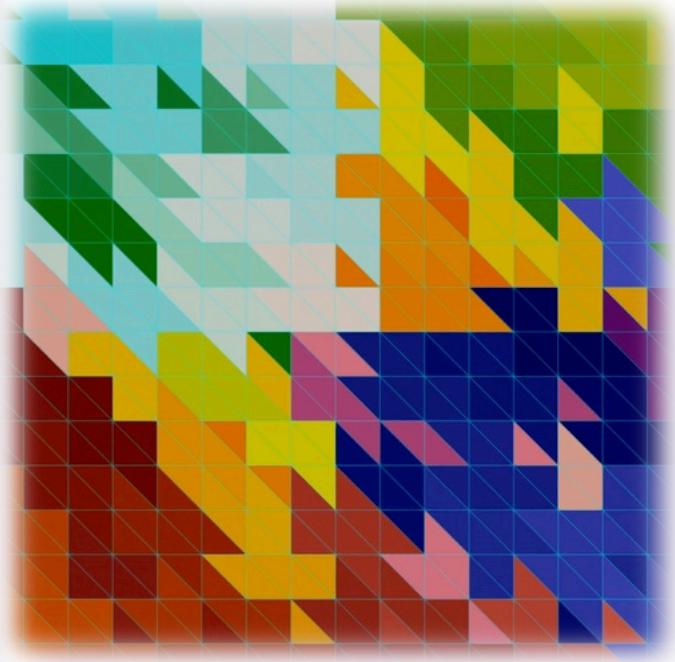


k=4, l=3

Adaptive HyTeG meshes for time dependent problem



- „Moving heat source“ simulation



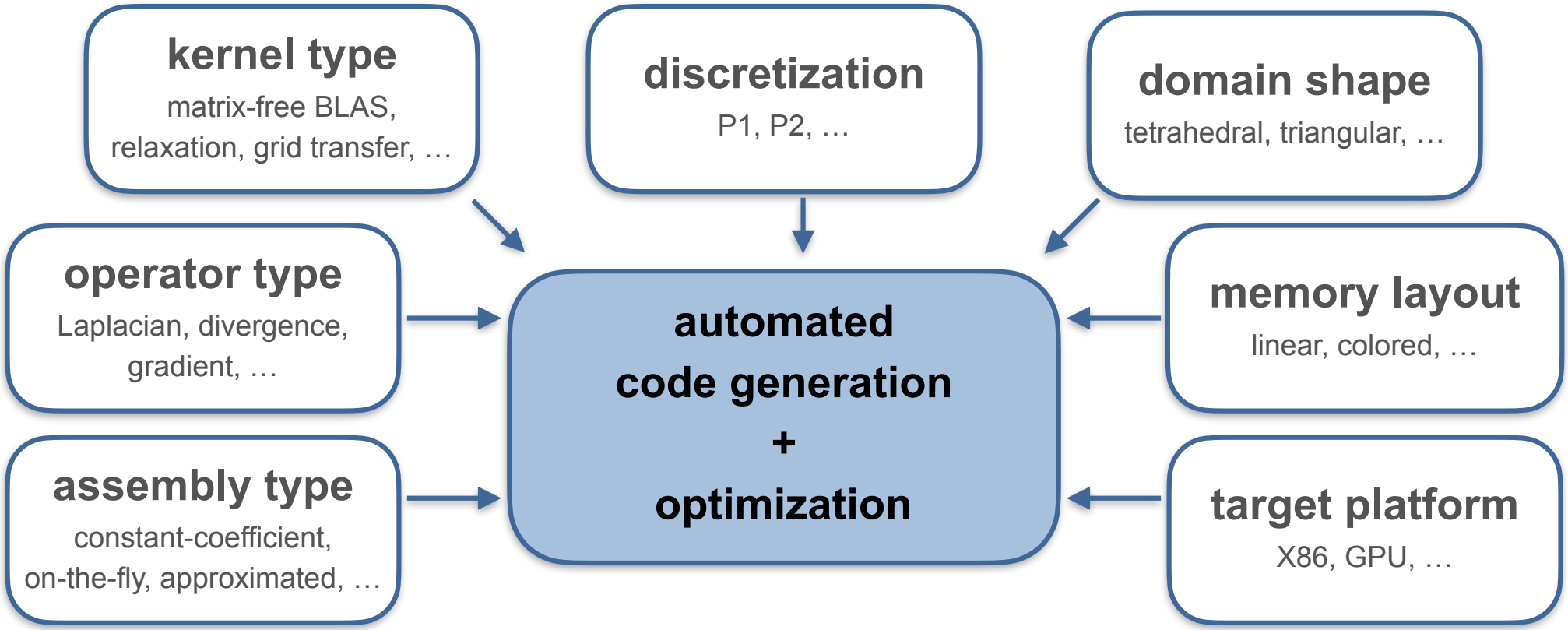
- Processor assignment
- Load balancing
- Data re-distribution

Automatic Code Generation Metaprogramming

Böhm, F., Bauer, D., Kohl, N., Alappat, C., Thönnes, D., Mohr, M., ... & Rüde, U. (2024). Code Generation and Performance Engineering for Matrix-Free Finite Element Methods on Hybrid Tetrahedral Grids. SISC 225, vol 47, pp. B131-B159

The HyTeG framework - code generation

Combinatorial explosion leads to many different kernels and would require **an enormous manual implementation and optimization effort!**



HyTeG Operator Generator (HOG)

Inputs to HOG

define performance properties

$$\int_\Omega G(\boldsymbol{x}, \boldsymbol{v}, \boldsymbol{w}) d\boldsymbol{x}$$

Weak form

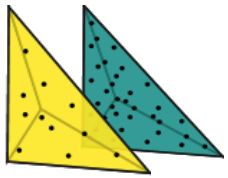
$\mathcal{P}_2$

FE space

AVX2

Architecture

Quadrature



Abstract syntax tree

Provides option to transform the program

For(…++elZ)

For(…++elY)

For(…++elX)

SrcDoF = ...

DstDoF = ...

Tmp = ...

Tailored optimizations

e.g. vectorization

Transformed AST

...

Tmp = ...

For(…++elZ)

For(…++elY)

For(…elX +=4)

For(…++elX)

Vec4 SrcDoF = ...

...

...

Python Code Generator

Print sweep

```
for (int64_t ctr_2 = 0; ctr_2 < mic
for (int64_t ctr_1 = 0; ctr_1 < -ct
for (int64_t ctr_0 = 0; ctr_0 < -ct
{
   const double tmp_WHITE_UP_0 = ma
   const double tmp_WHITE_UP_1 = po
   const double tmp_WHITE_UP_2 = tm
```

C++ Framework

HYTEG

Hybrid Tetrahedral Grids

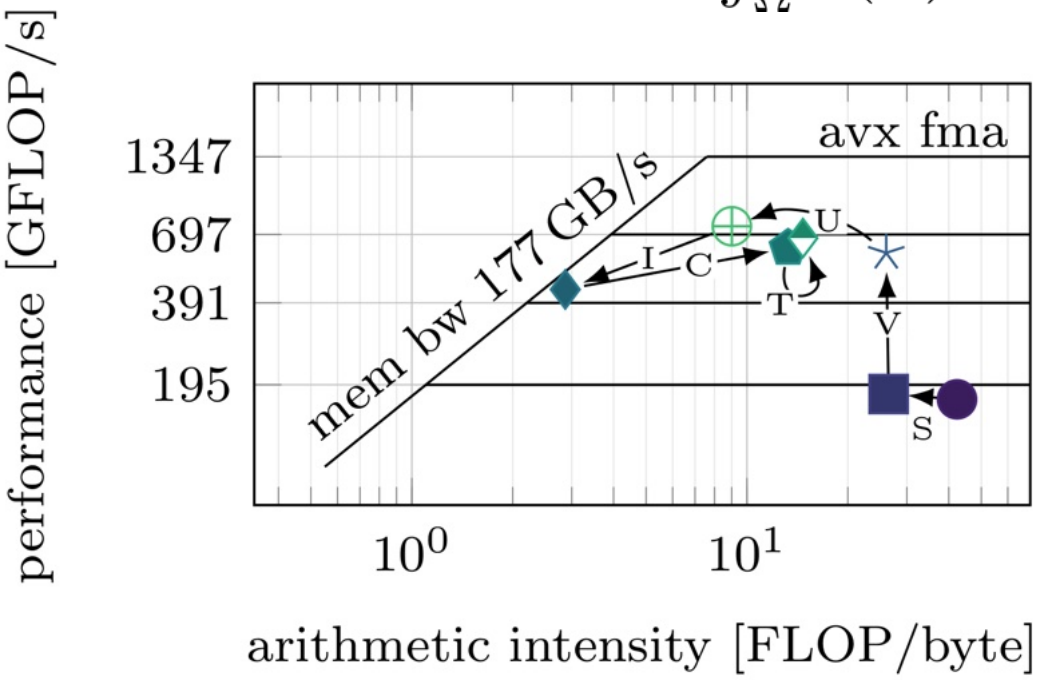
Output: C++ code

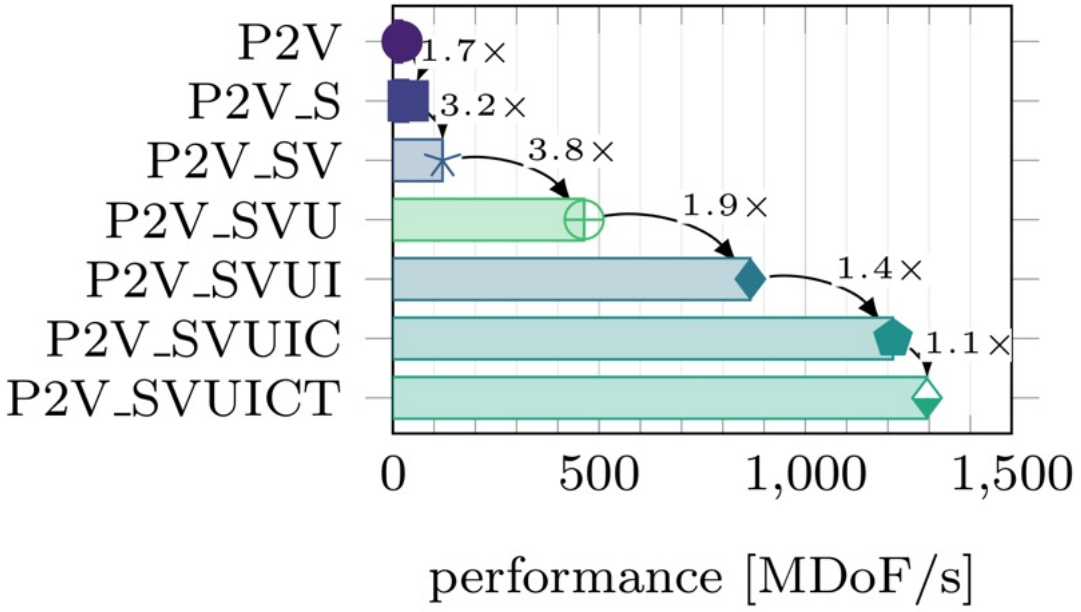
Scalable, node-level optimized, matrix-free FE operator

Optimization Path: P2V

Operator P2V: $\int_\Omega k(\boldsymbol{x}) \nabla v \cdot \nabla w, \mathcal{P}_2$

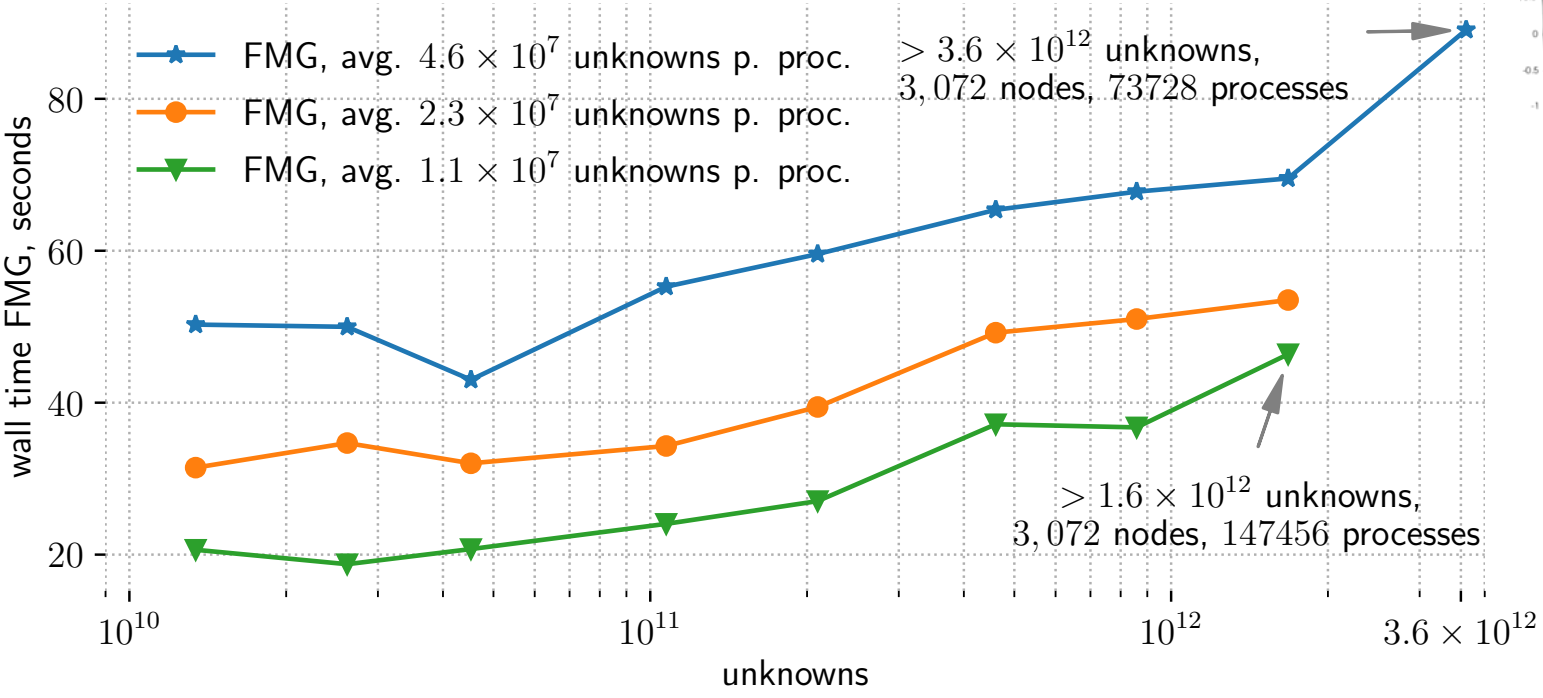
- Symmetry (S)
- Inter-element vectorization (V)
- Loop invariants (I)
- Cubes loop strategy (C)
- Under-integration (U)
- Fused quadrature loops (fQ)
- Tabulation (T)

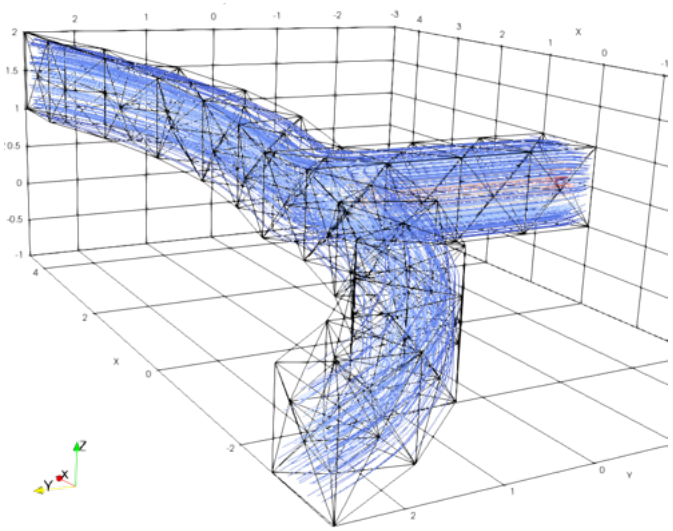




- Starting point: already compute-bound
- Series of opts reducing arithmetic intensity
- Compute-intense **P2V becomes memory-bound** with P2V_SVUI
- Cubes loop applicable -> more speed-up
- **58x accumulated speed-up, 50% peak, 1.4 GDoF/s**

HyTeG: Scaling for the Stokes Problem





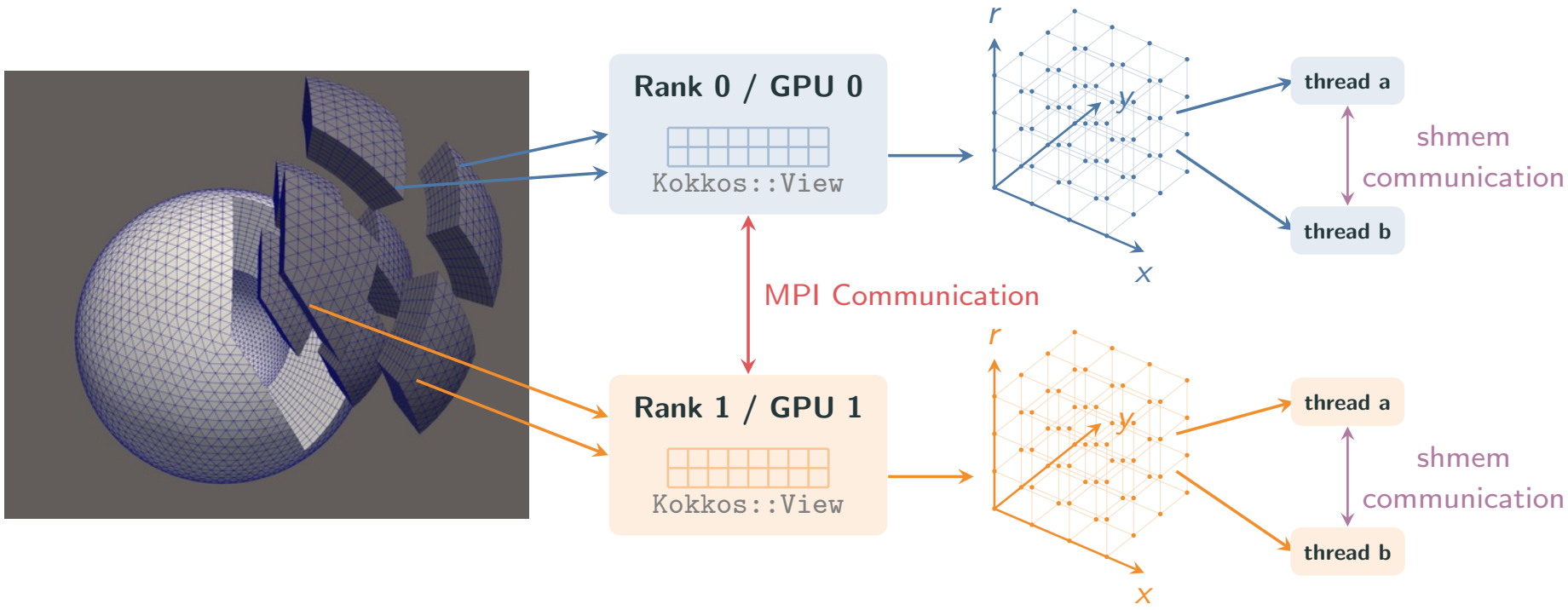
Kohl, N., & Rüde, U. (2022). **Textbook efficiency**: massively parallel matrix-free multigrid for the Stokes system. *SIAM Journal on Scientific Computing*, *44*(2), C124-C155.

Kohl, N., Mohr, M., Eibl, S., & Rüde, U. (2022). A Massively Parallel Eulerian-Lagrangian Method for Advection-Dominated Transport in Viscous Fluids. *SIAM Journal on Scientific Computing*, *44*(3), C260-C285.

Towards a GPU-ready Earth Mantle Simulation

(N. Kohl, F. Böhm, M. Mohr)

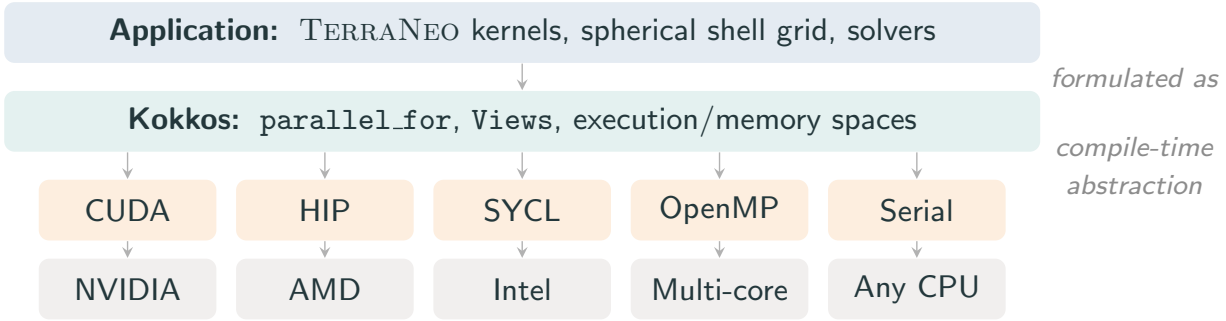
- Key Step: simplify mesh structures & specialize to single app scenario



Why Kokkos? Performance Portability!

- C++ performance portability library
- Single codebase for CPUs and GPUs
- Developed at Sandia, Oak Ridge & CEA
- >1200 developers, >150 institutions, ~ 300 HPC projects

In a nutshell:



JUWELS Booster

JSC, Jülich

LUMI-G

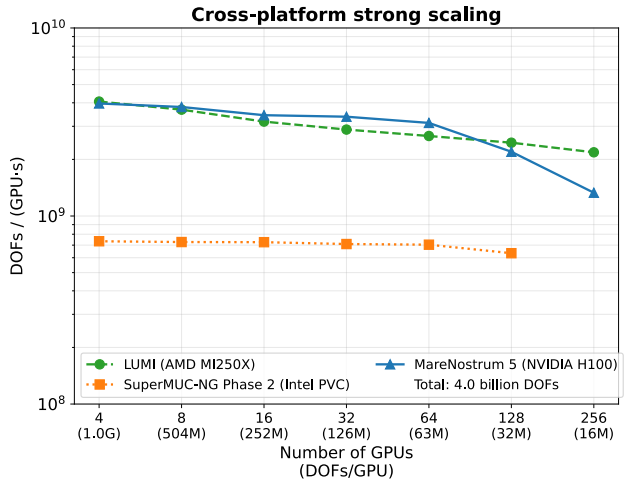
CSC, Kajaani

SuperMUC-NG Phase 2

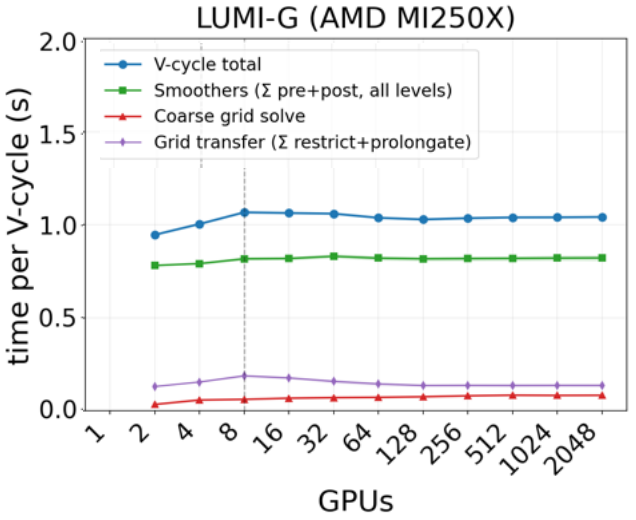
LRZ, Garching

	NVIDIA	AMD	Intel
GPU	A100	MI250X	PVC (Max 1550)
Toolkit	CUDA Toolkit	ROCm	oneAPI
Compiler	`nvcc_wrapper`	`hipcc`	`icpx`
Arch flag	`AMPERE80`	`AMD_GFX90A`	`INTEL_PVC`

Matrix-free SpMV kernel



V-Cycle Scaling



$$\frac{\mathfrak{W}(\mathrm{MG})}{\mathfrak{W}(A)} < 10$$

Textbook Multigrid Efficiency

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]

This is a programmatic claim - not a theorem.

Is it achievable?

For which types of PDE?

Work unit (WU)

- Linear system $Ax = b$
- Work unit (WU) to apply operator: $1\mathrm{WU} := \mathfrak{W}(A)$
  - or perform one sweep of relaxation
- TME achieved, if work for MG solver(!) less than 10 WU:

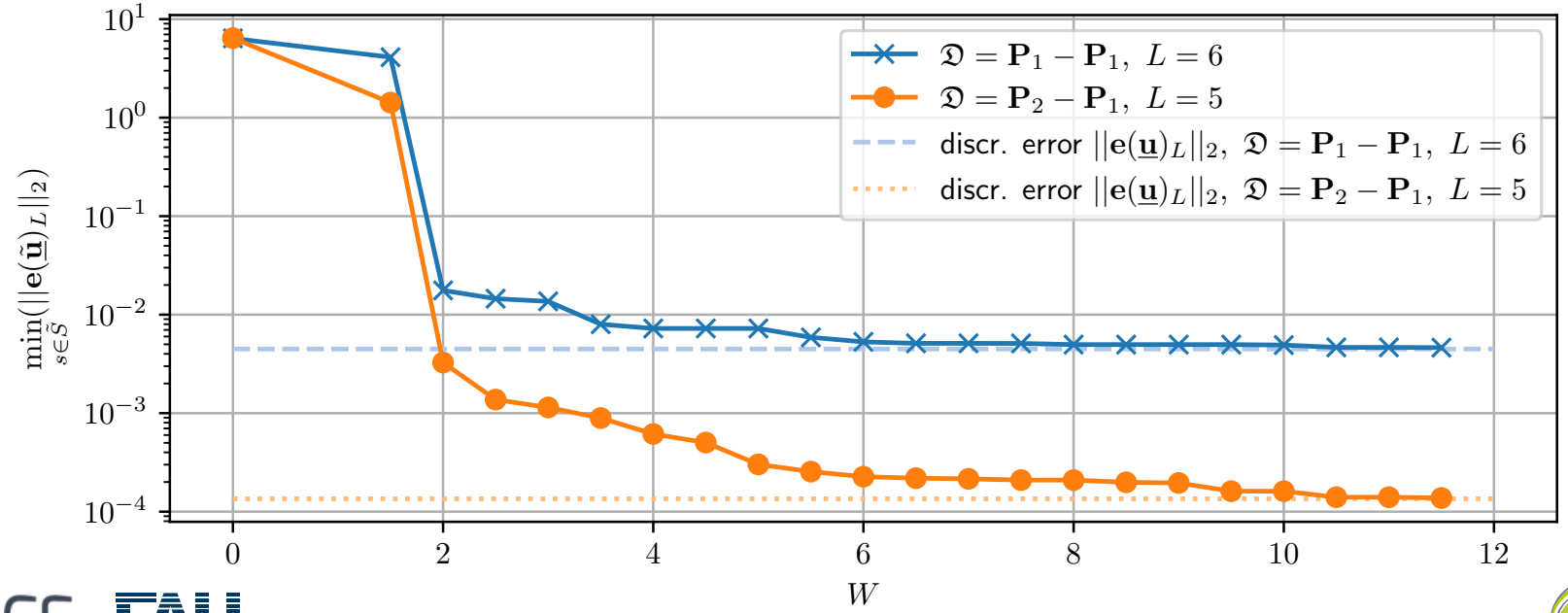
$$\frac{\mathfrak{W}(\mathrm{MG})}{\mathfrak{W}(A)} < 10$$

- TME defined wrt. to underlying differential equation
- TME is (much!) more ambitious than asymptotic optimality or mesh independent convergence of an iterative solver
- TME requires to quantify the constant
  - Hard to assess theoretically
  - But systematic numerical studies possible

Cost comparison for Stokes with stabilized P1-P1 vs. P2-P1

$$\lim_{\ell\to\infty} \frac{\mathfrak{W}(\mathbf{A}_\ell^{\mathbf{P}_2-\mathbf{P}_1})}{\mathfrak{W}(\mathbf{A}_{\ell+1}^{\mathbf{P}_1-\mathbf{P}_1})} = \frac{23}{12}, \qquad \lim_{\ell\to\infty} \frac{\mathfrak{W}(\mathbf{B}_\ell^{\mathbf{P}_2-\mathbf{P}_1})}{\mathfrak{W}(\mathbf{B}_{\ell+1}^{\mathbf{P}_1-\mathbf{P}_1})} = \frac{13}{24} \qquad \lim_{\ell\to\infty} \frac{\mathfrak{W}(\mathcal{A}_\ell^{\mathbf{P}_2-\mathbf{P}_1})}{\mathfrak{W}(\mathcal{A}_{\ell+1}^{\mathbf{P}_1-\mathbf{P}_1})} = \frac{9}{10}$$

- A WU for P2-P1 and for P1-P1 are roughly equivalent
- Velocity error after an FMG iteration with parameterization chosen to achieve minimal error



Supercomputers

Lets come to the old-man-story-telling part of the talk

SuperMUC-NG: Leibniz Supercomputing Center Garching/Munich

Jupiter at Forschungszentrum Jülich

A personal review of computer evolution

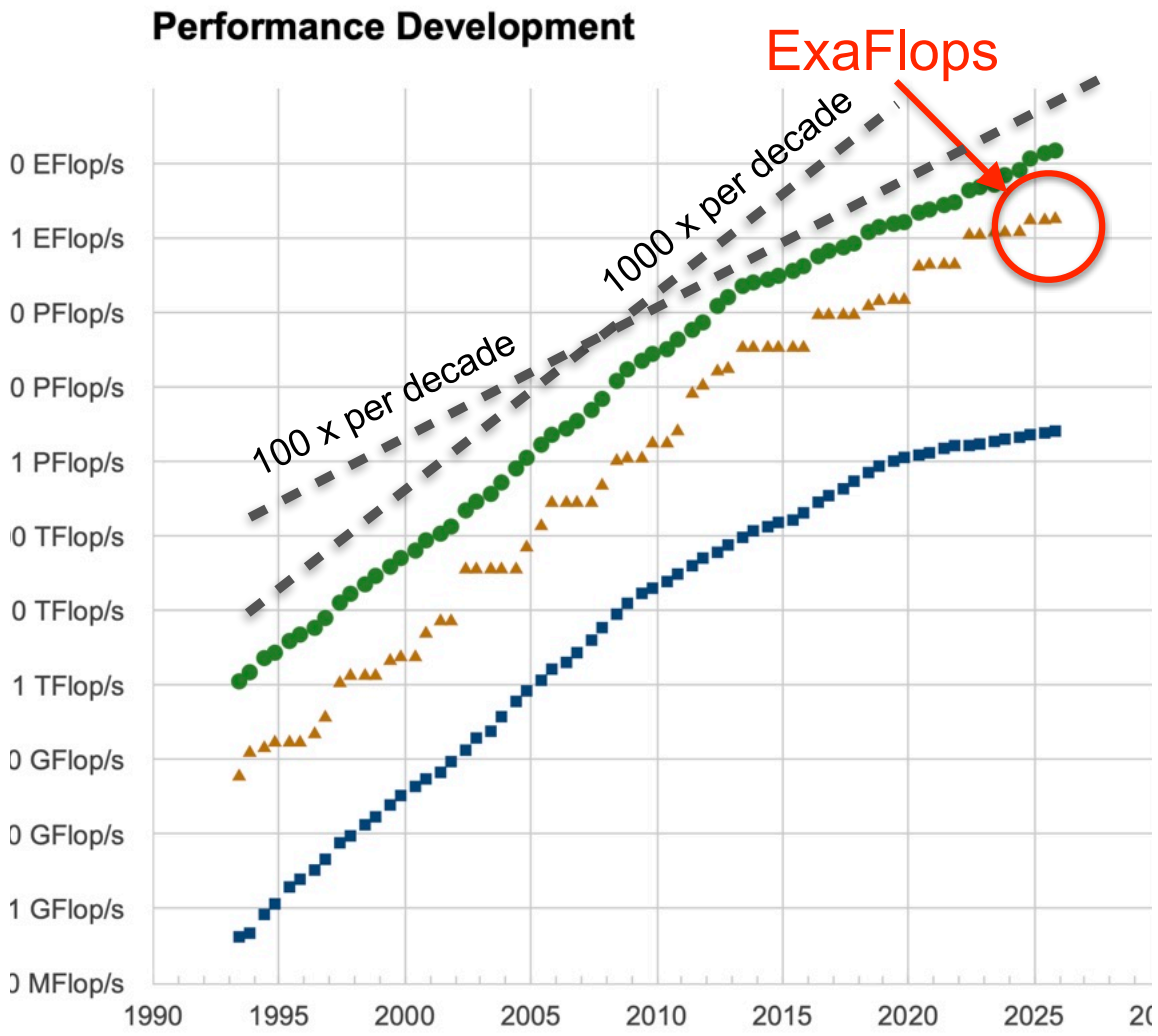
Let me take a look back …

Die PERM im Zustand von 1956 — bis Sommersemester 1974 in der Ausbildung eingesetzt, heute im Deutschen Museum

WE HAVE COME A LONG WAY!

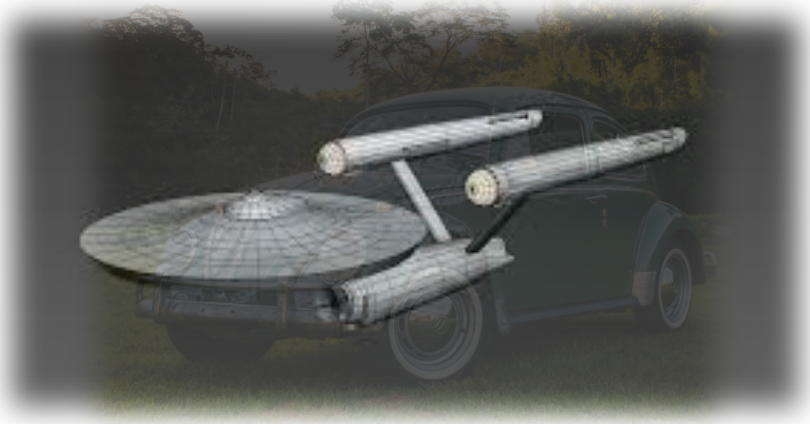
ExaFLOPS: TOP 500 List

- The fastest computers today (Frontier, ElCapitan) deliver >1 ExaFLOPS= 10^{18} FLOPS
- Will the deflation of computational cost continue?



Moore's Law: We are children of the golden age of computing!

- We are used to 1000x improvement per decade
  It slows down, but for now seems to continue with 100x improvement per decade
- We have seen improvement of speed by factors of
  - 10^7 since 1993 and
  - 10^{14} since 1963



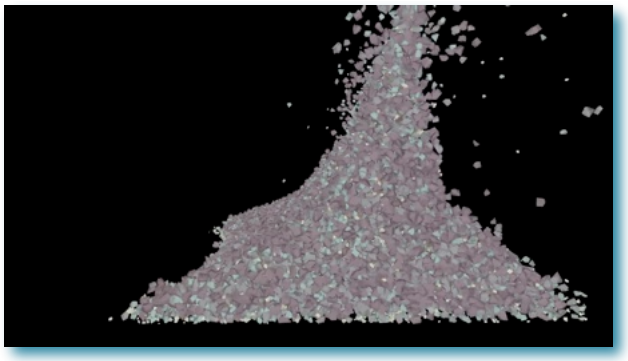
must ignore Einstein's speed limit at 10^9 km/h

Moore's Law for the Hitchhikers of the Galaxy

- If my car had seen similar improvements since 1993, it would drive instead of 10^2 km/h with 10^9 km/h
  - since the solar system has a diameter of 10^{10} km, we could reach Neptune for a summer holiday within approximately 5 hours.
- If my car had sped up by 10^{14} since 1963 it would drive at 10^{16} km/h
  - since our home galaxy has a diameter of 10^{18} km we could tour the galaxy by driving some 100 hours

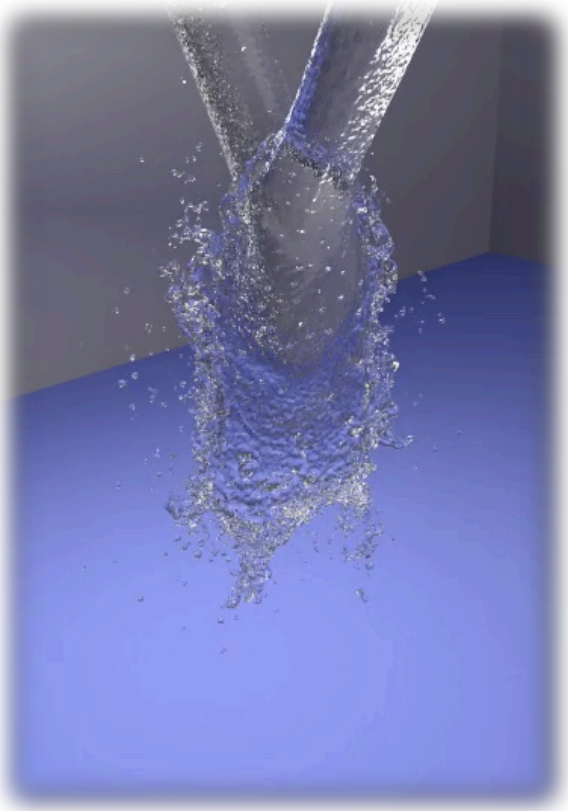
What is ExaScale possibly good for?

- 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
- Also think about ExaByte:
  - The Bible can be stored in approx. 1 MByte= 10^6 Byte
  - If every one of 10^{10} humans on earth
    wrote one bible/year
  - A single ExaByte system could store everything
    of relevance for humans in the next 100 years.



Simulation performed with our 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
Eibl, S., & UR (2019). A systematic comparison of runtime **load balancing** algorithms for massively parallel rigid particle dynamics. *Computer Physics Communications*, *244*, 76-85.



waLberla

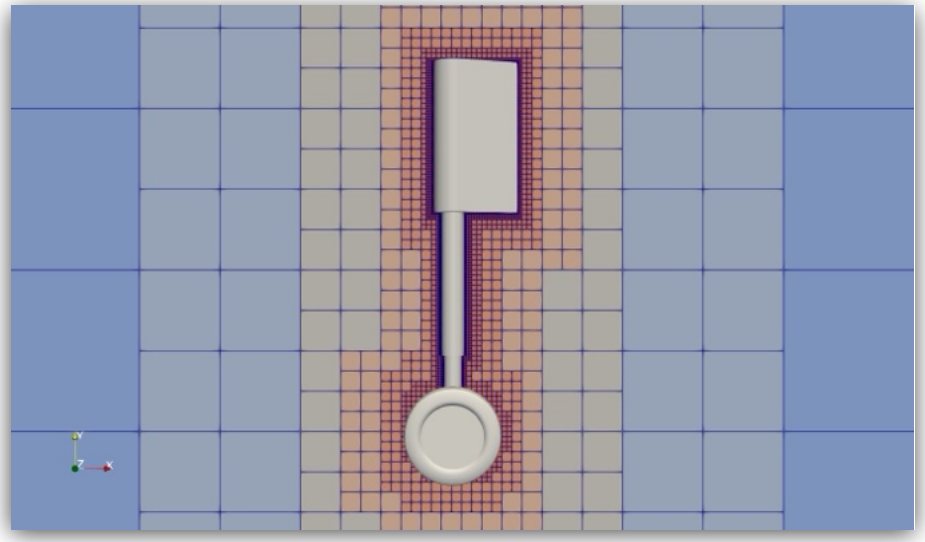
widely applicable Lattice Boltzmann from Erlangen

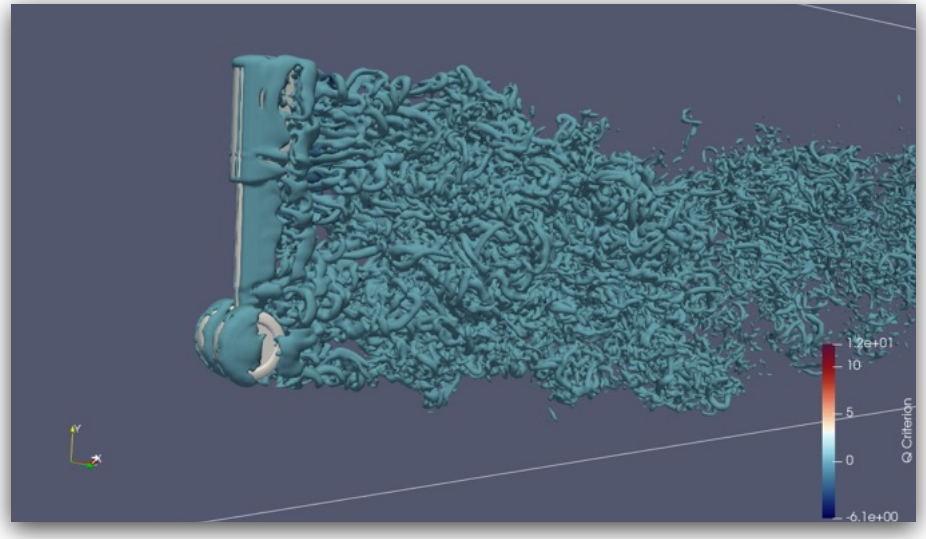
Feichtinger, C., Donath, S., Köstler, H., Götz, J., & Rüde, U. (2011). WaLBerla: HPC software design for computational engineering simulations. *Journal of Computational Science*, *2*(2), 105-112.

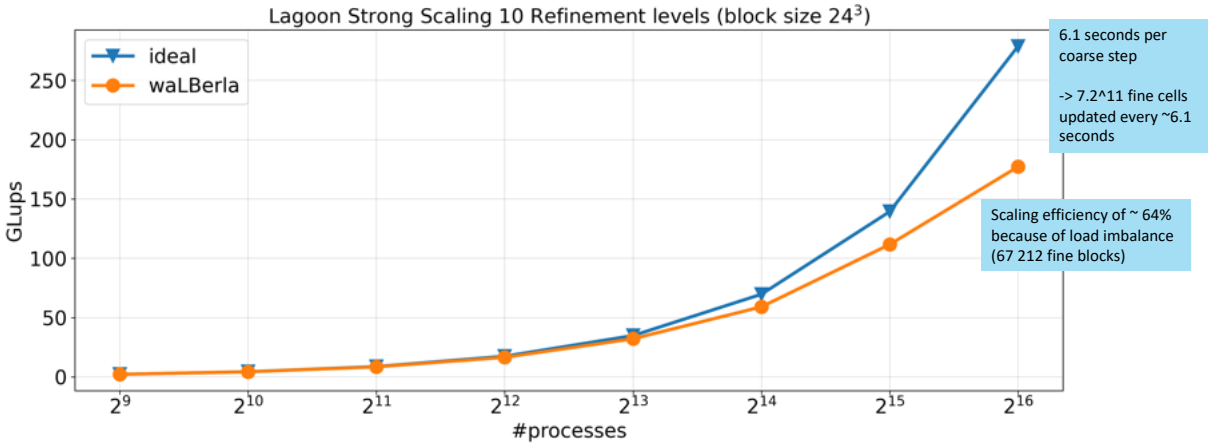
Bauer, M., Eibl, S., Godenschwager, C., Kohl, N., Kuron, M., Rettinger, C., ... & Rüde, U. (2021). waLBerla: A block-structured high-performance framework for multiphysics simulations. *Computers & Mathematics with Applications*, *81*, 478-501.

SCALABLE

- EuroHPC consortium including Airbus, FAU, IT4I Ostrava, CS Group, and CERFACS
- LBM for aerodynamics/aeroacoustics
- Up to 10 refinement levels
- Scaling up to 65000 cores
- 1.3 x 10^9 cells
- Resolution of 0.25 mm around landing gear

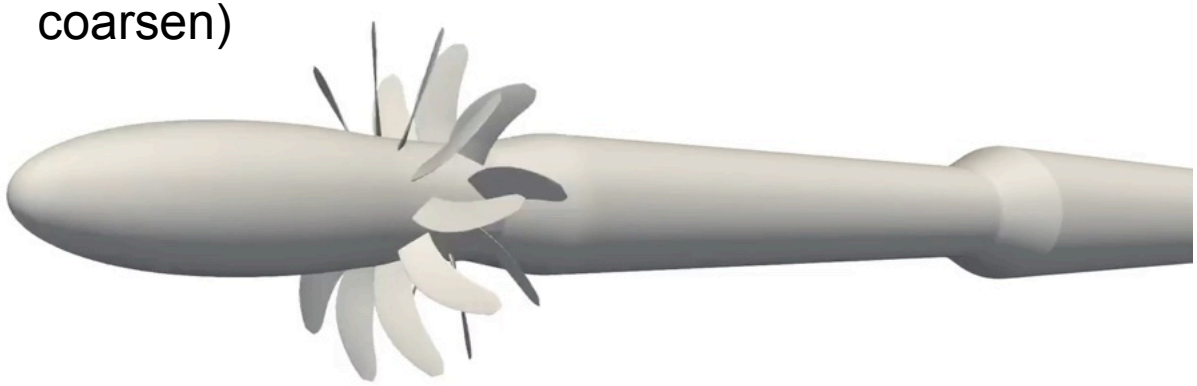






SCALABLE

- Test case of counter rotating rotor
- Partially saturated cells (PSM) method for moving boundaries/geometry
- realistic Re-number still problematic
- Adaptive refinement in parallel under development (refine/ coarsen)



Visualization/ performance optimization by P. Suffa, presented at DSFD conference. Best paper award.

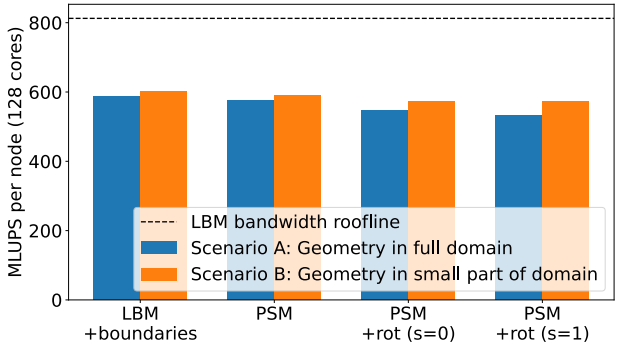


FIG. 10: Node level performance of PSM vs LBM on 2x AMD EPYC 7763 CPUs on LUMI-C with 128 blocks and 64^3 cells per block. Scenario A and B are visualized in Figure 9.

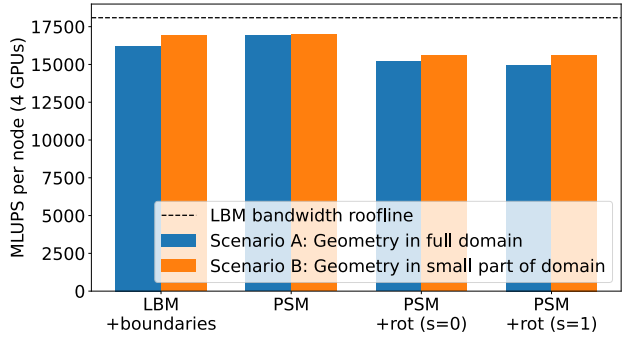


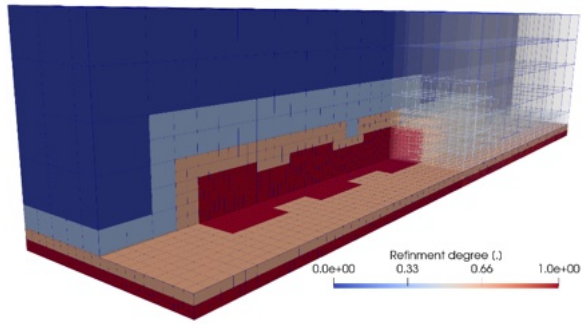
FIG. 11: Node level performance of PSM vs LBM on 4 NVIDIA A100 GPUs on JUWELS-Booster with 1 block per GPU and 256^3 cells per block. Scenario A and B are visualized in Figure 9.

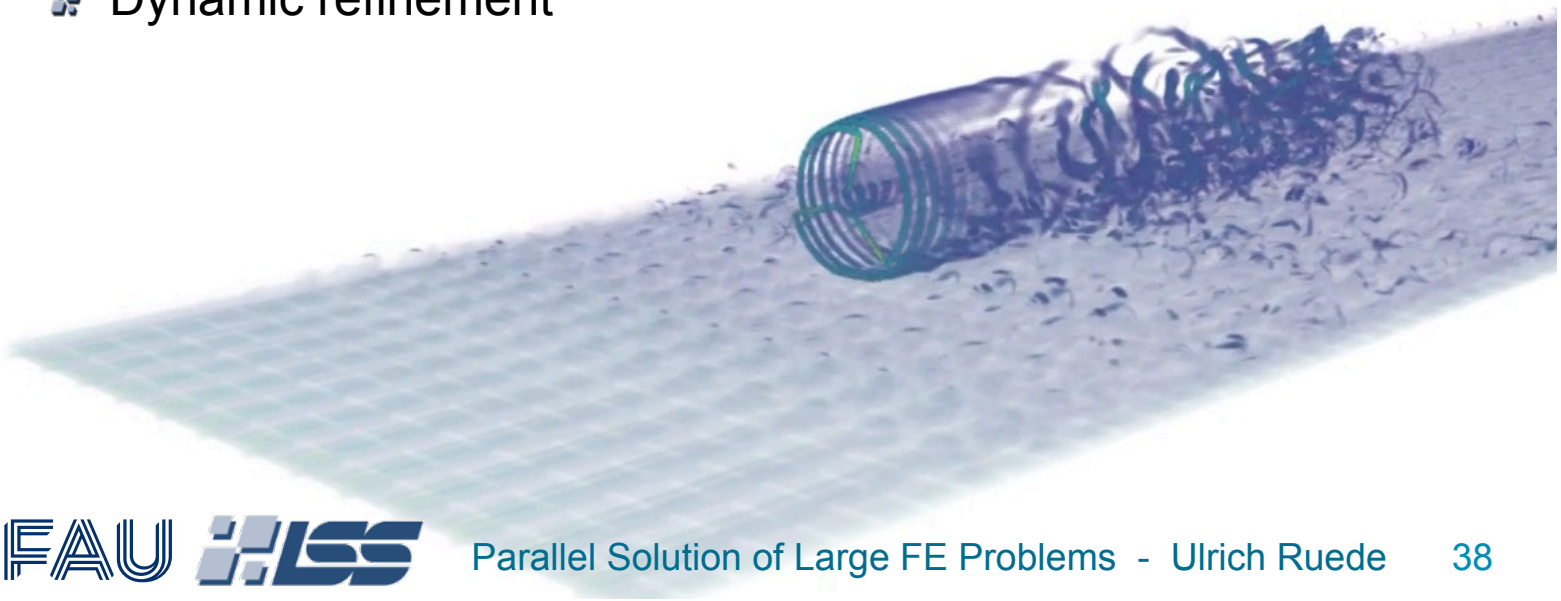
Holzer, M., Staffelbach, G., Rocchi, I., Badwaik, J., Herten, A., Vavrik, R., Vysocky, O., Riha, L., Cuidard, R., Ruede, U. (2023). Scalable flow simulations with the Lattice Boltzmann method. In *Proceedings of the 20th ACM International Conference on Computing Frontiers* (pp. 297-303).

Wind turbine simulations - mesh refinement

Schottenhamml, H., Anciaux Sedrakian, A., Blondel, F., Köstler, H., Rüde, U. (2024). waLBerla-wind: A lattice-Boltzmann-based high-performance flow solver for wind energy applications. *Concurrency and Computation: Practice and Experience*, *36*(16), e8117.

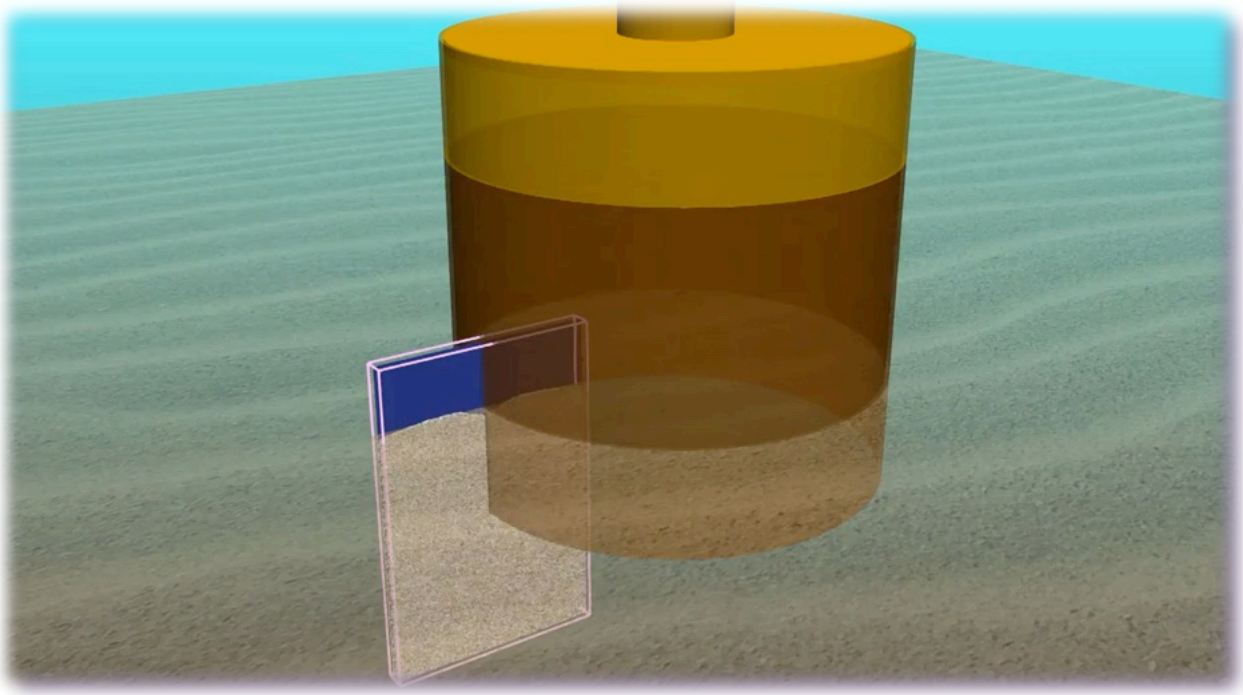
- Static refinement
  - Automatic refinement around wind turbines
  - Optionally: refinement at boundaries, user-defined boxes
  - Gradient-based vs. Vorticity-based
- Dynamic refinement
- Refinement criteria based on flow field
- AMR-GPU version available, continuing development





Simulation of a „suction bucket“

In collaboration with BAM
(Bundesanstalt für Materialforschung und -prüfung)



- Alternative method for building the foundations for off-shore wind turbines
- Suction bucket: Reverse cup with low pressure forces flow around edges
- Fluidization sucks the bucket into the sand
- Simulation domain:
  4000 x 200 x 6000 cells
- 4 million LBM time steps for ca 20 seconds physical time
- 500 000 particles
  (each approximate 20 cells diameter)
- 40h run time on 64 LUMI-G nodes = 256 GPUs

Suffa, P., Kemmler, S., Koestler, H., Ruede, U. (2025). Large-scale simulations of fully resolved complex moving geometries with partially saturated cells. *Physics of Fluids*, *37*(5).

Kemmler, S., Cuéllar, P., Artinov, A., Luu, L. H., Farhat, A., Philippe, P., Köstler, H. (2025). A fully-resolved micromechanical simulation of piping erosion during a suction bucket installation. *Computers and Geotechnics*, *186*, 107375.

Lets step back an ask:
where do we stand?

What is the fastest solver for Poisson's equation?

Are these basics trivial or not?

- When teaching linear algebra we insist that students learn:
  - Gaussian elimination costs $\sim \frac{2}{3}n^3$ FLOPS
- But for PDE? Even when we limit ourselves to:
  - Poisson's equation in the unit square with
  - 5-point discretization of the Laplace operator
  - Complexity metric: FLOPS
- With this: What is the cost of solving the discretized Poisson equation on a grid with $n = n_x \times n_y$ unknowns?
  - … what is the best algorithm known today?
- In any case: I insist on the constant, multiplying the dominating term
  - When the complexity is (almost) linear, the constant is the critical quantity

If we use full multigrid

The total computational work of MGØ1 in the FMG version (r=1) is less than

$22\mathcal{N}$ additions,	$8\mathcal{N}$ multiplications	(if V-cycles are used),	(10.5)
$32.5\mathcal{N}$ additions,	$11.5\mathcal{N}$ multiplications	(if W-cycles are used)	

From: Stüben, K., Trottenberg, U. Multigrid methods: Fundamental algorithms, model problem analysis and applications, in vol. 960 of Lecture Notes in Mathematics. Springer Verlag, 1982

- Summarizing: We should be solving the 2D Poisson equation
  - to discretization error accuracy
  - **with 30 Flops per unknown!**
  - in the model case, FMG-V(2,1) cycles are enough to achieve asymptotic optimality

So, what is the cost of solving the discrete Poisson equation?

- What is the best constant published?
  - For Poisson 2D, second order:
    #Flops ~ **30 n** (Stüben, 1982)
- assume computer with 1 ExaFLOPS, $n=10^{12}$
  - expected time to solution: Poisson 2D
    $30*10^{-6}$ sec (30 micro-seconds!)
- standard computational practice in 2026 misses this by **several orders of magnitude**!
- Why do we have a huge **gap** between theory and practice?
- We need a failure analysis!
- Has the deflation of computational cost lured us into mis-developments?

Thanks for your attention!

Happy Birthday Zdenek!

Vše nejlepší!