Massively parallel multigrid with direct coarse grid solvers

Markus Huber[1], Nils Kohl[2], Philippe Leleux[3] (leleux@cerfacs.fr), Ulrich Rüde[2,3], Dominik Thönnes[2], Barbara Wohlmuth[1]

[1]Chair for Numerical Mathematics, Technical University of München, Germany
[2]Chair for system simulation, Friedrich-Alexander University of Erlangen-Nuremberg, Germany
[3]Parallel Algorithms Team, CERFACS, Toulouse, France

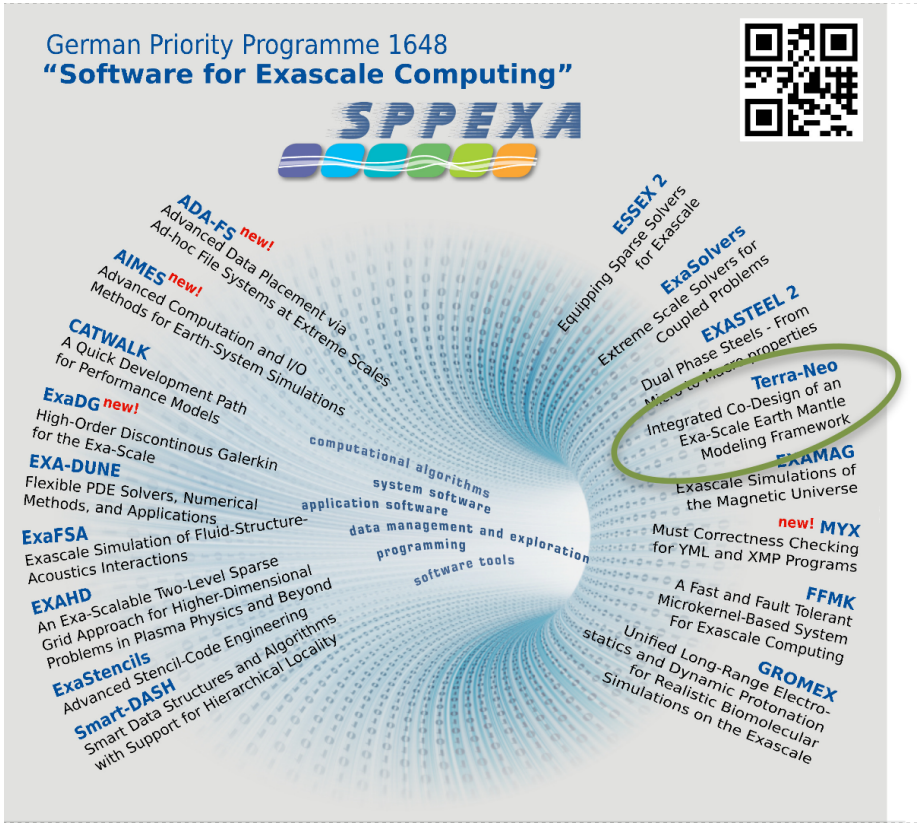
28th February, 2020

The Terra-Neo Project

Funded by the DFG programme 1648
Software for Exascale Computing - SPPEXA

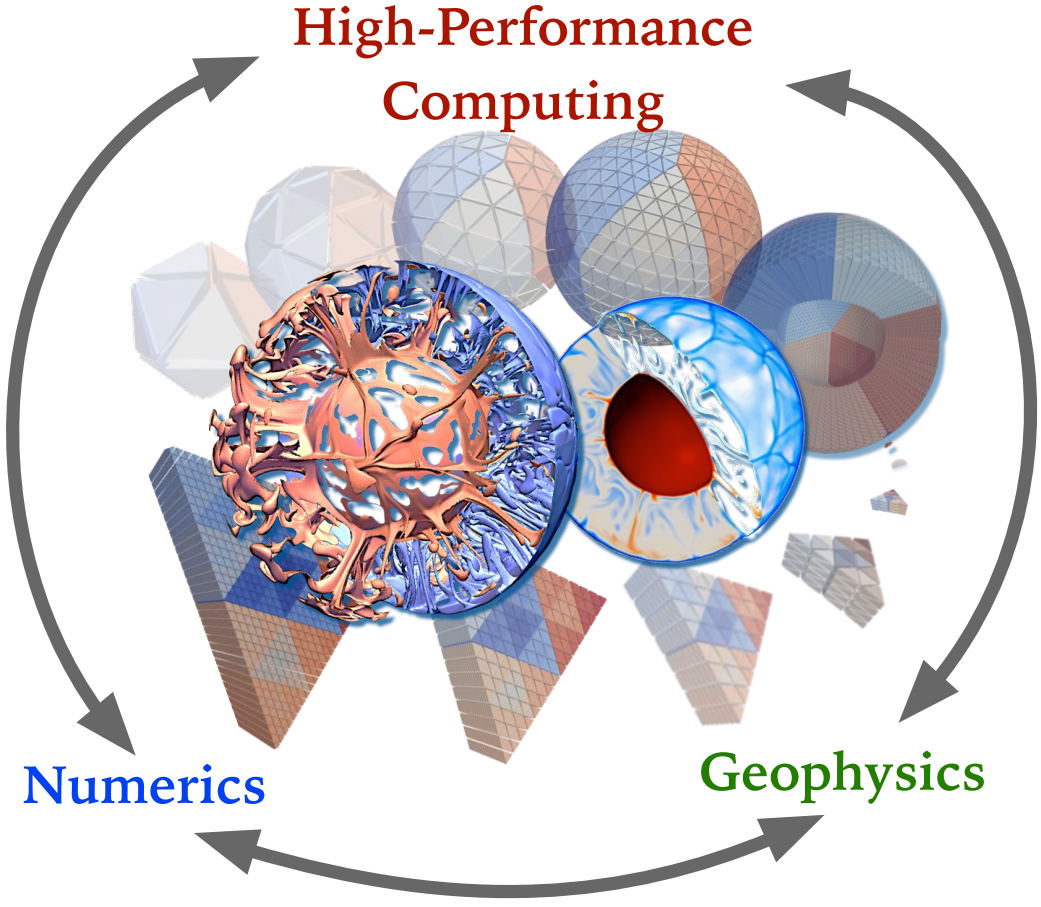
- Priority program of the German Research Fundation (DFG)
- Software for Exascale Computing
- 17 projects funded in Germany with collaborations all over the world

⟹ TerraNeo is one of them



The Terra-Neo Project

Goal: Creation of a HPC framework employing the concept of hybrid hierarchical grids for simulations of the Stokes flow problem



Motivated by the simulation of the Earth Mantle convection

Lead by
Hans-Peter Bunge (LMU Munich, Geophysics)
Barbara Wohlmuth (TUM Garching, Mathematics)
Ulrich Rüde (FAU Erlangen-Nürnberg, CS)

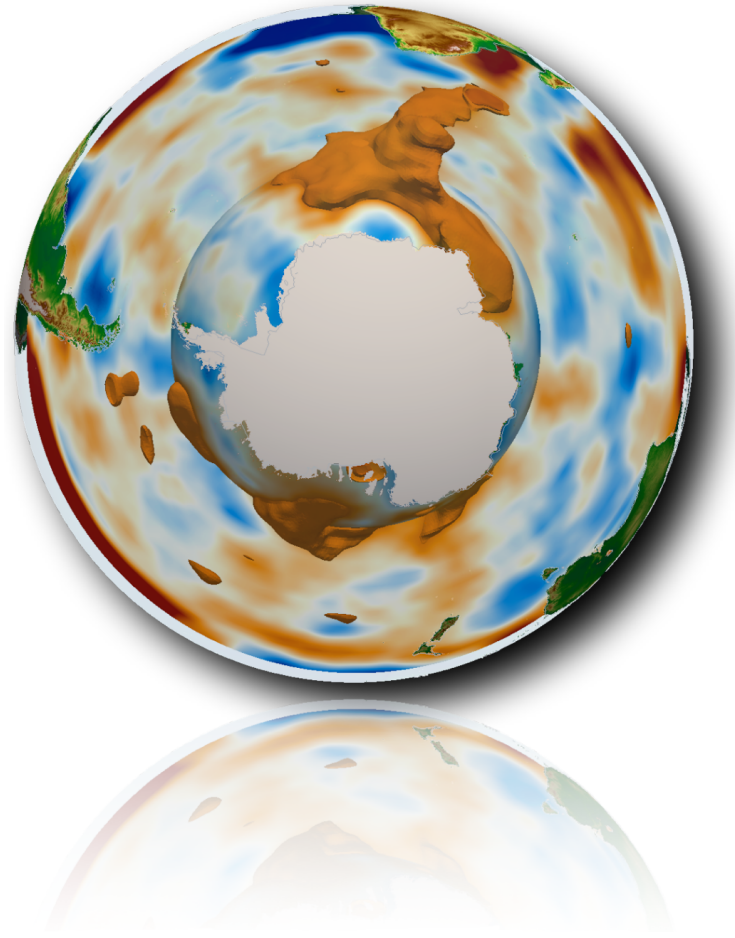
Funding period 2013-2019

terraneo.fau.de

Mantle Convection

Gaining insight from simulation of MC:

- driving force for plate tectonics
- cause of earthquakes and formation of mountains



Modeled by **Stokes eq.** coupled with energy transport:

$$\begin{aligned} -\nabla \cdot (2\eta\varepsilon(\mathbf{u})) + \nabla p &= \rho(T)g, \\ \nabla \cdot \mathbf{u} &= 0, \\ \frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T - \nabla \cdot (\kappa \nabla T) &= \lambda. \end{aligned} \tag{1}$$

Problem dimensions:

- linear systems with 10^{12} unknowns (1km global mantle resolution)
- time scale 10^8 **years** (interval $\sim$ 10 000 years)

Problematic

How can we solve such systems (efficiently)?

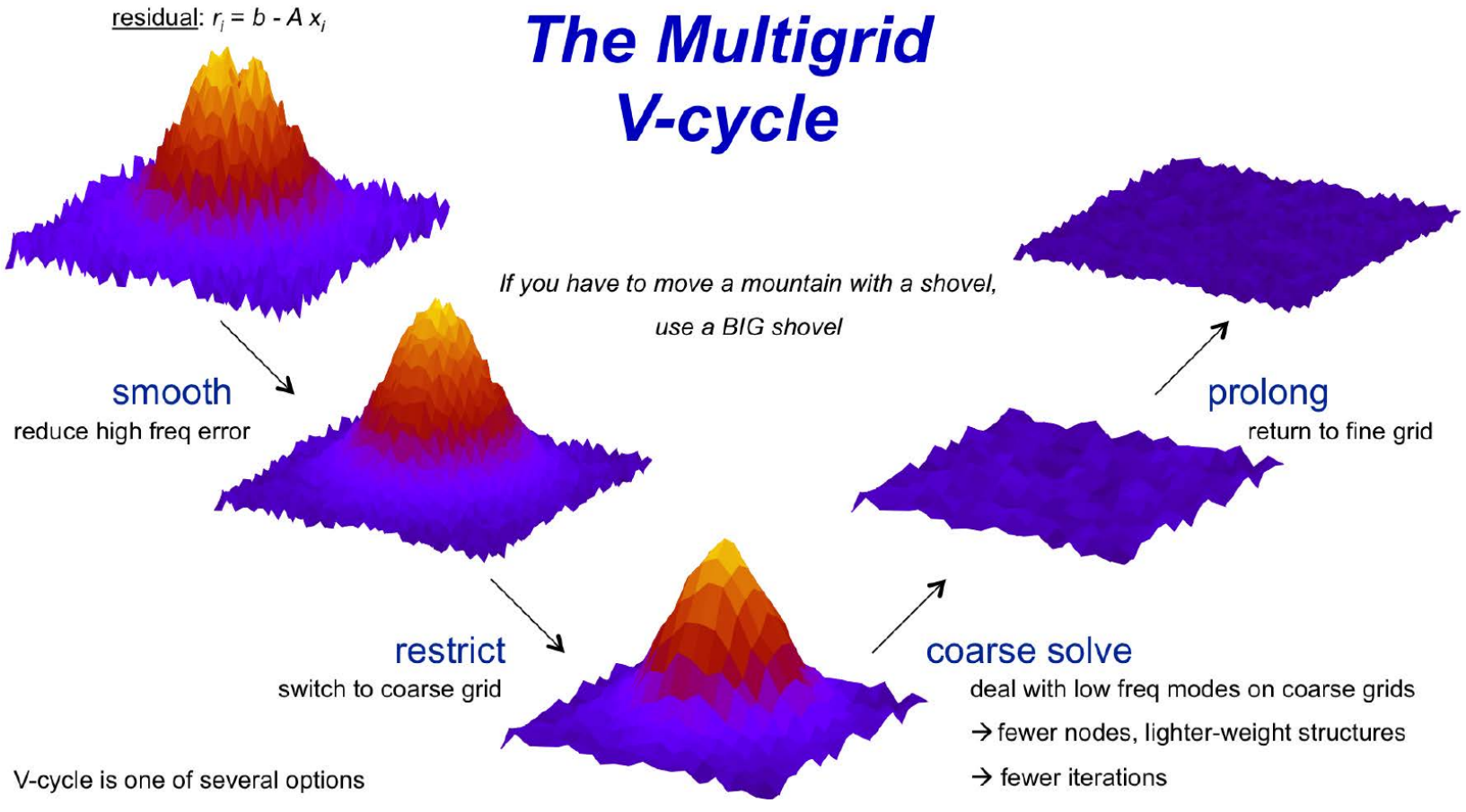
JUWELS, 2019 (31st on TOP500)

- 8 168 compute nodes 2x24 cores Dual Intel Xeon Platinum 8168,
- **264 TB main memory**: $\sim$ 3 double precision vectors of size $N = 10^{13}$.

Solution:

- System **matrix** requires **10-100x more** memory
  $\Longrightarrow$ **matrix-free implementations** are required
  (even smaller problems cannot afford explicit assembly for higher order discretizations)
- **Linear complexity** solvers are essential
  $\Longrightarrow$ **geometric multigrid**

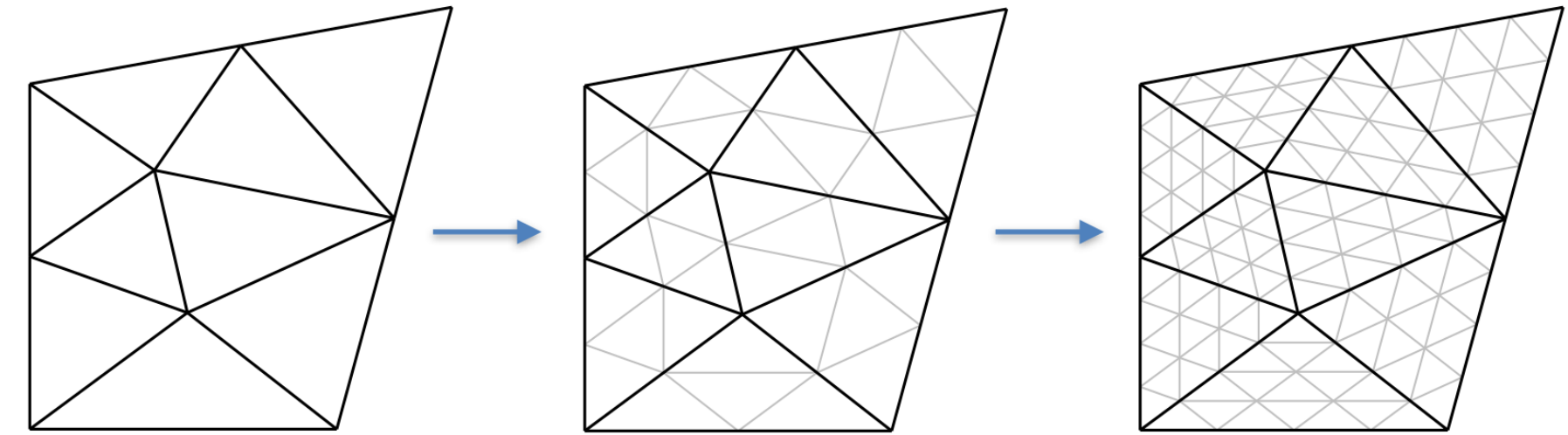
Geometric Multigrid



Hierarchical Hybrid Grids (HHG)

B. Bergen. ***Hierarchical Hybrid Grids:*** *Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers. Dissertation. 2005.*

- structured refinement of unstructured tetrahedral meshes
- multigrid hierarchy by design
- matrix-free, stencil-based kernels



Scalability

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

- monolithic, matrix-free **multigrid solver** with Uzawa smoother for **Stokes**
- JUQUEEN supercomputer – **450 TB** main memory (solution vector 80 TB)

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	$1.7 \cdot 10^{11}$	9	720.24	671.63	6.75
20 480	327 680	$1.1 \cdot 10^{13}$	9	776.09	681.91	12.14

$\Longrightarrow$ **Find scalable solutions on the coarse grid**

Outline

A Stokes problem

- Simplified model and discretization
- Solver

Coarse grid solver

- Iterative vs Direct solver
- simple Krylov solver
- Weak Scaling in HHG

Direct solver and agglomeration

- MUMPS
- Agglomeration
- Coarse grid experiments

Technische Universität München

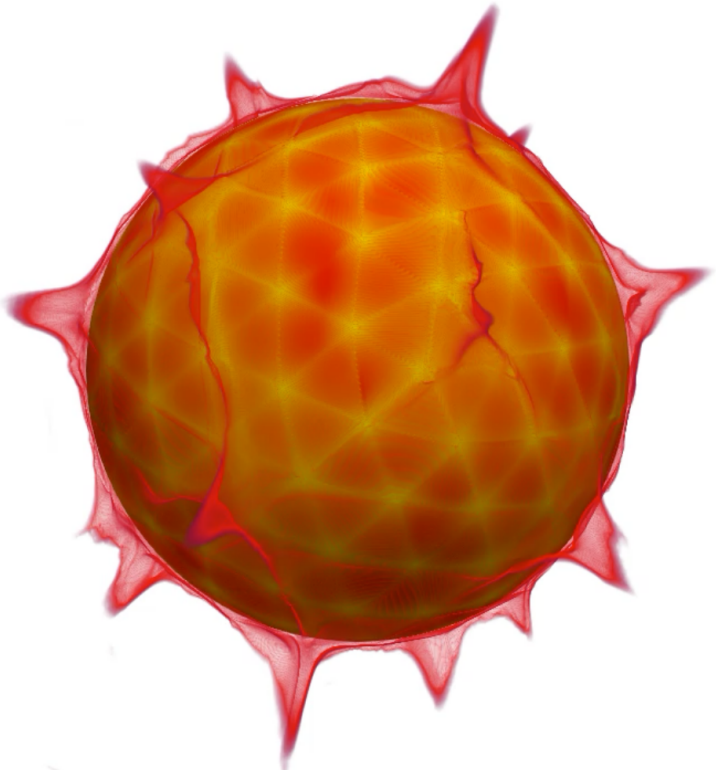
CERFACS

LSS

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

A Stokes problem

Stokes: the problem



- Stokes problem on a spherical shell

$$\begin{aligned} -\operatorname{div}\left(\frac{\nu}{2}(\nabla\mathbf{u}+(\nabla\mathbf{u})^{\top})\right)+\nabla p &= \mathbf{f} && \text{in } \Omega, \\ \operatorname{div}(\mathbf{u}) &= 0 && \text{in } \Omega, \\ \mathbf{u} &= \mathbf{g} && \text{on } \partial\Omega \end{aligned} \tag{2}$$

with **u** velocity, p pressure, **f** forcing term

- Boundary conditions
  - surface: Dirichlet B.C. from plate velocity data,
  - core-mantle: free-slip (simplified).

$\Longrightarrow$ Solution up to res. 1e-5 (model/measurement err.)

Stokes: the discretization

- lowest equal-order FE method + PSPG stabilization
- (component-wise) nodal basis functions

$$\begin{pmatrix} A_\ell & G_\ell \\ D_\ell & -C_\ell \end{pmatrix} \begin{pmatrix} \mathbf{u}_\ell \\ \mathbf{p}_\ell \end{pmatrix} = \begin{pmatrix} \mathbf{f}_\ell \\ \mathbf{g}_\ell \end{pmatrix} \tag{3}$$

- tetrahedral mesh hierarchy
  - HHG uniform refinement of input grid,
  - 2 levels for the coarse grid, 6 levels for the MG,

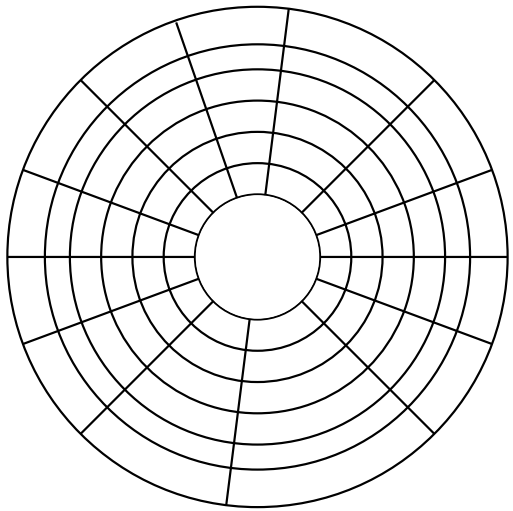
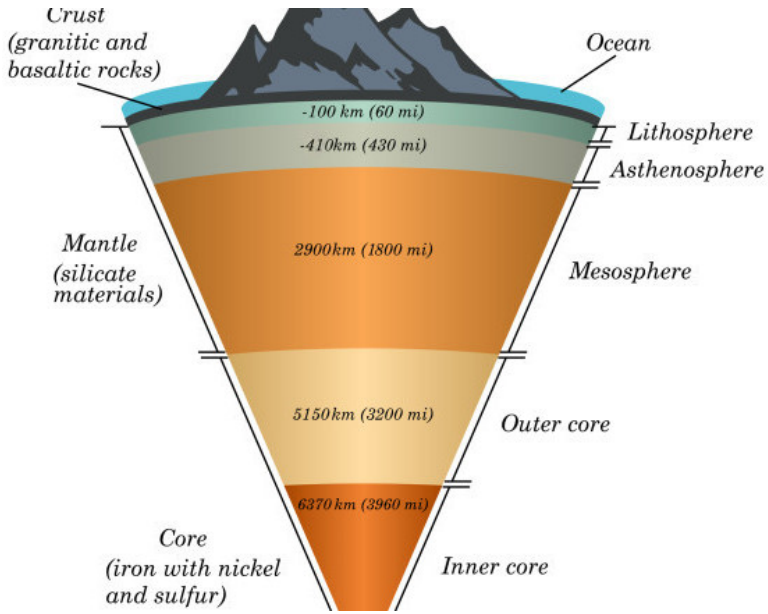


Figure: Unstructured input grid (rad: 6 div., tan: 13 div.)

rad	tan	Tets.	grid res. (km)	#Nodes	DOF	DOF coarse
3	5	1920	6.89	40	$5.37 \cdot 10^9$	$9.22 \cdot 10^4$
5	9	15360	3.44	320	$4.29 \cdot 10^{10}$	$6.96 \cdot 10^5$
6	13	43200	2.30	900	$1.21 \cdot 10^{11}$	$1.94 \cdot 10^6$

Stokes: into serious business



- MG scheme (res. $\sim$1e-5):
  - linear interpolation,
  - All-at-once Uzawa MG method:
    - Monolithic velocity-pressure solver,
    - Uzawa smoother in $V_{var} - cycle$ (+2 steps / level),
    - faster + lower mem Stokes flow
  - Coarse grid solver (res. $\sim$1e-5),
- Viscosity scenarios:
  - Velocity scenarios
    1. *iso-viscous*: $\nu(\mathbf{x}, T) \equiv 1$,
    2. *jump-410*: asthenosphere

$$\nu(\mathbf{x}, T) = \exp\left(2.99 \frac{1 - \|\mathbf{x}\|_2}{1 - r_{\text{cmb}}} - 4.61\, T\right) \begin{cases} \frac{1}{10} \cdot 6.371^3 d_a^3 & \text{for } \|\mathbf{x}\|_2 > 1 - d_a \\ 1 & \text{otherwise,} \end{cases} \tag{4}$$

- Big impact on the convergence of the MG scheme AND **coarse grid solver**

Technische Universität München

CERFACS

LSS

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Coarse grid solver

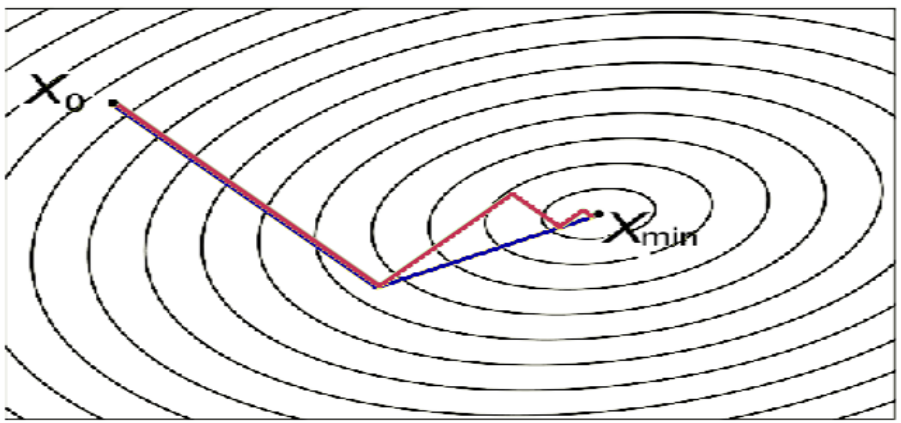
Coarse grid solver: choice

Iterative solvers:

- Only requires a MV product
  - Low cost in terms of memory,
  - An iteration is cheap in flops.
- Efficiency problem dependent
  - Cvgce – Preconditioning,
  - Special problems/structures,
  - starts from scratch for every RHS.

Direct solvers:

- Based on Gaussian elimination
  - robust method,
  - efficient parallelization,
  - $A = LU$ facto. kept for multiple RHS.
- Costly factorization
  - computation (num. pivoting, ...),
  - memory (fill-in, ...).



Coarse grid solver: choice

Iterative solvers:

- Only requires a MV product
  - Low cost in terms of memory,
  - An iteration is cheap in flops.
- Efficiency problem dependent
  - Cvgce – Preconditioning,
  - Special problems/structures,
  - starts from scratch for every RHS.

Direct solvers:

- Based on Gaussian elimination
  - robust method,
  - efficient parallelization,
  - $A = LU$ facto. kept for multiple RHS.
- Costly factorization
  - computation (num. pivoting, ...),
  - memory (fill-in, ...).

$$\begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{pmatrix}$$

Coarse grid solver: choice

Iterative solvers:

- Only requires a MV product
  - Low cost in terms of memory,
  - An iteration is cheap in flops.
- Efficiency problem dependent
  - Cvgce – Preconditioning,
  - Special problems/structures,
  - starts from scratch for every RHS.

Direct solvers:

- Based on Gaussian elimination
  - robust method,
  - efficient parallelization,
  - $A = LU$ facto. kept for multiple RHS.
- Costly factorization
  - computation (num. pivoting, ...),
  - memory (fill-in, ...).

$\Longrightarrow$ Choice will depend on the problem

Coarse grid solver: choice

Iterative solvers:

- Only requires a MV product
  - Low cost in terms of memory,
  - An iteration is cheap in flops.
- Efficiency problem dependent
  - **Cvgce** – Preconditioning,
  - Special problems/structures,
  - starts from scratch for every RHS.

Direct solvers:

- Based on Gaussian elimination
  - **robust** method,
  - efficient parallelization,
  - $A = LU$ facto. kept for multiple RHS.
- Costly factorization
  - computation (num. pivoting, ...),
  - memory (fill-in, ...).

$\Longrightarrow$ Choice will depend on the problem

$+$ Double precision accuracy not required

Coarse grid solver: Krylov iterative method

Standard method in HHG: PMINRES

1. Velocity: Jacobi-preconditioned conjugate gradient (res. 1e-2),
2. Pressure: lumped mass-matrix preconditioner,

Convergence: precond. residual of 1e-3

+++

- Easy implementation and parallelization,
- No matrix assembly in HHG,

- - -

- Convergence slows depending on problems size/complexity.

Coarse grid solver: Krylov iterative method

Table: Average time (in seconds) over the iterations of the mildly variable V–cycle with PMINRES on the coarse grid: total, fine and coarse grid timings for iso-viscous and 410 asthenosphere scenario.

scenarios			iso-viscous				jump-410				
proc.	DOFs		iter	avg. time(s)			par. eff.	iter	avg. time(s)			par. eff.
	fine	coarse		total	fine	coarse			total	fine	coarse	
1 920	$2 \cdot 10^{10}$	$9 \cdot 10^{4}$	8	58.6	57.6	1.0	1.00	15	61.0	57.9	3.1	1.00
15 360	$4 \cdot 10^{10}$	$7 \cdot 10^{5}$	4	66.1	63.2	2.9	0.89	13	83.0	62.0	21.0	0.73
43 200	$2 \cdot 10^{11}$	$2 \cdot 10^{6}$	4	68.7	65.3	3.4	0.85	14	92.0	63.7	28.3	0.66

Technische Universität München

CERFACS

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Direct solver and agglomeration

Alternative: use a direct solver

Why

- No convergence issues,
- Robust to viscosity scenarios,
- We have to pay the price of factorization...

GMG context

- Multiple RHS with same matrix,
- Price of facto paid only once, then spread through the V-cycle iterations,
- Can be ideal in our case !
- Would be different for a simple or dynamic problem.

MUMPS: MUltifrontal Massively Parallel Solver

Parallel sparse direct solver for **A x** = **b** based on the multifrontal scheme,

Three phases

1. Analysis: ordering, scaling, symbolic factorization,
2. Factorization: A=LU,
3. Solve: Ly=b, then Ux=y

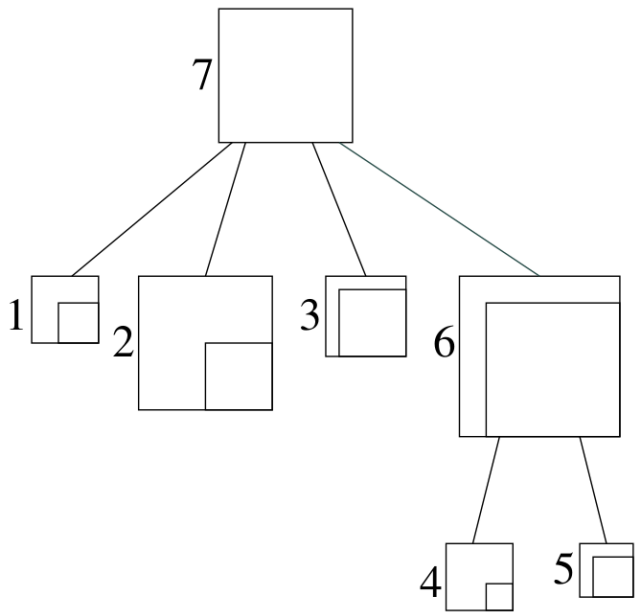
Features

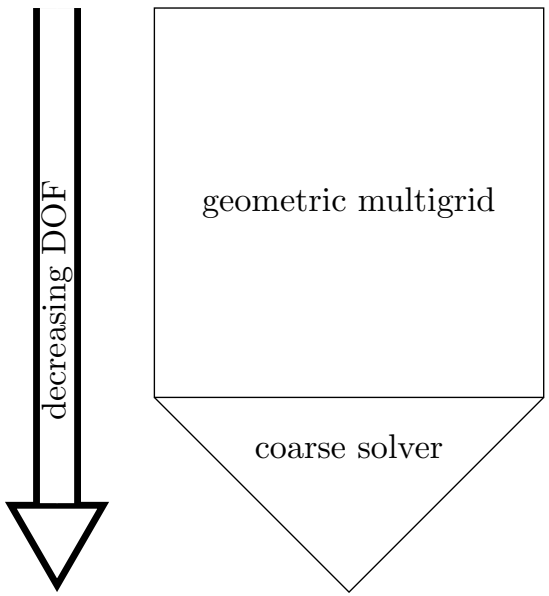
- \- Input in COO format: need a fully assembled matrix,
- ++ Analysis and Factorization (most of the cost) only required once in MG,
- ++ Robust.

MUMPS solver: `http://mumps-solver.org`

Issue known: more cores $\neq$ faster solve

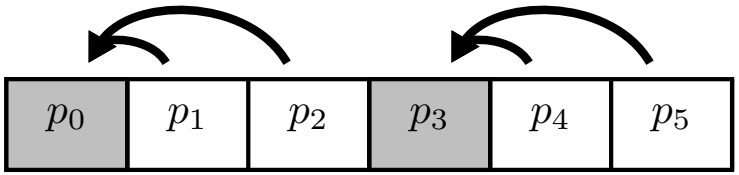
Crumbling of the granularity in sub-systems:
Communication $\gg$ Computation





Sometimes less is better... $\sim 50 DOFs$ on each process !

Solution: Agglomerating processes



Agglomeration

- Gather data to a subset of the processes: $m = |P|/r$ with r agglo. factor,
- *Master-Slave scheme*: use the topology of the architecture.

Agglomeration in practice

Table: *Reverse* strong scaling study of the MUMPS sparse direct solver separated into analysis+factorization and solve.

r	1920			15 360			43 200		
	proc.	ana.-fac.	solve	proc.	ana.-fac.	solve	proc.	ana.-fac.	solve
1	1 920	30.66	31.84	15 360	–	–	43 200	–	–
24	80	2.78	0.02	640	39.97	0.23	1 800	–	–
48	40	**2.08**	**0.02**	320	30.18	0.21	900	176.6	1.40
192	10	2.77	0.03	80	**29.69**	**0.18**	225	136.35	1.02
576	–	–	–	–	–	–	75	**136.78**	**0.97**

MUMPS vs. viscosity

Table: Study of the influence of the viscosity scenario on the accuracy and run-time of the direct solver. Run-times are separated in analysis, factorization and solve step. Each process runs on a separate node.

proc.	DOF coarse	scenario	ana.-fac.(s)	solve(s)	scaled residual
40	$9.22 \cdot 10^4$	iso-viscous	2.16	0.02	$1.8 \cdot 10^{-18}$
		jump-410	2.10	0.02	$1.9 \cdot 10^{-17}$
160	$6.96 \cdot 10^5$	iso-viscous	27.64	0.19	$5.7 \cdot 10^{-19}$
		jump-410	27.79	0.18	$1.2 \cdot 10^{-18}$
225	$1.94 \cdot 10^6$	iso-viscous	154.14	0.50	$5.31 \cdot 10^{-19}$
		jump-410	162.53	0.47	$1.27 \cdot 10^{-18}$

All-in-all

Table: Weak scaling of the V_{var}–cycle with a sparse direct and a simple Krylov coarse level solver. The run-times for total, fine and coarse are averages over the iterations. The run-times for analysis, factorization and data transfer are the total timing.

proc.	DOF		iter	time (s)					par. eff.
	fine	coarse		total	fine	coarse	ana.-fac.	trans.	
1 920	$2.1 \cdot 10^{10}$	$9.22 \cdot 10^{4}$	15	60.91	60.73	0.02	2.20	0.04	1.00
15 360	$4.3 \cdot 10^{10}$	$6.96 \cdot 10^{5}$	13	69.90	67.28	0.20	31.11	0.25	0.87
43 200	$1.7 \cdot 10^{11}$	$1.94 \cdot 10^{6}$	14	80.06	69.25	1.02	136.36	0.65	0.76

All-in-all

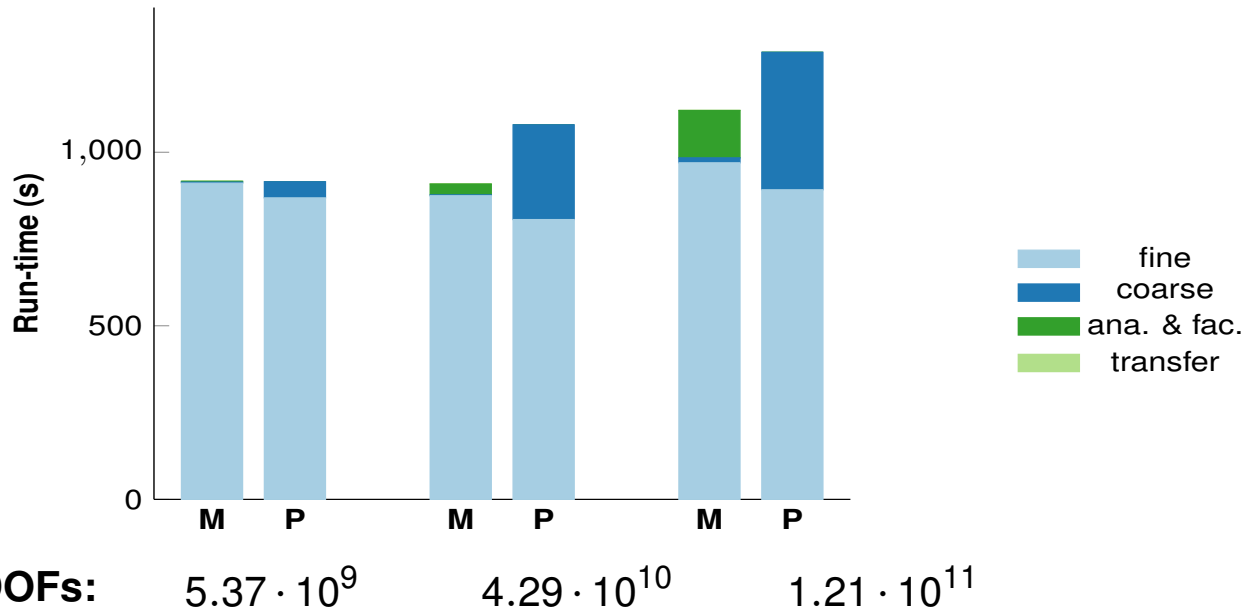


Figure: Difference between an HHG run with PMINRES (*P*) and MUMPS (*M*), using BLR ($\varepsilon = 10^{-3}$) and single precision, as solvers on the coarse grid for the three different sizes of problem. from using MUMPs.

proc.	PMINRES		MUMPS	
	total(s)	par. eff.	total(s)	par. eff.
1 920	61.0	1.00	60.91	1.00
15 360	83.0	0.73	69.90	0.87
43 200	82.0	0.66	80.06	0.76

Technische Universität München

CERFACS

LSS

FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
FACULTY OF ENGINEERING

Conclusion

Intermediary Conclusion

Summary

- Bringing together modern methods,
- matrix-free GMG with a focus on the coarse grid solution,
- parallel sparse direct solver,
- agglomeration methods.

In practice

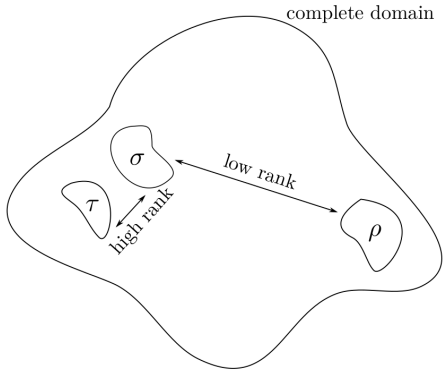
- working parallel implementation in the HHG framework,
- improvement of the parallel efficiency for the solver on the coarse grid with MUMPS+agglomeration: from 66% to 76%,
- However
  - the agglomeration is a very practical approach, requiring extensive tests,
  - in the end the improvement is not so large compared to the Krylov solver.

$\Longrightarrow$ Towards better results ?

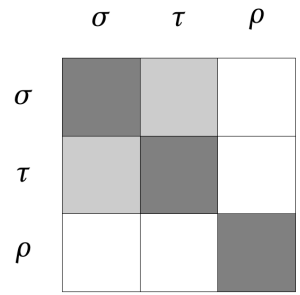
Approximate direct solver

Approximation on the coarse grid

double precision accuracy unnecessary:
$\Longrightarrow$ only a res. $1e-5$ required for PMINRES



(a) Strong and weak interactions between clusters in the geometric domain.



(b) Corresponding block-clusters in the matrix.

MUMPS approach

1. Block Low Rank approximation:
   - At facto: off-diagonal blocks of the fronts can be approximated with a controlled accuracy ε using a low-rank matrix,
   - Controlled solution accuracy,
   - Corr. reduction of memory and flops.
2. Single precision arithmetic.

Tests performed on HazelHen from HLRS, Stuttgart with the same problems and number of cores.

BLR+single precision effect

Table: Study of the influence of BLR ε parameter for the *jump-410* viscosity, with double and single precision, on the accuracy and the run-time of the direct solver. Run-times are separated in analysis, factorization and solve step. Each process runs on a separate node.

proc.	DOF coarse	BLR ε	analysis time (s)	factorization		solve	scaled res.
				Flops red.	time (s)	time (s)	
40	$9.22 \cdot 10^4$	Full Rank	1.55	100.0	0.88	0.03	$6.0 \cdot 10^{-18}$
		10^{-3}	1.81	28.5	0.91	0.02	$3.7 \cdot 10^{-4}$
		10^{-3} + single	1.74	26.0	0.67	0.01	$2.5 \cdot 10^{-4}$
160	$6.96 \cdot 10^5$	Full Rank	13.74	100.0	19.58	0.20	$4.8 \cdot 10^{-18}$
		10^{-3}	16.03	10.7	9.95	0.10	$2.1 \cdot 10^{-4}$
		10^{-3} + single	15.86	10.5	6.62	0.09	$7.5 \cdot 10^{-5}$
225	$1.94 \cdot 10^6$	Full Rank	41.02	100.0	134.61	0.56	$1.5 \cdot 10^{-18}$
		10^{-5}	47.56	13.0	36.98	0.30	$2.4 \cdot 10^{-6}$
		10^{-5} + single	47.65	13.2	25.63	0.27	$1.4 \cdot 10^{-6}$
		10^{-3}	47.55	7.5	31.11	0.24	$5.0 \cdot 10^{-5}$
		10^{-3} + single	47.62	7.6	21.16	0.19	$4.7 \cdot 10^{-5}$

All-in-All-in-All

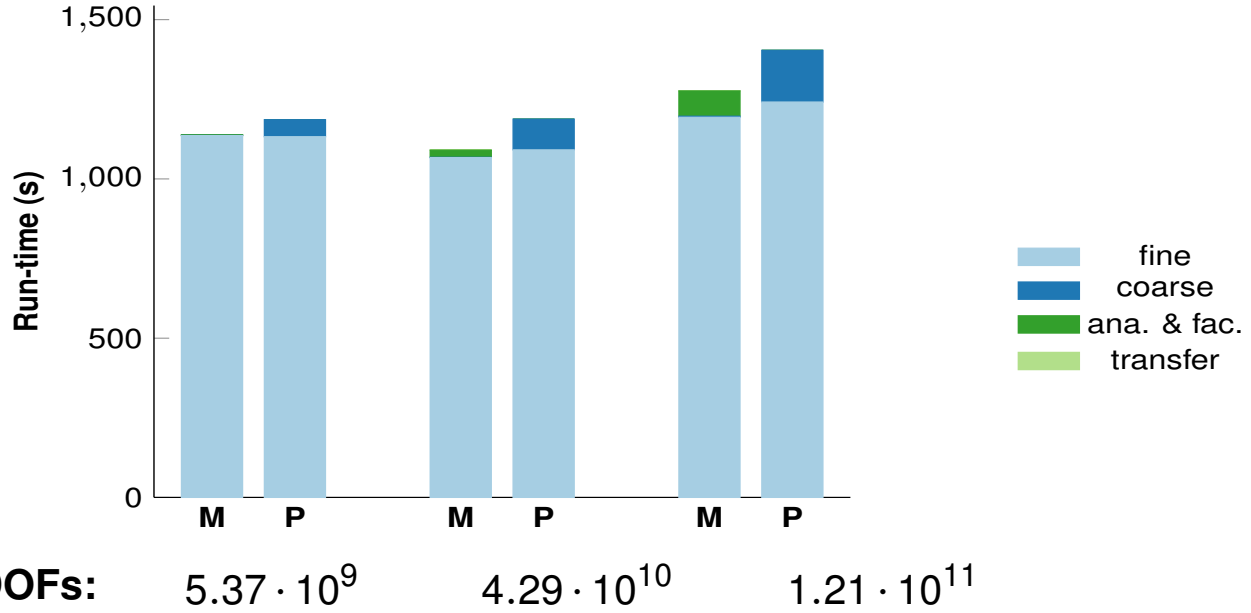


Figure: Difference between an HHG run with PMINRES (*P*) and MUMPS (*M*), using BLR ($\varepsilon = 10^{-3}$) and single precision, as solvers on the coarse grid for the three different sizes of problem. from using MUMPs.

proc.	PMINRES		MUMPS	
	total(s)	par. eff.	total(s)	par. eff.
1 920	79.07	1.00	75.93	1.00
15 360	91.38	0.87	83.92	0.93
43 200	100.29	0.79	91.58	0.85

Conclusion

Our method

- Efficient solution on the coarse grid, part. for slow converging MG/coarse grid solver,
- Approximating the solution is enough,
- First attempt at the use of BLR in large scale application,
- single precision enough for such approximation.

Future

- Larger systems,
- Fine tune MUMPS (e.g. geom. info) + OpenMP parallelism,
- comparison to other solvers (e.g. AMG),
- Use a hybrid parallel solver with controlled convergence.
- alternative agglomeration strategy (virtually remove the cost of analysis+factorization).

References

[1] P. AMESTOY, A. BUTTARI, J.-Y. L'EXCELLENT, AND T. MARY. *On the complexity of the block low-rank multifrontal factorization*. SIAM Journal on Scientific Computing, 39(4):A1710–A1740, 2017. **http://mumps-solver.org/**

[2] B. BERGEN. *Hierarchical hybrid grids: Data Structures and Core Algorithms for Efficient Finite Element Simulations on Supercomputers*. Dissertation, 2005.

[3] B. GMEINER, M. HUBER, L. JOHN, U. RÜDE, AND B. WOHLMUTH. *A quantitative performance study for Stokes solvers at the extreme scale*. J. Comput. Sci., 17:509–521, 2016.

[4] T. MARY. *Block Low-Rank multifrontal solvers: complexity, performance, and scalability*. PhD thesis, UT3, 2017.

[5] W. ZULEHNER *Analysis of iterative methods for saddle point problems: A unified approach*. Math. Comput., 71(238):479–505, Apr. 2002.

Thank you, any questions?

HHG primitives

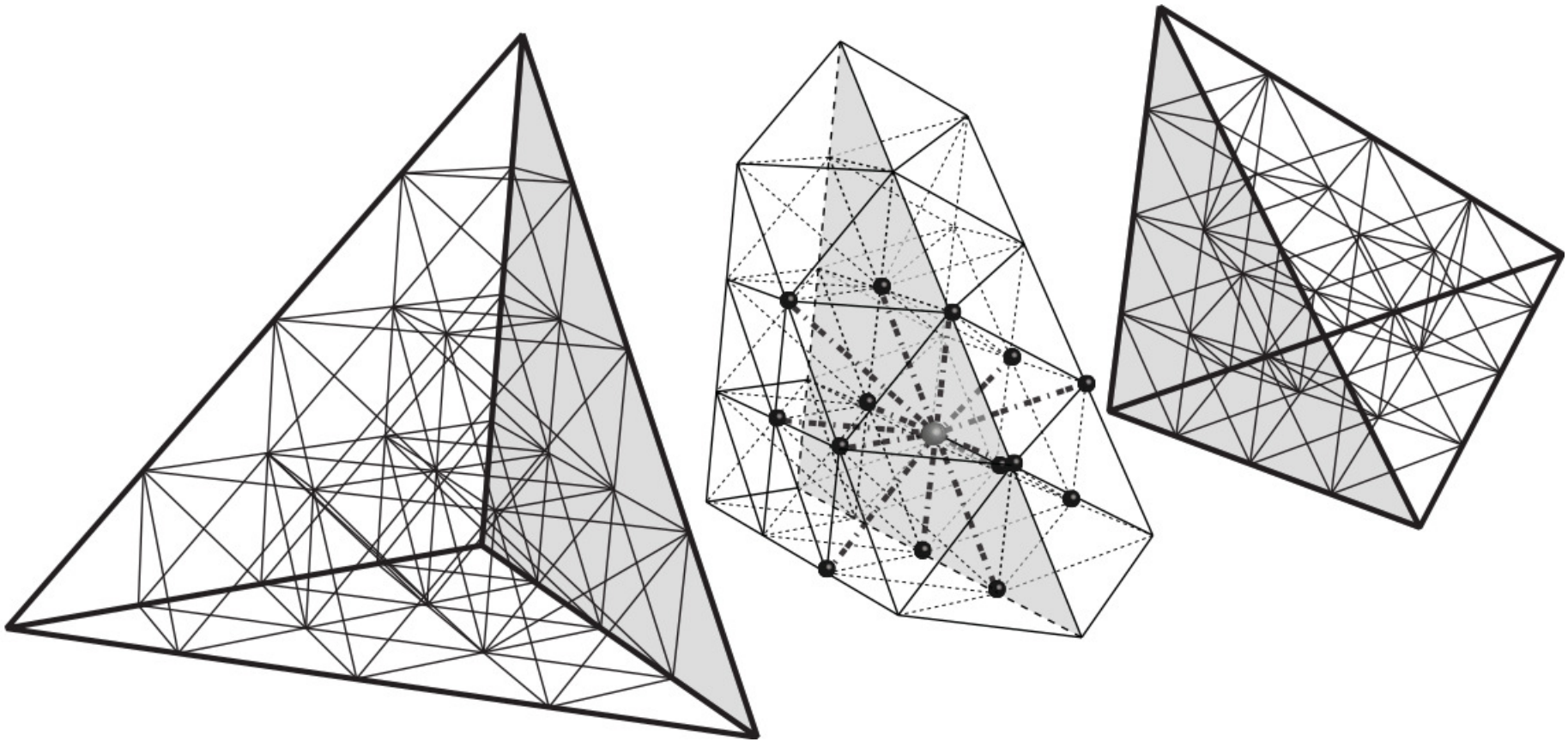
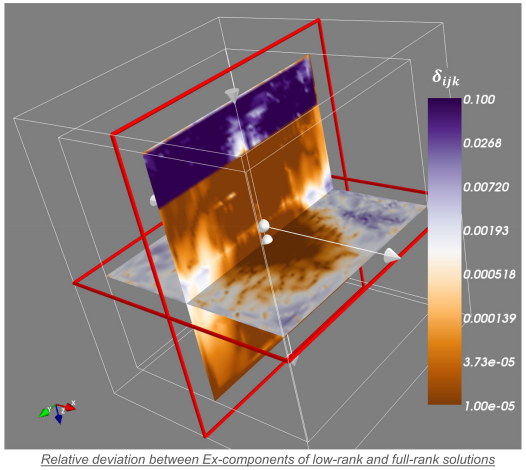


Figure: left: two refined input elements; right: ghost layer structure of two input elements.

MUMPS performance on 3D EM application on 900 cores

E_x, BLR STRATEGY 2, IR = 0, $\varepsilon_{BLR} = 10^{-7}$



$$\delta_{ijk} = \sqrt{\frac{|x_{\varepsilon,ijk} - x_{ijk}|^2}{\frac{|x_{\varepsilon,ijk}|^2 + |x_{ijk}|^2}{2} + \eta^2}}$$

(only for E_x)

Relative deviation between Ex-components of low-rank and full-rank solutions

emgs

3D Electromagnetic Modeling

(Double) Complex matrix

Factorization of matrix D4 requires:

3 TBytes of storage, 3 PetaFlops

Matrix from 3D EM problems (credits: EMGS)

matrix	n	nnz	MUMPS-(Full-Rank) time	sp-up**	$\%_{peak}$	BLR* time
D4	30M	384M	2221s	373	**33%**	566s

*$\varepsilon = 10^{-7}$; **estimated speedup on 90 $\times$ 10 cores

Superman agglomeration

$$
\begin{aligned}
t^{simu}_{ana.\&fac.} &= max\left(t_{ana.\&fac.} + \bar{t}_{coarse} + \bar{t}_{trans.} - \bar{t}_{fine}, 0\right) \\
t^{simu}_{total} &= t_{fine} + t^{simu}_{ana.\&fac.} + \left(\#it - 1\right) * \left(\bar{t}_{coarse} + \bar{t}_{trans.}\right)
\end{aligned}
\tag{5}
$$

Table: Weak scaling of the V_{var}–cycle with a sparse direct and a block low-rank coarse level solver with single precision arithmetic. These results are a simulation of what the Superman strategy would give if we allocate enough additional nodes to cover for the processes specialized for MUMPS. To compensate for the first cycle. the computation of the analysis and factorization is removed from the fine grid execution time. The parallel efficiency compares the average total run-time of each run to the average total run-time of the smallest case with no BLR.

proc.	DOFs		it	fine	coarse	trans.	Master-Slave			Superman		
	fine	coarse					ana.-fac.	total	par. eff.	ana.-fac.	total	par. eff.
1 920	$2.1 \cdot 10^{10}$	$9.2 \cdot 10^{4}$	15	1166.4	0.36	0.10	2.40	1169.3	1.00	0.00	1171.8	1.00
			15	1135.9	0.26	0.10	2.50	1138.7	1.03	0.00	1141.3	1.03
15 360	$4.3 \cdot 10^{10}$	$7.0 \cdot 10^{5}$	13	1080.9	2.75	0.26	36.30	1120.2	0.90	0.00	1156.8	0.93
			13	1066.6	1.08	0.70	22.30	1090.7	0.93	0.00	1113.7	0.95
43 200	$1.2 \cdot 10^{11}$	$1.9 \cdot 10^{6}$	14	1197.2	8.19	0.34	176.20	1382.0	0.79	91.29	1649.8	0.84
			14	1218.8	3.78	0.86	75.30	1298.7	0.84	0.00	1374.9	0.89
94 464	$2.58 \cdot 10^{11}$	$4.18 \cdot 10^{6}$	9	1302.1	4.66	0.19	185.18	1492.1	0.47	40.51	1342.6	0.52