Automated performance modeling of the UG4 simulation framework

Andreas Vogel[1], Alexandru Calotoiu[2], Arne Nägel[1],
Sebastian Reiter[1], Alexandre Strube[3], Gabriel Wittum[1],
and Felix Wolf[2]

[1]Goethe-Universität Frankfurt, Frankfurt am Main, Germany
[2]Technische Universität Darmstadt, Darmstadt, Germany
[3]Forschungszentrum Jülich, Jülich, Germany

Munich, January 2016

Introduction

- **Goal:** Find scaling issues in large parallel codes.
- **Target code: UG4** (*Unstructured Grids 4*), a simulation framework for solving partial differential equations on unstructured grids.
- **Idea:** Create performance models with fine granularity
  - **Measure** the scaling behavior of different ***code kernels***.
  - **Create scaling models** for each such kernel.
  - Do this for thousands of kernels $\rightarrow$ **automation required!**
- **Usage:**
  - Automated performance analysis to **identify current bottlenecks**.
  - **Prediction of possible bottlenecks** for even larger parallel runs.
  - Allows for the **prediction of resource consumption** for larger core counts.

Outline

- **UG4** — Overview and parallelization concepts
- **Application** — Drug diffusion through the human skin
- **Performance Modeling** — Ideas
- **Results**

UG4 overview

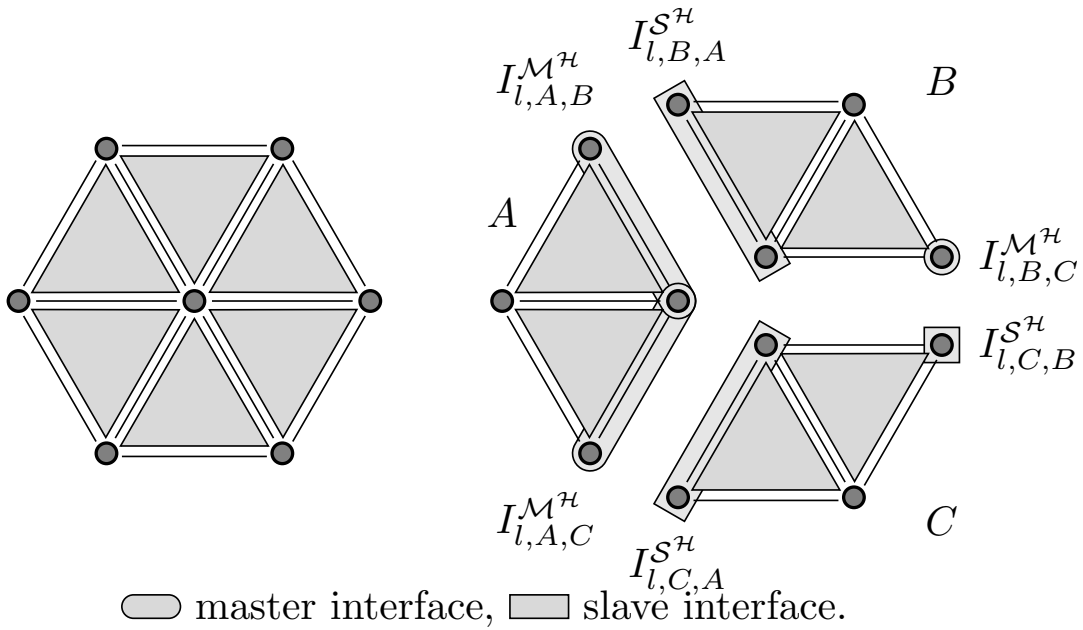
- **UG4** [1] is a simulation framework for the solution of ***PDE's***.
  - **Fast** Cross platform *C++* code.
  - **Modular** plugin based structure.
  - Full **scripting** support through *Lua* and *Java*.
  - **Flexible** distributed **unstructured grids**.
  - *FE-/FV*-discretizations.
  - **Various applications** such as *groundwater-flow, elasticity, neuroscience, biology*, and many more.
  - $\gg 100k$ lines of code.
- Developed with **massively parallel** applications in mind:
  - Multigrid **solvers** for **optimal complexity** (important!).
  - **Efficient distribution** of grid hierarchies.
  - **Very good scalability** shown for up to **262144 cores**[2].

1) Vogel, A., Reiter, S., Rupp, M., Nägel, A., Wittum, G.: UG 4: A novel flexible software system for simulating PDE based models on high performance computers. Comp. Vis. Sci. 16(4), 165–179 (2013)

2) Reiter, S., Vogel, A., Heppner, I., Rupp, M., Wittum, G.: A massively parallel geometric multigrid solver on hierarchically distributed grids. Com. Vis. Sci 16(4), 151-164 (2013)

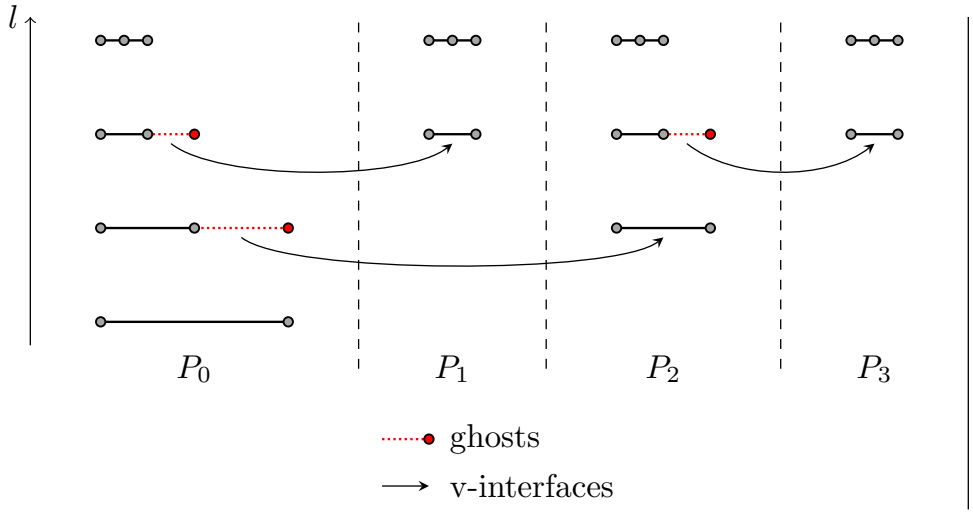
UG4 parallelization concepts

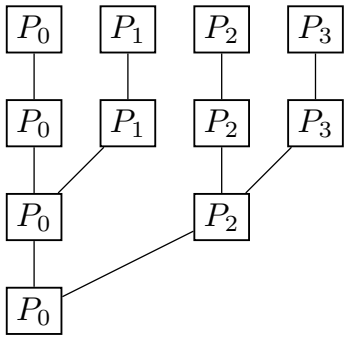
- **Interfaces** allow for data exchange between distributed grid objects.



UG4 parallelization concepts

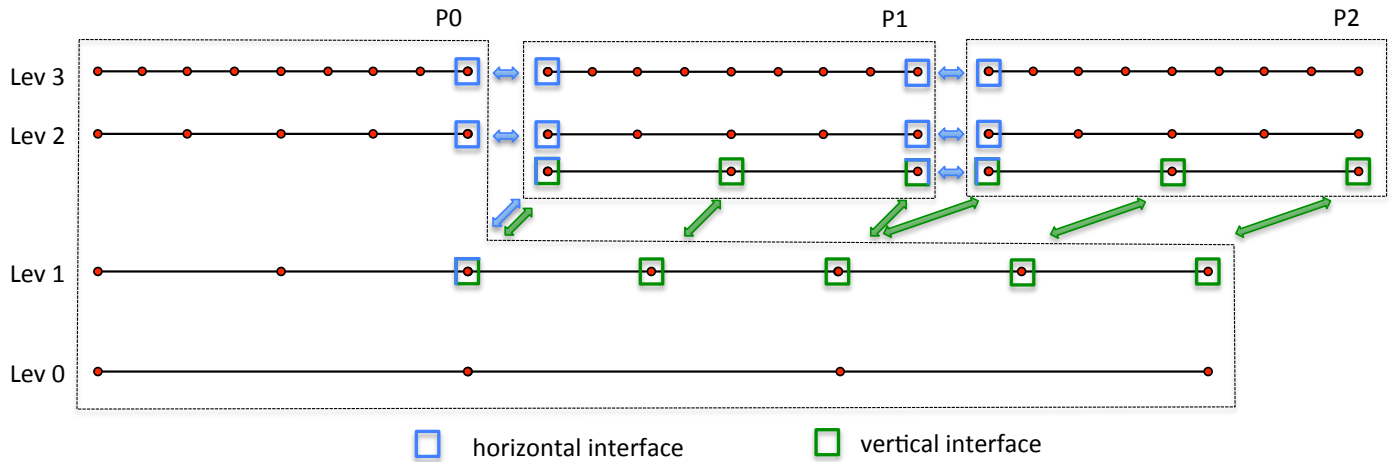
- **Interfaces** allow for data exchange between distributed grid objects.
- **Hierarchical distribution** guarantees good *comp/comm* ratio.





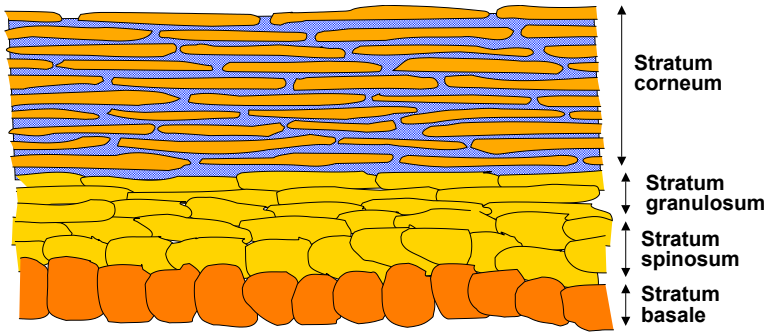
UG4 parallelization concepts

- **Interfaces** allow for data exchange between distributed grid objects.
- **Hierarchical distribution** guarantees good *comp/comm* ratio.
- ***Horizontal/vertical*** interfaces are used for multigrid communication.

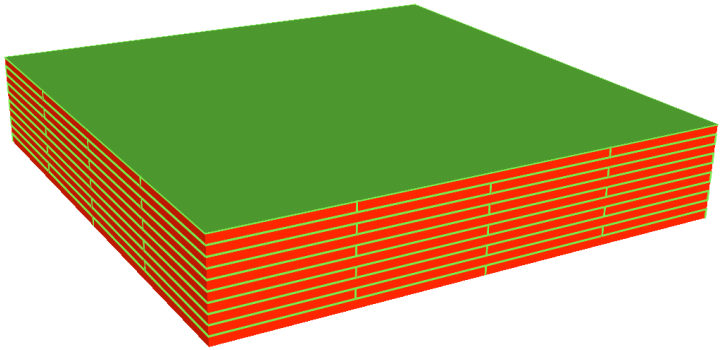


Application: Drug diffusion through human skin

Model of the human skin[1]:



Grid for Stratum corneum:



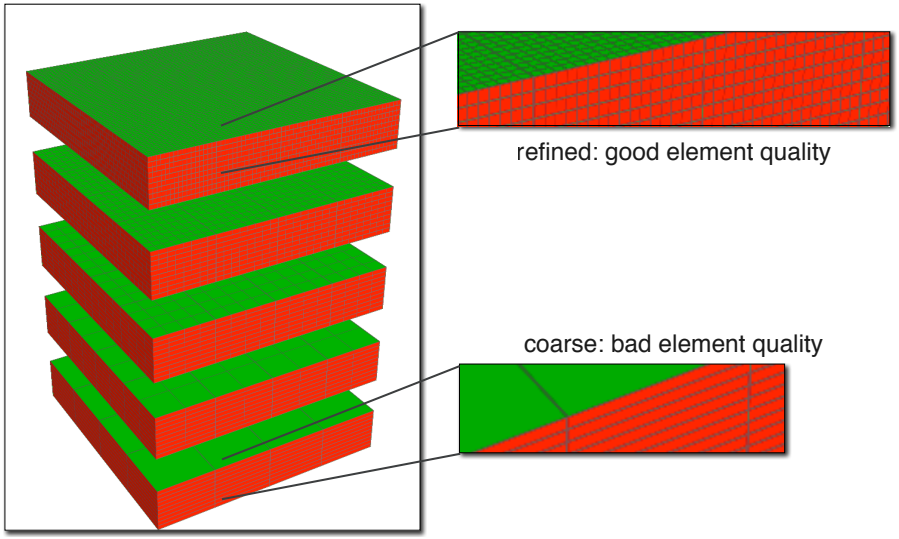
Diffusion equation:

$$\partial_t c_s(t,x) = \nabla \cdot (D_s \nabla c_s(t,x)),$$
$$s \in \{cor, lip\}$$

- *FV*-Simulation of drug diffusion through the *'Stratum corneum'*
- Hex-Grid with *'Corneocytes'* (red) and *'Lipid channels'* (green).
- Difficulties: **jumping coefficients** and **anisotropic elements**.

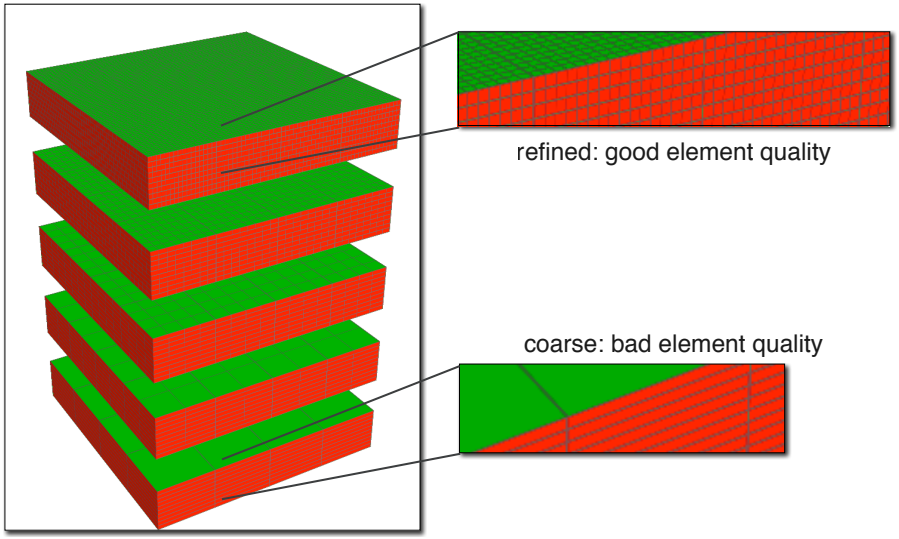
1) Nägel, A., Heisig, M., Wittum, G.: A comparison of two- and three-dimensional models for the simulation of the permeability of human stratum corneum. European Journal of Pharmaceutics and Biopharmaceutics 72(2),

Skin-Problem: Solution process

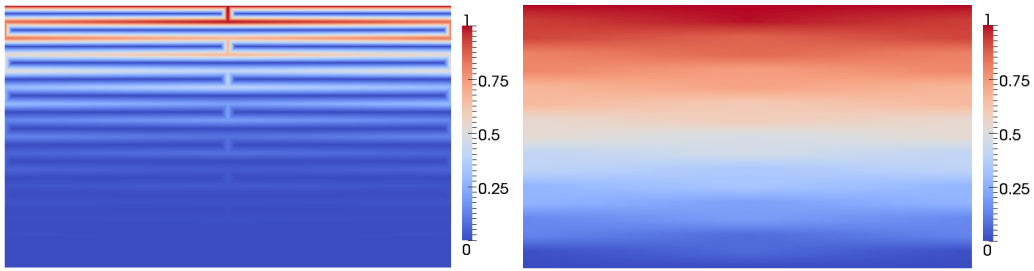


- Generation of MG-hierarchy through **parallel anisotropic refinement**.
- **Element quality improves** with each refinement $\rightarrow$ less solver steps.

Skin-Problem: Solution process



- Generation of MG-hierarchy through **parallel anisotropic refinement**.
- **Element quality improves** with each refinement $\rightarrow$ less solver steps.



Concentration of a substance at two different timesteps.

Performance Modeling for **UG4** - Setting

Setting:

- **UG4** 's code contains *'PROFILE'* calls at many crucial points.
- Different profile-backends exist. Here *ScoreP*[1] is used.
- **Weak scaling study** for the steady state drug diffusion problem.
- Solver: **Geometric Multigrid**, Jacobi-smoother, outer CG.

1) http://www.vi-hps.org/projects/score-p/

2) A. Vogel, A. Calotoiu, A. Strube, S. Reiter, A. Nägel, F. Wolf, and G. Wittum. 10,000 performance models per minute – scalability of the UG4 simulation framework. In J. L. Träff, S. Hunold, and F. Versaci, editors, Euro-Par 2015: Parallel Processing, vol. 9233 of Theoretical Computer Science and General Issues, pages 519–531. Springer International Publishing, 2015

Performance Modeling:

- Run simulations at different process numbers, record timings.
- Generate performance-model for each code-kernel (100-1000) and each metric (5-10) by finding the best fit in *PMNF**:

$$f(p) = \sum_{k=1}^{n} c_k \cdot p^{i_k} \cdot log_2^{j_k}(p)$$

- Sort and analyze models by asymptotic behavior.
- Complexity of $\mathcal{O}(\log p)$ is considered fine.

(*): *Performance Model Normal Form*

1) Calotoiu, A., Hoefler, T., Poke, M., Wolf, F.: Using automated performance modeling to find scalability bugs in complex codes. In: Proc. of the ACM/IEEE Conference on Supercomputing (SC13), Denver, CO, USA. ACM (November 2013)

2) Calotoiu, A., Hoefler, T., Wolf, F.: Mass-producing insightful performance models. In: Workshop on Modeling & Simulation of Systems and Applications, University of Washington. Seattle, Washington (Aug 2014)

3) Picard, R.R., Cook, R.D.: Cross-validation of regression models. Journal of the American Statistical Association 79(387), 575–583 (1984)

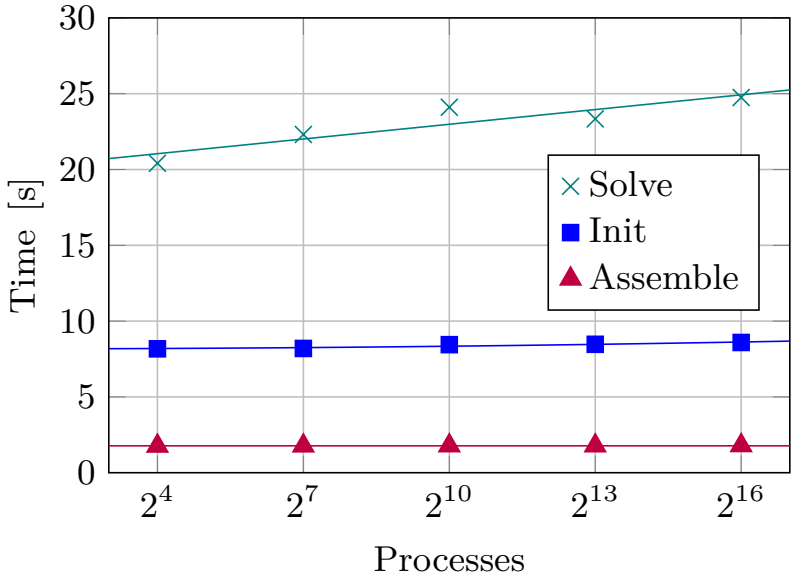
Results I — Identifying performance bottlenecks

Kernel	Time		Bytes sent	
	Model time $= f(p)$ [ms]	$\lvert 1 - R^2 \rvert$ $[10^{-3}]$	Model bytes $= f(p)$	$\lvert 1 - R^2 \rvert$ $[10^{-3}]$
`LoadUGScript` $\rightarrow$ `MPI_Allreduce`	$9.33 + 0.91 \cdot \log p$	42.6	$4 \cdot \mathcal{O}$(`MPI_Allreduce`)	0.000
`init_levels` $\rightarrow$ `MPI_Allreduce`	$27.3 + 1.3 \cdot \log p^2$	19.6	$80.03 \cdot p \cdot \mathcal{O}$(`MPI_Allreduce`)	0.003
`init_top_surface` $\rightarrow$ `MPI_Allreduce`	$3.71 + 5.18 \cdot p^{1/4}$	9.88	$4 \cdot p \cdot \mathcal{O}$(`MPI_Allreduce`)	0.000

- ***Found issue:*** `MPI_Allreduce` used for array of length p.
- ***Where:*** `InitSolver`: Processes have to signal whether they want to take part in the communication on certain multigrid levels.
- ***Now:*** Using `MPI_Comm_split` which scales with $\mathcal{O}(\log^2 p)$.[1]
- Fast location of issue possible thanks to fine grained performance models.

1) Siebert, C., Wolf, F.: Parallel sorting with minimal data. In: Recent Advances in the Message Passing Interface, pp. 170.177. Springer, 2011

Results II — Validating optimal complexity of *GMG*

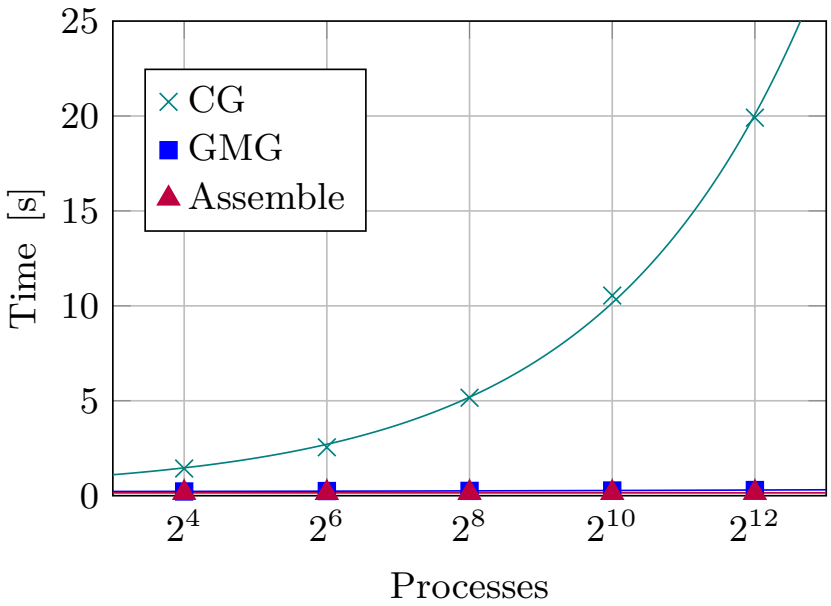


p	L	DoF	n_{gmg}
16	6	290,421	25
128	7	2,271,049	27
1024	8	17,961,489	29
8192	9	142,869,025	29
65536	10	1,139,670,081	29

Kernel	Model for time [s]
Solve	$19.75 + 0.32 \cdot \log_2 p$
Init	$8.17 + 0.002 \cdot \log_2^2 p$
Assemble	1.78

- ***Weak* scaling study for 3d skin problem:**
  - When the number of processes grows by a factor of 8, refine once more.
    $\rightarrow$ Number of elements per process is the same in all runs (*weak scaling*).
- ***Known:*** Number of MG iterations independent of mesh size.
  $\rightarrow$ crucial for good *weak scalability*!
- ***Observed:***
  - Slight increase in iteration numbers (anisotropy in lipid layers).
  - No GMG code kernel scales worse than $\mathcal{O}(\log p)$!
    $\rightarrow$ **Very good overall scaling behavior.**

Results III — Comparison of *GMG* and *CG* scalability



p	L	DoF	n_{cg}	n_{gmg}
16	7	66,049	524	14
64	8	263,169	1003	14
256	9	1,050,625	1977	13
1024	10	4,198,401	3875	13
4096	11	16,785,409	7588	13

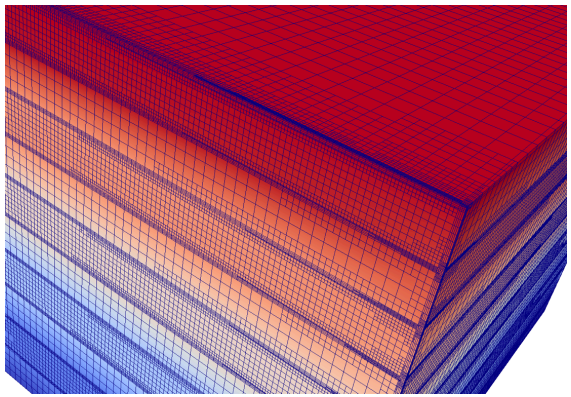
Kernel	Model (time [s])
CG	$0.227 + 0.310 \cdot \sqrt{p}$
GMG	$0.219 + 0.0006 \cdot \log_2^2 p$
Assemble	0.1498

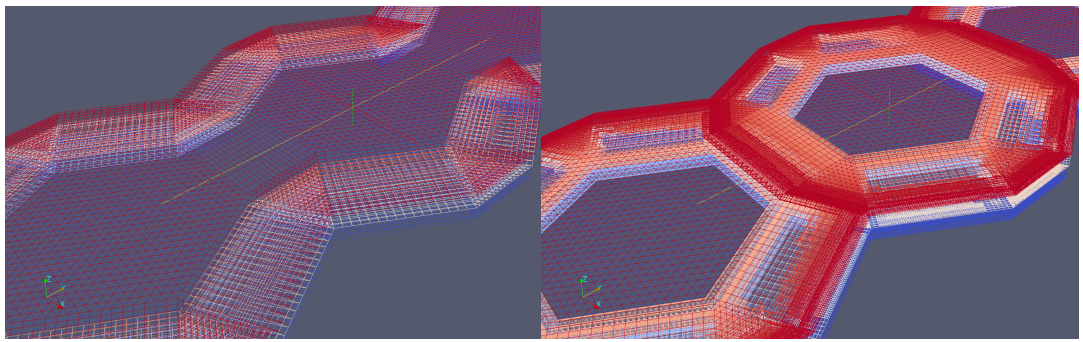
- ***Weak* scaling study for 2d Laplace problem:**
  - When the number of processes grows by a factor of 4, refine once more.
- ***Observed:***
  - GMG iteration numbers stay constant.
  - CG iteration numbers increase by a factor of 2 (expected from theory)

$$\text{GMG: } \mathcal{O}(\log_2^2 p) \quad \leftrightarrow \quad \text{CG: } \mathcal{O}(\sqrt{p})$$

Conclusions and Outlook

- **Automated performance modeling:**
  - Helps to **find scalability issues** in existing codes.
  - **Fine granularity** helps to easily identify responsible code kernels.
  - Suited for large code-bases and massive parallelism.
  - ***Plus:*** Automated unit tests, resource consumption, etc.
- **Results for UG4** :
  - **UG4** features **solvers** with **nearly optimal *weak* scalability**.
  - Efficient **massively parallel** runs on flexible **unstructured grids**.
- **Next steps**:
  - Detailed analysis of **massively parallel adaptive** runs.
  - Special focus on domain partitioning, redistribution, ...





Thank You

Financial support from the DFG Priority Program 1648 *Software for Exascale Computing* (SPPEXA) is gratefully acknowledged.

The Gauss Centre for Supercomputing (GCS) is gratefully acknowledged for providing computing time on the GCS share of the supercomputer *JUQUEEN* at Jülich Supercomputing Centre (JSC).