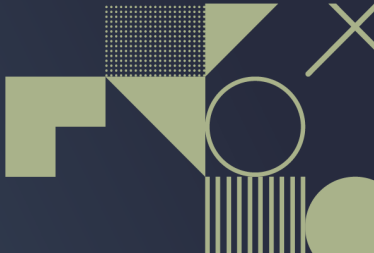
A Next-Generation Discontinuous Galerkin Fluid Dynamics Solver with Application to High-Resolution Lung Airflow Simulations

Martin Kronbichler[1,2], Niklas Fehn[3], Peter Munch[1,4], Maximilian Bergbauer[1], Karl-Robert Wichmann[1,5], Carolin Geitner[1], Momme Allalen[3], Martin Schulz[1], and Wolfgang A. Wall[1]

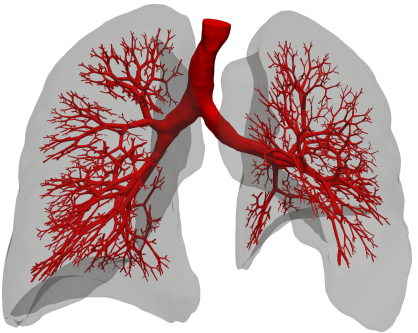
[1] Technical University of Munich [2] Uppsala University
[3] Leibniz Supercomputing Centre [4] Helmholtz-Zentrum hereon GmbH
[5] Ebenbuild GmbH



Outline

Application background: Simulation of air flow in human lung

- Scientific goal: Understand the flow of air in human respiratory system to find better medical treatments
- Especially relevant for mechanical ventilation
- Today: Ventilation done mostly by medical intuition
- Increasing use of **reduced-order computational models**
  - CT/MRI scan of lung
  - Create geometrical model with “network” character
  - Predict response to treatment within minutes to hours
- Real flow is 3D and turbulent in parts of airways
  - PDE model by incompressible Navier–Stokes equations
  - Might use **detailed 3D simulations** to “train” reduced order models



- 3D simulations resolving all relevant features
- Need high resolution to make reliable predictions
- Millions to billions of unknowns
- Use supercomputer power!

Geometry for lung airflow simulations

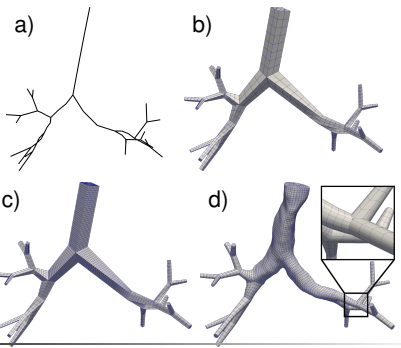
- Human lung starts from main trachea and bifurcates into increasingly smaller airways
- Weibel model: Around 20–23 generations of bifurcations into alveolar network
- Around 10^7 geometrical features to be resolved
- Fluid flow becomes slow and regular for generations $\geq$ 10–13 $\rightarrow$ boundary condition
  - Reduced order model assume Poiseuille flow
  - Model absorption in alveolar structure
- Flow in upper airways much more irregular, especially with local changes in geometry
- Workflow: CT/MRI scan $\rightarrow$ segmented image $\rightarrow$ geometry for large upper airways $\rightarrow$ tree-generation algorithm via volumes for lower airways $\rightarrow$ patient-specific geometry

a) Create 1D network model

b) Transform in 3D volume mesh for hexahedral elements

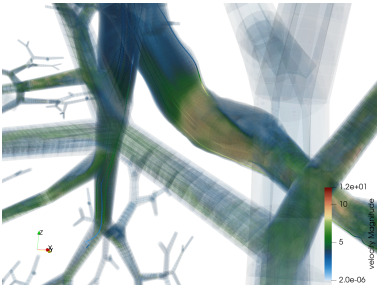
c) Adapt mesh for big upper airways by local mesh refinement

d) Apply patient-specific deformation & cylinders in lower airways



Space discretization: high-order discontinuous Galerkin methods

- High-Reynolds number incompressible flows develop fine-scale features
  - Need to track fine-scale solution components over long times
  - Need accurate “dispersive” behavior
  - Discontinuous Galerkin with upwind fluxes good fit for **velocity**
  - High-order methods, $k = 3, \ldots, 7$, most efficient[1]
- For stable schemes we must fulfill
  - inf–sup condition (e.g. Taylor–Hood finite element)
  - point-wise divergence-free velocity field needed for energy stability and pressure robustness[2]



- For $H(\mathrm{div})$ velocity field, L^2 conforming **discontinuous Galerkin discretization** of pressure is the **natural space**

[1] Fehn, Wall, Kronbichler, Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows, *Int. J. Numer. Meth. Fluids* 88:32–54, 2018

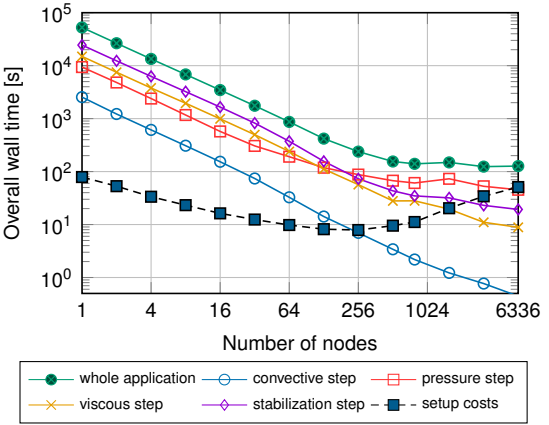
[2] Fehn, Kronbichler, Lehrenfeld, Lube, Schroeder, High-order DG solvers for underresolved turbulent incompressible flows: A comparison of L^2 and $H(\mathrm{div})$ methods, *Int. J. Numer. Meth. Fluids* 91, 2019

Time discretization of Navier–Stokes equations

Dual splitting method in time[3]

- Convective step: explicit
- Pressure step: Poisson equation
  - Multigrid method
- Viscous step: Helmholtz-like equation
  - symmetric positive definite
  - well-conditioned due to small viscosity
- Stabilization step: Symmetric positive definite equation to ensure $H(\text{div})$ conformity
- All linear solvers of optimal complexity $\mathcal{O}(n)$
- Size of linear systems n very large $\rightarrow$ supercomputers

Breakdown of times on 512^3 resolution
$k_u = 3$ for 5k time steps



Arndt, Fehn, Kanschat, Kormann, Kronbichler, Munch, Wall, Witte: ExaDG: High-order discontinuous Galerkin for the exa-scale, LNCSE 136, 2020

[3] Karniadakis, Israeli, Orszag, High-order splitting methods for the incompressible Navier–Stokes equations. *J. Comput. Phys.* 97(2), 1991

Outline

Challenge in lung simulation: strong scaling

- Airway network very long compared to diameter
- Involves $\mathcal{O}(10k)$ "grid points" in length direction
- A single breathing cycle involves $\mathcal{O}(10)$ flow-through periods of represented geometry
- With CFL $= 0.4$, this yields time step sizes of $\sim 1.5 \cdot 10^{-6} \ldots 3 \cdot 10^{-6}$ seconds
- Simulation time for a single breathing cycle is 3 seconds
- $\mathcal{O}(10^6)$ time steps per breathing cycle
- Use parallelism mainly to drive wall time per time step as low as possible

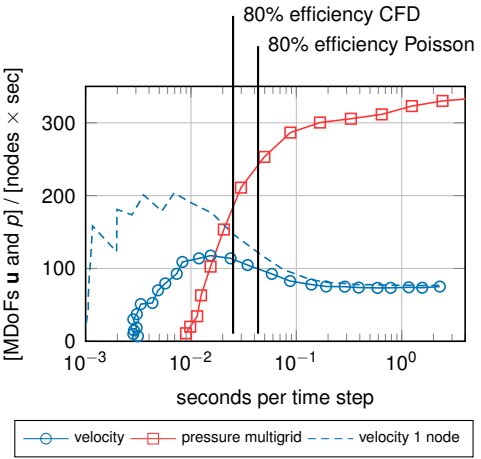
- Simulation on as many compute nodes of SuperMUC-NG (Intel Xeon Skylake) as to reach strong scaling limit
- Coarsest useful mesh: 12 cells in cross section at small outlets
- Polynomial degree $k_u = 3$
- Number of lung generations g

g	#node	#DoF	$N_{\Delta t}$	$t_{\text{wall}}/N_{\Delta t}$	h/cycle
3	2	4.4e5	1.8e5	0.0174 s	0.9
5	16	3.6e6	5.2e5	0.0232 s	3.4
7	32	9.2e6	1.0e6	0.0229 s	6.4
9	128	4.5e7	1.6e6	0.0419 s	19
11	128	7.7e7	2.0e6	0.0451 s	25

Scaling limit in CFD simulation

Example: throughput vs scaling limit

- **Throughput per time step** relative to velocity + pressure DoFs
- Model for velocity: CEED BP4 benchmark[4] with FEM of degree $k_u = 5$ ($\sim$ DG $k_u = 4$)
- Assume 4 iterations for pressure MG with $k_p = 3$ (DG), 15 iterations for velocity
  - Sophisticated extrapolations
- Combination of pressure Poisson + CG solver influences scaling limit
  - Velocity solver with cache effect on velocity $\rightarrow$ scaling limit of Poisson solver
  - Velocity solve must be optimized for **throughput**!



System: **512 nodes** of Intel Skylake

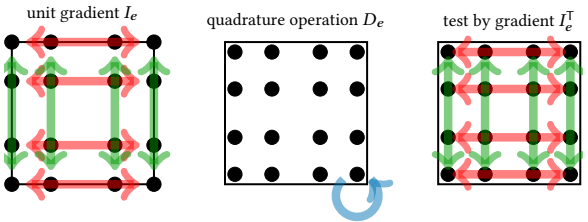
[4] Fischer, Min, Rathnayake, Dutta, Kolev, Dobrev, Camier, Kronbichler, Warburton, Świrydowicz, Brown: Scalability of high-performance PDE solvers. *Int. J. High Perf. Comput. Appl.*, 34(5):562–586, 2020

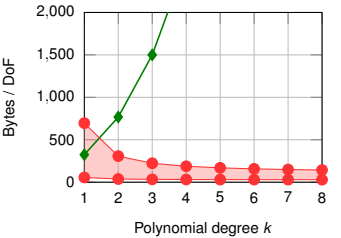
Implications on CFD efficiency

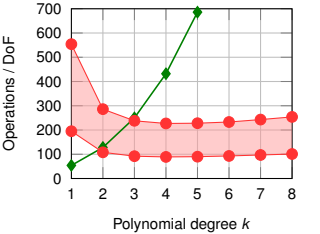
- Velocity ingredients need to be **optimized for throughput**
  - Matrix-free methods with sum factorization to optimize efficiency for higher polynomial degrees
  - Optimal use of hardware resources by optimizing for **node-level performance**
  - Data locality optimizations in conjugate gradient solver to reduce memory transfer
- Pressure Poisson solver needs to be **optimized for latency**
  - Obtain low number of iterations and few sweeps through MPI network
  - Balance between simple smoothers and cost of synchronization
  - Still need reasonably high throughput
  - Also use matrix-free methods and optimizations made for velocity
- Hardware choice:
  - SuperMUC-NG, based on Intel Xeon Platinum 8174 (Skylake)
  - 2×24 cores
  - 205 GB/s of memory bandwidth per node
  - Arithmetic performance with SIMD vectorization up to 3.5 TFlop/s

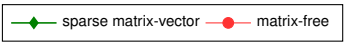
Fast evaluation of high-order discretizations: matrix-free methods

- Matrix-vector product and substitutions 60–95% of cost of multigrid (smoothers, residual, transfer) or velocity solvers
- Large structured/unstructured meshes and adaptivity: traditionally solved with sparse matrices at 0.16–0.25 Flop/Byte $\rightarrow$ 1–5 % of arithmetic peak
- Change algorithm: read less from memory, re-compute
- Matrix-free methods target **high orders** with sum-factorization techniques to compute integrals





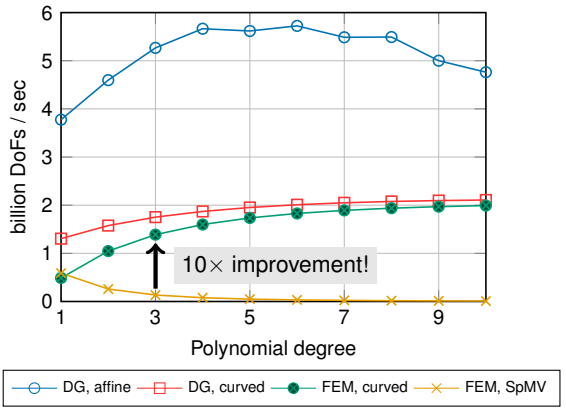




Fast evaluation of high-order discretizations: matrix-free methods

- Look at **run time** of matrix-vector product
- Problem sizes around 10 million DoFs
- Discretization DG: Symmetric interior penalty method
- Matrix-free algorithm
  - Series of small matrix-matrix multiplications and operation at quadrature points
  - Arithmetic intensity 1–8 Flop/byte
  - Use SIMD: vectorization across cells
    - SIMD abstraction class to reliably vectorize
  - Reaches memory bandwidth limit
- Sparse matrix is inappropriate format for higher polynomial degrees, $p \geq 2$

Throughput of matrix-vector product in 3D



System: 1 node of 2 × 24 cores of Intel Xeon Platinum 8174 (Skylake)
Memory bw: 205 GB/s, arithmetic peak 3.5 TFlop/s

Kronbichler, Wall: A performance comparison of continuous and discontinuous Galerkin methods with fast multigrid solvers. *SISC* 40(5):A3423–48, 2018
Kronbichler, Kormann: Fast matrix-free evaluation of discontinuous Galerkin finite element operators. *ACM TOMS* 45(3):29/1–40, 2019

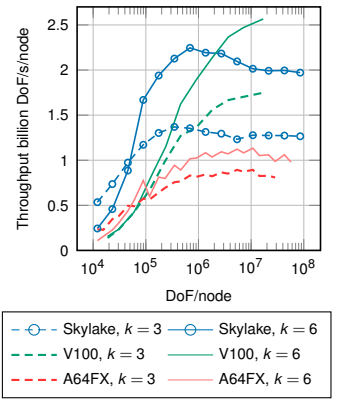
Implementation

- Algorithms implemented in C++
- Finite element algorithms, sum factorization, HPC infrastructure by deal.II finite element library `https://dealii.org`, hosted on `https://github.com/dealii/dealii`
- CFD application solver ExaDG, hosted on `https://github.com/exadg/exag`
- Precise artifact description available with paper
- Algorithms executed on SuperMUC-NG, Intel Xeon Platinum 8174 (Skylake), using 1–6400 nodes

Arithmetic optimizations: Result

- On CPUs: Optimizations of arithmetic operations in matrix-free methods make operator evaluation almost as cheap as streaming vectors
- Consequence: Vector operations in conjugate gradient solver take significant time
- Solution: Data access optimizations
  - Overlap vector updates with matrix-vector product
  - Compute metric terms on the fly
- CPU code for conjugate gradient method loads around 11 doubles / DoF in an iteration
- Un-optimized algorithm on CPU or best GPU code must load $\sim$ 33 doubles / DoF for CEED benchmark
- Intel Skylake CPU can almost catch up with one NVIDIA V100 GPU despite 3.5$\times$ lower memory bandwidth

Evaluation of CEED benchmark BP3 on Intel Skylake of SuperMUC-NG, Nvidia V100 (Summit, results by Kolev et al., Parallel Computing 2021), and Fujitsu A64FX



Outline

Application background

Fast airflow simulations

Scalable multigrid solvers

Summary

Matrix-free methods within multigrid solvers

- Matrix-free implementation essential to reach good throughput for higher orders
- But **no matrix entries available** – options for multigrid solvers?

Low performance / impossible

- Substitutions on matrix entries such as (S)SOR, ILU $\rightarrow$ little benefit from using these smoothers
- Algebraic multigrid hierarchy generation
- AMG applied to high-order space leads to high iteration counts

High performance

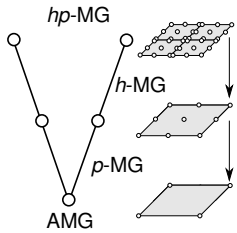
- Polynomial smoothers (point-Jacobi within Chebyshev iteration)
- Overlapping Schwarz smoothers[5]
- Multigrid transfer operations
- Algebraic multigrid on low-order refined discretization

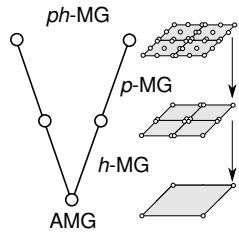
- Select point-Jacobi with Chebyshev acceleration despite somewhat higher iteration counts than Schwarz methods due to better arithmetic optimizations

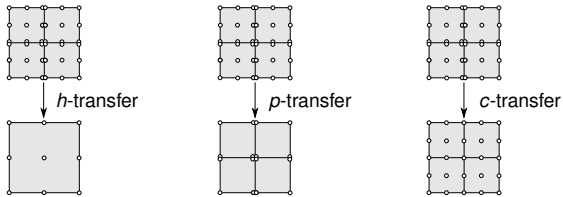
[5] Fischer, Lottes, Hybrid Schwarz-Multigrid Methods for the Spectral Element Method, 2004

Multigrid: Work on hierarchy of coarser representations

- Setup for lung: Large coarse mesh + some adaptive refinements
- Question: How to create coarser problems needed for multigrid?
- Classical approach 1: reduce polynomial degree down to linear methods (*p* multigrid)
- Hybrid multigrid: after *p* coarsening, coarsen also mesh (*ph* MG)
- Innovation[6]: transfer DG $\rightarrow$ continuous elements as first step in hierarchy (*cph* coarsening)
- Combined with AMG on coarse level
- Multigrid V-cycle in **single precision**







[6] Fehn, Munch, Wall, Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *J Comput Phys* 415:109538, 2020

Best multigrid coarsening strategy by # iterations

Solve Poisson equation on deformed cube, 8^3 elements, tolerance 10^{-10}, Chebyshev(5,5) with point-Jacobi smoother, coarse solver AMG V-cycle, report fractional iteration counts

- p multigrid with coarsening $k_{l-1} = \lfloor k_l/2 \rfloor$

MG type	Polynomial degree			
	1	3	5	9
h	3.4	15.3	18.5	25.1
ph	17.7	16.4	18.6	25.1
phc	17.7	16.4	14.1	25.1
cp	8.5	5.8	6.5	9.5
cph	8.5	5.9	6.5	9.5

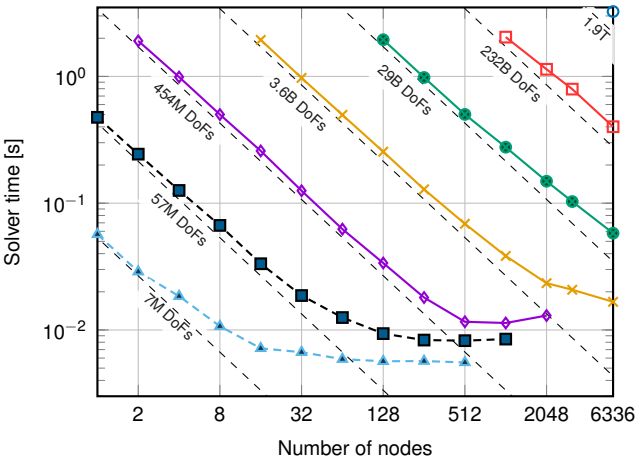
- Robustness with respect to penalty parameter τ in SIPG
- Polynomial degree $k = 4$

IP factor	*hp* MG	*ph* MG	*cph* MG
$10^0(k+1)^2/h$	12.2	12.0	4.9
$10^1(k+1)^2/h$	39.7	33.2	5.3
$10^2(k+1)^2/h$	83.7	52.2	5.4
$10^3(k+1)^2/h$	123	70.9	5.4

[6] Fehn, Munch, Wall, Kronbichler, Hybrid multigrid methods for high-order discontinuous Galerkin discretizations, *J Comput Phys* 415:109538, 2020

Multigrid on simple meshes: Extremely fast

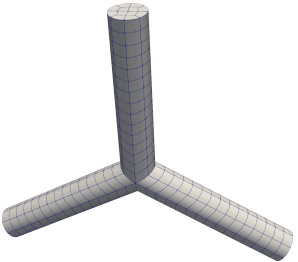
- Discontinuous elements, $p = 5$, affine mesh
- Conjugate gradient solver preconditioned by *ch* multigrid
  - DG: Chebyshev (6,6) iteration around block-Jacobi (fast diagonalization)
  - FEM: Chebyshev (6,6) + pJac
  - Multigrid V-cycle in single precision
- 2 CG iterations (tolerance 10^{-3})
- Serial performance very high: **2.5 million DoFs / core / sec**
- Intel Xeon Platinum 8174, 48 cores per node, up to 6336 nodes (full SuperMUC-NG)

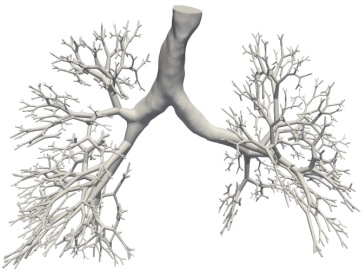


Arithmetic performance 1.9 trillion DoFs:
5.9 PFlop/s (5.7 PFlop/s in SP, 0.27 PFlop/s in DP)
180 GB/s per node (STREAM: 210 GB/s)

Multigrid on more complicated setups

- Generic bifurcation: Simple mesh
  - Cylinders bifurcating at 60 degree angle
  - Nice aspect ratio elements
  - $3-7$ global refinements to obtain larger problems
  - Coarsest mesh with 686 DoFs
  - Coarse solver with BoomerAMG on 1 MPI process
- Lung geometry: Complicated domain
  - Various types of angles
  - Patient-specific geometry deformation
  - Adaptive refinement in upper airways to create good CFD resolution
  - $0-3$ global refinements to obtain larger problems (in practice: 0–2 most important)
  - Coarsest problem with 300k DoFs
  - Coarse solver with BoomerAMG on ≤ 710 MPI processes

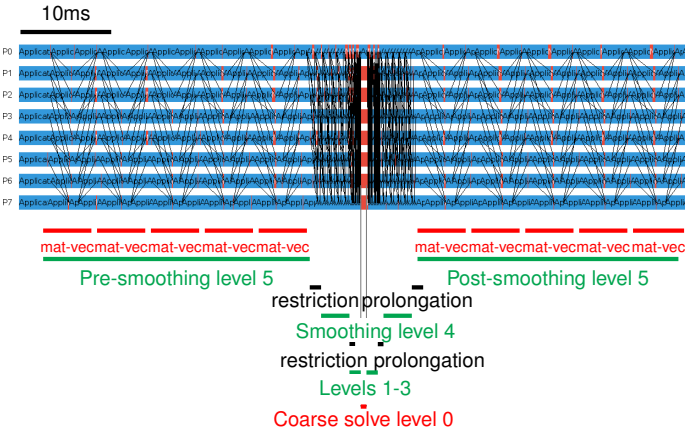




Communication pattern of multigrid

- Multigrid involves communication on coarser levels
- Near strong scaling limit, only coarsest 1–2 levels in throughput regime, rest latency-dominated
- Coarse solver is serial $\rightarrow$ multigrid implies **global communication**
  - Restriction $\approx$ global reduction
  - Prolongation $\approx$ broadcast
  - Reduction structure implied by mesh coarsening (GMG + AMG)
  - Cannot be optimized to machine, as opposed to `MPI_Allreduce`

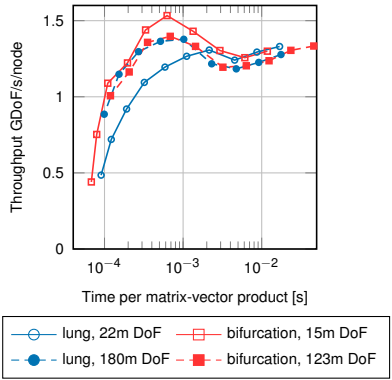
Analysis of MPI messages between 8 ranks with Intel trace analyzer



Ingredients to good multigrid scalability

- Matrix-free evaluation operated in throughput-limited regime for large sizes (SIMD vectorization)
- Matrix-free evaluation does nearest-neighbor communication in MPI $\rightarrow$ look at communication latency
- Analysis: Scaling limit of mat-vec
- Observation: Cannot scale much below 10^{-4} seconds
  - Network latency around 10^{-6} seconds
  - $\mathcal{O}(10)$ messages, twice per step, some computations
  - Overlap communication and computation
- Multigrid with Chebyshev (3,3) smoother does 6 matrix-vector products per level
- Level transfer has similar latency limit
- On 7 levels, scaling limit of 1 V-cycle is around $5 \cdot 10^{-3}$ second

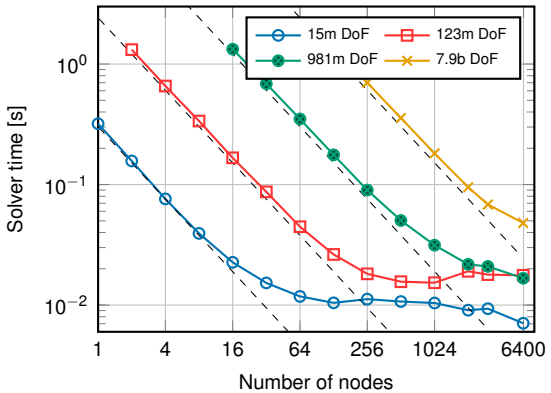
- Two fixed problem sizes
- Number of nodes increases from right to left
- Observe scaling limit



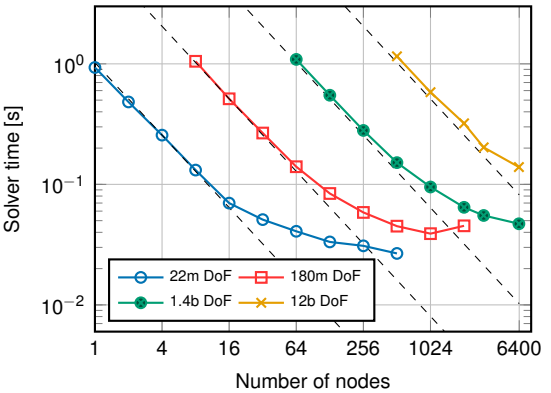
Scalability of multigrid solver on SuperMUC-NG (Intel Skylake)

Chebyshev (3,3) smoother, tolerance 10^{-3}, *cph* coarsening

- Generic bifurcation
- Converges in 3 CG iterations



- Lung geometry
- Converges in 7 CG iterations



Largest computations on 6,400 nodes subject to noise – despite dynamic load balancing

Scalability of multigrid solver on SuperMUC-NG (Intel Skylake)

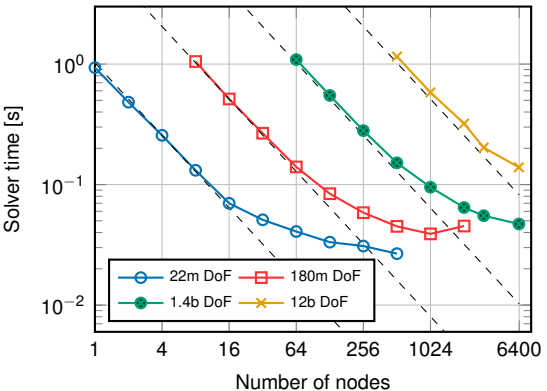
Chebyshev (3,3) smoother, tolerance 10^{-3}, *cph* coarsening

- Generic bifurcation
- Converges in 3 CG iterations

Reasons for worse scaling:

- Increase in iterations due to strongly deformed elements
- Coarse AMG solver takes 3.5×10^{-3} second per solve, or 2.4×10^{-2} seconds latency for full solve
- Slight load imbalance in matrix-free evaluation on hanging nodes

- Lung geometry
- Converges in 7 CG iterations



Largest computations on 6,400 nodes subject to noise – despite dynamic load balancing

Outline

Application background

Fast airflow simulations

Scalable multigrid solvers

Summary

Summary

- Lung simulations with high-order DG method
- CPU-based system very good for strong scaling limit
  - High performance when run from caches
  - Gains high throughput for smaller problem sizes than GPU
- Reach wall time per time step between 0.025 seconds and 0.05 seconds for complicated geometry
  - Many groups struggle to get below 0.1 second per time step (especially on GPUs)
- Presented ingredients
  - Use of SIMD units by cross-element vectorization
  - Multigrid coarsening strategies with low iteration count: *cph* multigrid
    - Multigrid latency-sensitive due to many levels
    - Intermediate and coarse levels much smaller in size — do **not need to be efficient, just fast**
  - Data locality optimizations for velocity CG solver
- Ongoing challenge to strong scaling and interaction with coarse AMG solver
- Better multigrid smoothers