High-performance Computing for Flows in Porous Media

Peter Bastian and Steffen Müthing

Universität Heidelberg | Zukunft. Seit 1386.

NM2PorousMedia-2014, Dubrovnik, Oct. 3, 2014

Contents

EXA-DUNE Project

A software framework for the numerical solution of PDEs

- Joint project in Dortmund (Göddeke, Turek), Münster (Engwer, Ohlberger), Kaiserslautern (Iliev), Heidelberg (B., Ippisch)
- Open-source reusable exa-scale PDE solver components
- Exploit modern hardware without sacrificing generality
- Application domain: flow in porous media
- Multiscale methods and uncertainty quantification

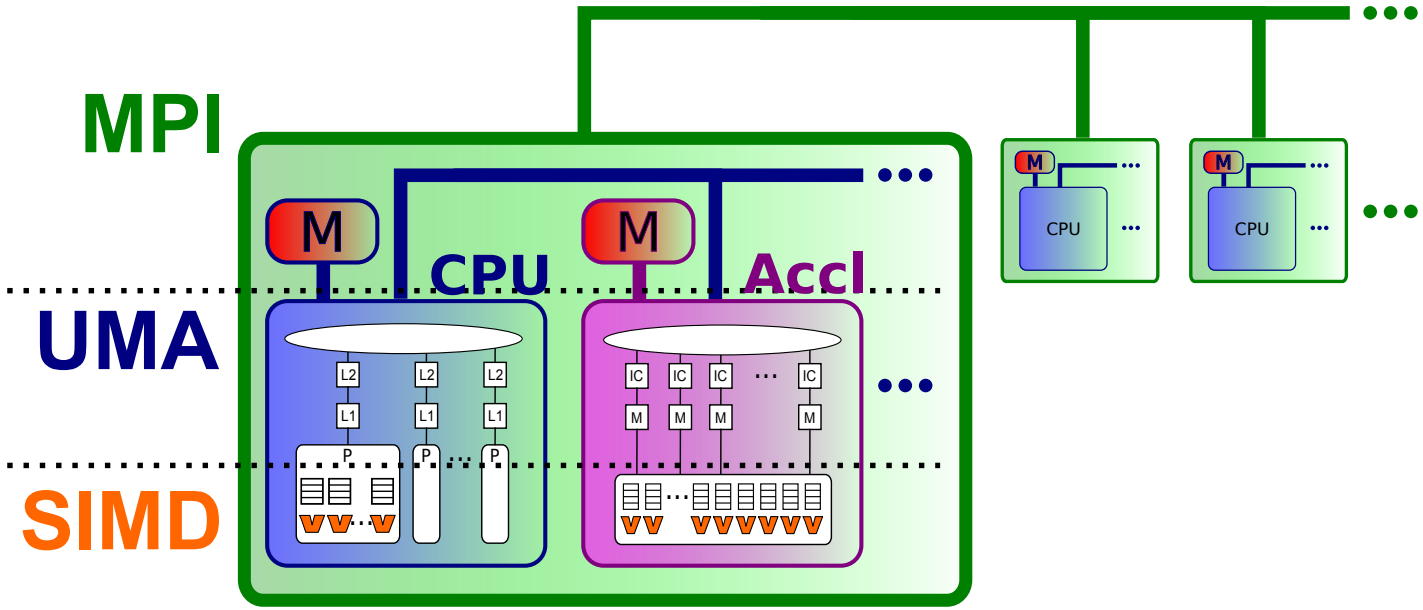
Dune

- Generality and reuse
- Seperation of data structures and algorithms
- Static polymorphism in C++
- State-of-the-art numerics
- MPI-based parallelism

FEAST

- Hardware-oriented numerics
- CFD / solid mechanics
- Locally structured, globally unstructured
- Robust geometric multigrid
- Multicore/GPGPU/MPI

The Exascale Challenge



(Figure by Jö Fahlke)

Trends in evolving HPC architectures

- Ever increasing amount of parallelism on several levels
- Heterogeneous node designs
- Growing importance of SIMD vector instructions
- Deep memory hierarchies (NUMA)

PDE Numerics and Exascale

- Exploit hybrid parallelism: MPI + shared memory + vectorization
  i.e. Intel TBB + auto vectorization
- Restrict threading to UMA nodes
- Transparent accelerator support (mostly Intel Phi)
- FLOPS are for free, memory access is expensive (in time and energy)
- Matrix-based methods
  - Robust preconditioners available (AMG)
  - Poor performance, bound by memory bandwidth
- Matrix-free methods
  - Requires explicit time-stepping or certain solvers, e.g. CG, GMG
  - Low-order FE schemes:
    - ★ Should exploit problem structure: linear transformation, regular grids, constant coefficients, ...
    - ★ Vectorization over several elements
  - High-order FE schemes:
    - ★ Complexity reduction through tensor-product approaches possible
    - ★ Vectorization within one element
- Mixed precission, asynchronicity, MS and UQ embarassingly parallel

Outline

This talk concentrates mainly on algorithmic improvements

Shared memory parallelization is in progress ...

1. Introduction to applications in porous media flow
2. Low-order fully-implicit DG method for two-phase flow
   - Accuracy versus computational effort
   - h-scalability (mesh size)
   - p-scalability (number of processors)
3. High-order DG methods exploiting tensor product structure
   - q-scalability (polynomial degree)
   - Floating point performance (vectorization)
   - Explicit time-stepping methods, some hint on implicit schemes

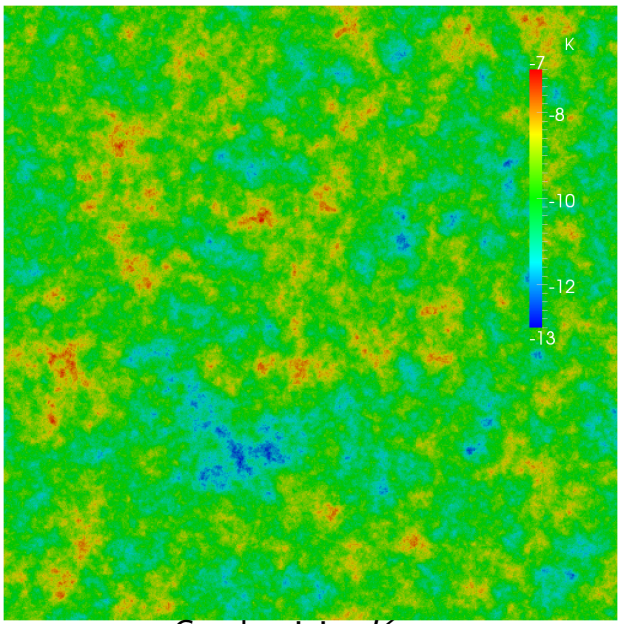
Single Phase Flow and Transport

Parabolic/hyperbolic problem coupled to elliptic problem:

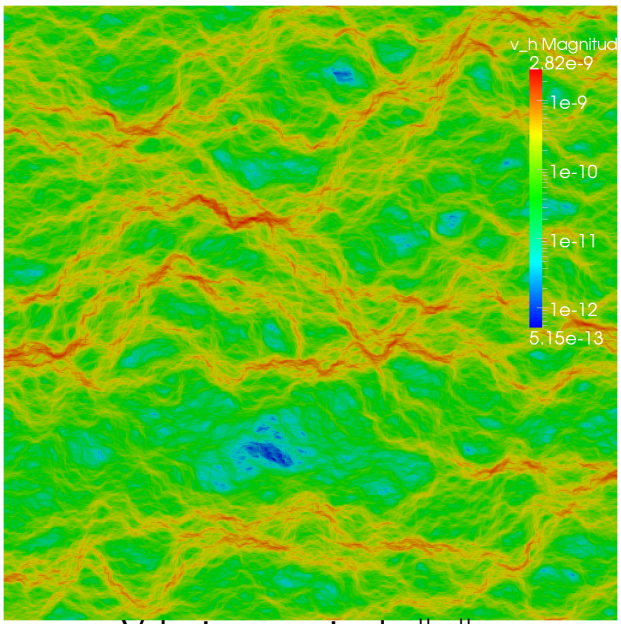
$$\nabla \cdot v = f, \qquad v = -\frac{K}{\mu}(\nabla p - \rho g),$$

$$\partial_t(\Phi c) + \nabla \cdot (cv - D(v)\nabla c) = q.$$

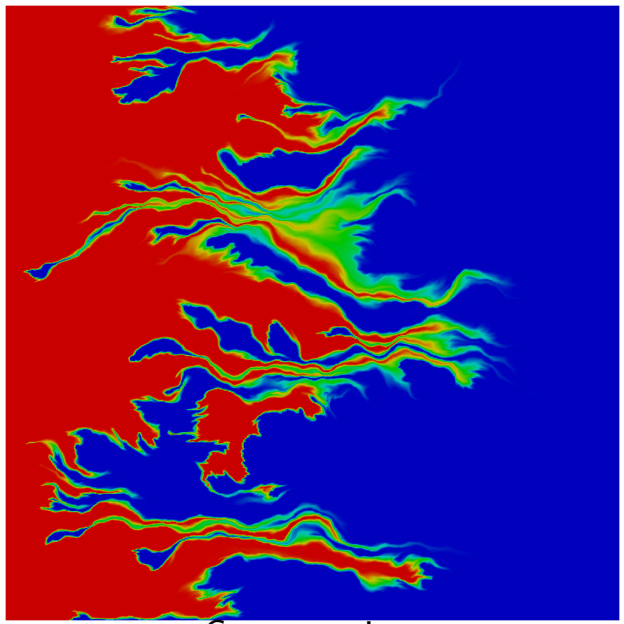
Tensor K, D, heterogeneity, convection-dominated
effective macro-scale descriptions, uncertainty of parameters



Conductivity K



Velocity magnitude $\|u\|$



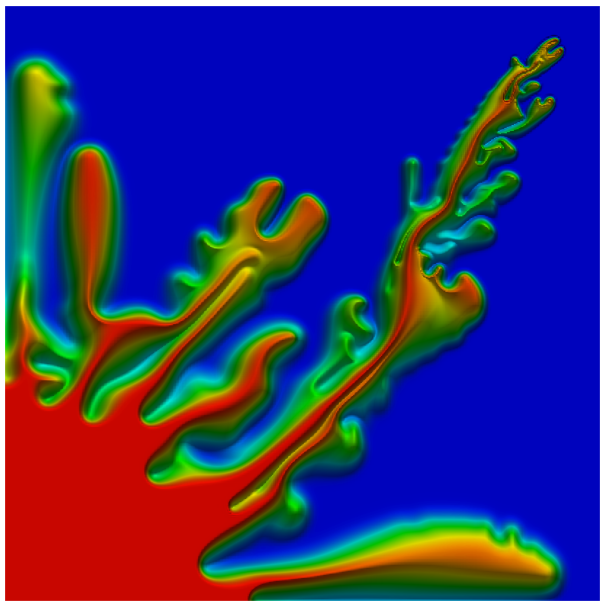
Concentration c

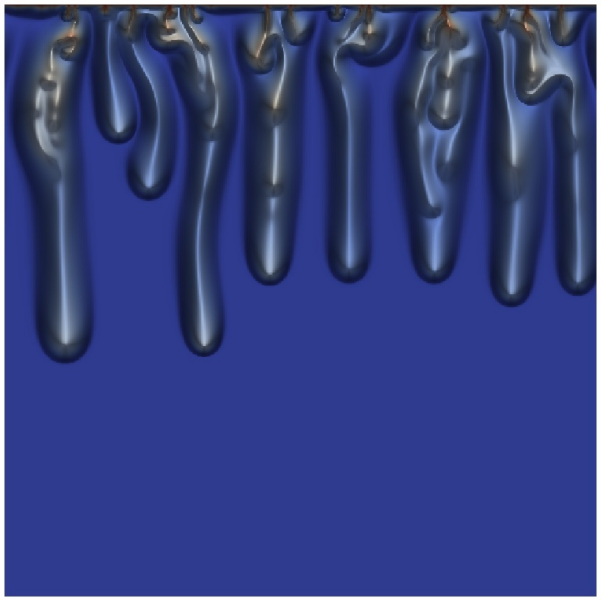
Unstable Flows in Porous Media

$$\nabla \cdot v = 0, \qquad v = -\frac{K}{\mu(c)}(\nabla p - \rho(c)g)$$

$$\partial_t(\Phi c) + \nabla \cdot (cv - D(v)\nabla c) = 0$$

Density driven flow, miscible discplacement
Requires high resolution schemes





Incompressible Immiscible Two-phase Flow

Total fluid conservation:

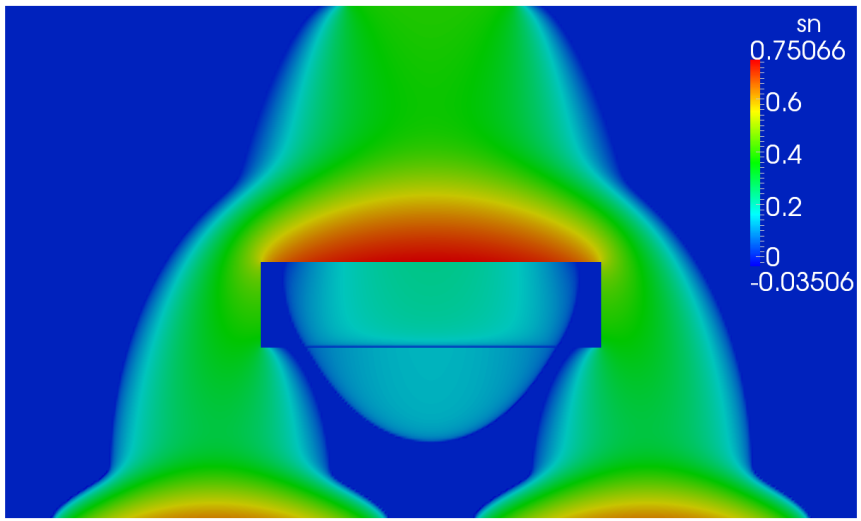
$$\nabla \cdot (v_a - \lambda_n K \nabla \phi_c) = q, \qquad v_a = -\lambda_t K \nabla \phi_w.$$

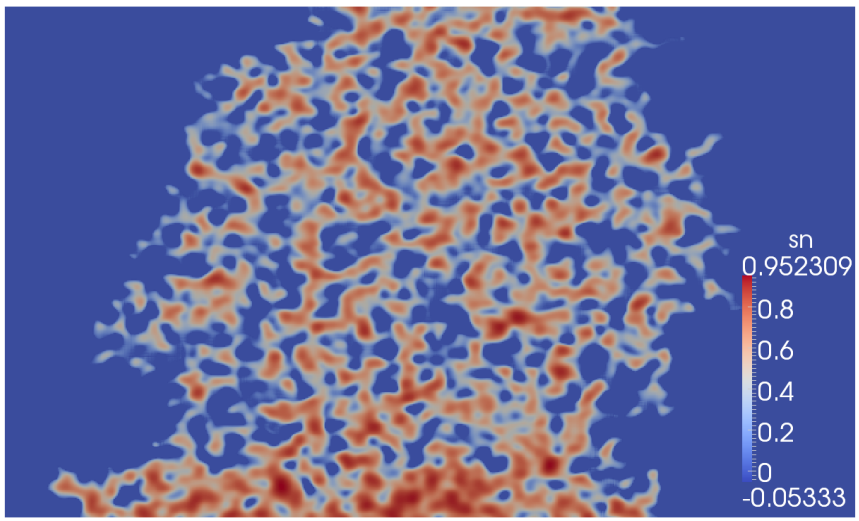
Conservation of non-wetting phase:

$$\Phi \partial_t (1 - \psi(\phi_c)) + \nabla \cdot (f_n(1 - \psi(\phi_c))v_a - \lambda_n(1 - \psi(\phi_c))K\nabla\phi_c) = q_n.$$

Primary variables: $\phi_w = p_w - \rho_w g d$, $\phi_c = p_c - (\rho_n - \rho_w) g d$

$\psi(\phi_c)$, $\lambda_n(\phi_c)$, $\lambda_t(\phi_c)$, $f_n(\phi_c)$ are nonlinear functions





Contents

Discussion of Discontinuous Galerkin

Advantages

- Potentially higher order of convergence
- Higher-order methods beneficial even without sufficient regularity
- Element-wise conservative
- Full permeability tensors, discontinuous coefficients
- Handles elliptic, parabolic and hyperbolic equations equally well
- Unstructured grids, nonconforming refinement
- Increased arithmetic intensity: locally dense matrix blocks: less indirect access, BLAS level 3 operations

Disadvantages

- No monotonicity
- Free parameter to tune (some methods)
- Higher number of DOFs compared to standard FE or FV
- Linear systems are more difficult to solve

Total Fluid Conservation Equation

$$a(\phi_w, \phi_c, w) = \sum_{T\in\mathcal{T}_h} \int_T (\lambda_t K\nabla\phi_w + \lambda_n K\nabla\phi_c)\cdot\nabla w - \sum_{F\in\mathcal{F}_h^{iDw}} \int_F \nu_F \cdot \{\lambda_t K\nabla\phi_w + \lambda_n K\nabla\phi_c\}_\omega [\![w]\!]$$

$$- \sum_{F\in\mathcal{F}_h^{iDw}} \int_F \theta\nu_F \cdot \{\lambda_t K\nabla w\}_\omega [\![\phi_w]\!] + \sum_{F\in\mathcal{F}_h^{iDw}} \int_F \gamma_{F,w} [\![w]\!][\![\phi_w]\!]$$

$$l(w) = \sum_{T\in\mathcal{T}_h} \int_T qw - \sum_{F\in\mathcal{F}_h^{Nw}} \int_F jw \qquad \theta = 1 \text{ (SIPG)}, \ \theta = 0 \text{ (OBB)}, \ \theta = -1 \text{ (NIPG)}$$

SIPG + weighted averages (*Di Pietro, Ern, Guermond '2008*):

$$\delta_{Kn}^{\pm} = \nu_F^t K^{\pm} \nu_F, \qquad \omega^- = \frac{\delta_{Kn}^+}{\delta_{Kn}^- + \delta_{Kn}^+}, \qquad \omega^+ = \frac{\delta_{Kn}^-}{\delta_{Kn}^- + \delta_{Kn}^+}.$$

Penalty term ($\langle\cdot,\cdot\rangle$ denotes harmonic average):

$$\gamma_{F,w} = \langle \lambda_t^- \delta_{Kn}^-, \lambda_t^+ \delta_{Kn}^+ \rangle M, \quad M = \alpha k(k+n-1)\frac{|F|}{\min(|T^-(F)|, |T^+(F)|)}$$

Non-wetting Fluid Conservation Equation

$$b(\phi_w, \phi_c, z) = -\sum_{T\in\mathcal{T}_h}\int_T (f_n\hat{v}_a - \lambda_n K\nabla\phi_c)\cdot\nabla z$$
$$+\sum_{F\in\mathcal{F}_h^{iDn}}\int_F \langle f_n^{\uparrow,-}, f_n^{\uparrow,+}\rangle\, \nu_F\cdot\hat{v}_a\,[\![z]\!] + \sum_{F\in\mathcal{F}_h^{iDn}}\int_F \nu_F\cdot\{-\lambda_n K\nabla\phi_c\}_\omega [\![z]\!]$$
$$+\sum_{F\in\mathcal{F}_h^{iDn}}\int_F \theta\nu_F\cdot\{-\lambda_n K\nabla z\}_\omega J(\phi_c) + \sum_{F\in\mathcal{F}_h^{iDn}}\int_F \gamma_{F,n}[\![z]\!]J(\phi_c)$$

$$r(z) = \sum_{T\in\mathcal{T}_h}\int_T q_n z - \sum_{F\in\mathcal{F}_h^{Nn}}\int_F j_n z$$

Upwinding of fractional flow and penalty parameter:

$$p_c^{\uparrow} = \begin{cases} p_c^- & \nu_F\cdot\hat{v}_a \geq 0 \\ p_c^+ & \text{else} \end{cases}, \quad s_n^{\uparrow,\pm} = 1-\psi^{\pm}(p_c^{\uparrow}), \quad f_n^{\uparrow,\pm} = f_n^{\pm}(s_n^{\uparrow,\pm}).$$

$$\gamma_{F,n} = \frac{\lambda_n^- + \lambda_n^+}{2}\langle\delta_{Kn}^-, \delta_{Kn}^+\rangle M \quad \text{for all interior faces}$$

Fully-Coupled Solution Technique

- Method of lines approach:
  - Discretize with DG in space
  - Discretize ODE system with diagonally implicit Runge-Kutta schemes
- Available schemes:
  - One step θ
  - Fractional step θ
  - Alexander schemes of order 2 and 3
- Solve nonlinear system per stage with inexact Newton method
- Solve linear system with algebraic multigrid

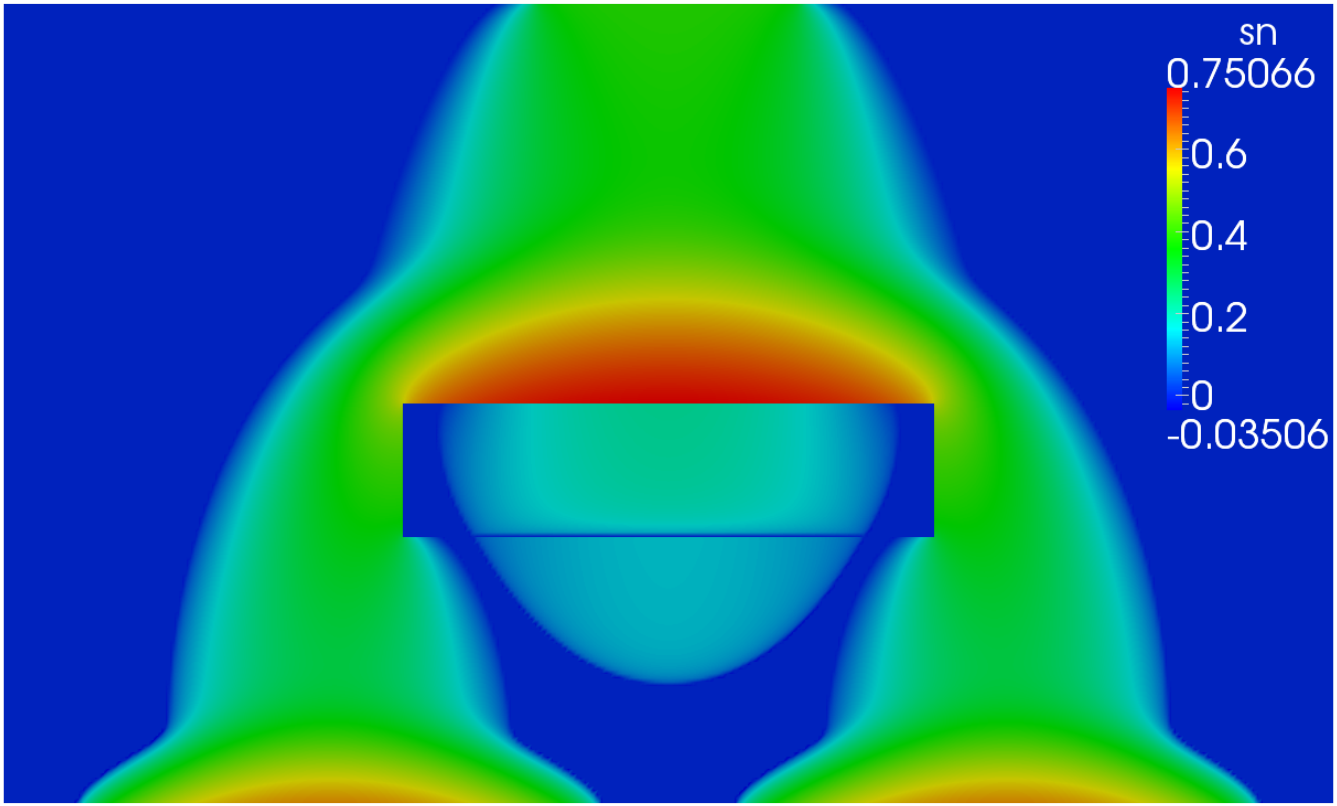
AMG For DG

Idea: Subspace correction in the lowest-order continuous Galerkin subspace [*B., Blatt, Scheichl, NLAA, 2012*]:

$$E_{AMG4DG} = (I - B_{SG}A_{DG})^{\nu_2}(I - R_{CG}^T B_{AMG} R_{CG} A_{DG})(I - B_{SG}A_{DG})^{\nu_1}$$

- B_{SG}: single grid preconditioner for DG system, e.g. block SSOR
- R_{CG} given by $\varphi_i^{CG} = \sum_{j=1}^{n_{DG}} r_{ij}\varphi_j^{DG}$
- Compute $A_{CG} = R_{CG}A_{DG}R_{CG}^T$ via sparse matrix multiplication
- B_{AMG} is one V-cycle of agglomeration AMG applied to A_{CG}
- Our agglomeration AMG scales weakly up to one million cores
- Purely algebraic, except the definition of R_{CG}

A DNAPL Infiltration Model Problem

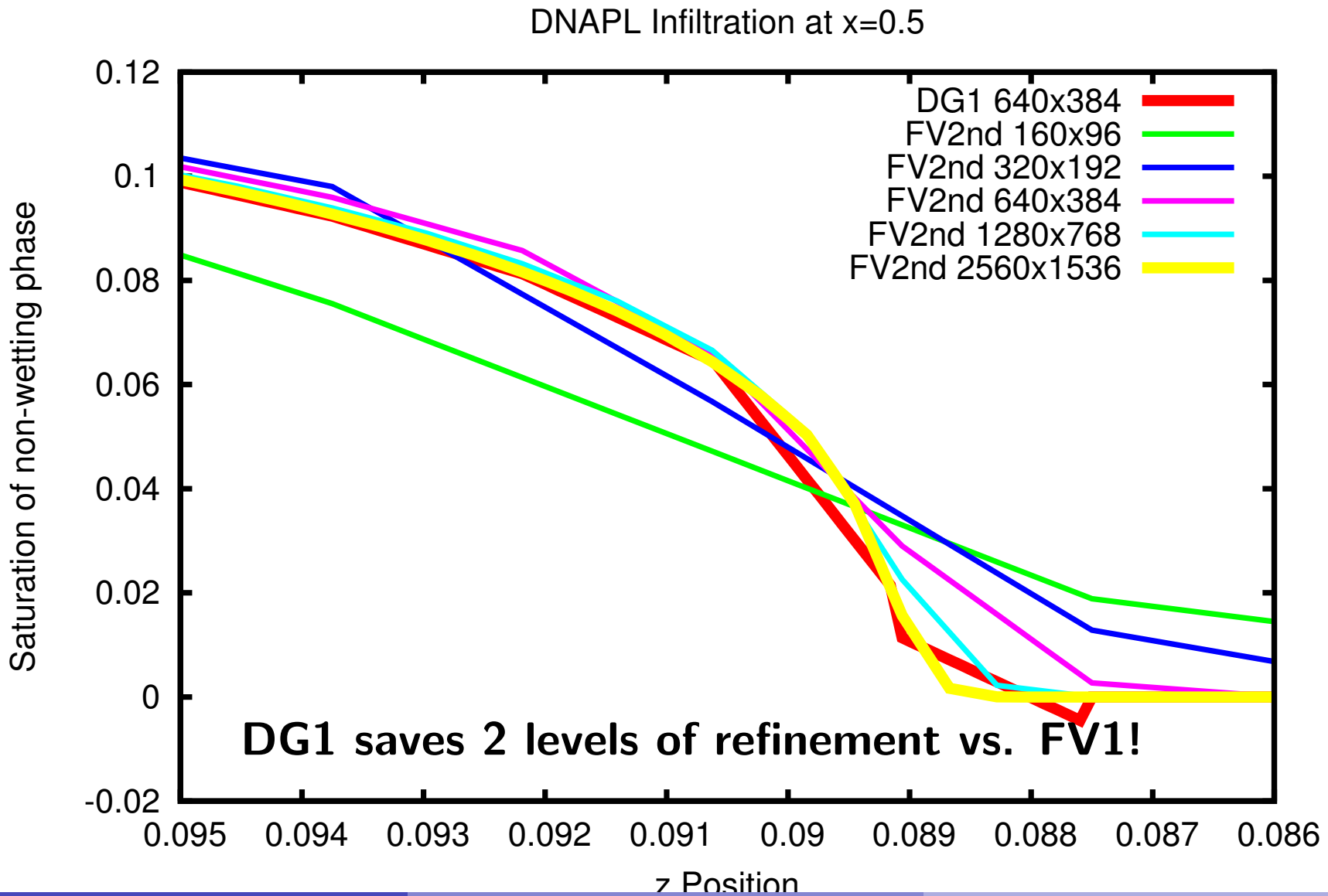


- $k_{r\alpha}$: quadratic
- p_c: Brooks-Corey, $\lambda = 2$
  $p_e^{bulk} = 755$, $p_e^{lense} = 1163$

- DG1 / Alexander(2)
- CCFV0: full upw., impl. Euler
- CCFV1: central, Alex.(2)

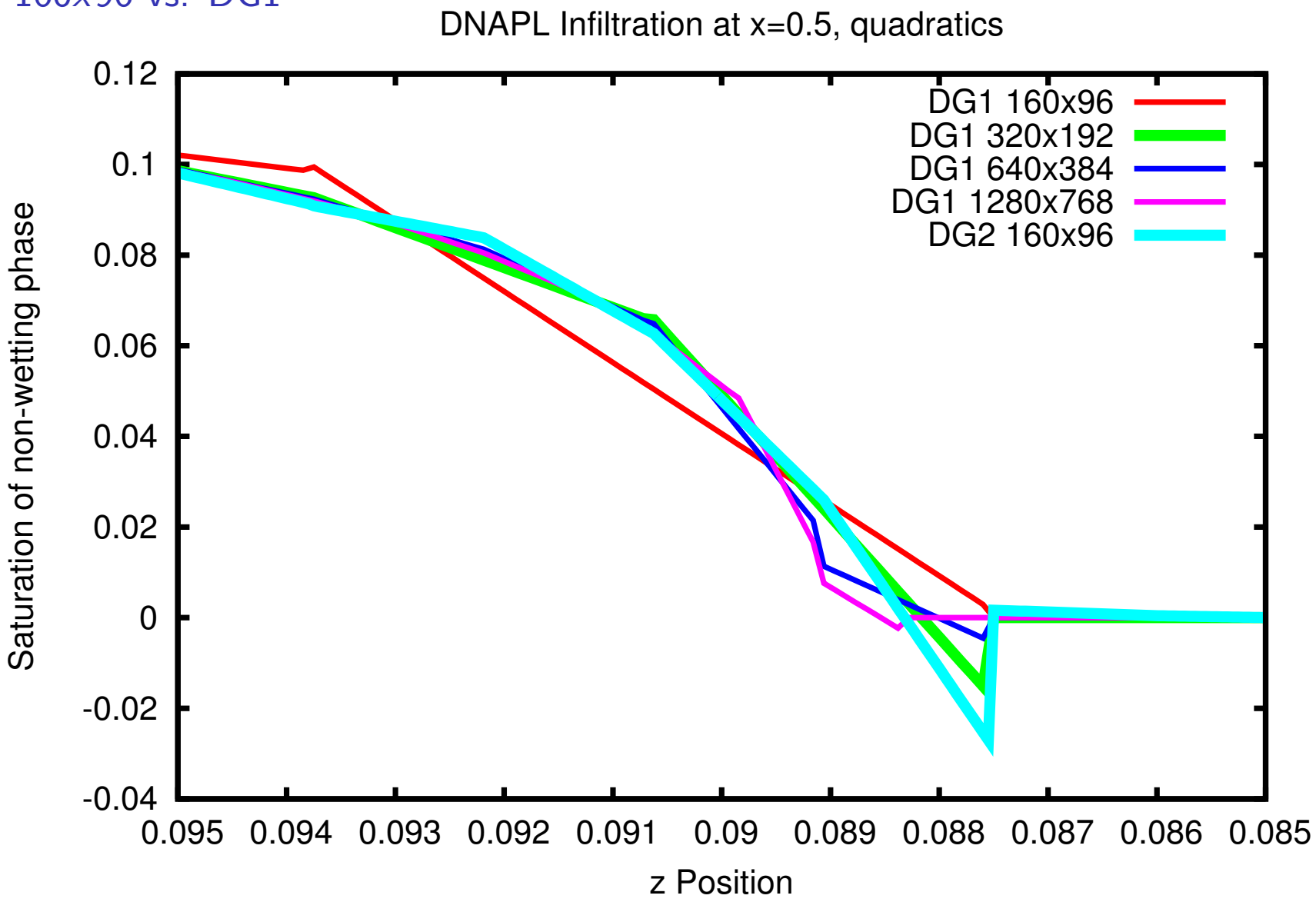
At Free Boundary

DG1 640x384 vs. FV1



At Free Boundary

DG2 160x96 vs. DG1



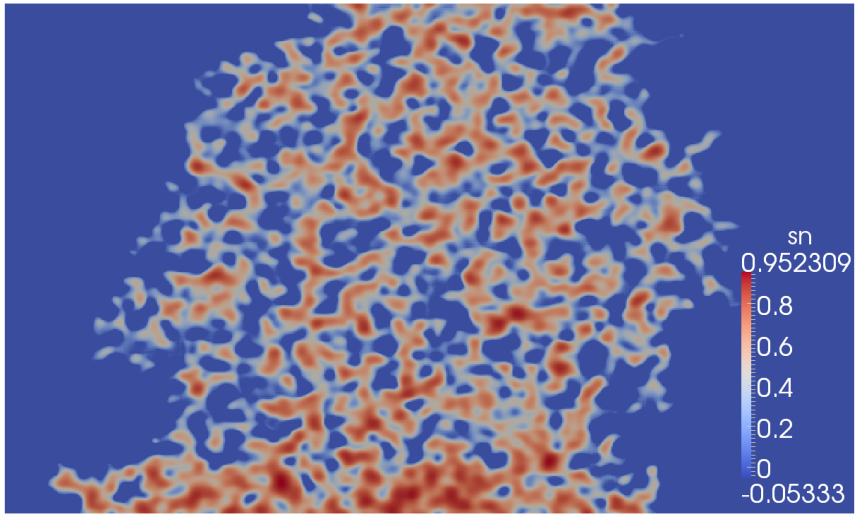
Weak Scalability for DNAPL Infiltration

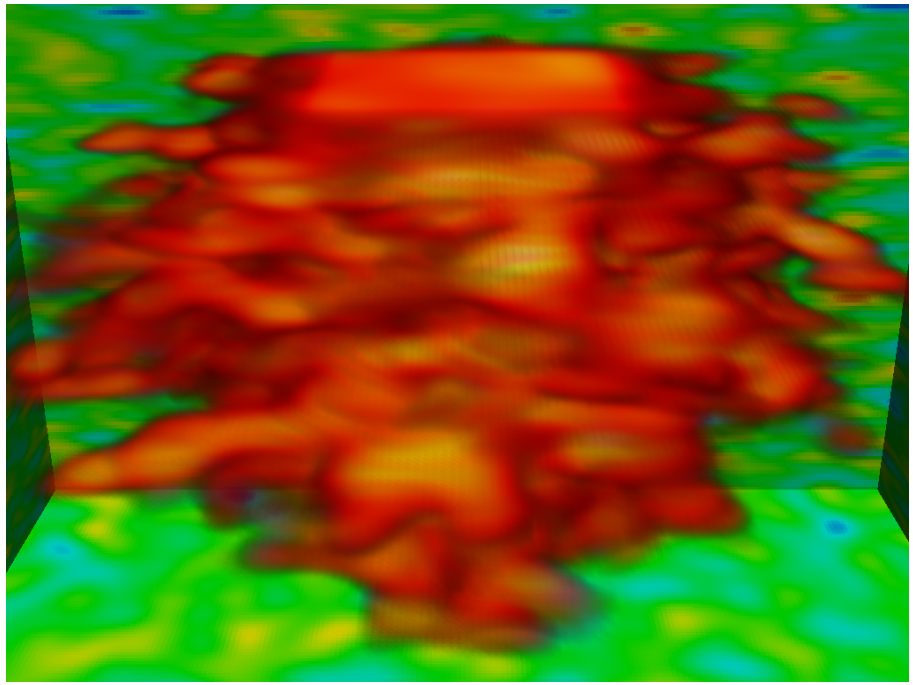
$T = [0, 3600s]$, DG1/Alexander(2) vs. 2nd order FV/Alexander(2)

	P	h^{-1}	DOF	TS	ANL	ALIN	TT	LTIT
DG	1	40	$7.7 \cdot 10^3$	30	11.4	1.8	156.8	0.144
	4	80	$3.1 \cdot 10^4$	71	11.0	3.4	466.1	0.052
	16	160	$1.2 \cdot 10^5$	125	12.0	5.7	1187.3	0.062
	64	320	$4.9 \cdot 10^5$	263	11.4	9.1	3250.5	0.067
	256	640	$2.0 \cdot 10^6$	563	10.7	15.3	11233.0	0.082
FV	1	160	$3.1 \cdot 10^4$	60	7.7	1.8	191.2	0.043
	4	320	$1.2 \cdot 10^5$	120	8.2	2.2	592.1	0.069
	16	640	$4.9 \cdot 10^5$	246	8.4	2.6	1453.7	0.078
	64	1280	$2.0 \cdot 10^6$	491	8.9	3.3	6496.1	0.151
	256	2560	$7.9 \cdot 10^6$	1021	9.7	4.5	12774.9	0.156

Accuracy(DG1) = Accuracy(CCFV) on 2 times refined mesh
Accuracy(DG2) = Accuracy(CCFV) on 3 times refined mesh

Cell-wise Entry Pressure





Problem	P	DOF	TS	ANL	ALIN	TT [h]	ASS/SLV
2D	512	$2 \cdot 10^6$	12019	4.1	2.5	8.3	0.5
3D	1024	$88 \cdot 10^6$	1026	6.0	4.3	107.0	1.4

Contents

Exploiting Tensor Product Structure I

- Tensor product basis: $\phi_i(x) = \phi_{i_1,\ldots,i_d}(x_1,\ldots,x_d) = \theta_{i_d}(x_d)\ldots\theta_{i_1}(x_1)$
- Tensor product quadrature: $\Xi_j = (\xi_{j_1},\ldots,\xi_{j_d})^T$
- $i = (i_1,\ldots,i_d) \in I^d = \{1,..,m\}^d$, $j = (j_1,\ldots,j_d) \in J^d = \{1,..,n\}^d$
- As in spectral finite element methods

Evaluate finite element function on ref elem at quadrature points:

$$
\begin{aligned}
u_h(\Xi_i) &= \sum_{j\in J^d} c_j \phi_j(\Xi_i) && (i \in I^d) \\
&= \sum_{j_d\in J} \ldots \sum_{j_1\in J} \theta_{j_d}(\xi_{i_d})\ldots\theta_{j_1}(\xi_{i_1})\ c_{j_1,\ldots,j_d} && \text{(tensor product structure)} \\
&= \sum_{j_d\in J} \ldots \sum_{j_1\in J} A^{(d)}_{i_d,j_d}\ldots A^{(1)}_{i_1,j_1}\ c_{j_1,\ldots,j_d} && (A^{(k)} \in \mathbb{R}^{m\times n})
\end{aligned}
$$

All basis functions at all quadrature points as matrix-vector product:

$$
y = Ax = (A^{(d)} \otimes \ldots \otimes A^{(1)})\ x, \qquad A_{(i_1,\ldots,i_d),(j_1,\ldots,j_d)} = \Pi_{k=1}^{d} A^{(k)}_{i_k,j_k}
$$

Exploiting Tensor Product Structure II

Elementwise operator evaluation:

$$
\begin{aligned}
r_i = a(u_h, \phi_i) &= \int_T (K(x)\nabla u_h(x)) \cdot \nabla \phi_i(x)\ dx \\
&= \int_{\hat{T}} (K(\hat{x})S(\hat{x})\hat{\nabla}\hat{u}_h(\hat{x})) \cdot S(\hat{x})\hat{\nabla}\hat{\phi}_i(\hat{x})\gamma(\hat{x})\ d\hat{x} \\
&= \sum_{\alpha,\beta=1}^{d} \int_{\hat{T}} \hat{\partial}_\beta \hat{\phi}_i(\hat{x})(S^T K S)_{\alpha,\beta}(\hat{x})\hat{\partial}_\alpha \hat{u}_h(\hat{x}))\gamma(\hat{x})\ d\hat{x} \\
&= \sum_{\alpha,\beta=1}^{d} \sum_{j_d \in J} .. \sum_{j_1 \in J} \underbrace{\tilde{\theta}_{i_d}^{(d,\beta)}(\xi_{j_d}) .. \tilde{\theta}_{i_1}^{(1,\beta)}(\xi_{j_1})}_{\hat{\partial}_\beta \hat{\phi}_i(\hat{\Xi}_j)} \underbrace{\hat{K}_{\alpha,\beta}(\Xi_{j_1..j_d})\hat{\partial}_\alpha \hat{u}_h(\Xi_{j_1..j_d}))\hat{\gamma}(\Xi_{j_1..j_d})}_{X_{j_1..j_d}} \\
&= \sum_{\alpha,\beta=1}^{d} \left(\sum_{j_d \in J} \dots \sum_{j_1 \in J} A_{i_d,j_d}^{(d,\alpha,\beta)} \dots A_{i_1,j_1}^{(1,\alpha,\beta)} \quad X_{j_1,\dots,j_d} \right)
\end{aligned}
$$

For every α, β the same abstract matrix vector product as before!

Sum Factorization I

Extract common factors:

$$
\begin{aligned}
Y_{i_1,\ldots,i_d} &= \sum_{j_d \in J} \cdots \sum_{j_1 \in J} A^{(d)}_{i_d,j_d} \cdots A^{(1)}_{i_1,j_1} X^{(0)}_{j_1,\ldots,j_d} \\
&= \sum_{j_d \in J} \cdots \sum_{j_2 \in J} A^{(d)}_{i_d,j_d} \cdots A^{(2)}_{i_2,j_2} \underbrace{\left(\sum_{j_1 \in J} A^{(1)}_{i_1,j_1} X^{(0)}_{j_1,\ldots,j_d} \right)}_{X^{(1)}_{j_2,\ldots,j_d}} \\
&= \sum_{j_d \in J} \cdots \sum_{j_3 \in J} A^{(d)}_{i_d,j_d} \cdots A^{(3)}_{i_3,j_3} \underbrace{\left(\sum_{j_2 \in J} A^{(2)}_{i_2,j_2} X^{(1)}_{j_2,\ldots,j_d} \right)}_{X^{(2)}_{j_3,\ldots,j_d}} \\
&= \ldots = \sum_{j_d \in J} A^{(d)}_{i_d,j_d} X^{(d-1)}_{j_d}
\end{aligned}
$$

Do this for all $(i_1, \ldots, i_d)$ simultaneously

Sum Factorization II

- Algorithm by *Buis, Dyksen (1996)*
- For $k = 1, \ldots d$ do
  - compute **matrix-matrix product** $X^{(k)} = A^{(k)} X^{(k-1)}$
  - "rotate" $X^{(k)}$ to make j_{k+1} the row index (transposition in 2d)
- Overall complexity: $O(n^{d+1})$ instead of $O(n^{2d})$ $(n = m = q + 1)$!
- Problem: matrices are rather small, e.g. $(4 \times 4) \cdot (4 \times 16)$ for $\mathbb{Q}_3$ in 3d
- Finite element application by e.g. *Melenk, Gerdes, Schwab (2001)*, *Kronbichler, Korman (2012)*:
  - compute $\hat{u}_h$, $\nabla \hat{u}_h$ at quadrature points, $O(n^{d+1})$
  - compute coefficients, geometry factors at quad. points, $O(n^d)$
  - compute residuals $O(n^{d+1})$
- **Matrix free schemes** can be implemented in $O(n^{d+1})$ ops per cell
- Storage is substantially reduced compared to standard implementation
- **DG**: complexity of face terms is $O(dn^d)$ instead of $O(n^{2d-1})$
- But there are $2d$ faces. For all practical n face terms dominate!

Density Driven Flow in Porous Media

Model

$$\nabla \cdot v = 0, \quad v = -(\nabla P - \omega_s \mathbf{1}_z), \quad \partial_t \omega_s + \nabla \cdot \left(v\omega_s - \frac{1}{Ra} \nabla \omega_s \right) = 0$$

- Unstable flow, leads to enhanced mixing
- Smooth solutions (no heterogeneity)

Numerical Scheme

- Operator splitting approach
- Flow: CCFV, two-point flux, AMG solver, RT_0 interpolation
- Transport space: Weighted SIPG-DG (*Di Pietro, Ern, Guerm. 2008*)
- Transport time: 2nd order explicit time stepping
- Gauß-Lobatto tensor product basis
- Underintegration for mass matrix $\rightarrow$ diagonal matrix

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

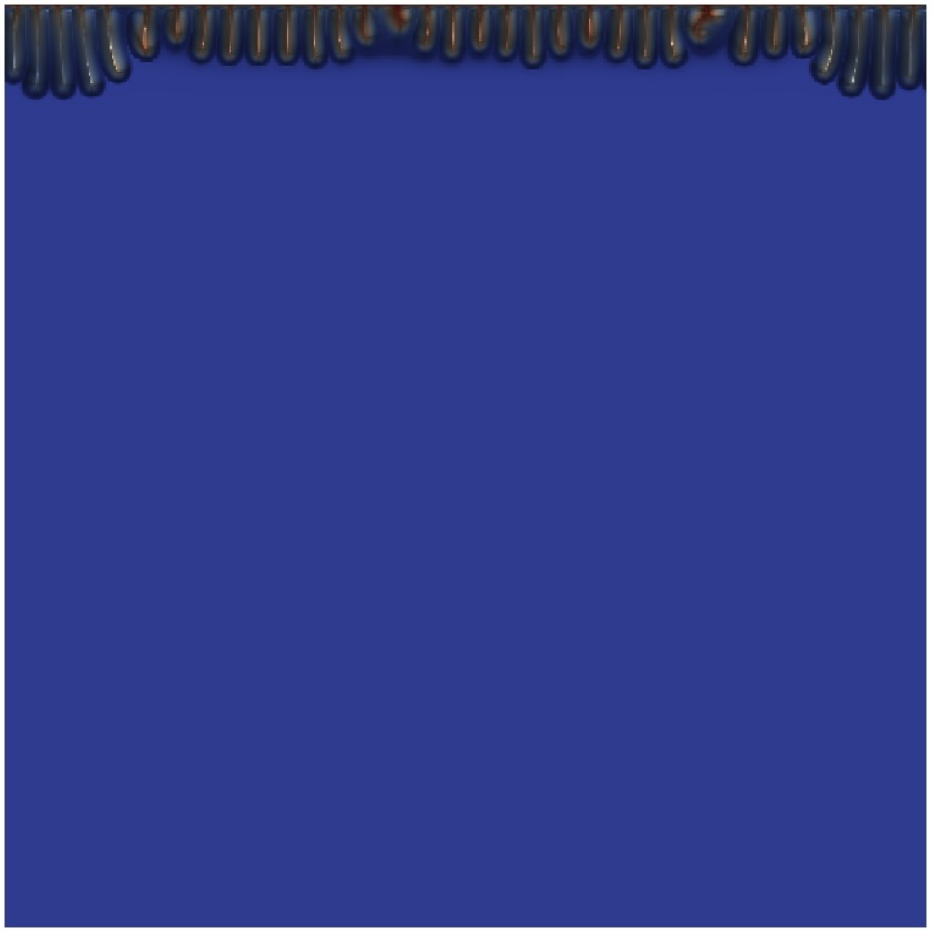
"Long" time simulation, $Ra = 8000$

25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$

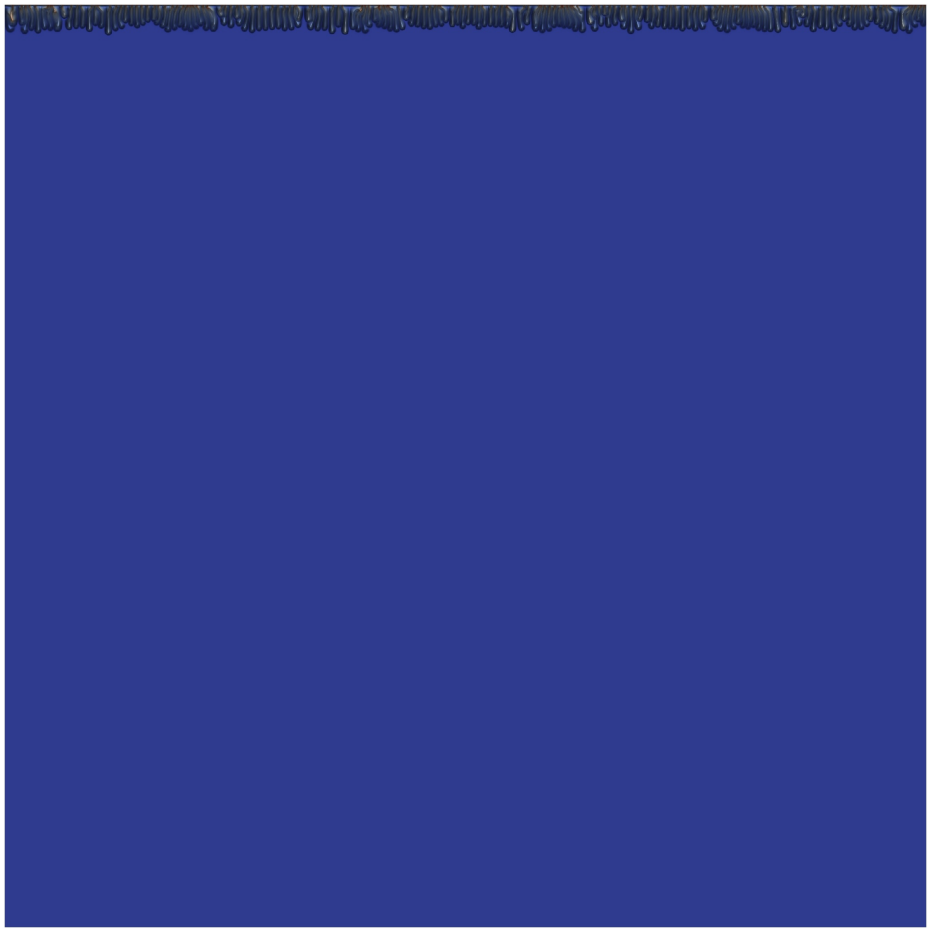


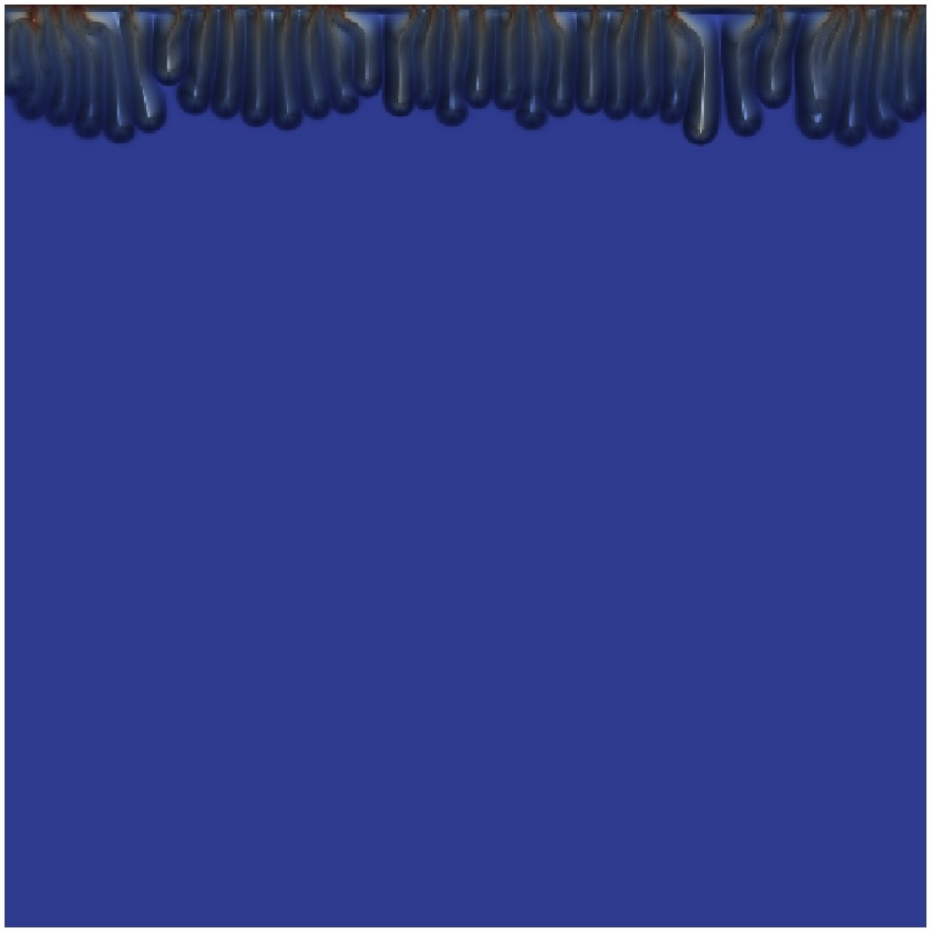
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



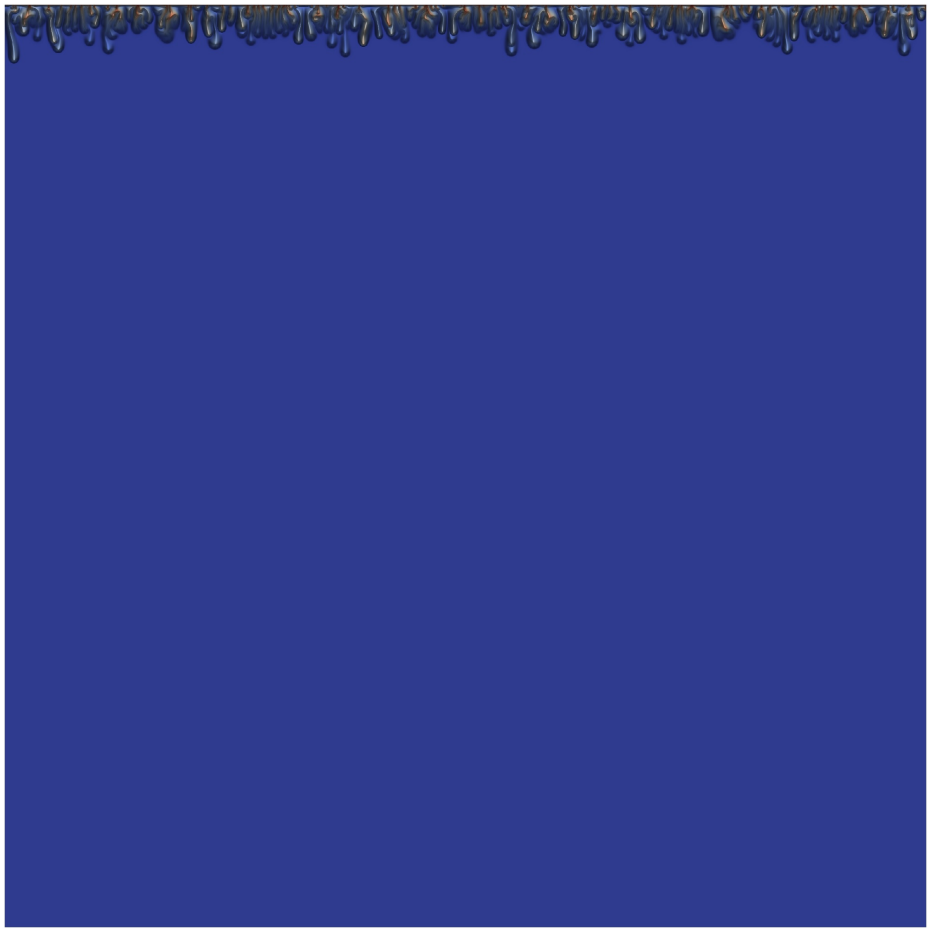


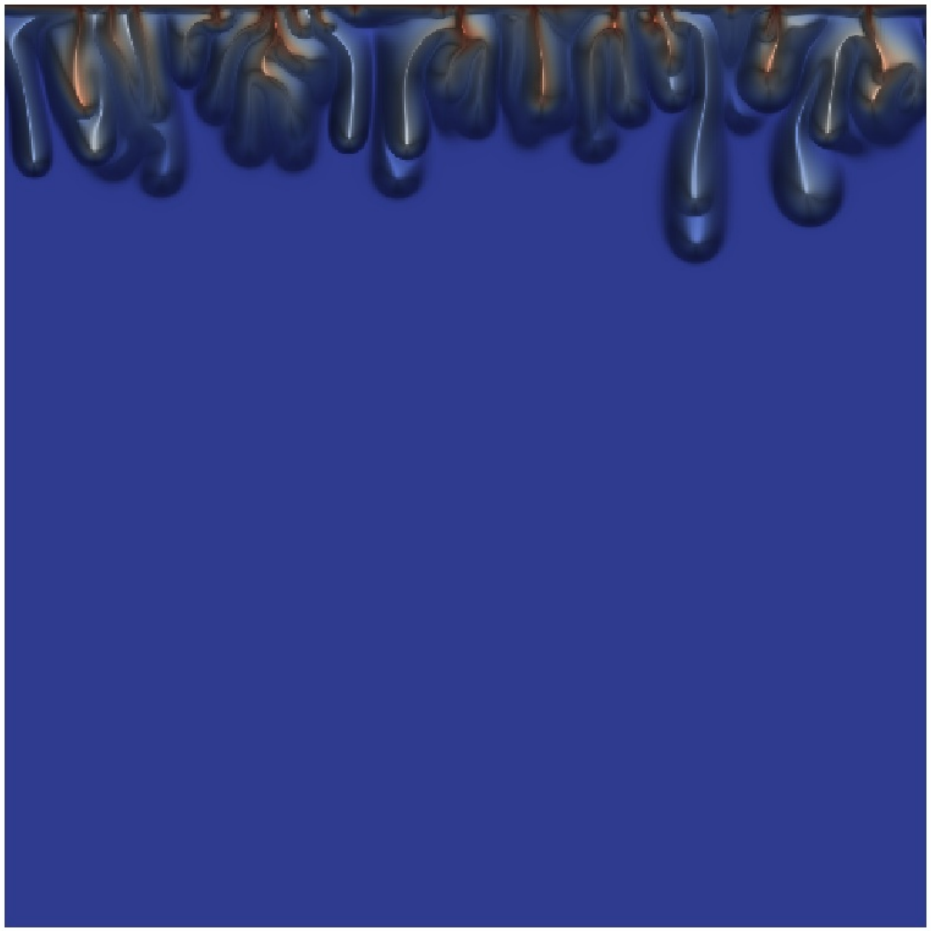
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



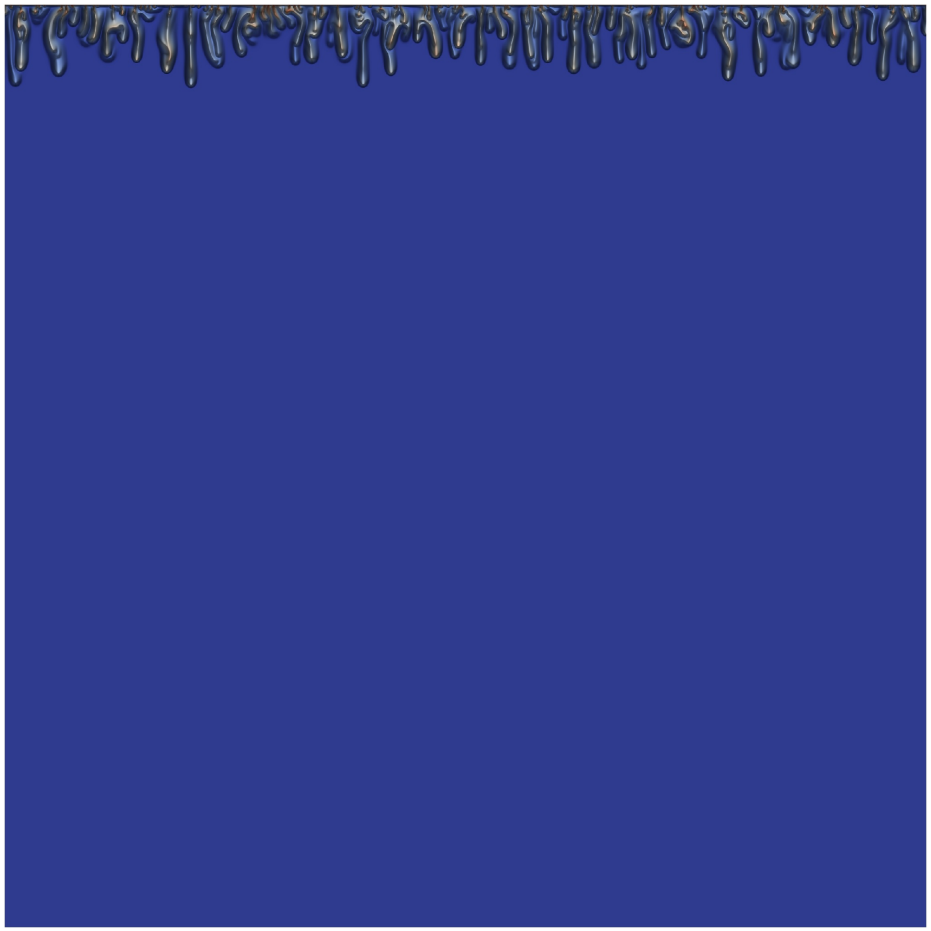


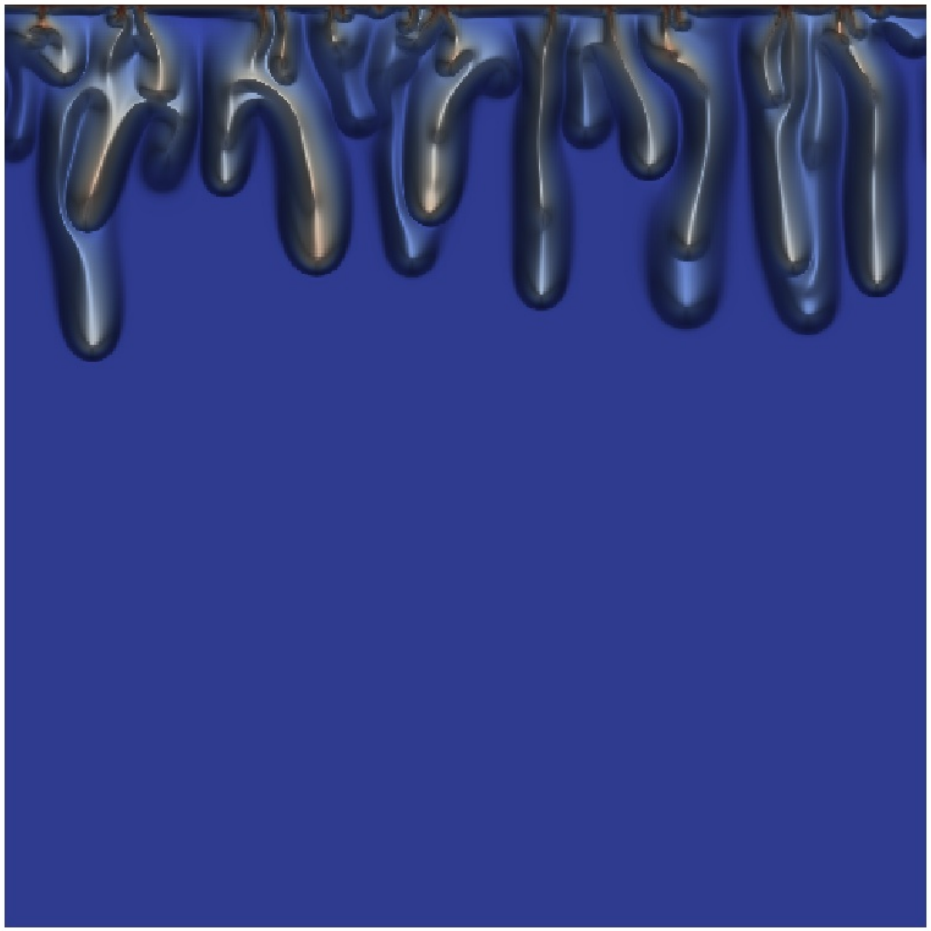
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



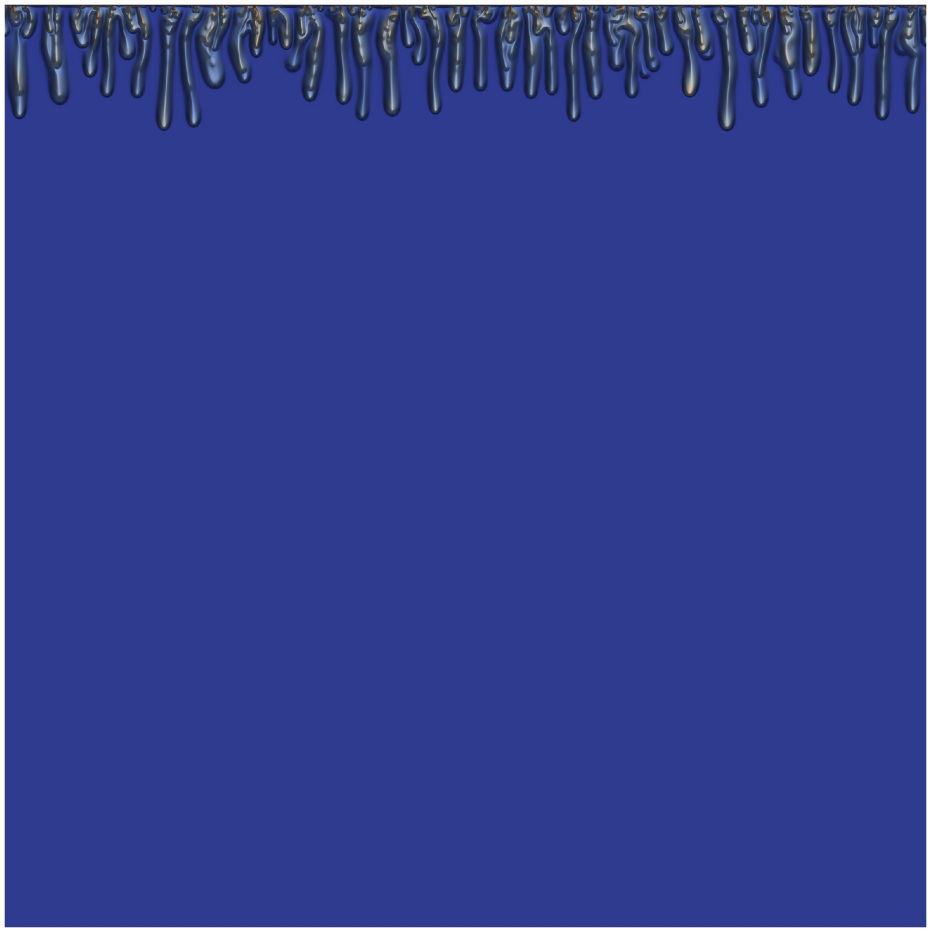


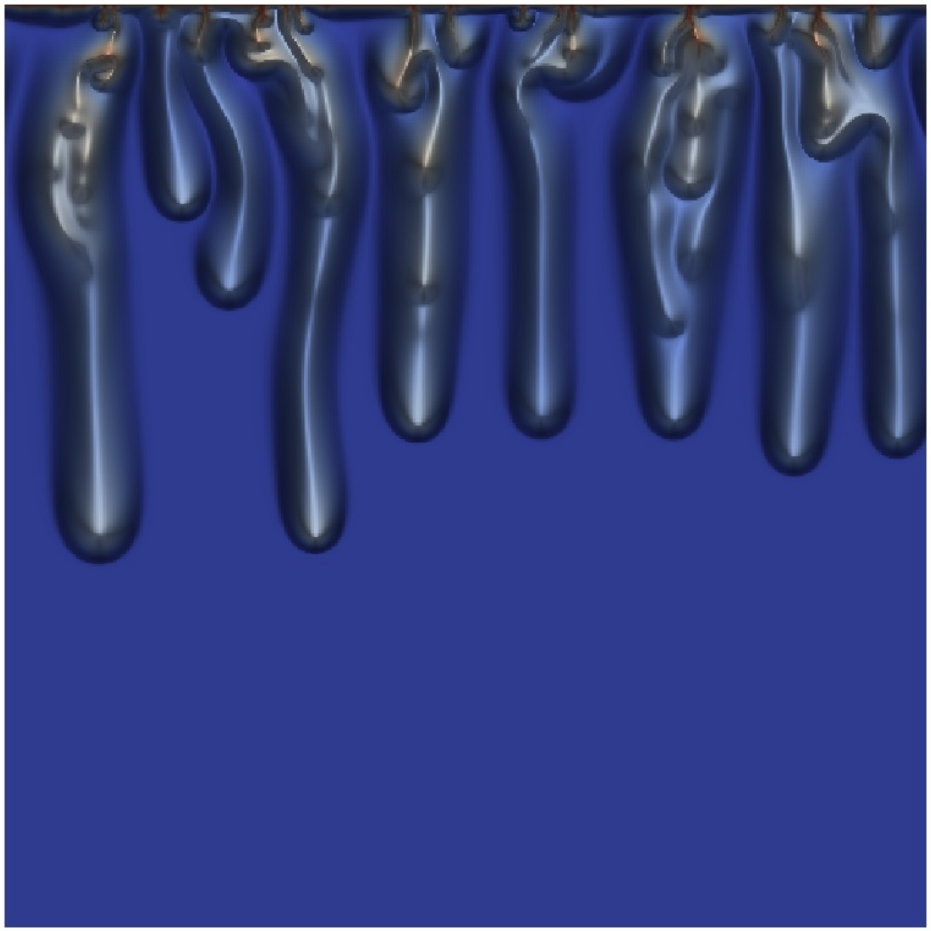
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



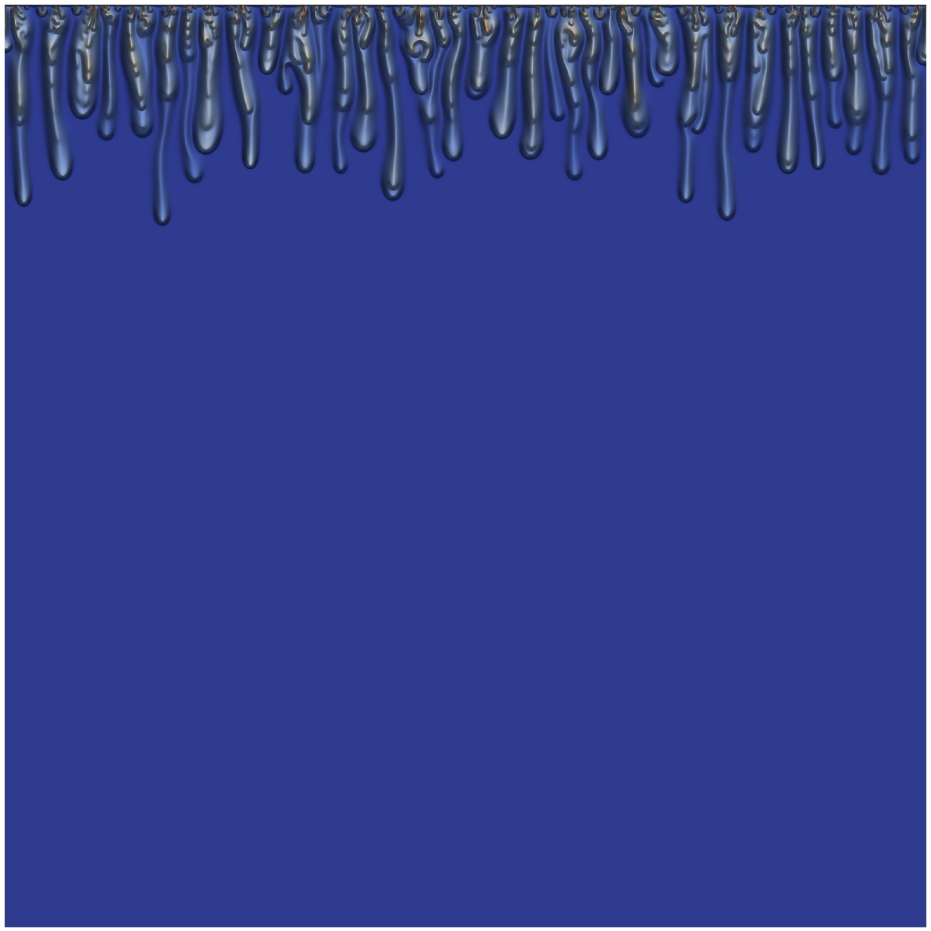


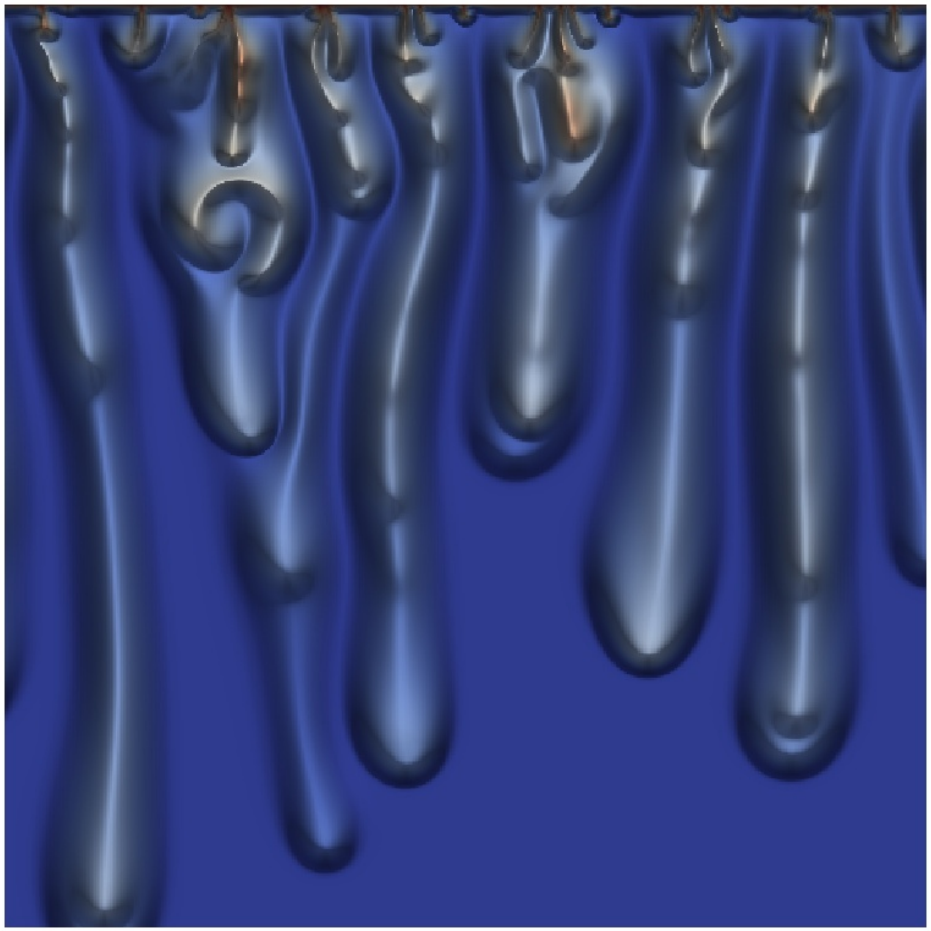
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



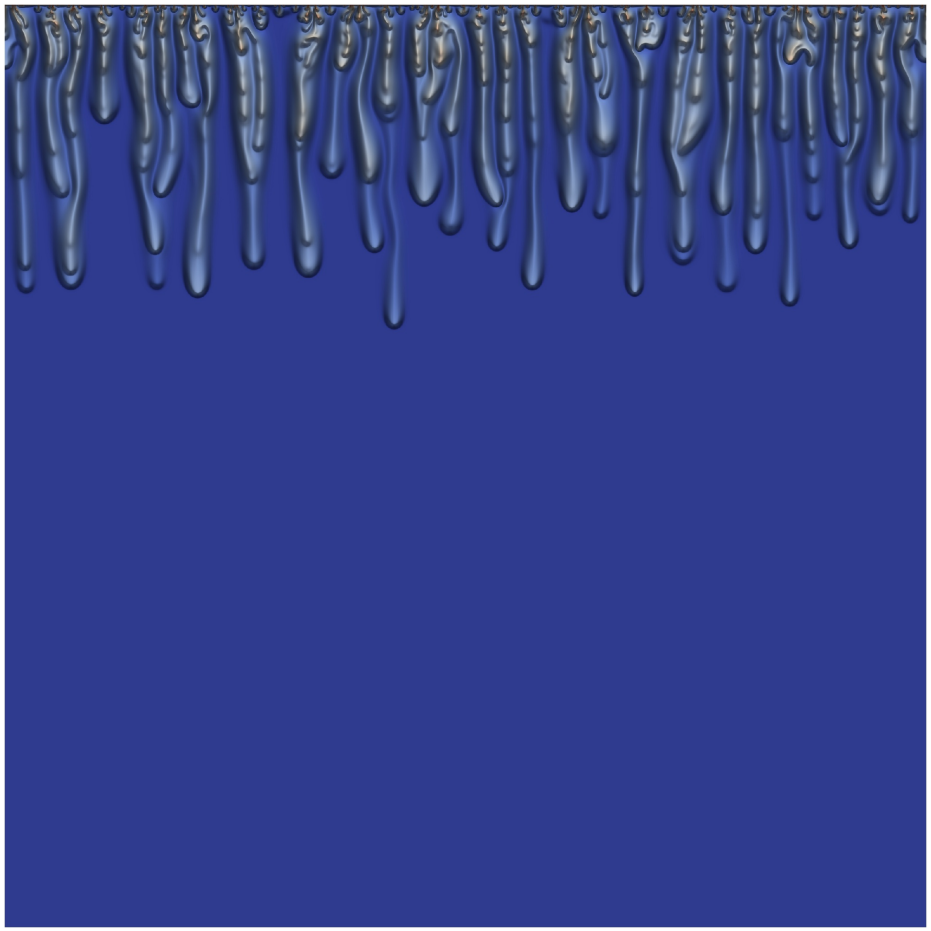


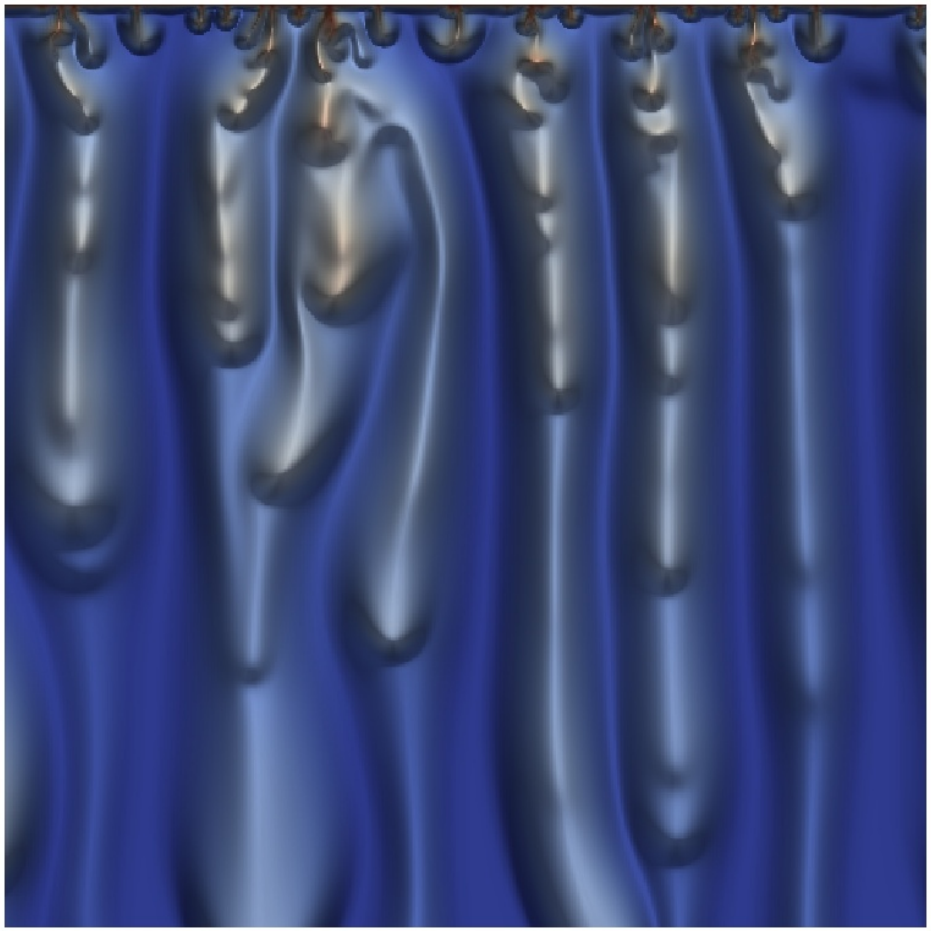
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



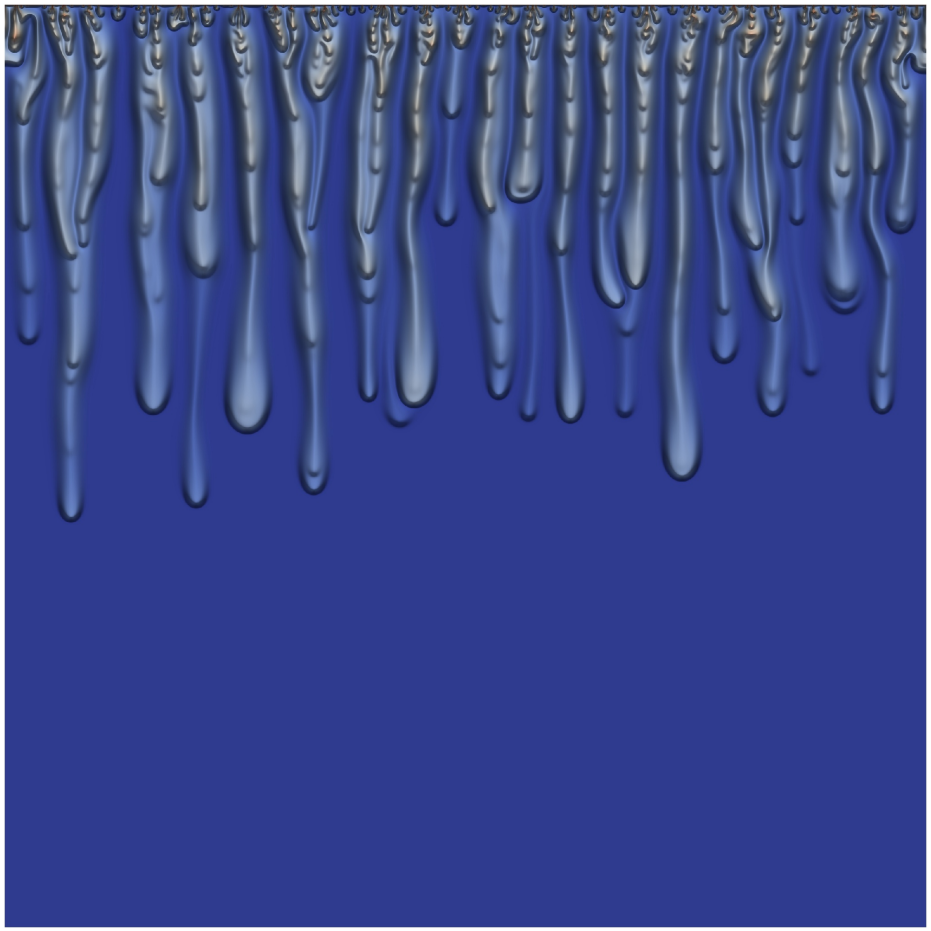


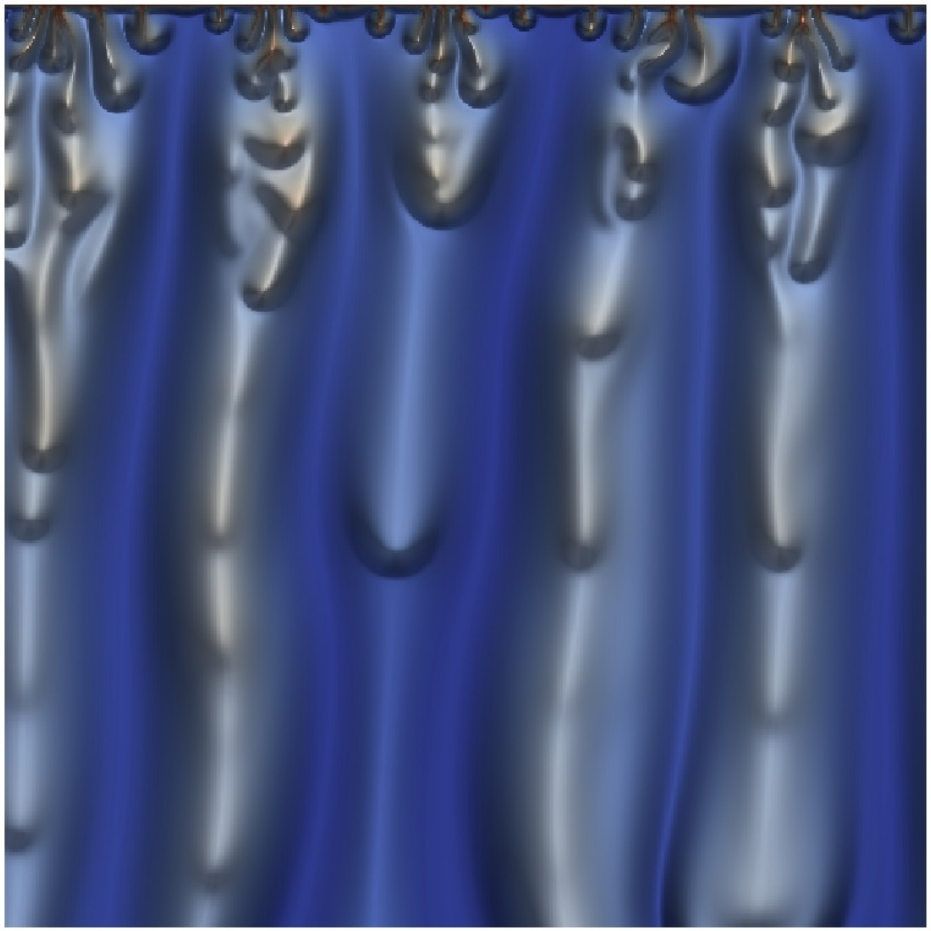
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



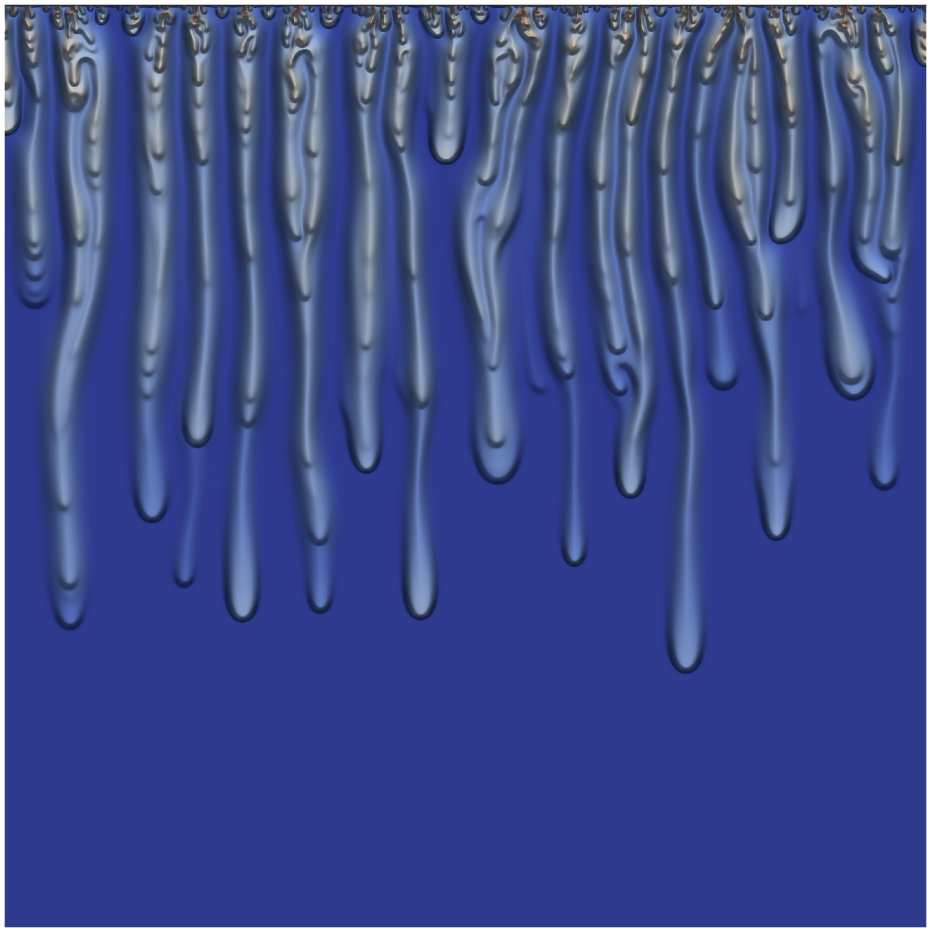


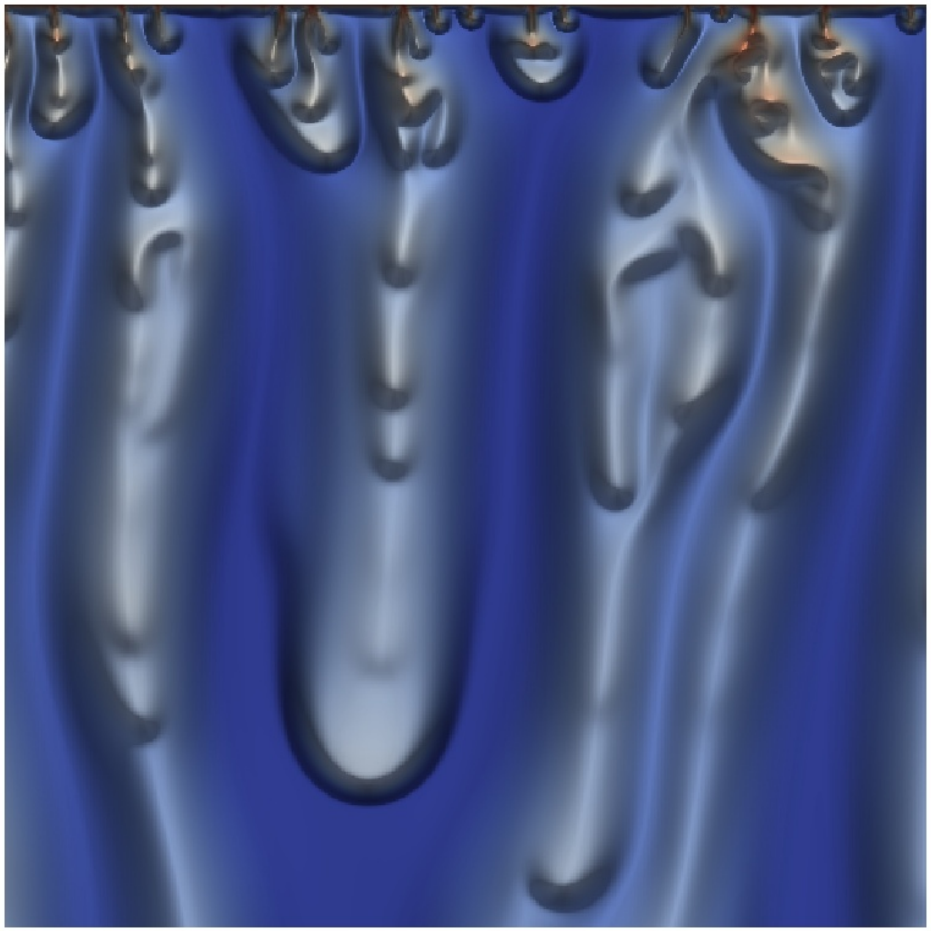
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



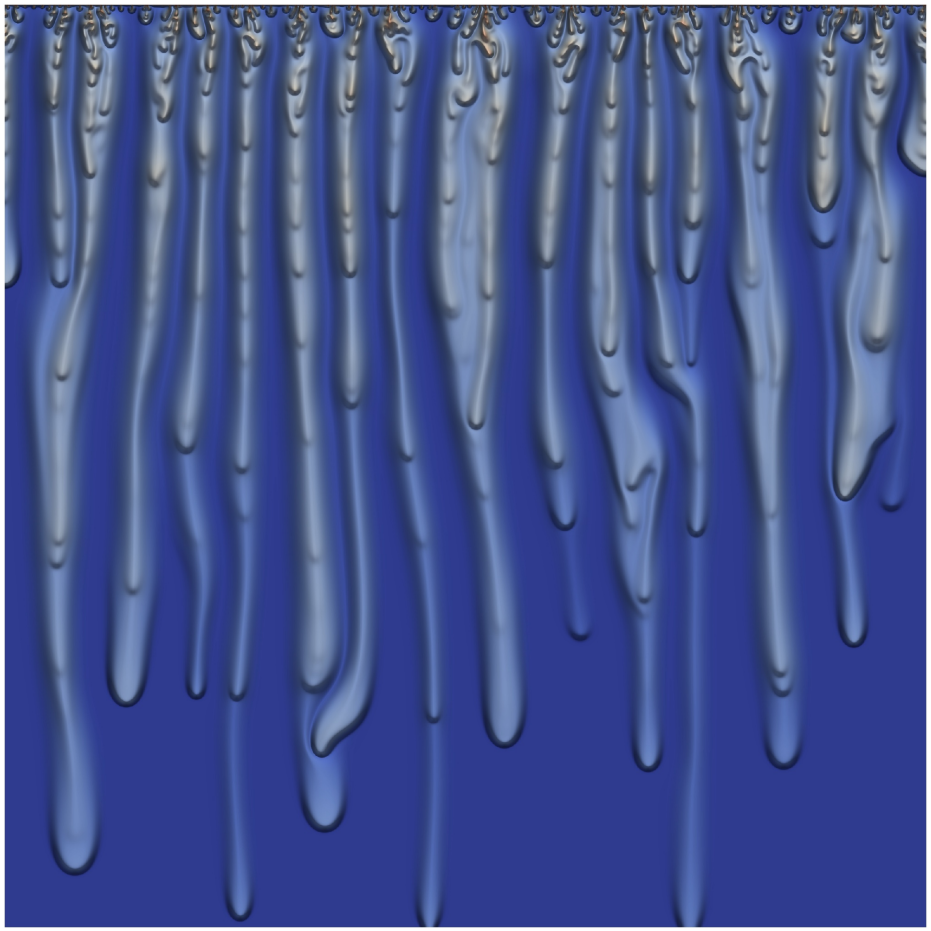


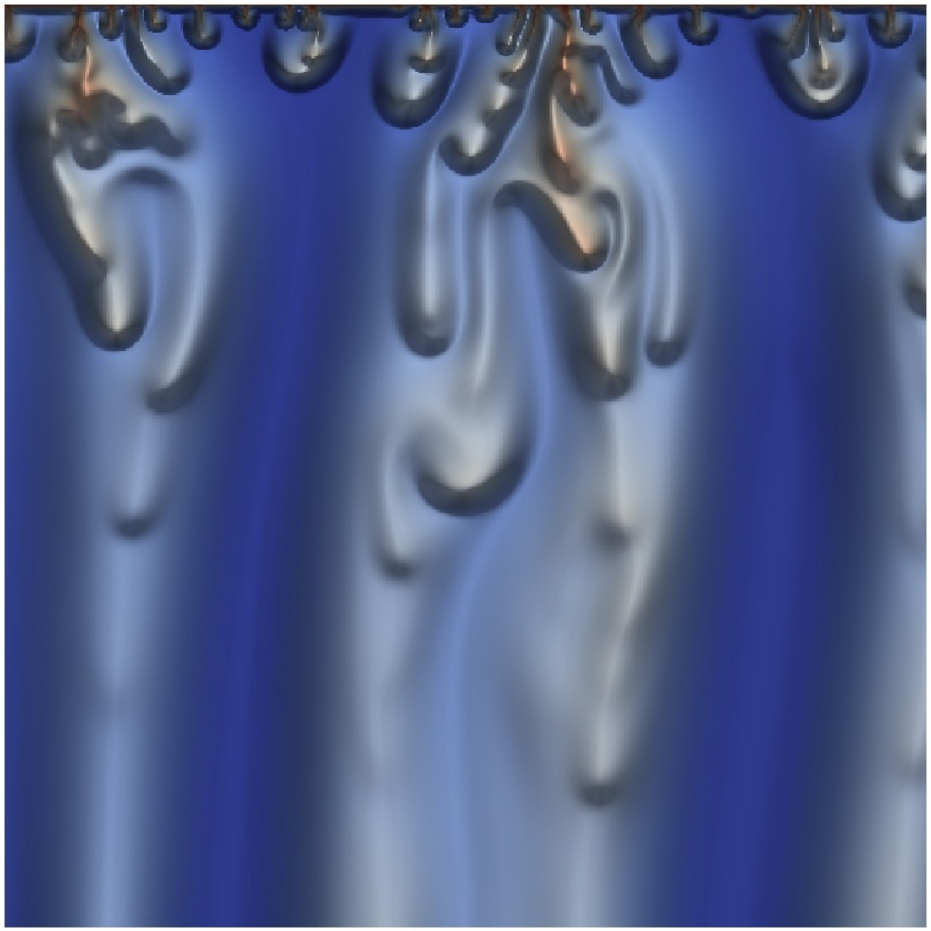
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



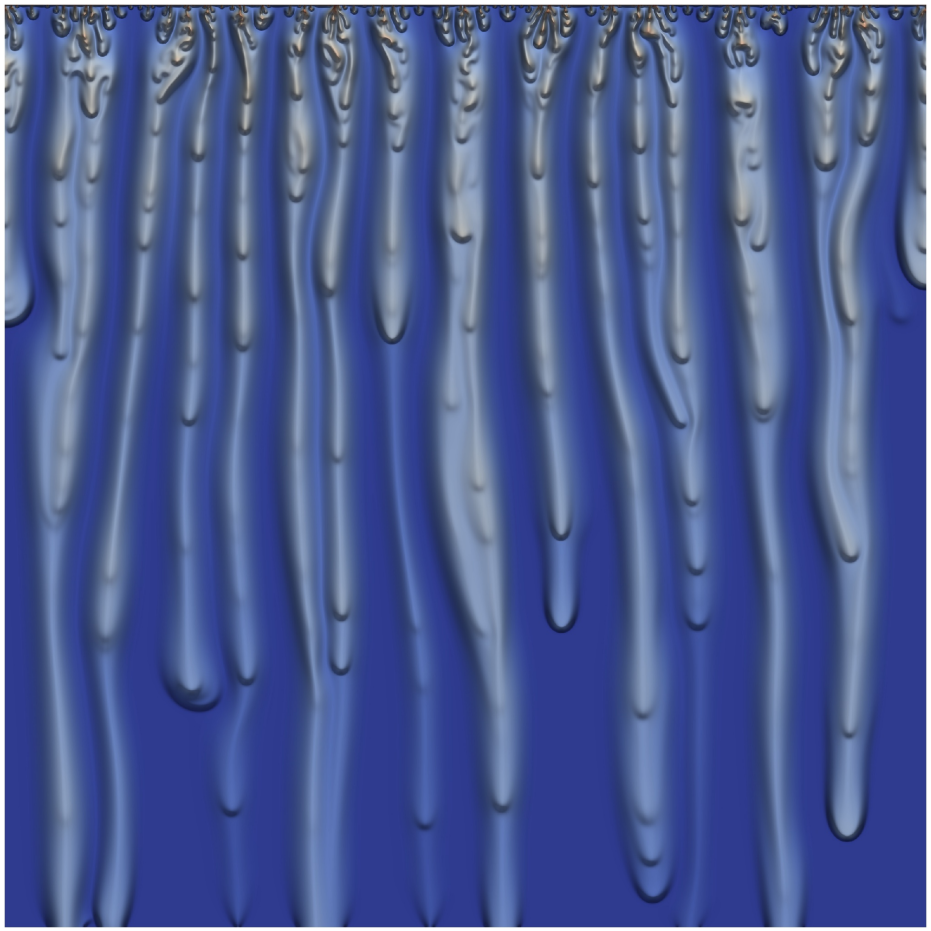


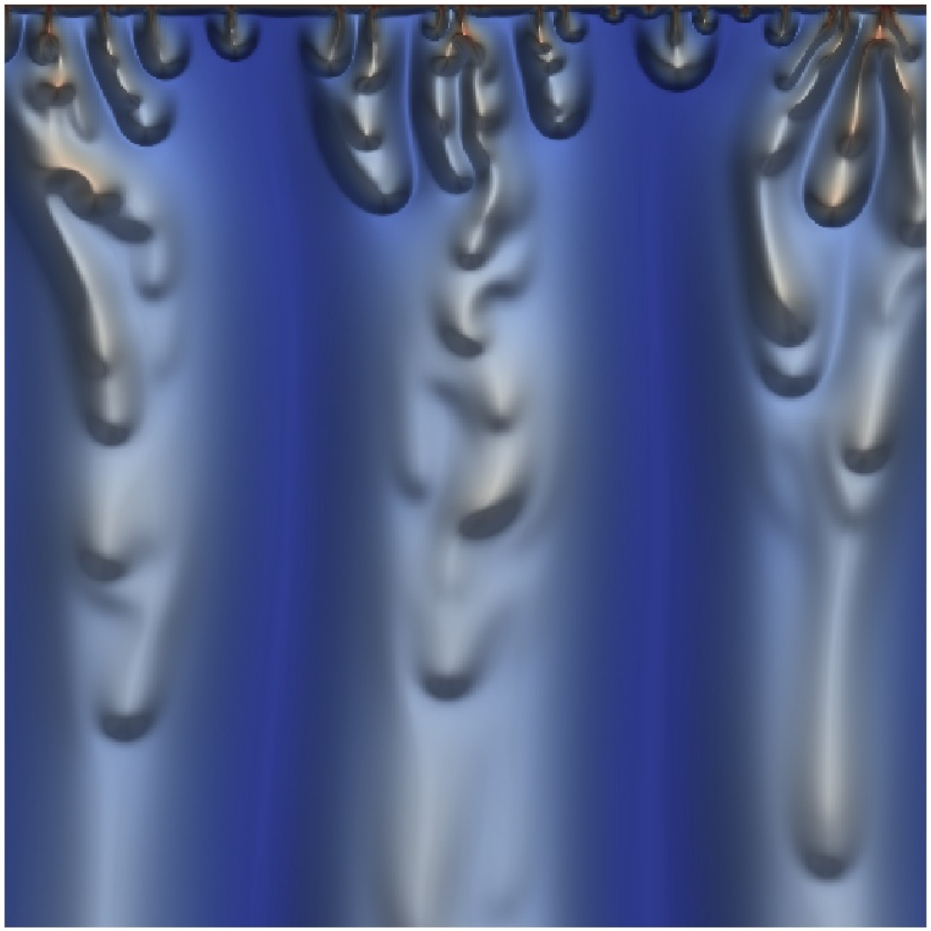
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$



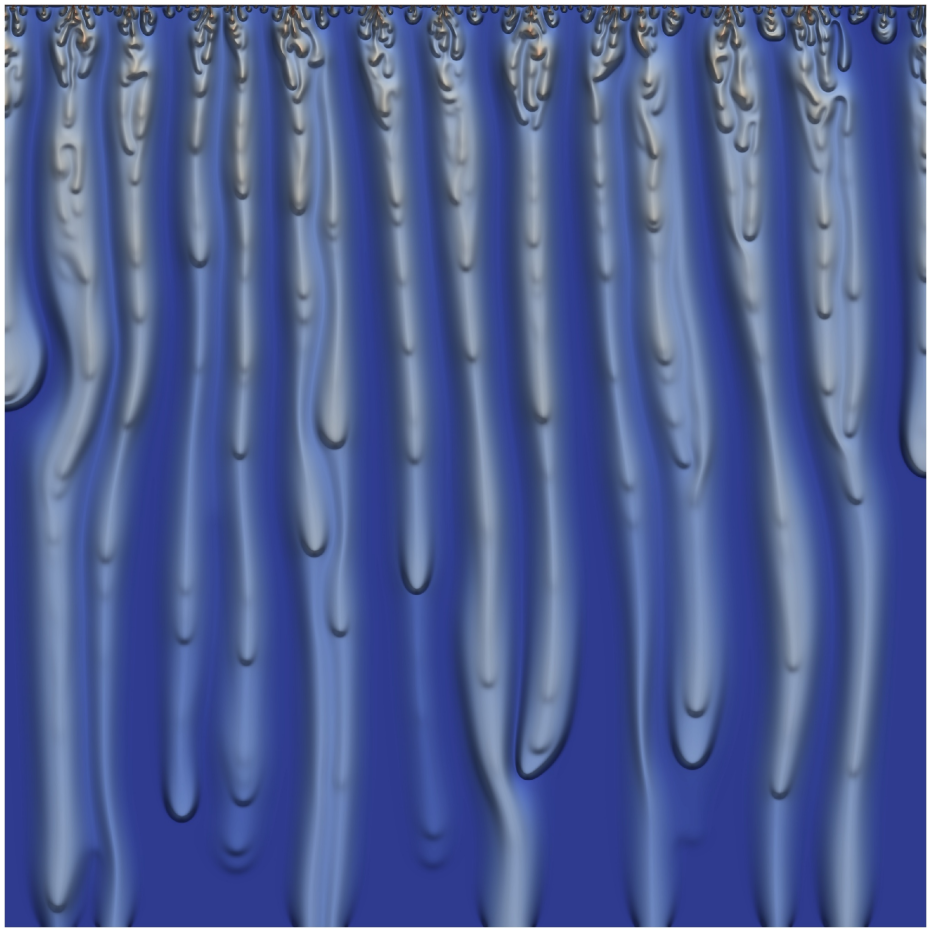


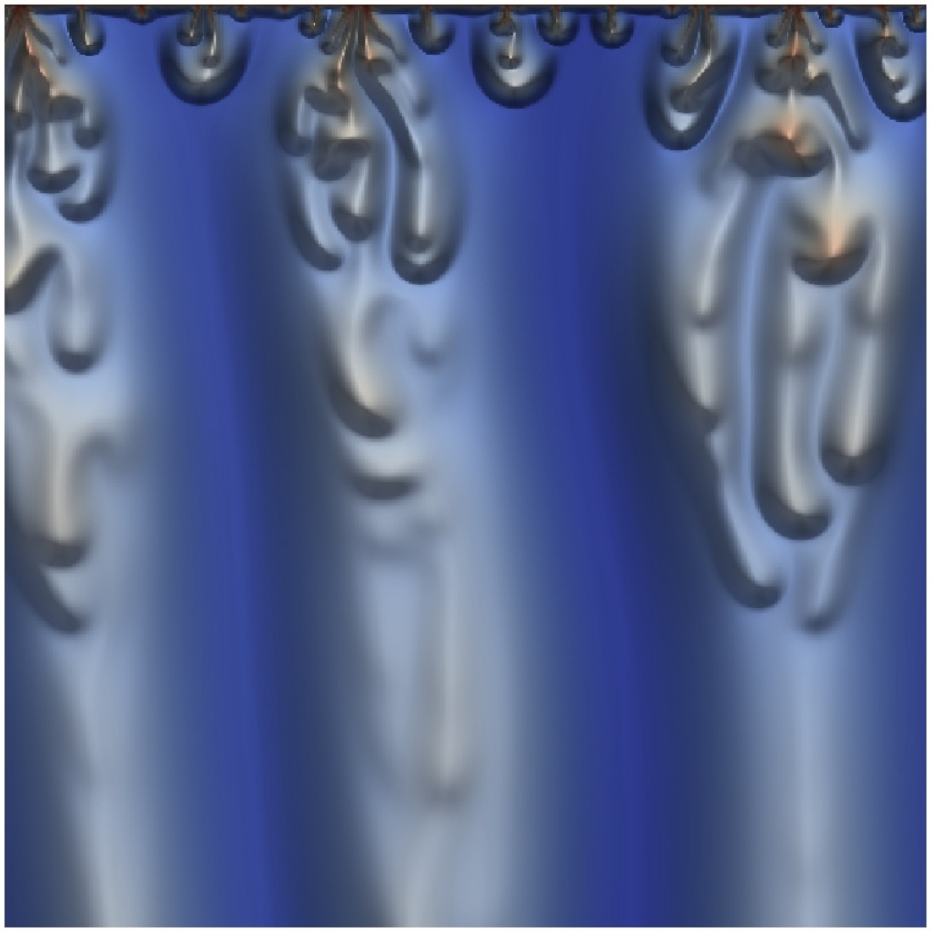
25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

Unstable small fingers feed relatively stable long fingers

2d Density Driven Flow $\Omega = (0,5)^2$, $T = 60$

"Long" time simulation, $Ra = 8000$

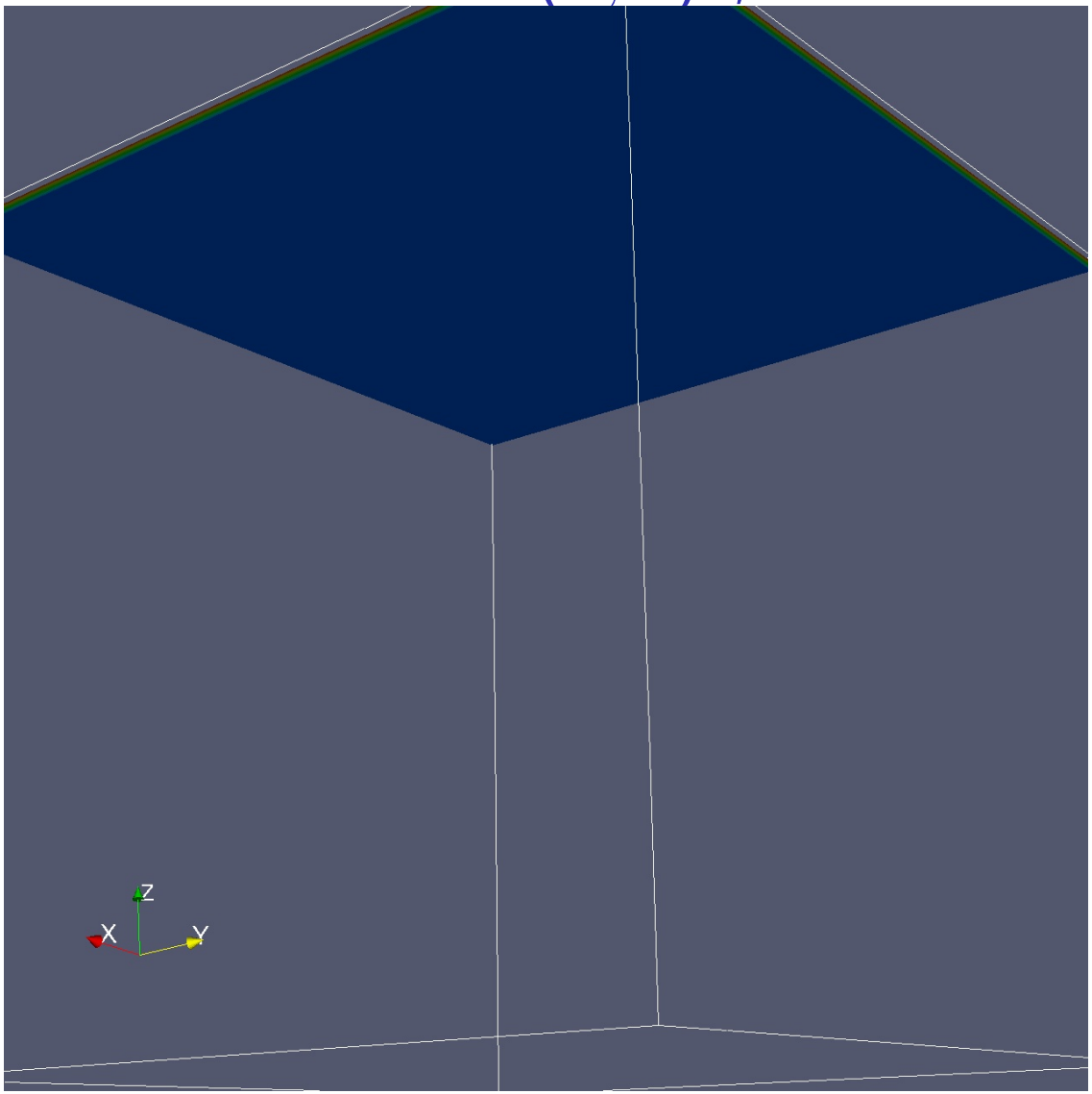




25 cores of Xeon E5-2680v2, 1200^2, Q_2, $1.3 \cdot 10^7$ DOF

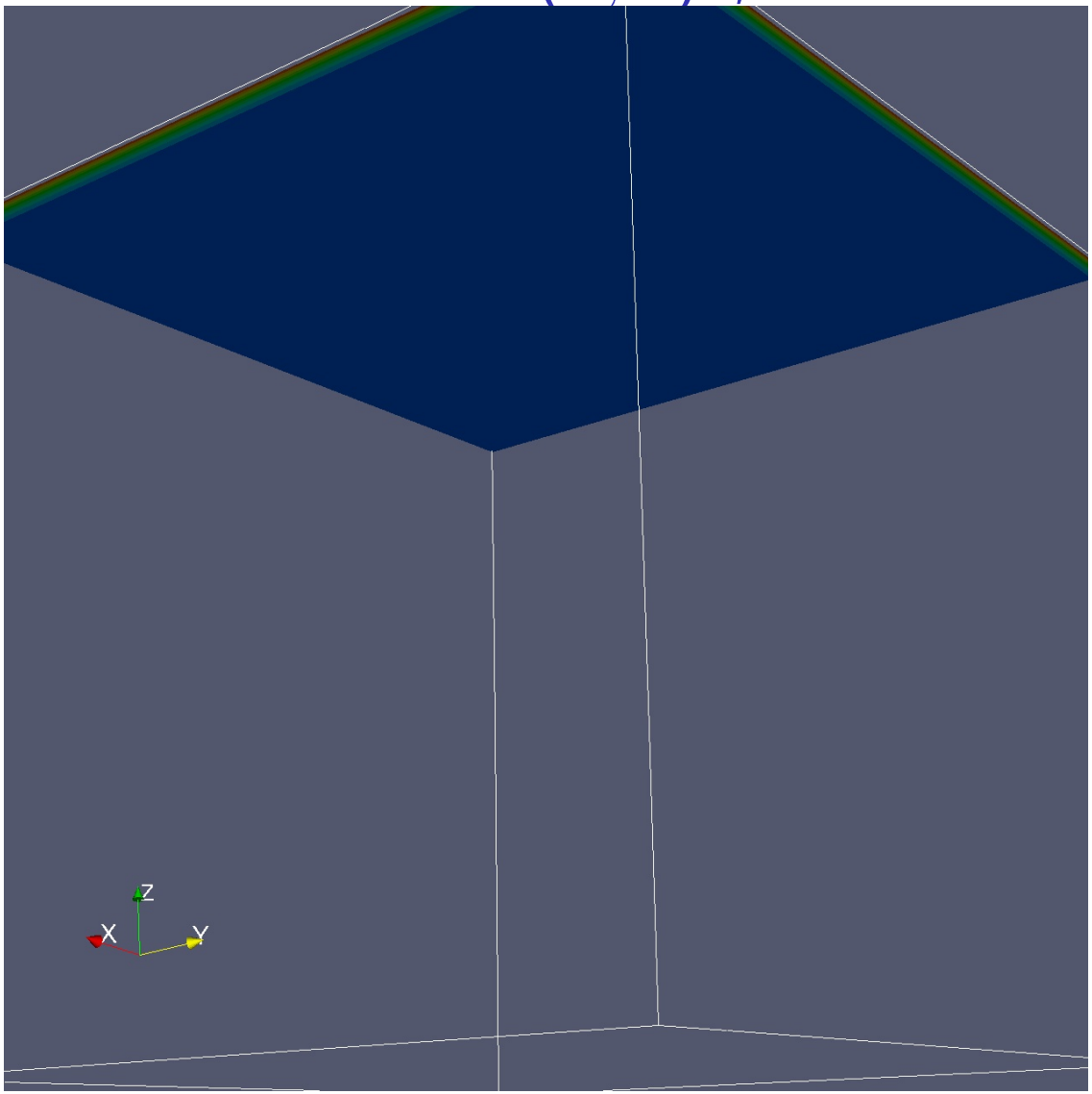
Unstable small fingers feed relatively stable long fingers

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



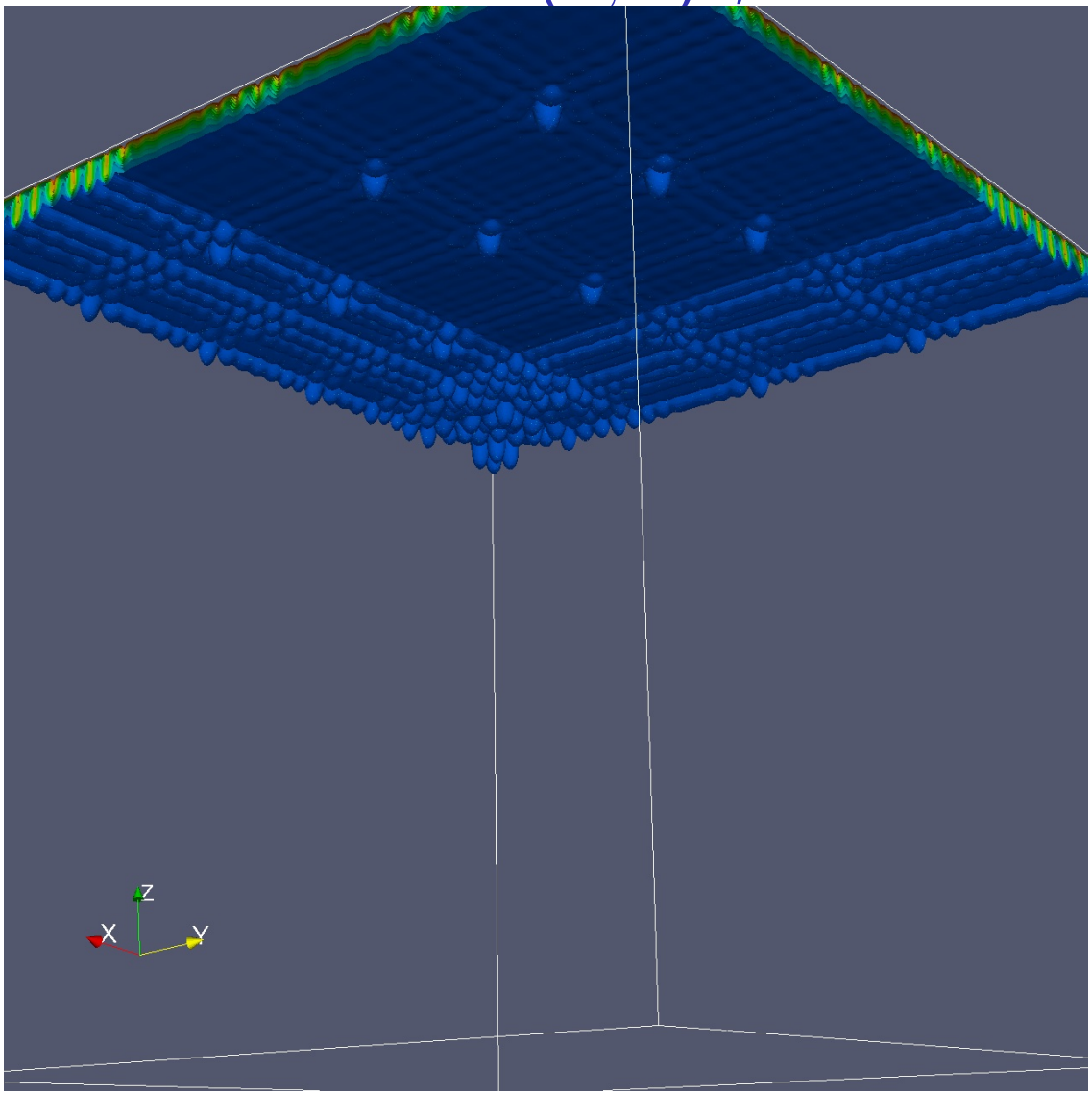
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



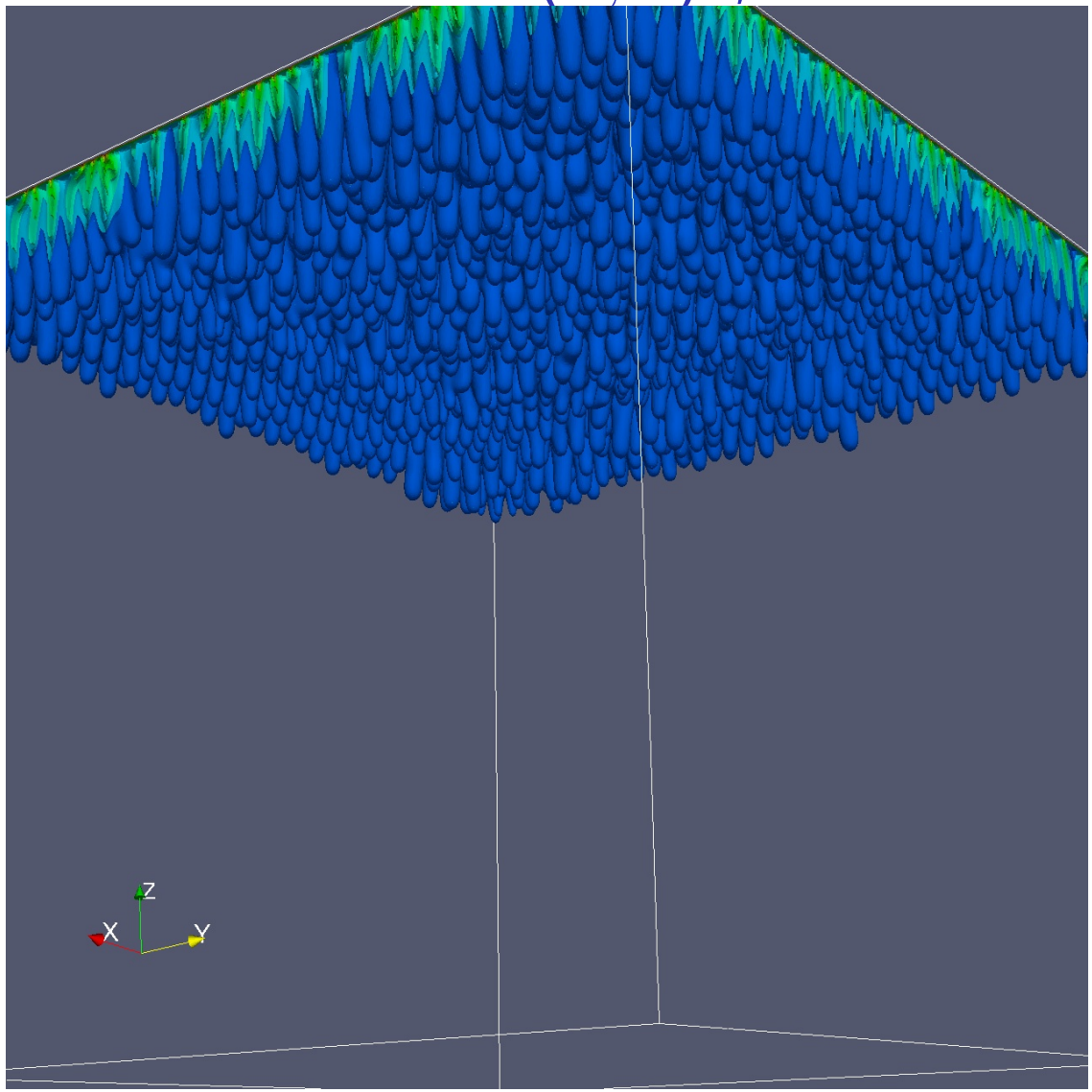
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



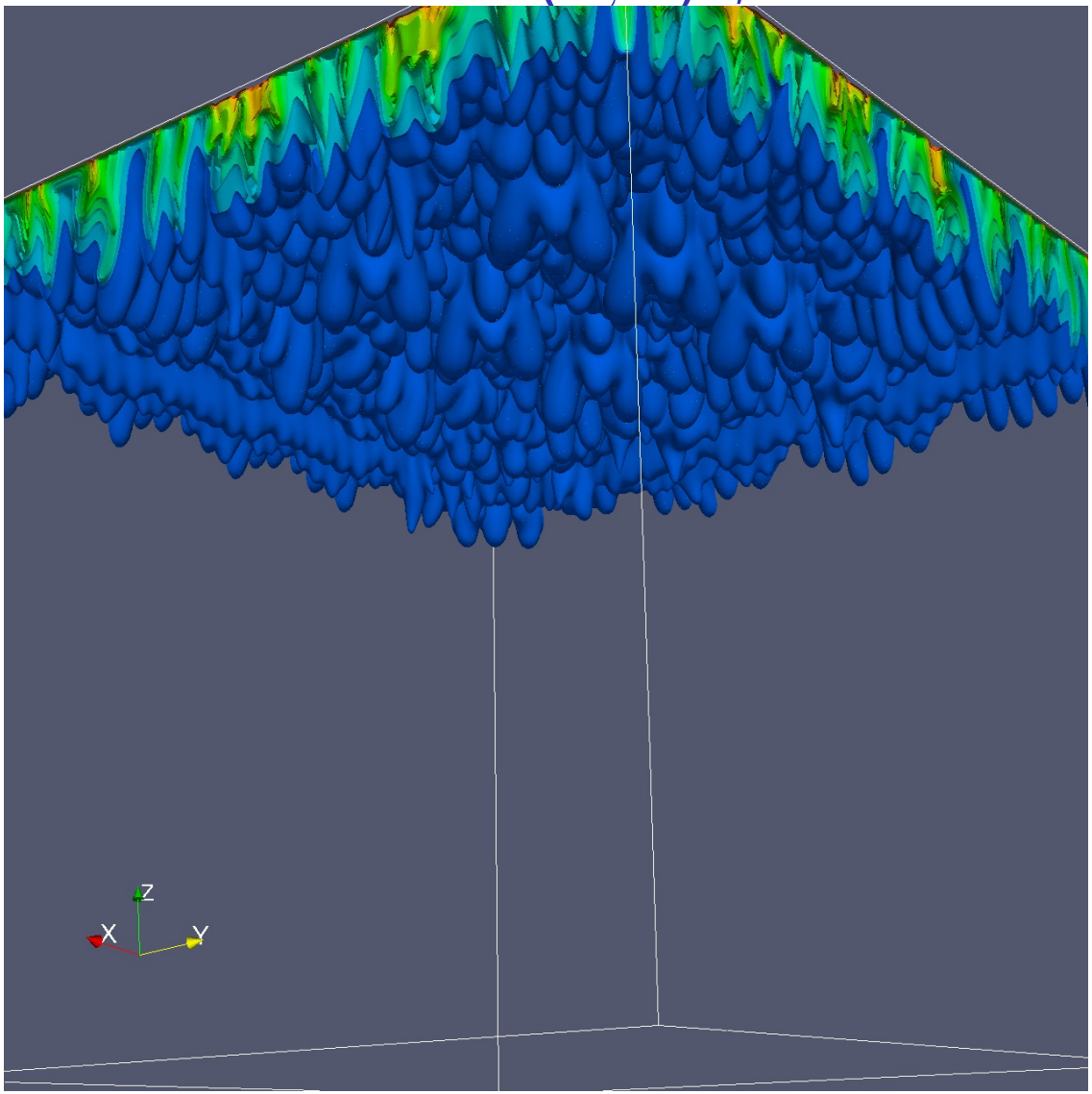
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



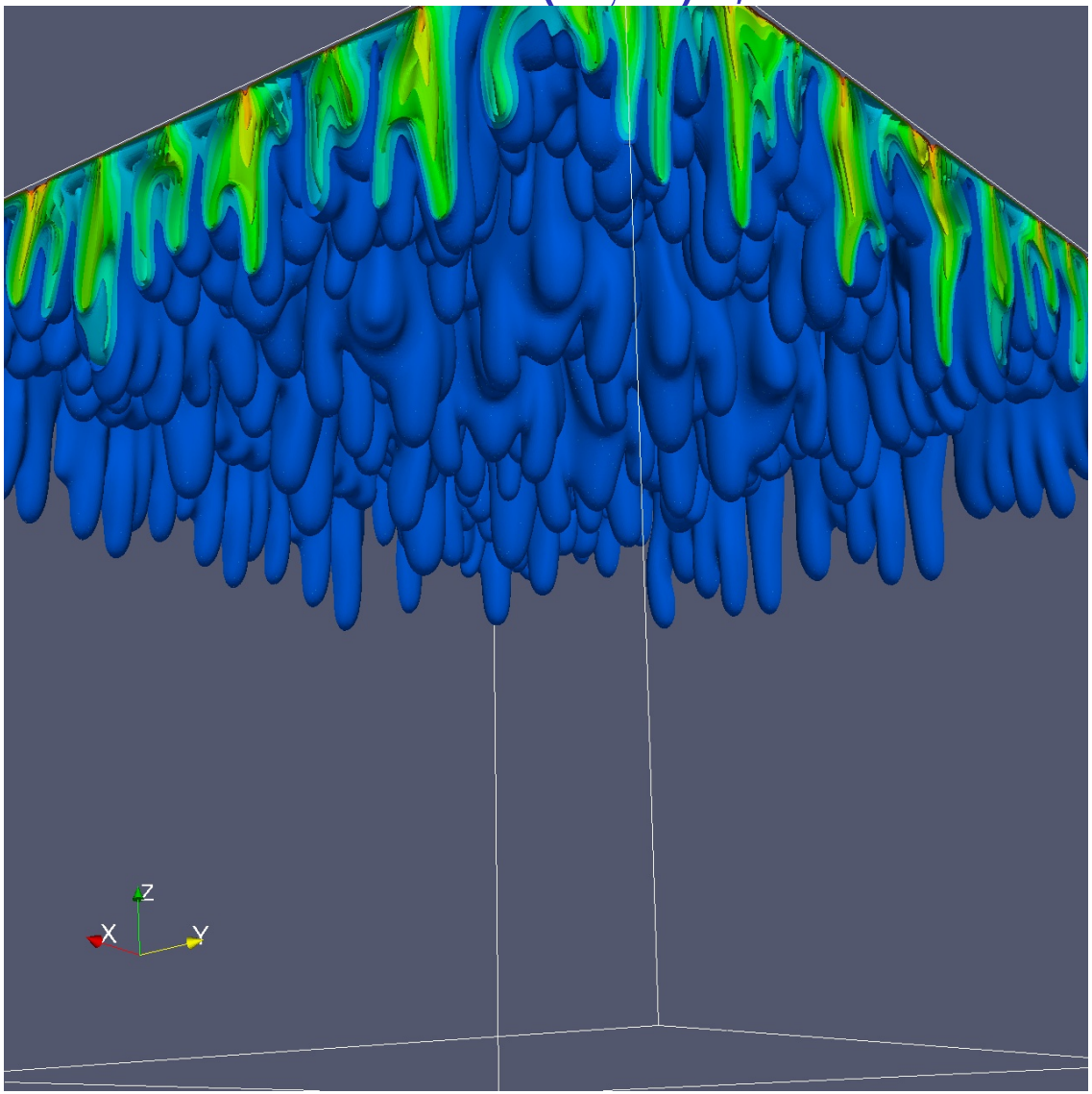
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



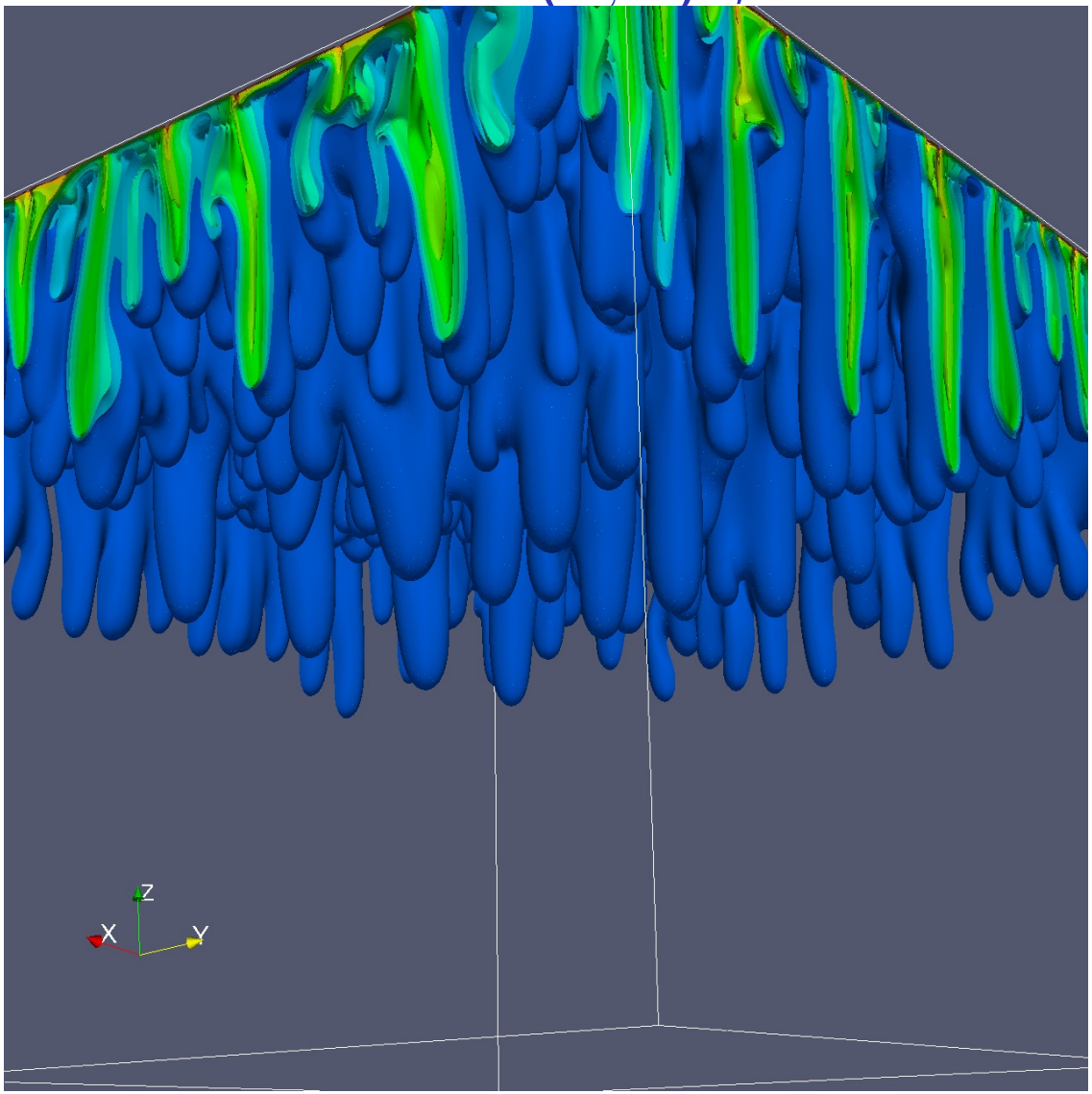
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



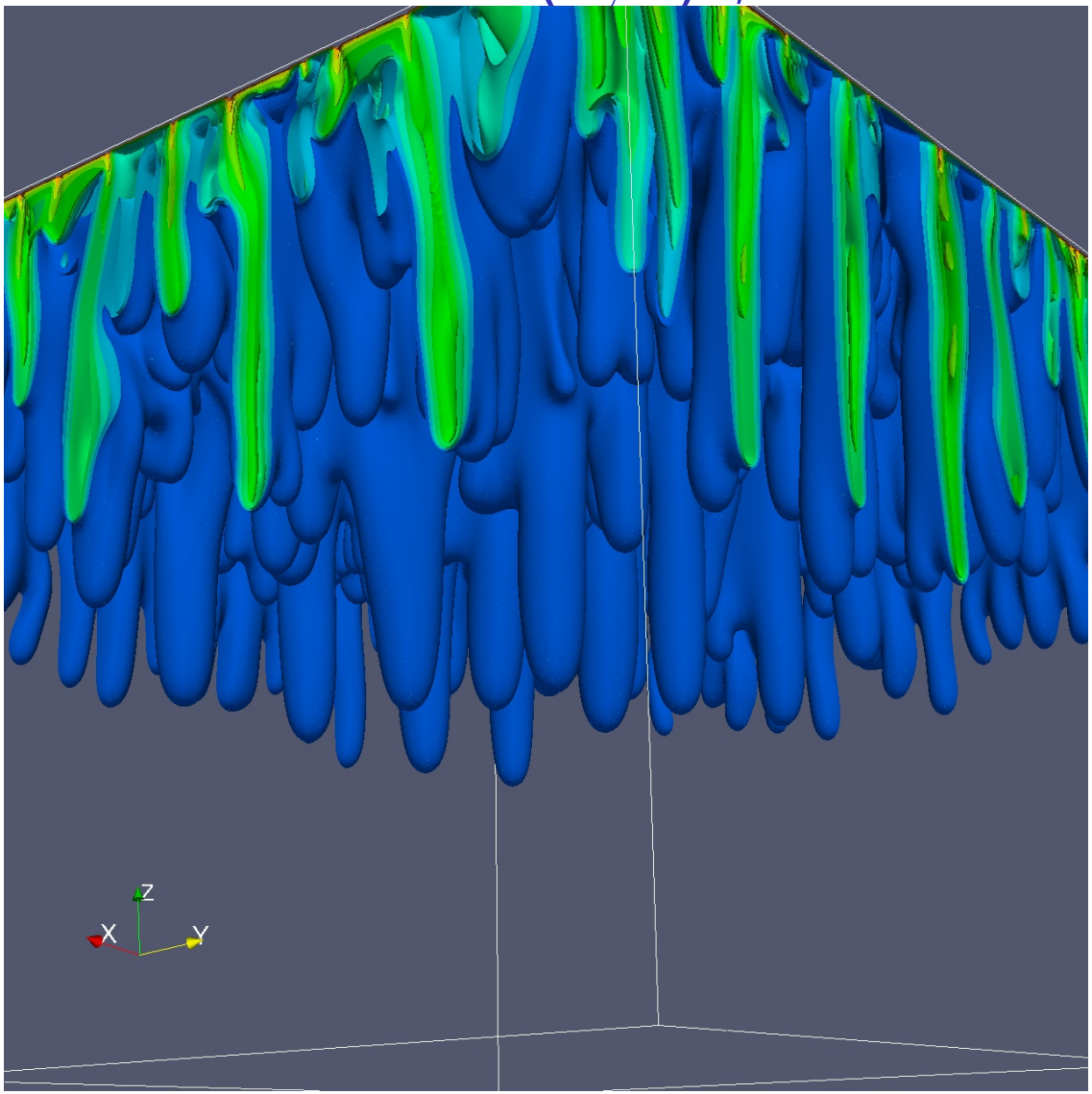
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



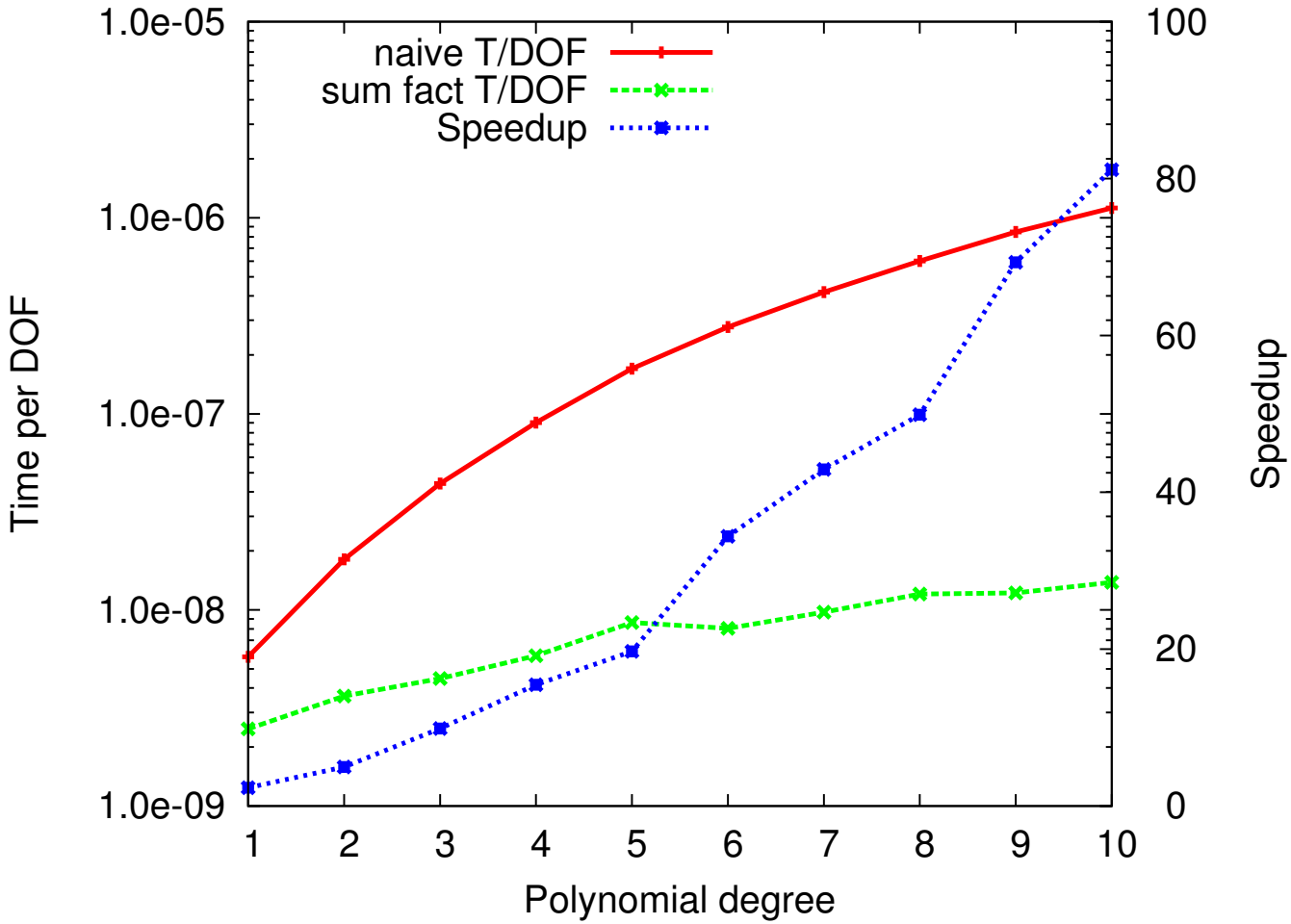
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

3d Density Driven Flow $\Omega = (0,1)^3$, $T = 5$



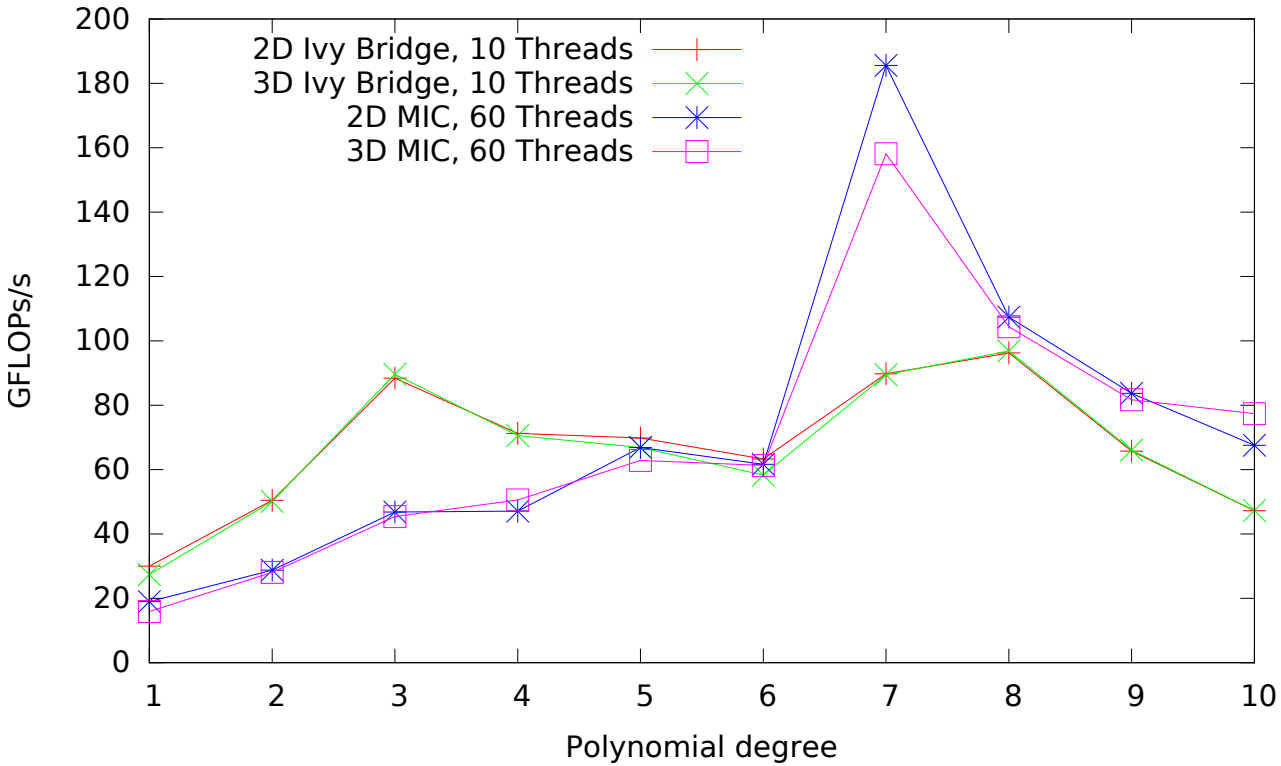
$8 \times$ Xeon E5-2680v2 10 core, 240^3, Q_2, $3.7 \cdot 10^8$ DOF, 16000 steps, 64h.

Evaluation of u_h at Quadrature Points (3D)



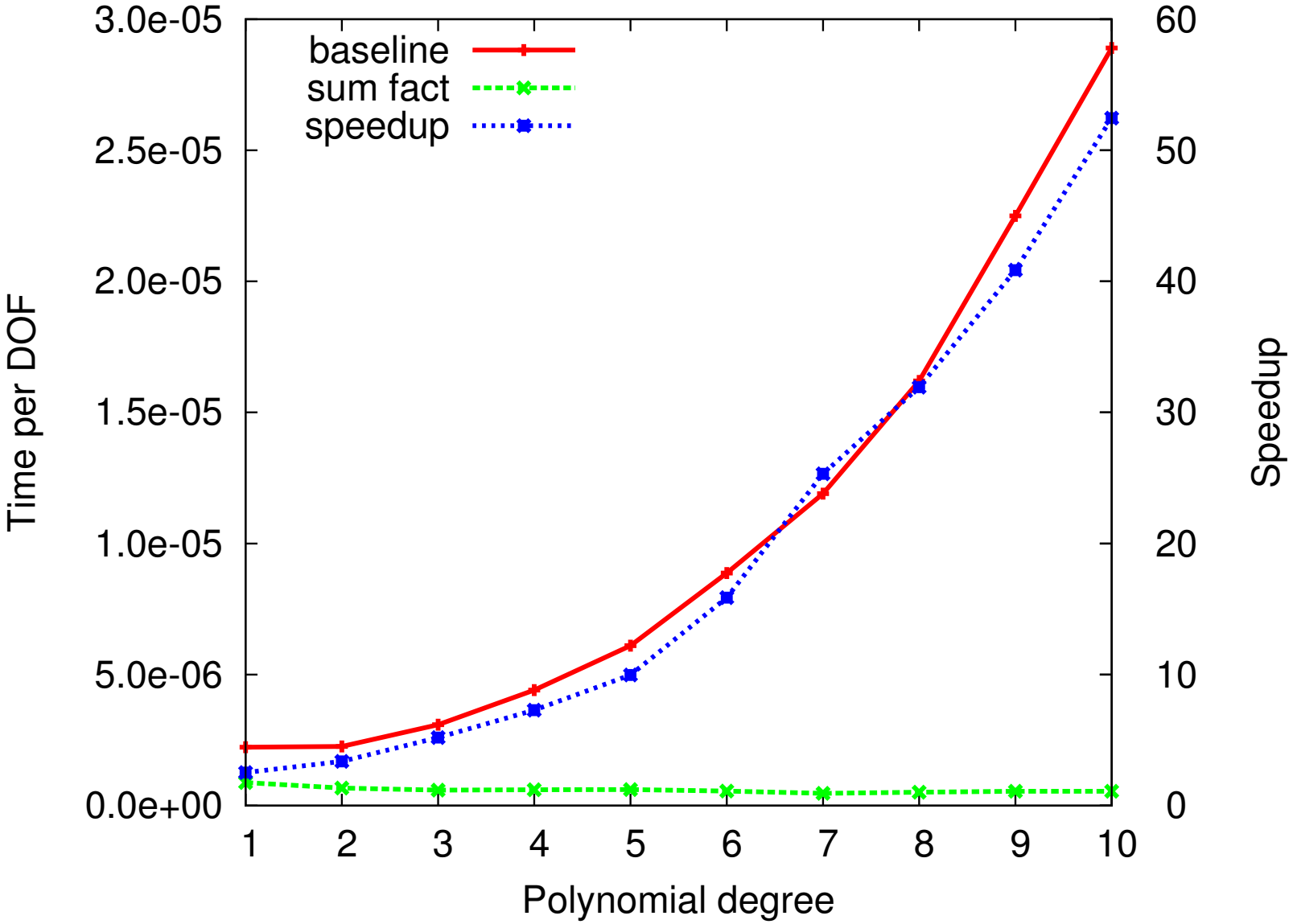
Except $\mathbb{Q}_1$ in 2d, sum factorization is always faster!

Floating-Point Performance: Auto vectorizer, threaded



- Multithreaded version simulating many elements
- icc with auto vectorization / SIMD hints
- Intel MIC (61 cores) vs. Xeon E5-2680v2 (Ivy Bridge, 10 cores)

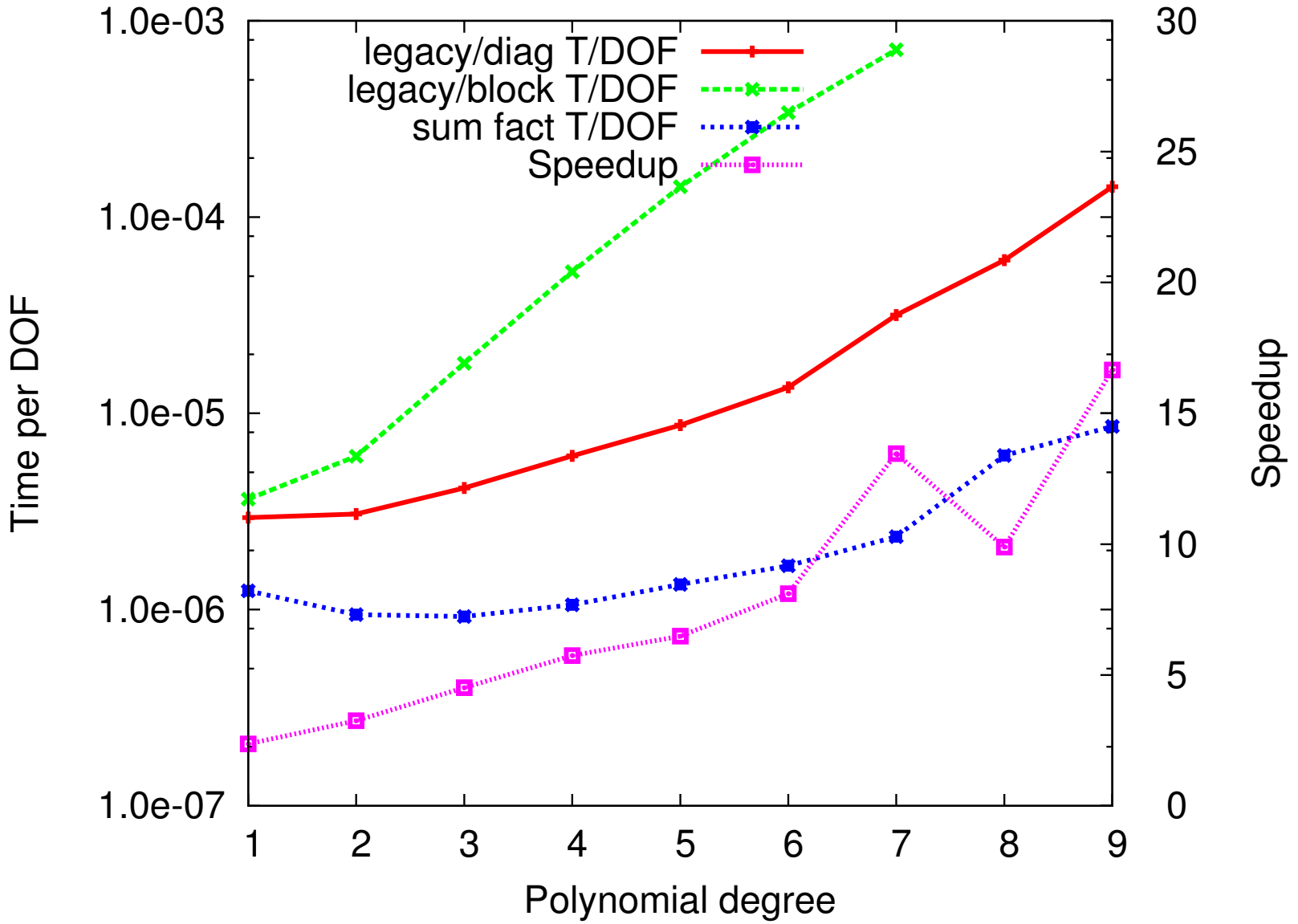
DG Spatial Residual Evaluation (3D)



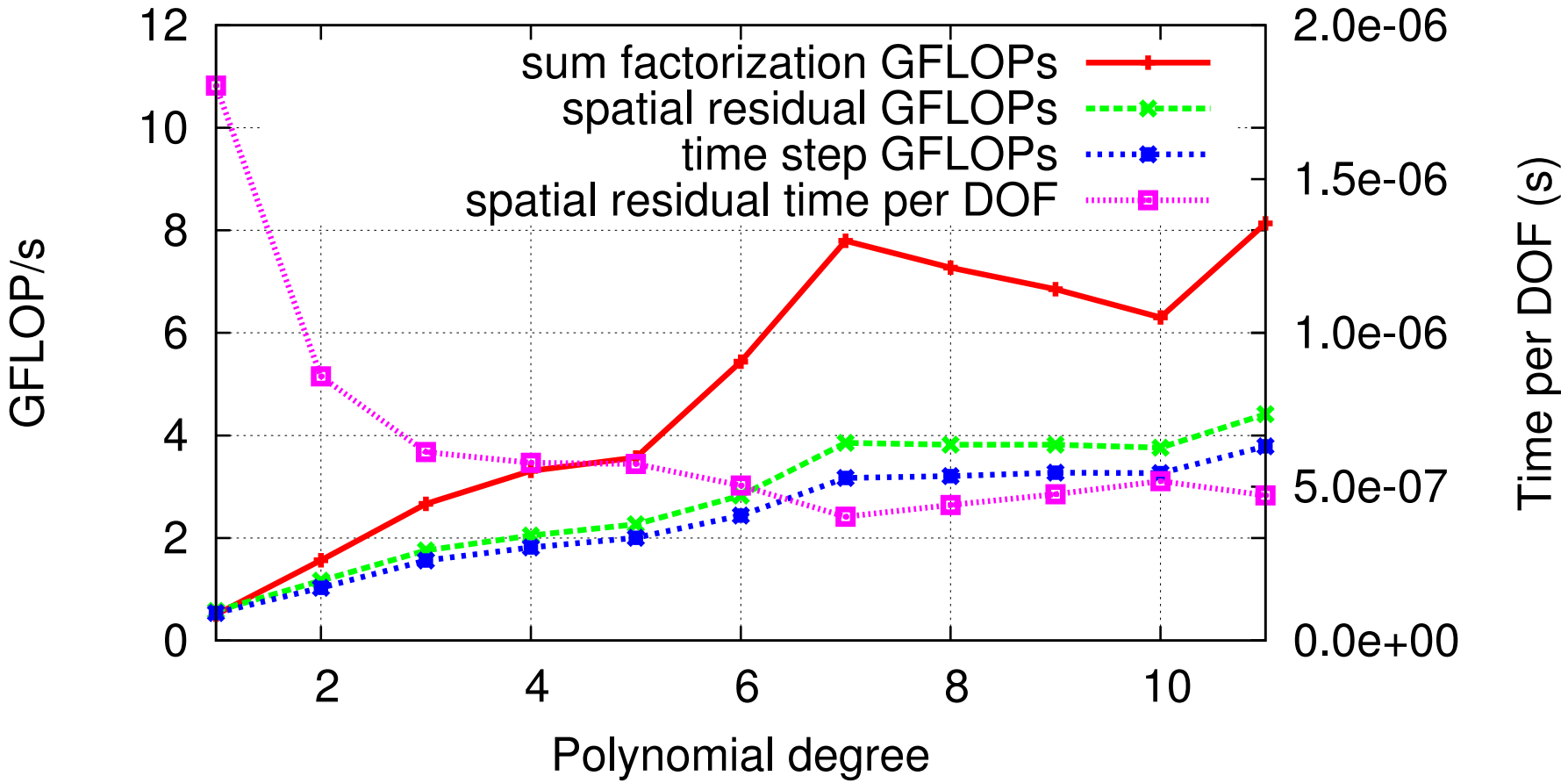
Discussion of Results

- There is no complexity reduction for 1d edges $\Rightarrow$ speedup in 2d is reduced
- Time/DOF for sum factorization version does not increase with polynomial degree
- $d = 3$, polynomial degree 6 breakdown:
  - $\approx$ 30 % grid traversal, solution/residual read/write, scales $O(n^d)$
  - $\approx$ 20 % coefficient/geometry evaluation, scales $O(n^d)$
  - $\approx$ 33 % sum factorization on faces, scales $O(n^d)(!)$
  - $\approx$ 10 % sum factorization on volumes, scales $O(n^{(d+1)})$
- Large part of run-time scales linear in # DOFs
- **Summary**: speedup of sum factorization kernel can be almost recovered in complete DG variable coefficient residual evaluation!

Complete Explicit Time Step 3d



GFLOP Rates Per Core



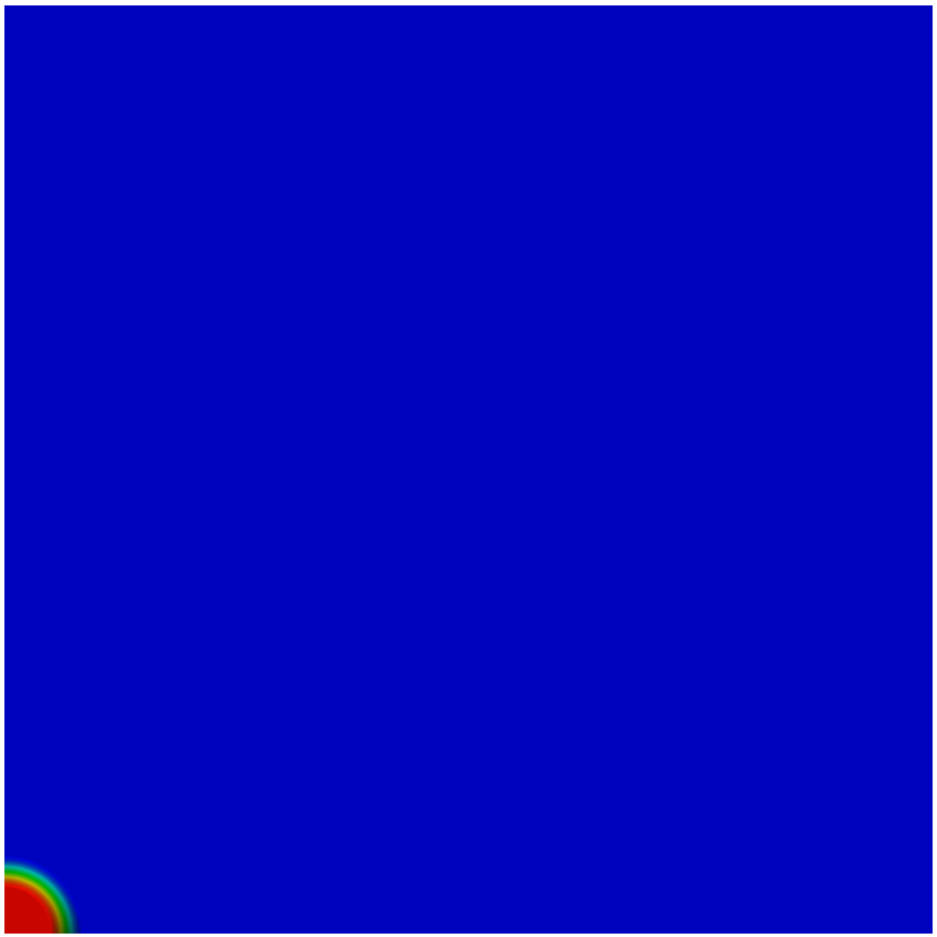
Xeon E5-2680v2 10 core processor (22 GFLOPS peak / core)

Residual: 4 GFLOPS

Sum factorization alone: 8 GFLOPS

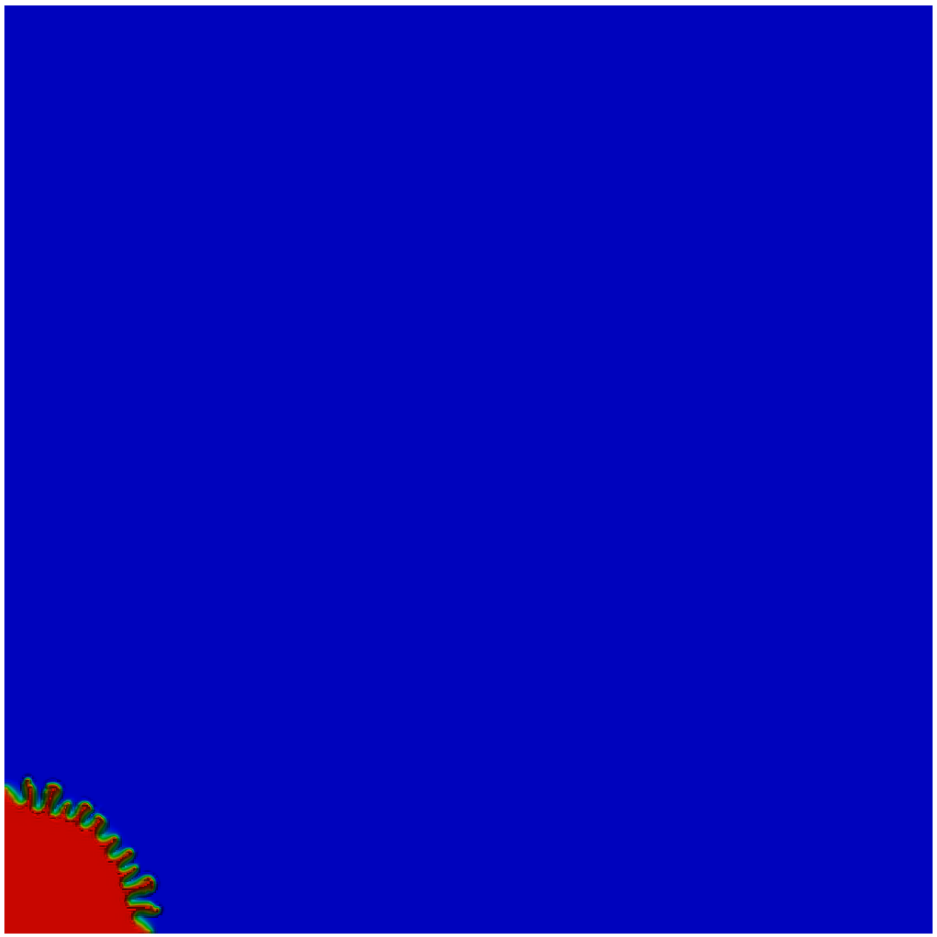
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



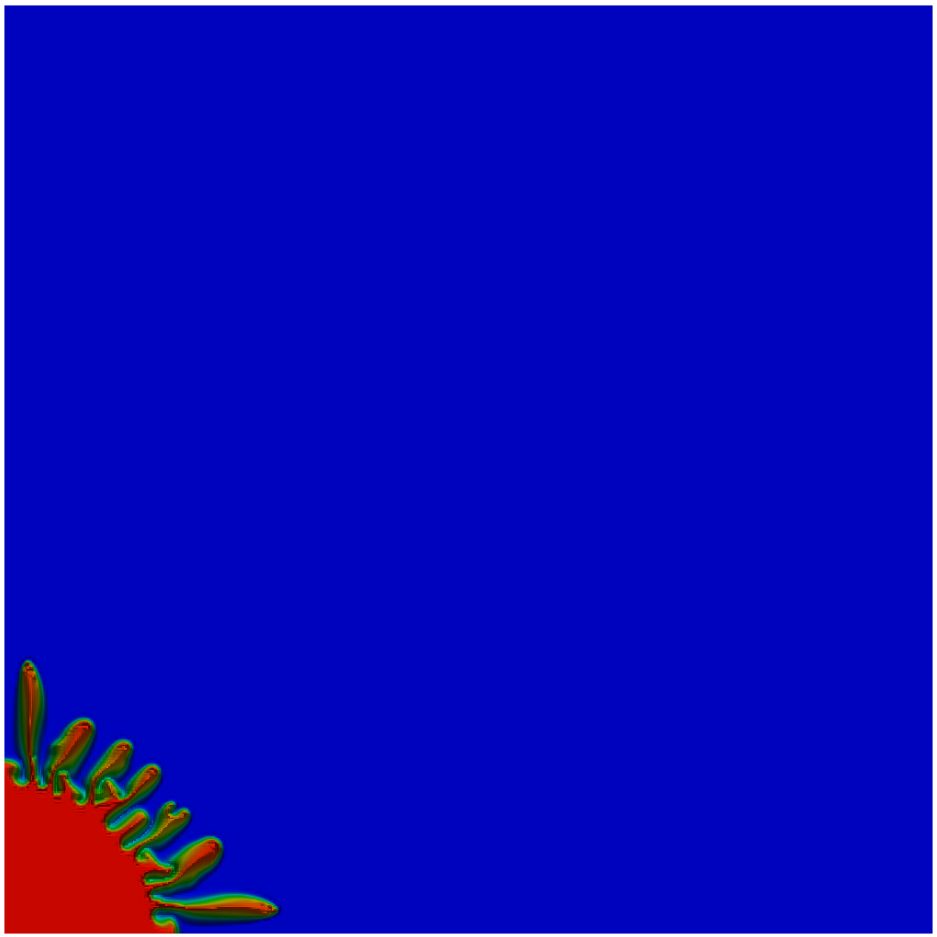
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



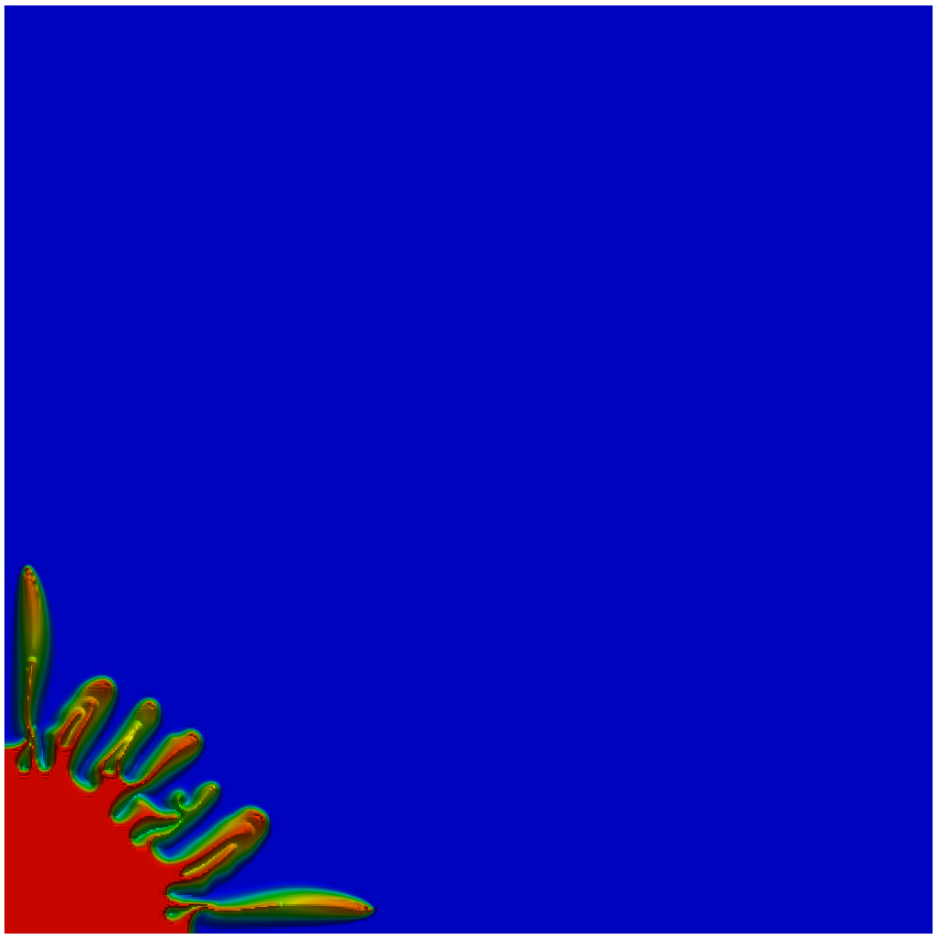
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



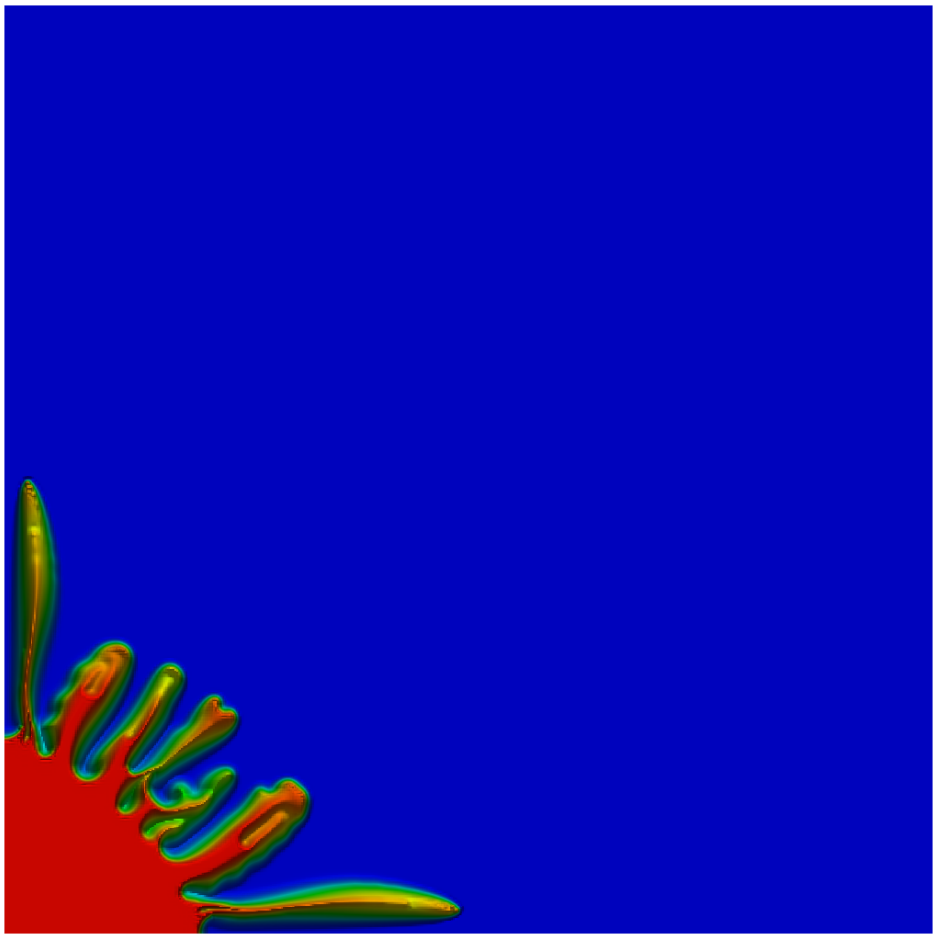
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



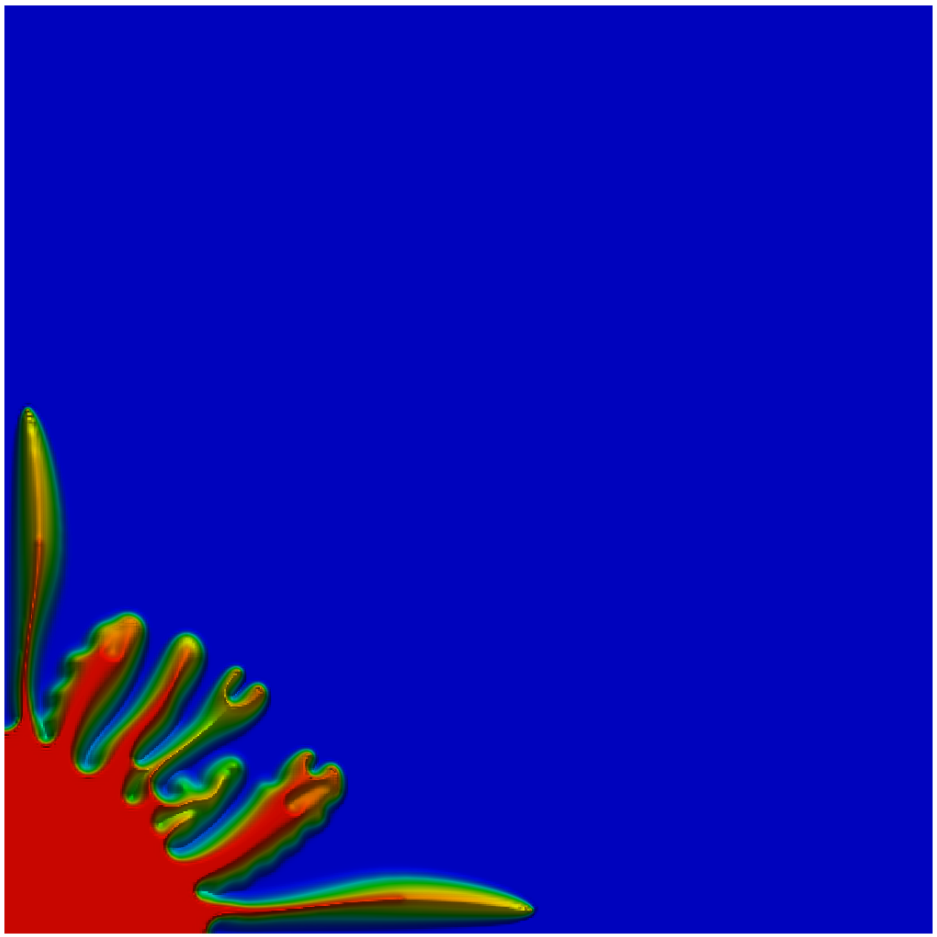
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



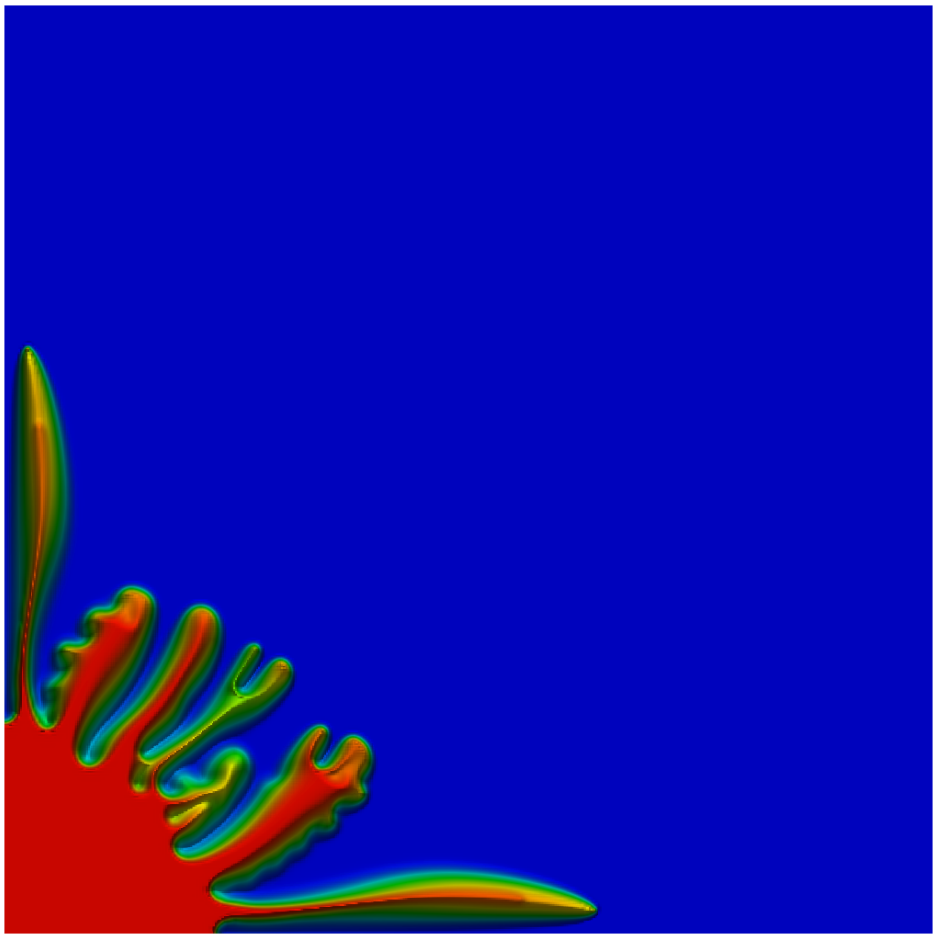
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



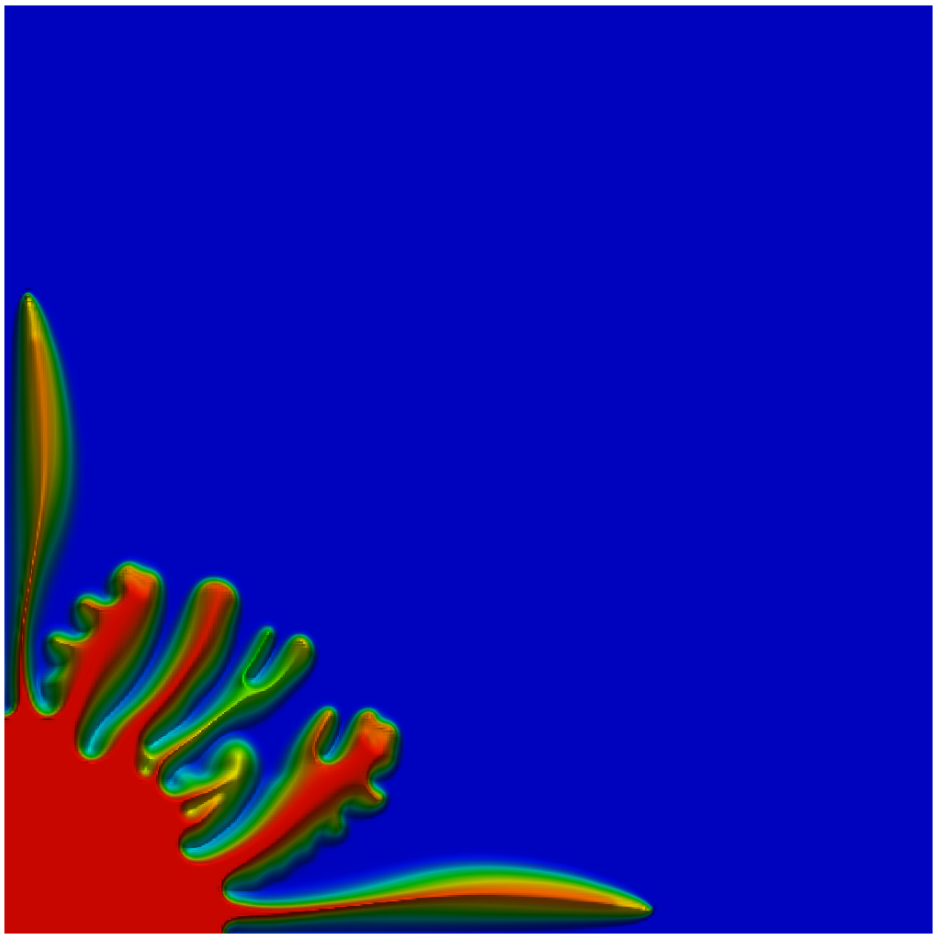
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



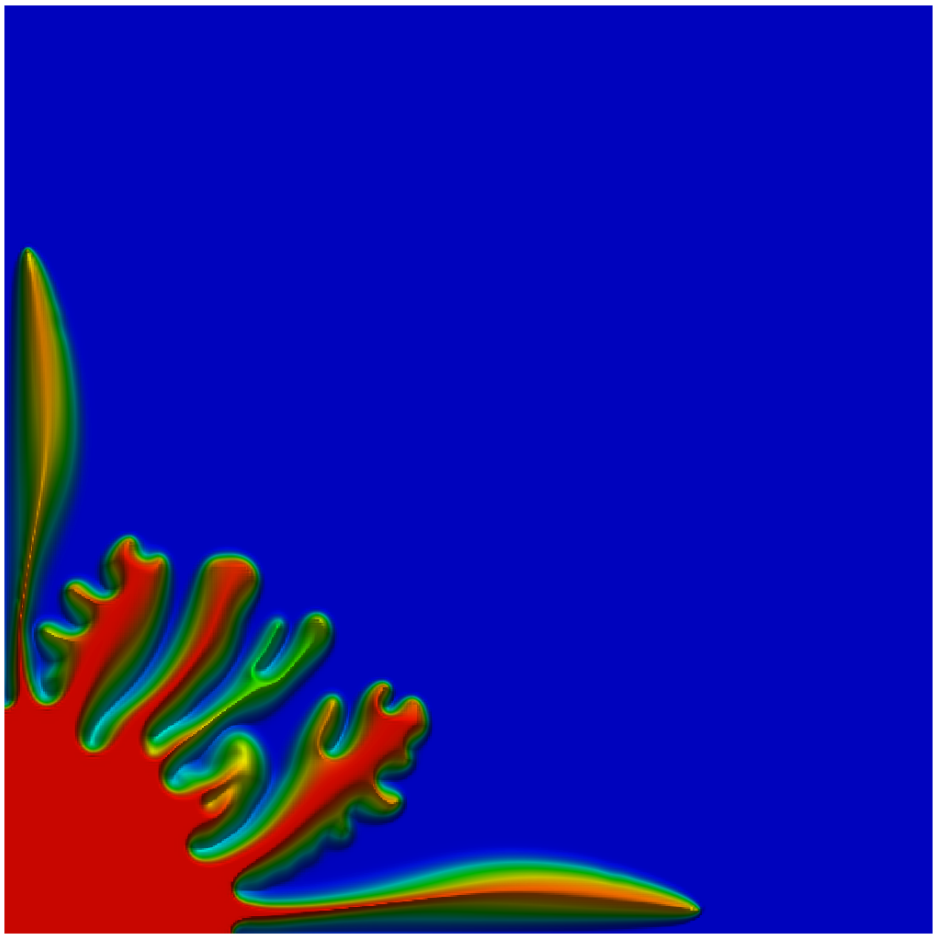
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



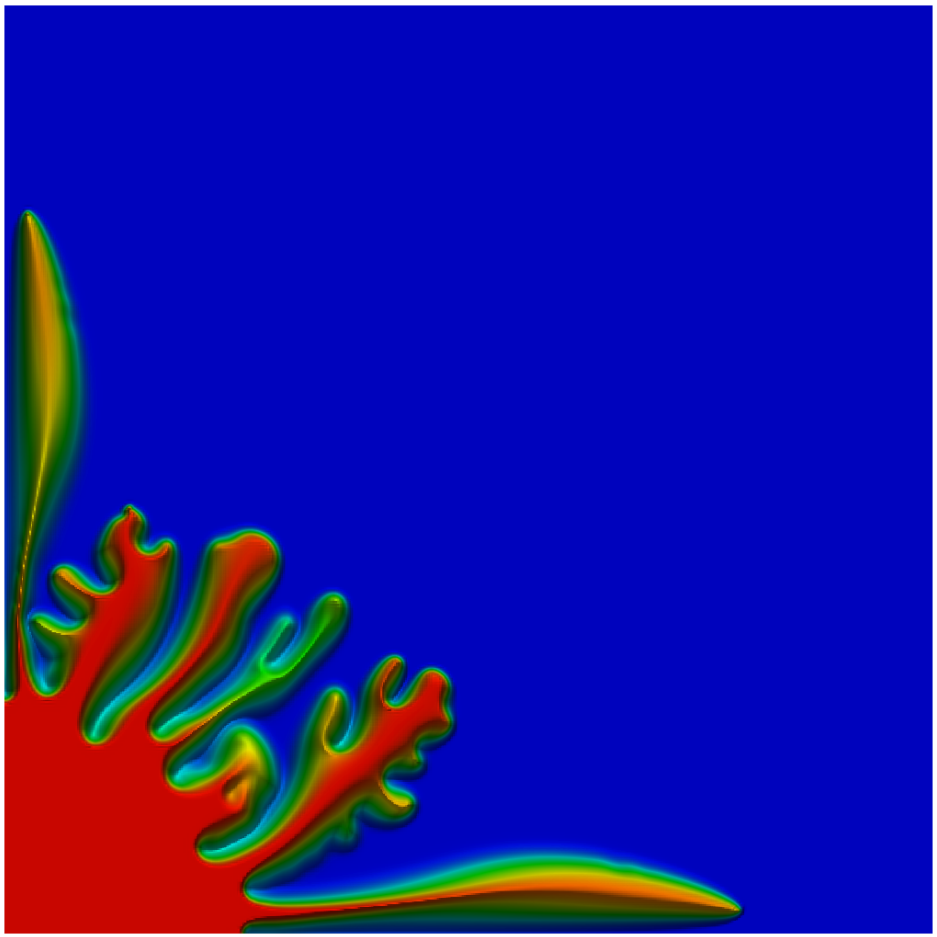
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



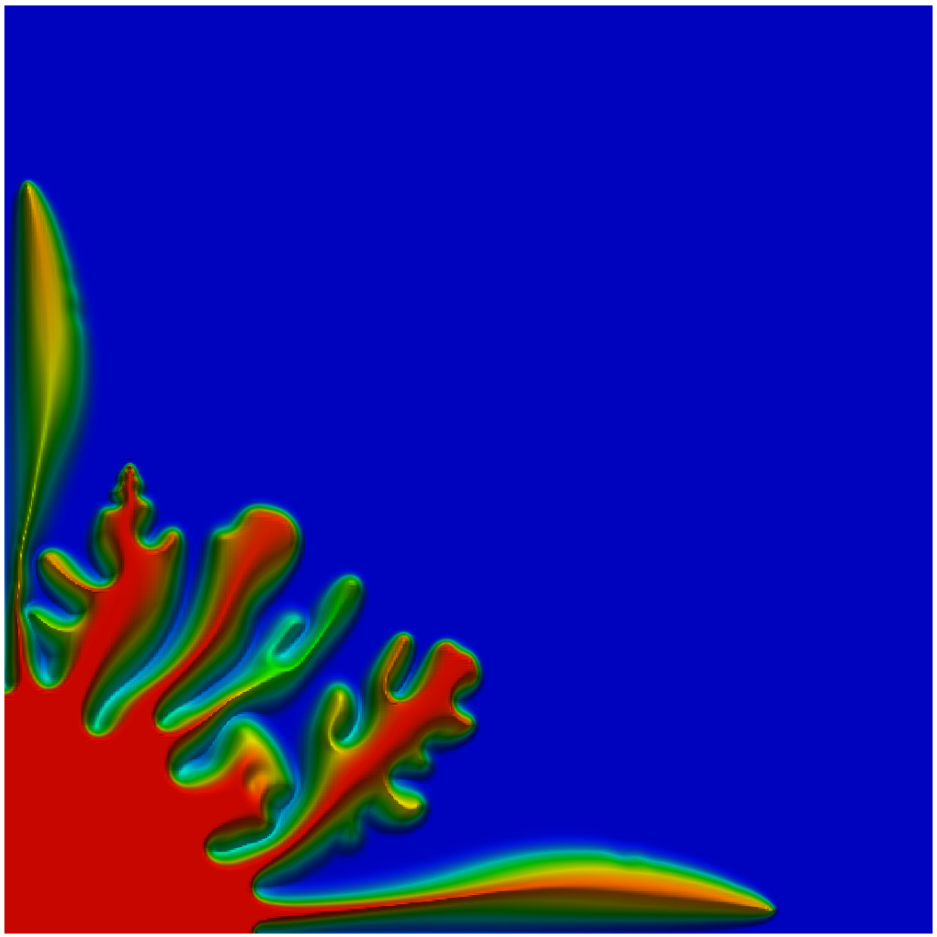
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



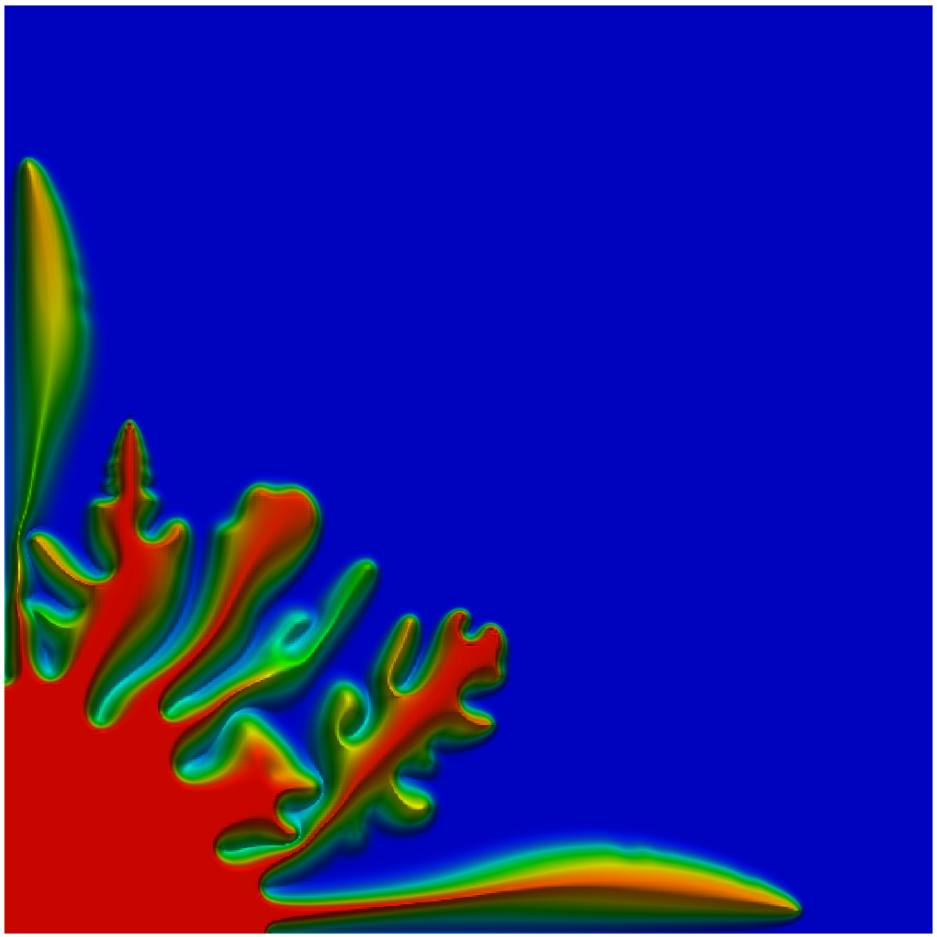
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



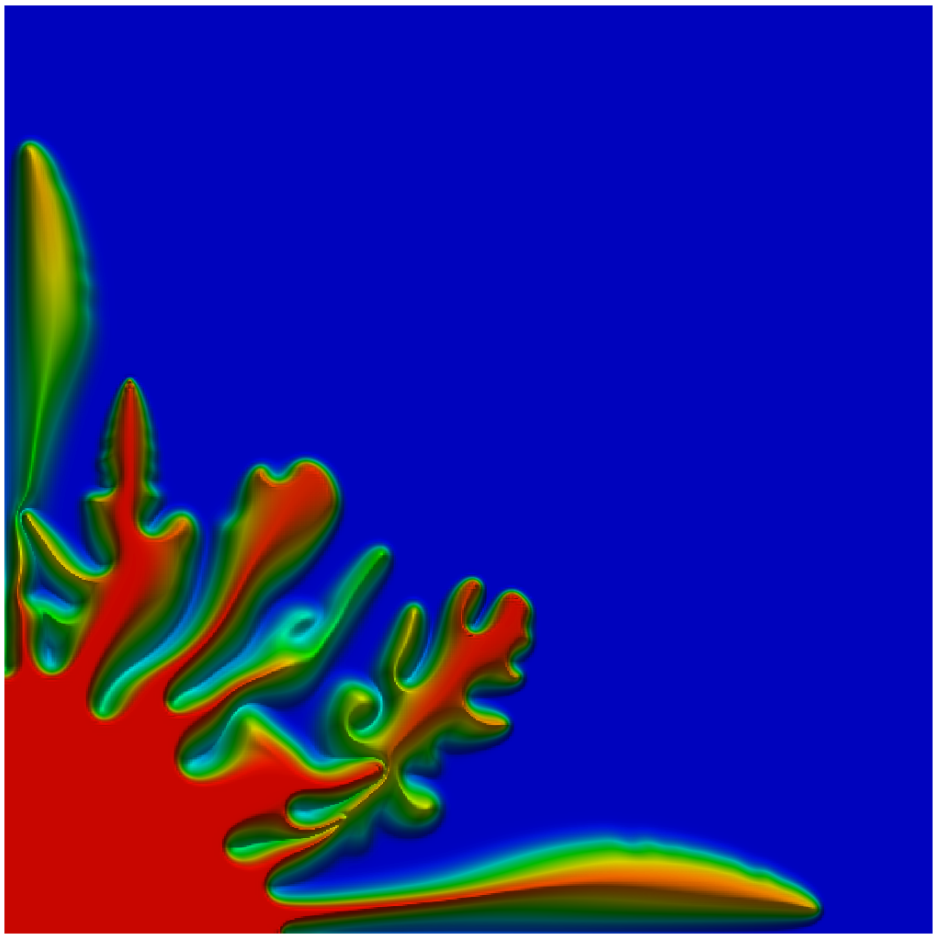
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



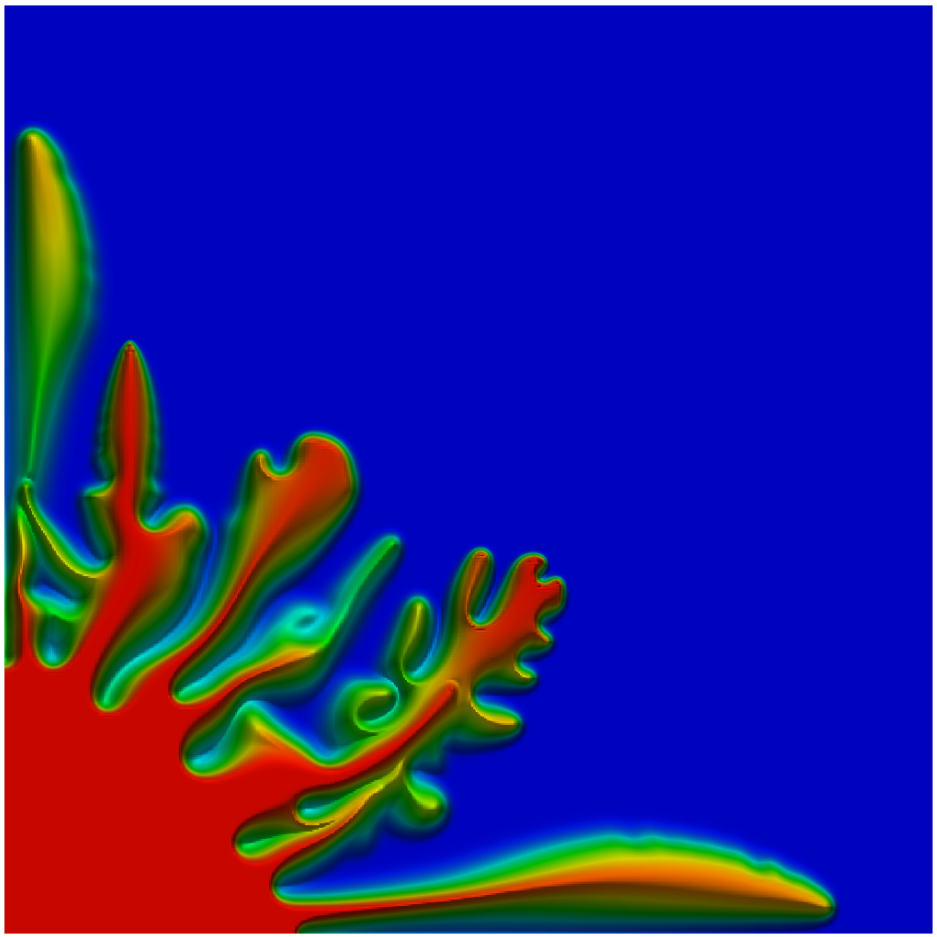
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



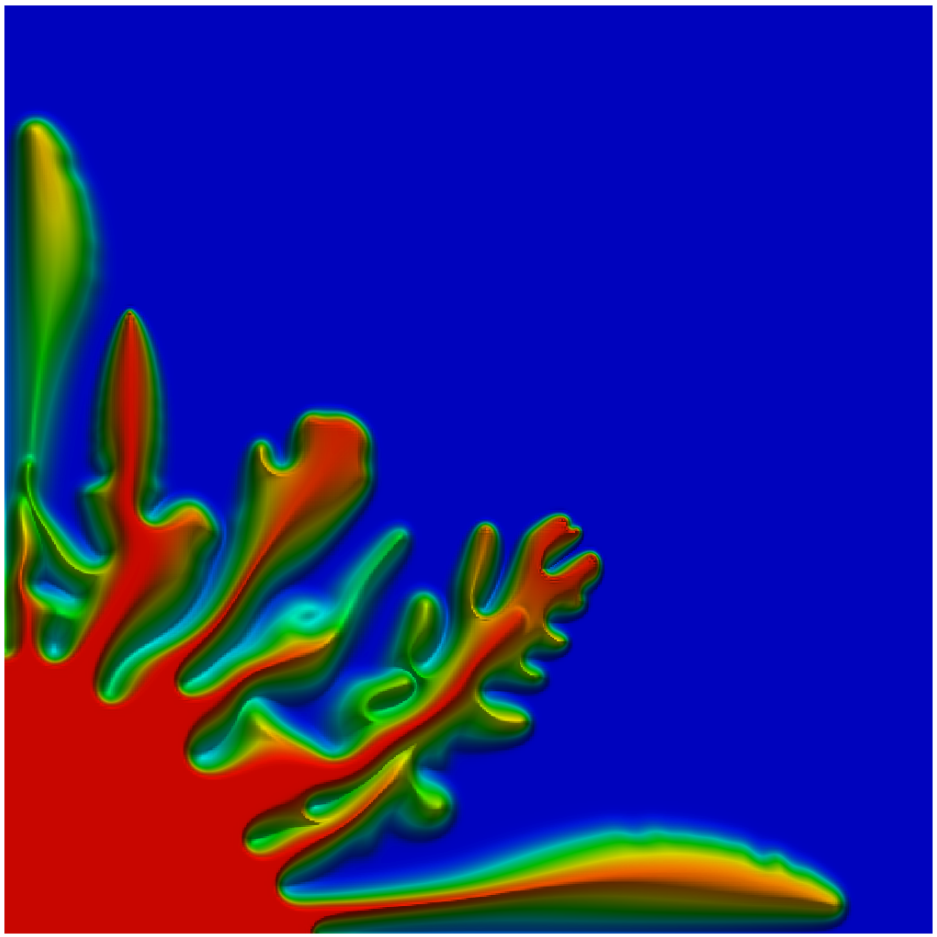
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



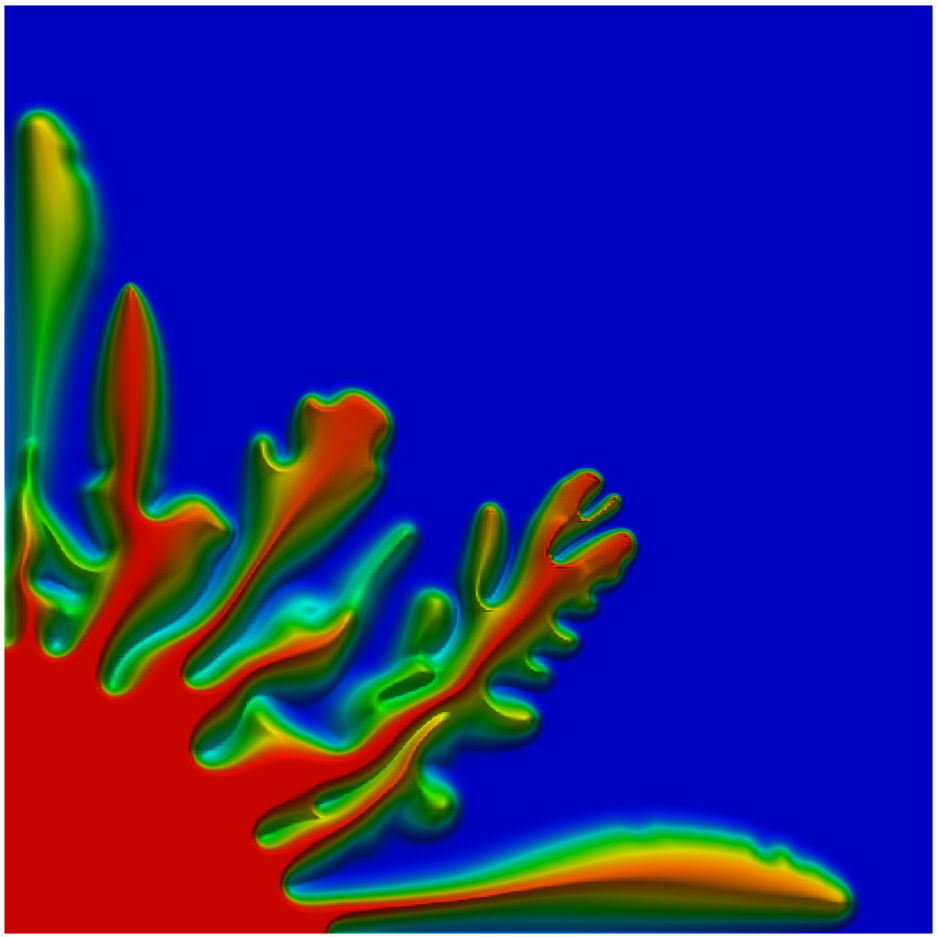
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



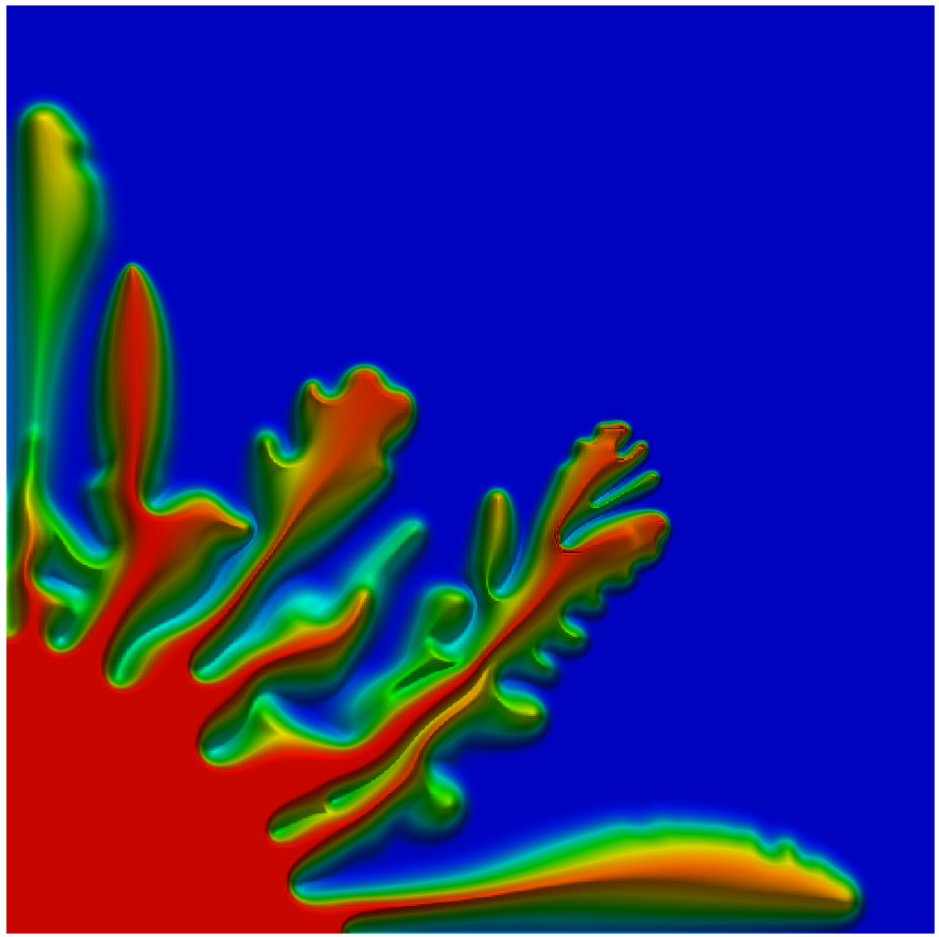
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



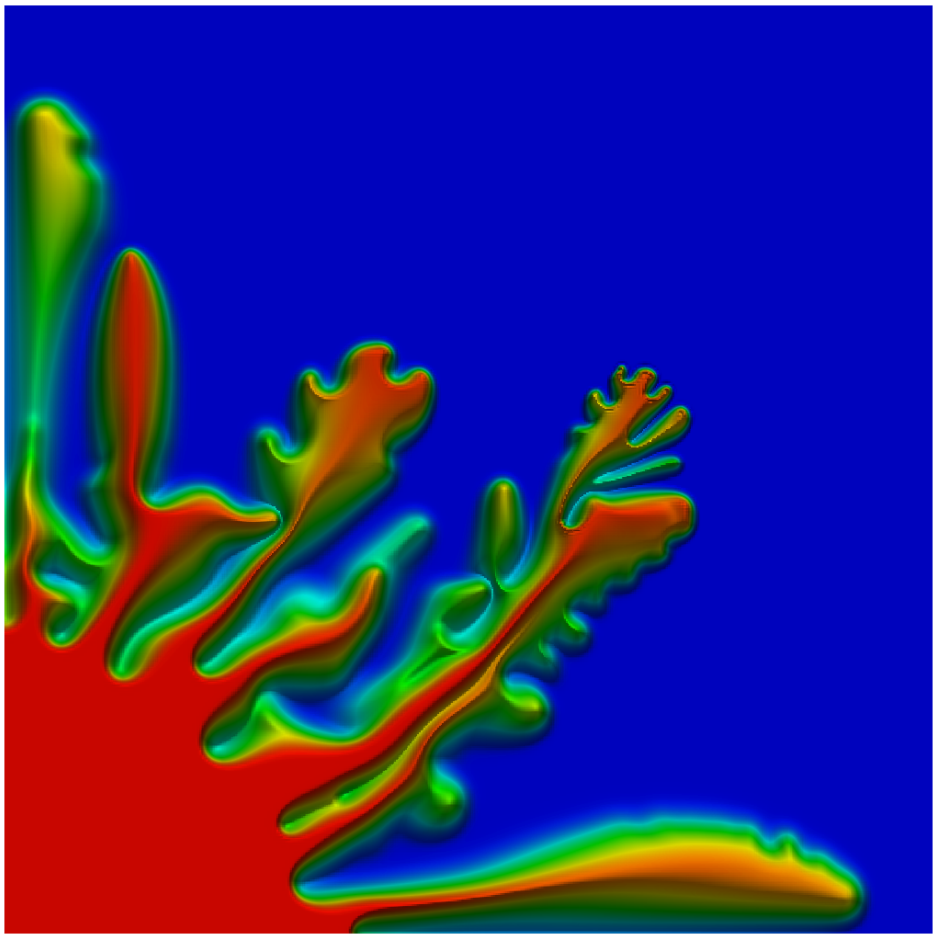
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



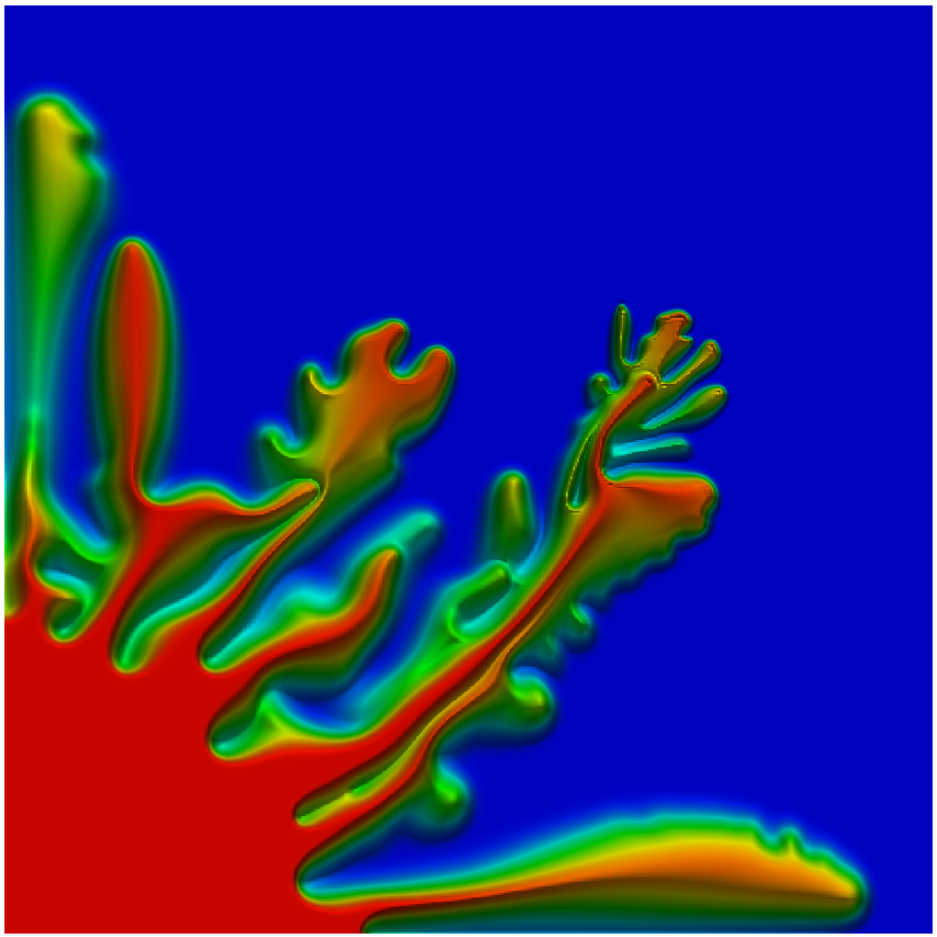
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



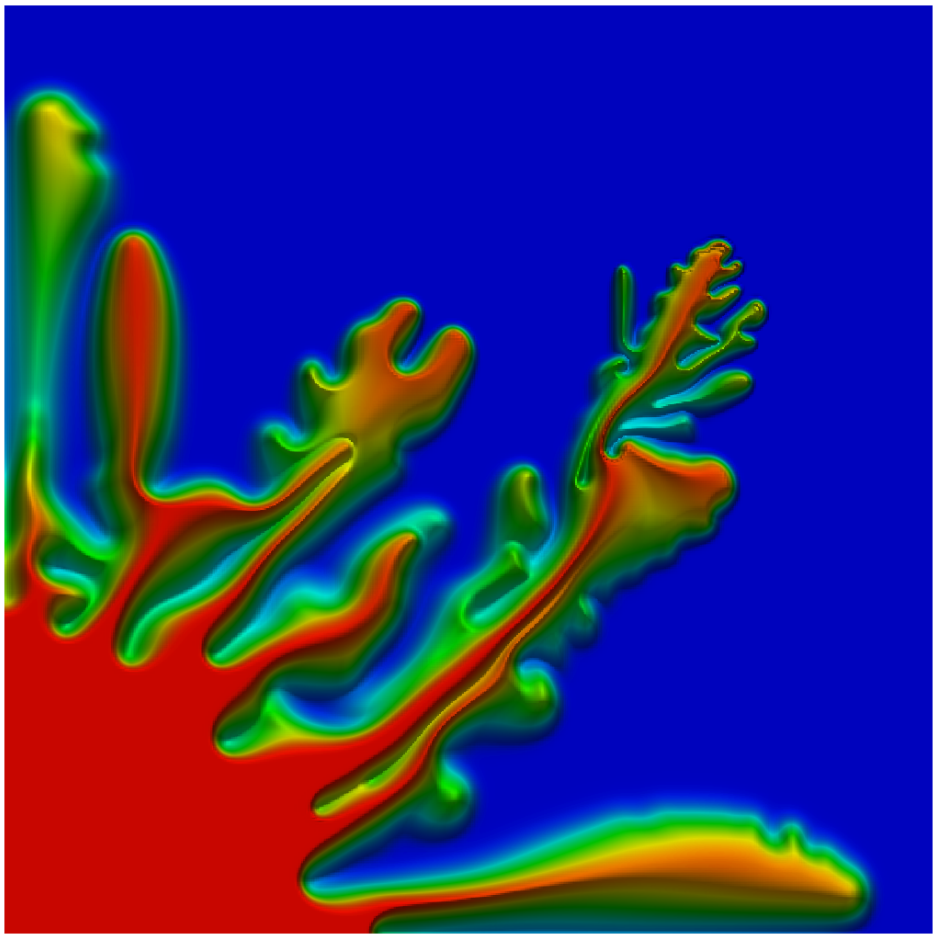
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



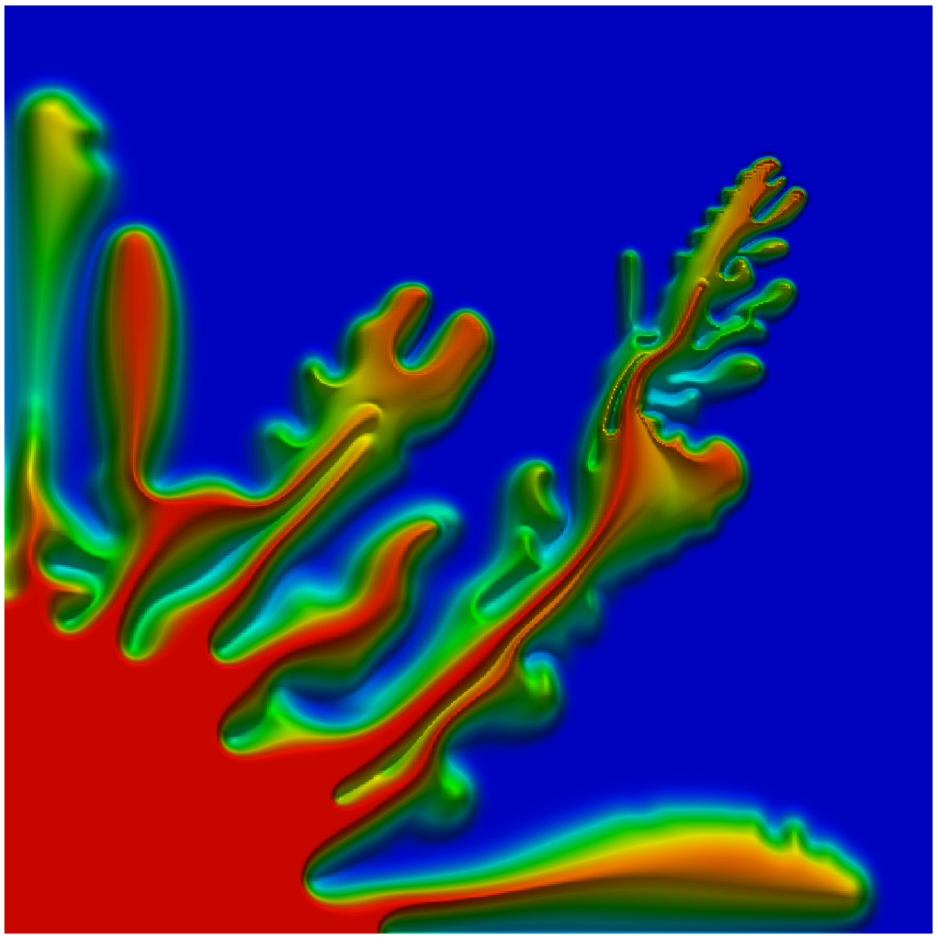
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



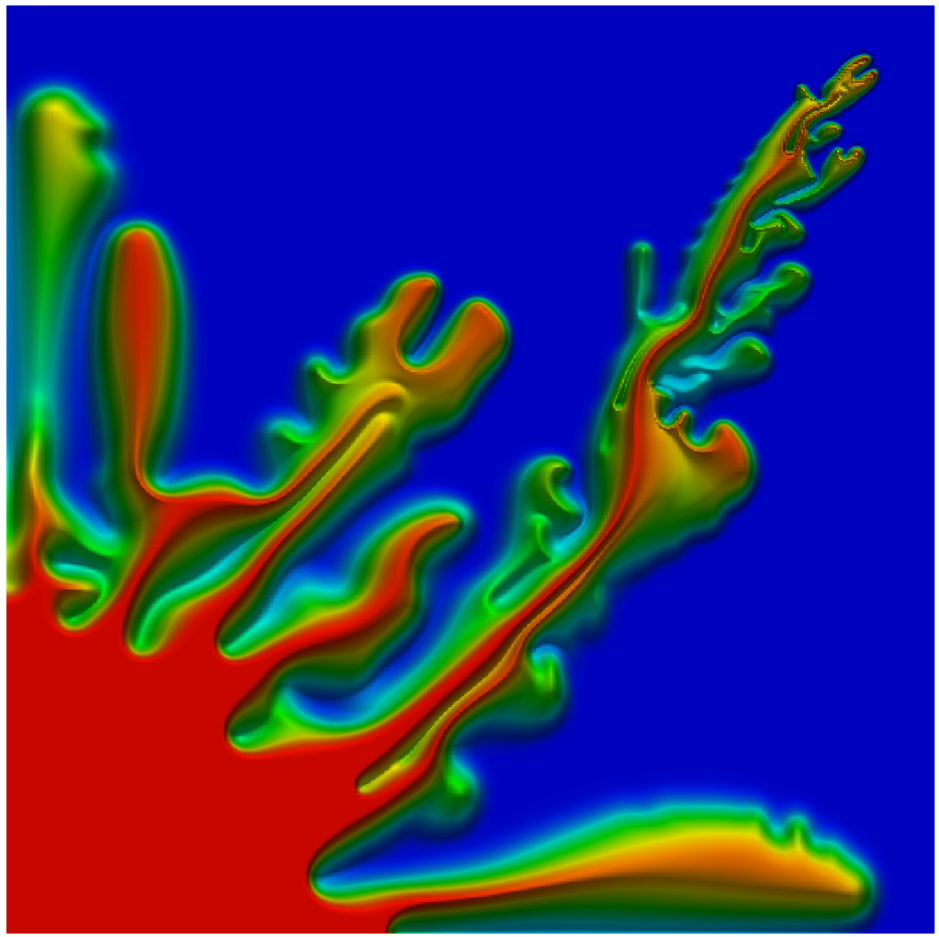
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



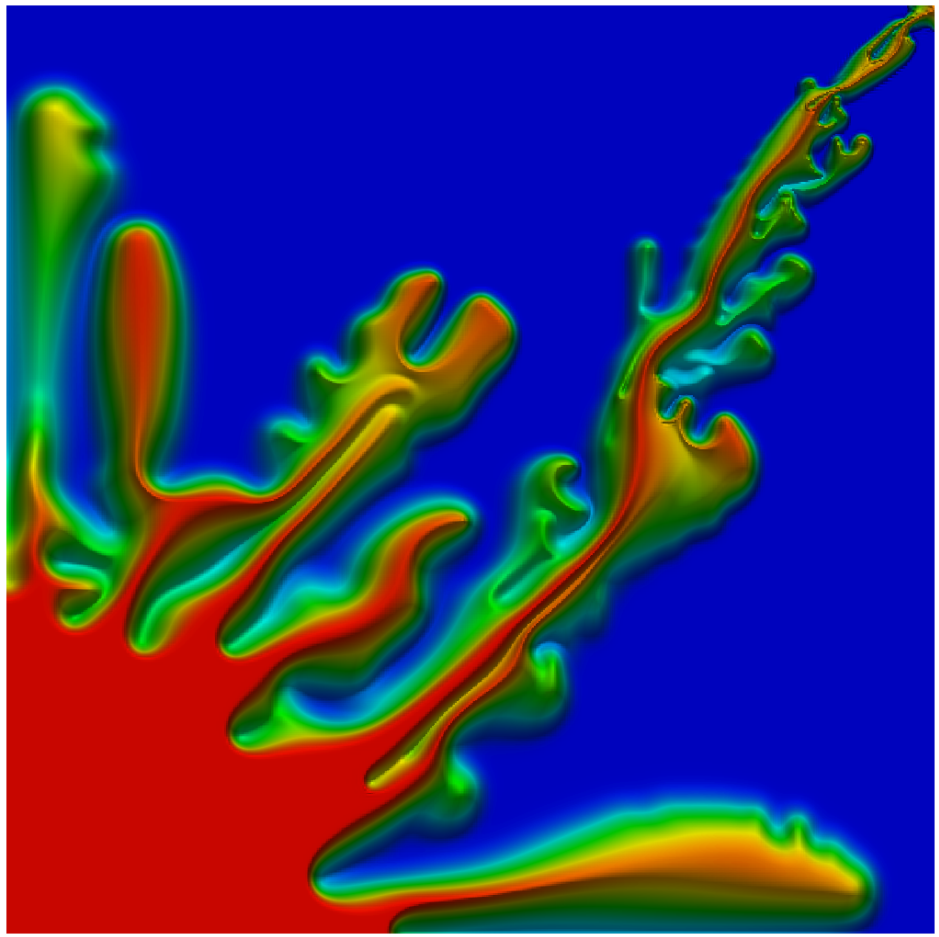
2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



2d Miscible Displacement

- Mobility ratio 100: $\mu(c) = \exp(R(1-c))$, $R = 4.6$, $C_{in} = 0.95$
- Uses same scheme as before: CCFV for pressure, Q_2 for concentration
- No limiter, molecular diffusion, grid Peclet close to one



Implicit Case: To Assemble or not to Assemble?

SF for element stiffness matrix: $O(n^{2d+1})$ instead of $O(n^{3d})$
Matrix-free SF-based solver has $O(n^{d+1})$ per element per step
3D Laplace, 8^3 elements

degree	1	2	3	4	5	6	7
DOF	4096	13824	32768	64000	110592	175616	262144[1]
TASS *old*	0.02	0.39	3.6	21.2	91.8	319.1	971.2
TASS *new*	0.02	0.17	0.8	2.8	8.2	25.1	59.3
#IT BILU[1]	20	25	30	31	34	35	38
TSOLVE	0.01	0.09	0.57	2.3	8.3	18.9	48.8
#IT JAC	97	103	117	142	171	203	242
TSOLVE[3]	0.3	0.9	2.0	3.4	7.1	13.9	20.8

[1] Matrix has 10^9 nonzeroes!
[2] does not include ILU decomposition!
[3] estimated from time for residual evaluation

$\rightarrow$ Matrix-free on DG + AMG on Q_1 conforming subspace

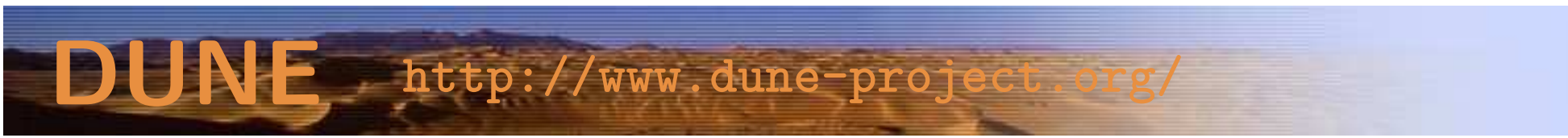
Contents

DUNE Software Framework

- All methods in this talk have been implemented in DUNE
- Developed since 2002 by groups/individuals in Berlin, Bergen, Aachen, Freiburg, Heidelberg, Münster, Stuttgart, Warwick, ...
- Available under GNU LGPL license with linking exception.
- Implementation effort in `dune-pdelab`:

Component	LOC
AMG/DG hybrid parallel preconditioner	375
Two-phase flow DG discretization	980
Sum factorization kernels	1600
SF based advection-diffusion DG discretization	2000

- DUNE courses given every spring (at least).



Summary

- All methods in this talk have been implemented in the DUNE software framework (`www.dune-project.org`)
- Part I
  - Fully-coupled DG competitive in accuracy/runtime with cell-centered finite volumes *while being much more flexible*
  - Efficient, robust, scalable hybrid AMG preconditioner for DG system
- Part II
  - Sum factorization based DG implementation has almost constant time per DOF *independent of polynomial degree*
  - SF can handle variable coeffs, non-affine geometry, nonlinear problems
  - Always faster than standard implementation
- Future work
  - Hybrid parallel implementation
  - Preconditioners for high-order implicit DG formulations
  - **Disclaimer**: Use moderate polynomial degree in practice!

Thank you for your attention !

Numerical Methods on High-Performance Computers

International SPPEXA Workshop

Invited Speakers:
Assyr Abdulle, EPFL, Lausanne, Switzerland
Alexandre Ern, CERMICS, Paris, France
Rob Falgout, Lawrence Livermore National Laboratory, California, USA
Mike Giles, University of Oxford, UK
Mark Hoemmen, Sandia National Laboratories, USA
Claus-Dieter Munz, Universität Stuttgart, Germany
Martin Kronbichler, Technische Universität München, Germany
Philip L. Roe, University of Michigan in Ann Arbor, Michigan, USA
Chi-Wang Shu, Brown University, Rhode Island, USA
Carol S. Woodward, Lawrence Livermore National Laboratory, California, USA

December 1 – 3, 2014

IWR • Im Neuenheimer Feld 368 • 69120 Heidelberg
http://conan.iwr.uni-heidelberg.de/sppexa-workshop
Register Online