

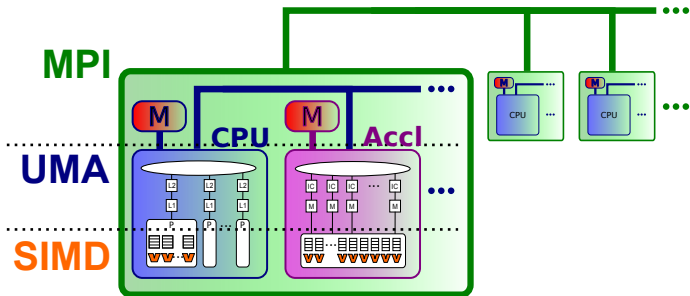
# High-performance Computing for Flows in Porous Media

Peter Bastian and Steffen Müthing

UNIVERSITÄT  
HEIDELBERG | Zukunft. Seit 1386.

MSPM-JRJJ2014, December 8-9, 2014

# The Exascale Challenge



(Figure by Christian Engner)

## 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 + SIMD  
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
  - ▶ Restricted to explicit time-stepping or certain solvers, e.g. CG, GMG
  - ▶ Low-order FE schemes:
    - ★ Exploit problem structure: linear transformation, regular grids, constant coefficients, ...
    - ★ Vectorization over several elements
  - ▶ High-order FE schemes:
    - ★ Complexity reduction through tensor-product approaches
    - ★ Medium high order: Vectorization within one element
    - ★ Very high order: Several threads to one element

# Outline

- 1 Discontinuous Galerkin Methods for flows in porous media
  - ▶ Porous media flow problems
  - ▶ Definition of DG schemes
  - ▶ Increasing polynomial degree may be beneficial even for problems with low regularity
  
- 2 High-order DG methods exploiting tensor product structure
  - ▶  $q$ -scalability (polynomial degree)
  - ▶ Floating point performance (SIMD instructions)
  - ▶ Explicit time-stepping methods
  - ▶ First results on Intel Phi
  - ▶ Some remarks on implicit schemes

# Contents

- 1 Discontinuous Galerkin Methods for flows in porous media
- 2 High Order Tensor Product DG Methods

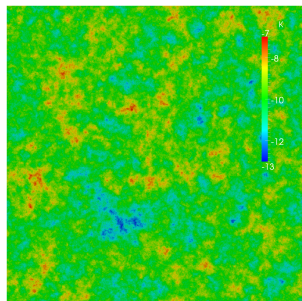
# Single Phase Flow and Transport

Parabolic/hyperbolic problem coupled to elliptic problem:

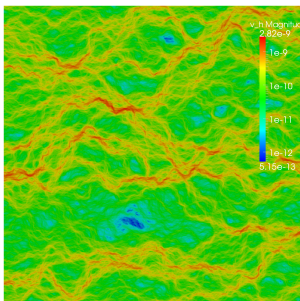
$$\nabla \cdot \mathbf{v} = f, \quad \mathbf{v} = -\frac{K}{\mu}(\nabla p - \rho \mathbf{g}),$$

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

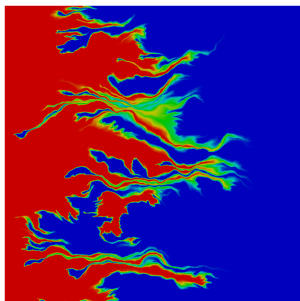
Tensor  $K$ ,  $D$ , heterogeneity, convection-dominated effective macro-scale descriptions, uncertainty of parameters



Conductivity  $K$



Velocity magnitude  $\|u\|$



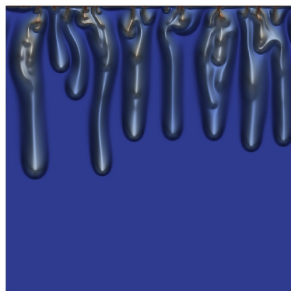
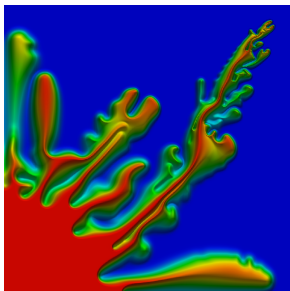
Concentration  $c$

# Unstable Flows in Porous Media

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

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

Density driven flow, miscible displacement  
Can exploit high resolution schemes



# Incompressible Immiscible Two-phase Flow

**Total fluid conservation:**

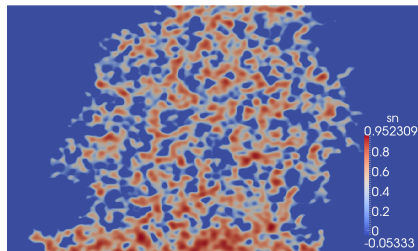
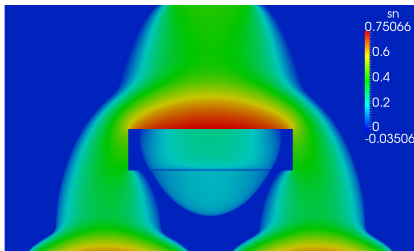
$$\nabla \cdot (v_a - \lambda_n K \nabla \phi_c) = q, \quad v_a = -\lambda_t K \nabla \phi_w.$$

**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





# Total Fluid Conservation Equation

$$\begin{aligned}
 a(\phi_w, \phi_c, w) &= \sum_{T \in \mathcal{T}_h} \int_T (\lambda_t K \nabla \phi_w + \lambda_n K \nabla \phi_c) \cdot \nabla w - \sum_{F \in \mathcal{F}_h^{iDw}} \int_F \nu_F \cdot \{\lambda_t K \nabla \phi_w + \lambda_n K \nabla \phi_c\}_\omega [w] \\
 &\quad - \sum_{F \in \mathcal{F}_h^{iDw}} \int_F \theta \nu_F \cdot \{\lambda_t K \nabla w\}_\omega [\phi_w] + \sum_{F \in \mathcal{F}_h^{iDw}} \int_F \gamma_{F,w} [w] [\phi_w] \\
 l(w) &= \sum_{T \in \mathcal{T}_h} \int_T q w - \sum_{F \in \mathcal{F}_h^{Nw}} \int_F j w \quad \theta = 1 \text{ (SIPG)}, \theta = 0 \text{ (OBB)}, \theta = -1 \text{ (NIPG)}
 \end{aligned}$$

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

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

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

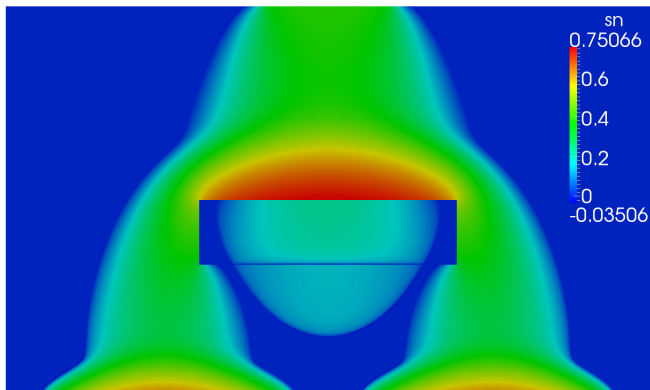
$$\begin{aligned}
 b(\phi_w, \phi_c, z) = & - \sum_{T \in \mathcal{T}_h} \int_T (f_n \hat{\nu}_a - \lambda_n K \nabla \phi_c) \cdot \nabla z \\
 & + \sum_{F \in \mathcal{F}_h^{iDn}} \int_F \langle f_n^{\uparrow, -}, f_n^{\uparrow, +} \rangle \nu_F \cdot \hat{\nu}_a [z] + \sum_{F \in \mathcal{F}_h^{iDn}} \int_F \nu_F \cdot \{-\lambda_n K \nabla \phi_c\}_\omega [z] \\
 & + \sum_{F \in \mathcal{F}_h^{iDn}} \int_F \theta \nu_F \cdot \{-\lambda_n K \nabla z\}_\omega J(\phi_c) + \sum_{F \in \mathcal{F}_h^{iDn}} \int_F \gamma_{F,n} [z] J(\phi_c) \\
 r(z) = & \sum_{T \in \mathcal{T}_h} \int_T q_n z - \sum_{F \in \mathcal{F}_h^{Nn}} \int_F j_n z
 \end{aligned}$$

Upwinding of fractional flow and penalty parameter:

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

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

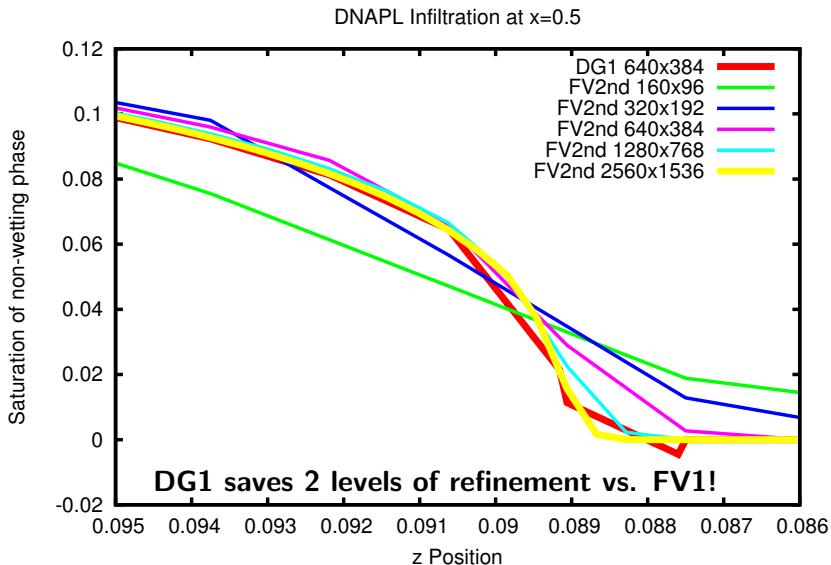
# A DNAPL Infiltration Model Problem



- $k_{r\alpha}$ : quadratic
- $p_c$ : Brooks-Corey,  $\lambda = 2$
- $p_e^{bulk} = 755$ ,  $p_e^{lense} = 1163$
- CCFV1: central, Alex.(2)
- DG1 and Alexander(2)
- DG2 and Alexander(3)

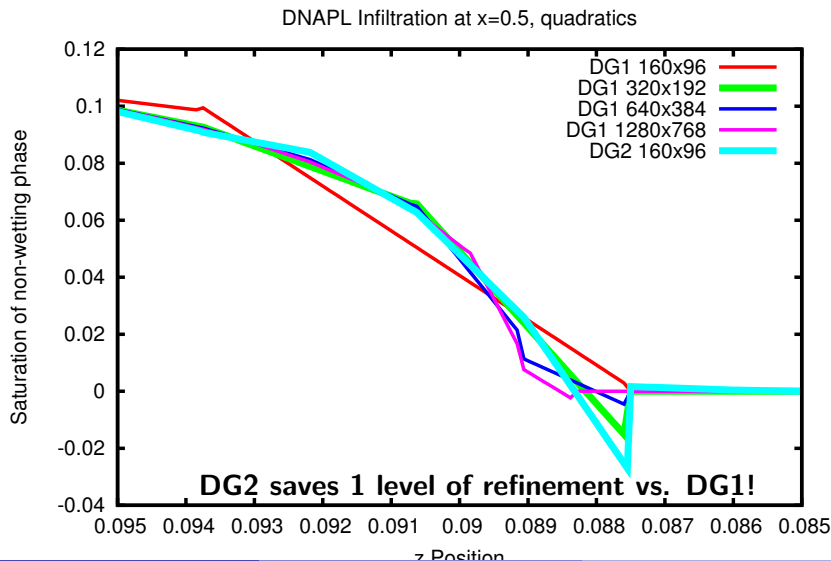
# At Free Boundary

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
DG1	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
FV1	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**

# Contents

- 1 Discontinuous Galerkin Methods for flows in porous media
- 2 High Order Tensor Product DG Methods

## Exploiting Tensor Product Structure I

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

Evaluate finite element function on ref elem at quadrature points:

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

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

$$y = Ax = (A^{(d)} \otimes \dots \otimes A^{(1)}) x, \quad A_{(i_1, \dots, i_d), (j_1, \dots, j_d)} = \prod_{k=1}^d A_{i_k, j_k}^{(k)}$$



## Exploiting Tensor Product Structure II

Elementwise operator evaluation:

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

For every  $\alpha, \beta$  the same abstract matrix vector product as before!

# Sum Factorization I

Extract common factors:

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

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

## Sum Factorization II

- Algorithm by *Buis, Dyksen (1996)*
- For  $k = 1, \dots, d$  do („dim by dim approach”)
  - ▶ 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  $O(n^{2d})$  ( $n = m = q + 1$ ) for the steps above!
- 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  $O(n^{d+1})$
- Can be applied to nonlinear problems on curved elements
- Only tensor product structure of basis functions and quadrature is needed

## Sum Factorization III

- *Discontinuous Galerkin*: Effort for face terms dominates computation time for say  $q < 10$
- Summary of complexities *per element* for  $n = m = q + 1$  and  $d = 3$

Operation	CG	DG (moderate $q$ )
Matrix-free operator evaluation	$n^4$	$n^3$
Assembled operator evaluation	$n^6$	$n^6$
Sum factorized matrix assembly	$n^7$	$n^6$
Naive matrix assembly	$n^9$	$n^9$
Matrix-free point Jacobi application	$n^4$	$n^3$
Block SSOR preconditioner	$n^9$	$n^9$

- Storage needed per element is reduced since only  $1d$  basis functions need to be evaluated

# Density Driven Flow in Porous Media

## Model

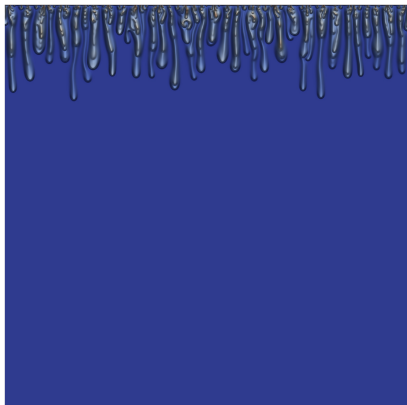
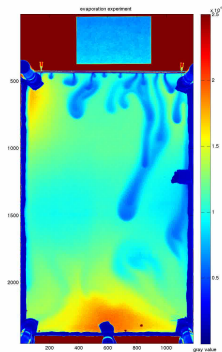
$$\nabla \cdot v = 0, \quad v = -(\nabla P - \omega_s \mathbf{1}_z), \quad \partial_t \omega_s + \nabla \cdot \left( v \omega_s - \frac{1}{Ra} \nabla \omega_s \right) = 0$$

- 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

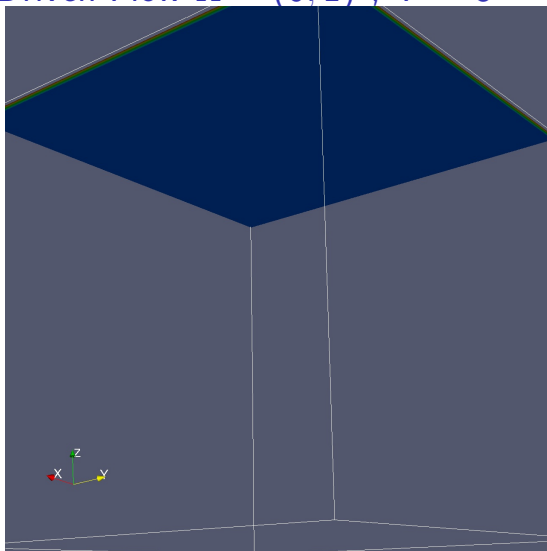
## 2d Density Driven Flow



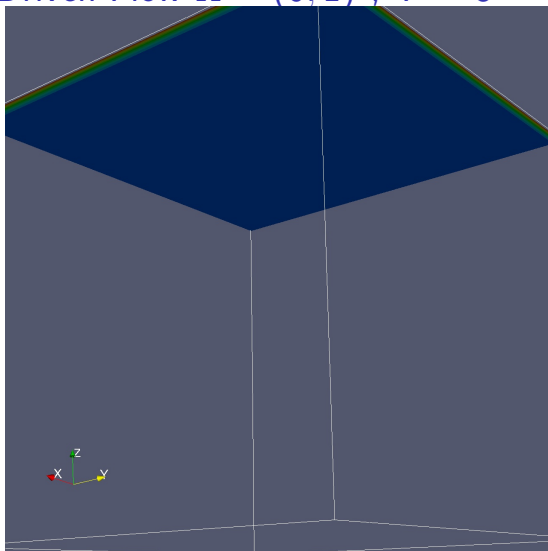
Experiment by Philip Kreyenberg  
(IUP, Heidelberg)

$$Ra = 8000$$

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

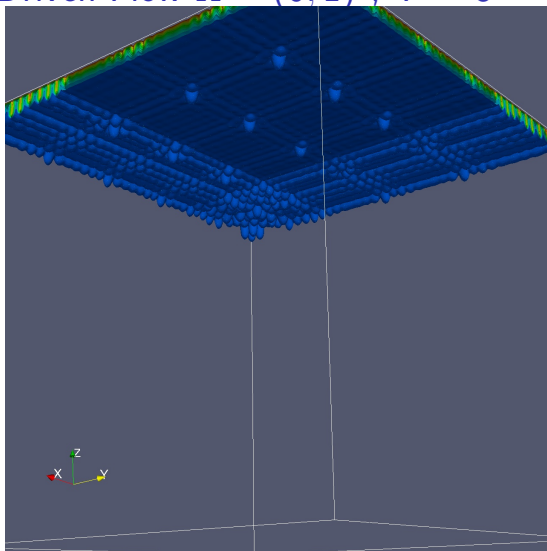
3d Density Driven Flow  $\Omega = (0, 1)^3$ ,  $T = 5$ 

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

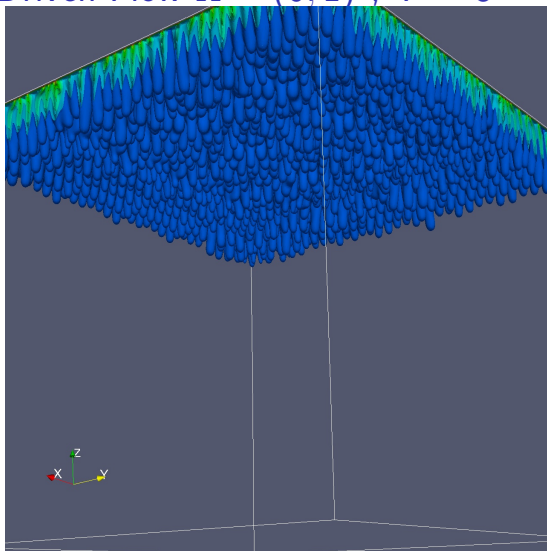
3d Density Driven Flow  $\Omega = (0, 1)^3$ ,  $T = 5$ 

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

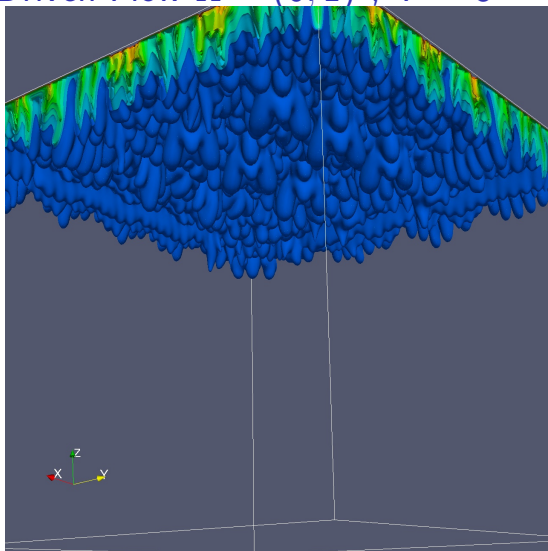


3d Density Driven Flow  $\Omega = (0, 1)^3$ ,  $T = 5$ 

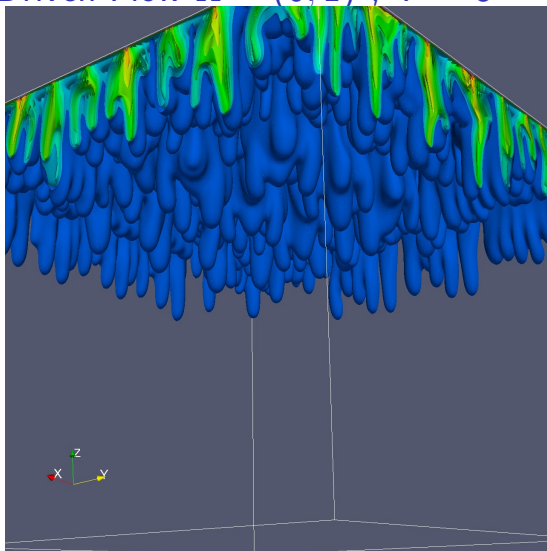
8 × Xeon E5-2680v2 10 core,  $240^3$ ,  $Q_2$ ,  $3.7 \cdot 10^8$  DOF, 16000 steps, 64h.

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

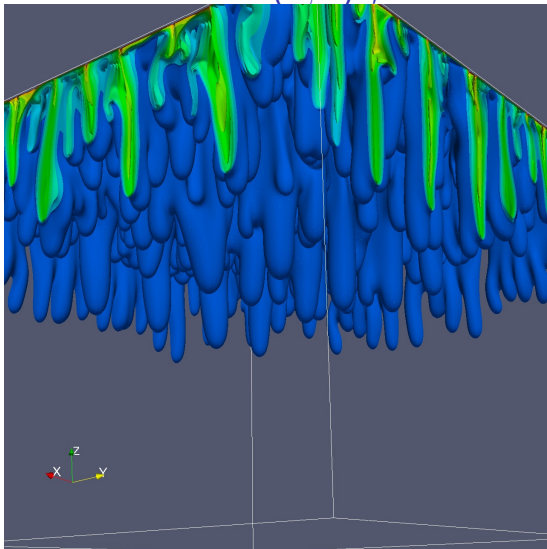
8 × Xeon E5-2680v2 10 core,  $240^3$ ,  $Q_2$ ,  $3.7 \cdot 10^8$  DOF, 16000 steps, 64h.

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

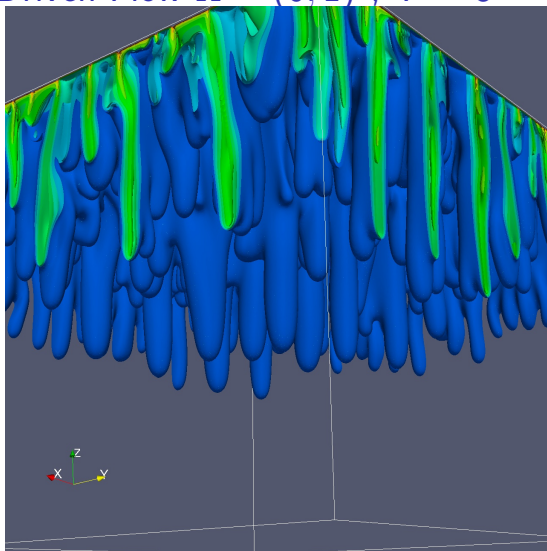
$8 \times$  Xeon E5-2680v2 10 core,  $240^3$ ,  $Q_2$ ,  $3.7 \cdot 10^8$  DOF, 16000 steps, 64h.

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  $\Omega = (0, 1)^3$ ,  $T = 5$ 

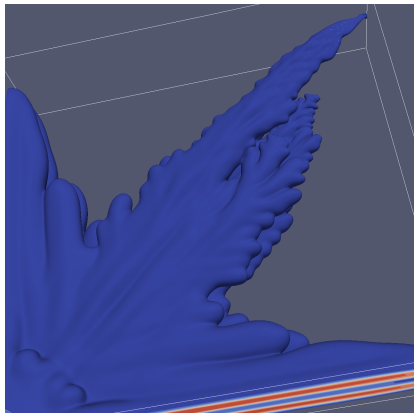
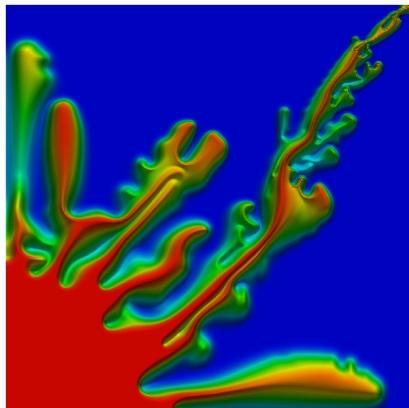
$8 \times$  Xeon E5-2680v2 10 core,  $240^3$ ,  $Q_2$ ,  $3.7 \cdot 10^8$  DOF, 16000 steps, 64h.

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

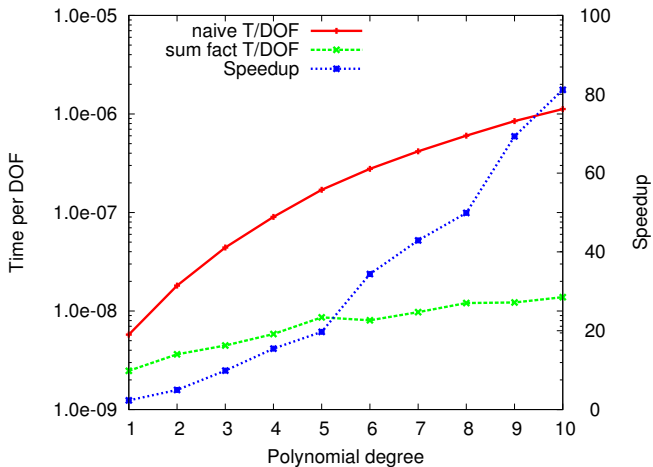
$8 \times$  Xeon E5-2680v2 10 core,  $240^3$ ,  $Q_2$ ,  $3.7 \cdot 10^8$  DOF, 16000 steps, 64h.

## 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



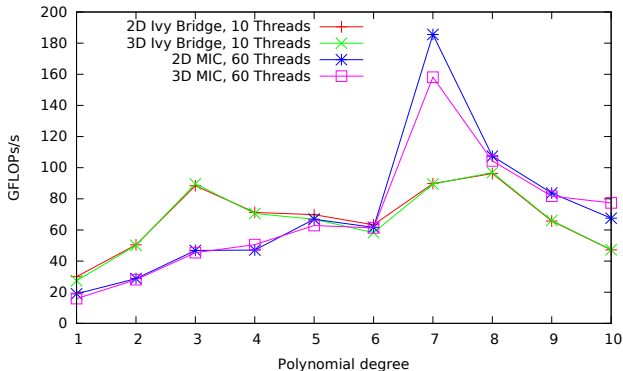
# Sum Factorization in 3D



Except  $Q_1$  in 2d, sum factorization is always faster!

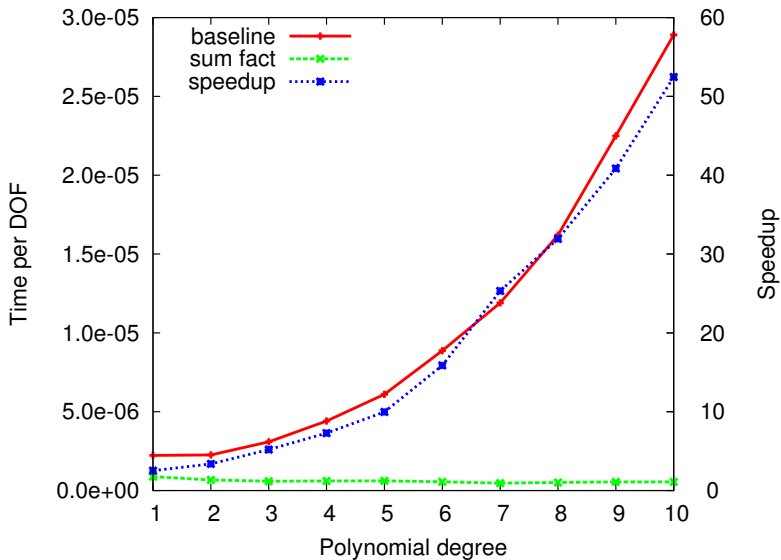


# Floating-Point Performance: Auto vectorizer, threaded

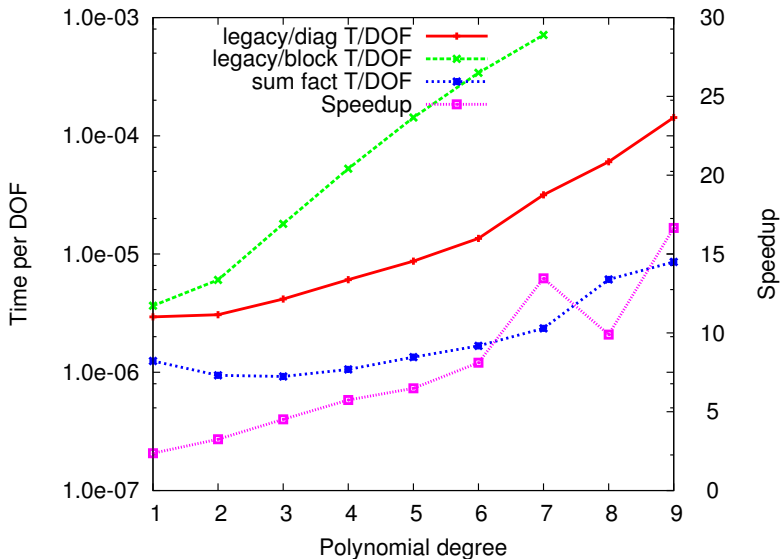


- Multithreaded version simulating many elements
- icc with auto vectorization / SIMD hints
- Intel MIC (61 cores) vs. Xeon E5-2680v2 (Ivy Bridge, 10 cores)

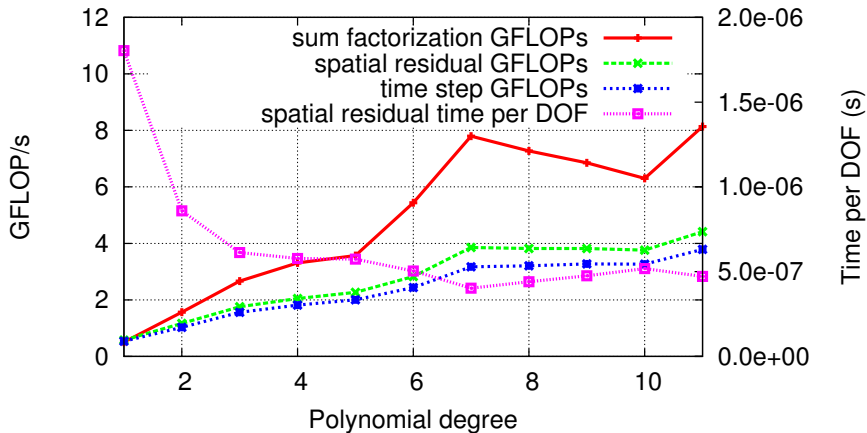
# DG Spatial Residual Evaluation (3D)



# Complete Explicit Time Step 3d



# GFLOP Rates Per Core



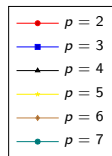
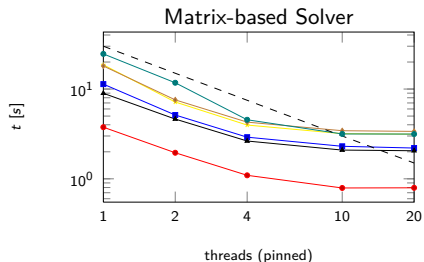
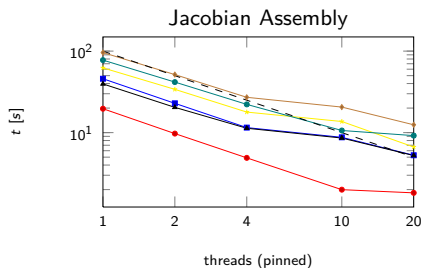
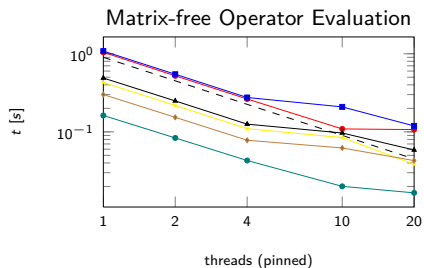
Xeon E5-2680v2 10 core processor (22 GFLOPS peak / core)

Residual: 4 GFLOPS

Sum factorization alone: 8 GFLOPS

# Strong Scaling Multithreaded Version on Intel Xeon

Xeon E5-2680v2 – Ivy Bridge, 10 cores, 2.8 GHz, Hyperthreading

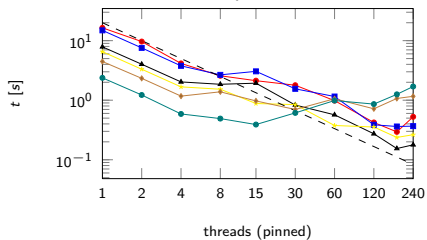


3D Poisson,  $\Omega = (0, 1)^3$

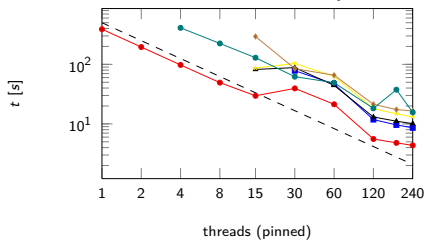
# Strong Scaling Multithreaded Version Xeon Phi

Xeon Phi 7120P – 61 cores (60 used), 1.2 GHz, 16 GB RAM

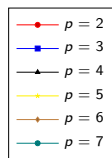
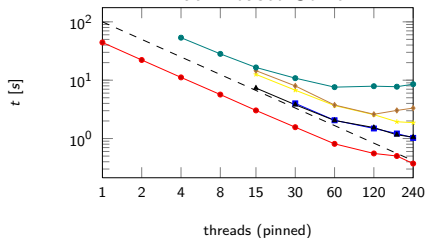
### Matrix-free Operator Evaluation



### Jacobian Assembly



### Matrix-based Solver



3D Poisson,  $\Omega = (0, 1)^3$

## 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 <sup>1</sup>
TASS <i>old</i>	0.02	0.39	3.6	21.2	91.8	319.1	971.2
TASS <i>new</i>	0.02	0.17	0.8	2.8	8.2	25.1	59.3
#IT BILU <sup>1</sup>	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 <sup>3</sup>	0.3	0.9	2.0	3.4	7.1	13.9	20.8

<sup>1</sup> Matrix has  $10^9$  nonzeros!

<sup>2</sup> does not include ILU decomposition! → *takes about TASS old!*

<sup>3</sup> estimated from time for residual evaluation

→ Matrix-free on DG + AMG on  $Q_1$  conforming (or  $P_0$ ) subspace

# Summary

- All methods in this talk have been implemented in the DUNE software framework ([www.dune-project.org](http://www.dune-project.org))
- SF based DG implementation gives huge savings for matrix-free and matrix based approaches
- SF can handle variable coeffs, non-affine geometry, nonlinear problems
- Good SIMD performance is achieved for larger polynomial degrees
- First results on Intel Phi look promising
- Future work
  - ▶ Code generation or intrinsics might be necessary to achieve still better performance
  - ▶ Hybrid (matrix-free/matrix-based) preconditioners for implicit DG formulations
  - ▶ **Disclaimer:** Use moderate polynomial degree in practice!



All the best to Jean and Jérôme ...



... hopefully you may visit some nice conferences also in retirement!