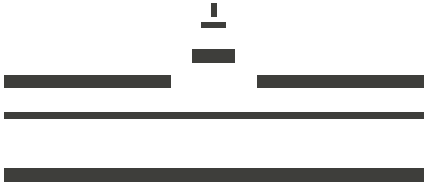


Westfälische
Wilhelms-Universität
Münster

Advances concerning multiscale methods and uncertainty quantification in Exa-Dune

living.knowledge
WWU Münster

P. Bastian, C. Engwer, J. Fahlke, M. Geveler, D. Göddeke, O. Iliev, O. Ippisch, R. Milk, J. Mohring, S. Müthing, M. Ohlberger, D. Ribbrock, S. Turek

January 25th, 2016

Outline

- Abstraction of Multiscale Methods
- Implementation + Results
- MLMC
- Integration with MLMC Framework

EXA-DUNE:

"Flexible PDE Solvers, Numerical Methods, and Applications"

- 7 groups from 5 german universities
- Aim: develop new numerical, algorithmic and computational techniques to enable exa-scale computing for PDEs on heterogeneous massively parallel architectures
- DUNE and FEAST (TU Dortmund, hardware-oriented numerics)

Problem Statement

Assume we are looking for $u^\epsilon \in U$ solving

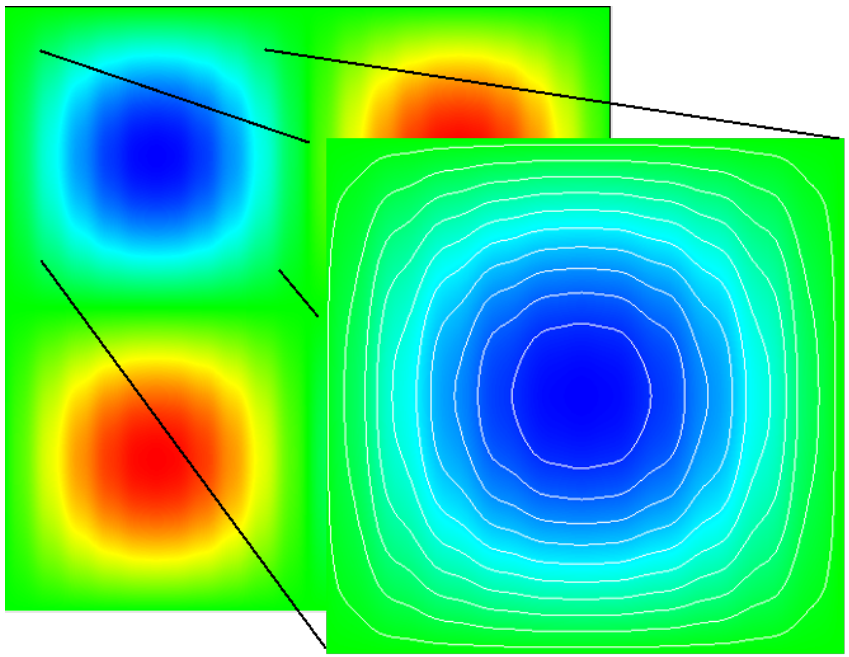
$$R^\epsilon[u^\epsilon](v) = 0 \quad \forall v \in U$$

where $R^\epsilon : U \mapsto U'$
Example:

$$U = H_0^1(\Omega), \qquad R^\epsilon[u] = b^\epsilon[u] - l$$
$$b^\epsilon[u](v) = \int_\Omega A^\epsilon \nabla u \nabla v, \qquad l(v) = \int_\Omega fv$$
$$A^\epsilon(x) = A\left(\frac{x}{\epsilon}\right) : \Omega \mapsto \mathbb{R}^{d\times d}$$

Scales

- Idea: Make use of possible scale-separation
- Macroscopic scale represented by coarse-scale space $U_H \subset U$ on coarse partition $\mathcal{T}_H$ of Ω
- Microscopic scale represented by fine-scale space $U_h \subset U$ on fine partition $\mathcal{T}_h$ of Ω
- Spaces should be nested $U_H \subset U_h \subset U$



Abstract Method

- Let $\pi_{U_H} : U \mapsto U_H$ denote a projection onto the coarse space with

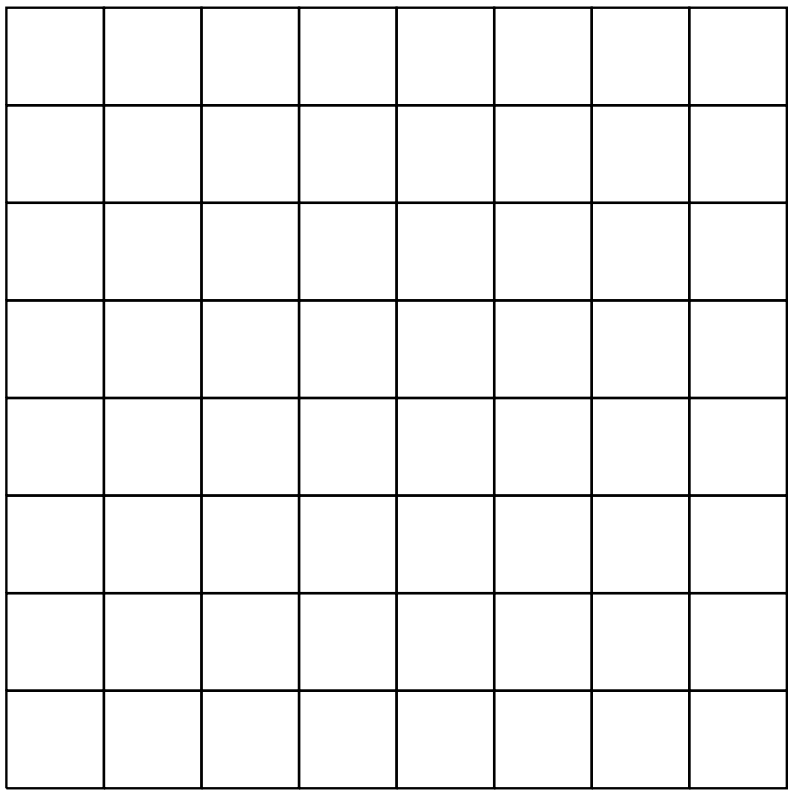
$$U_{f,h} = \{u_h \in U_h : \pi_{U_h}(u_h) = 0\}$$

- Compute and approximate discrete solution

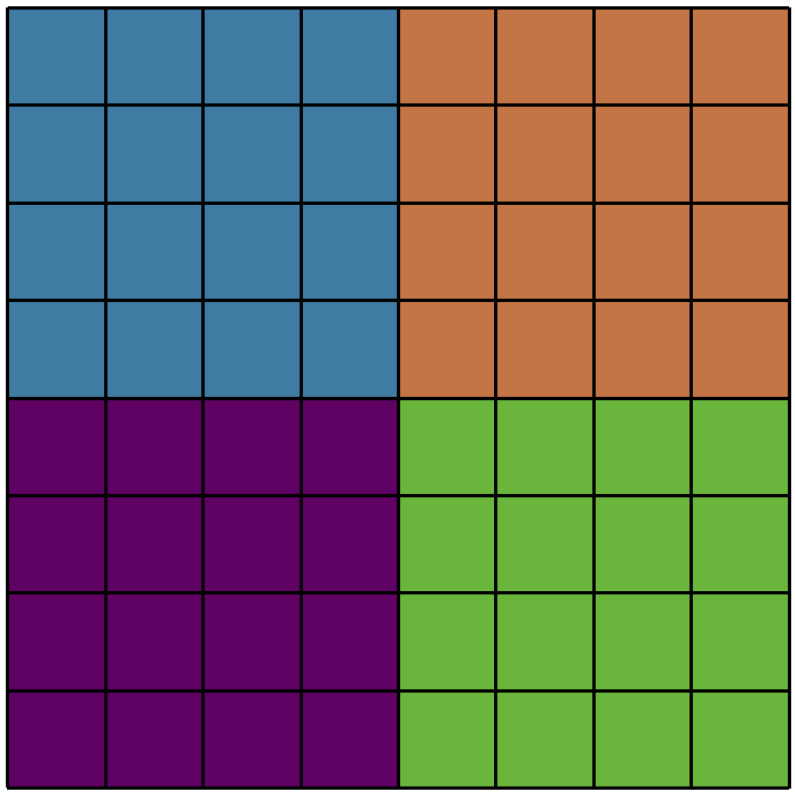
$$\begin{aligned} u^{\epsilon}_{\mu,h} = u_H + u_{f,h} \in & U_H \oplus U_{f,h} \text{ satisfying} \\ R^{\epsilon}_{\mu}[u_H + u_{f,h}](v_H) &= 0 \quad \forall v_H \in U_H \\ R^{\epsilon}_{\mu}[u_H + u_{f,h}](v_{f,h}) &= 0 \quad \forall v_{f,h} \in U_{f,h} \end{aligned}$$

- Concrete choices for U_H, U_h, π_{U_H} and further localization of fine scale equation yield various multiscale methods.

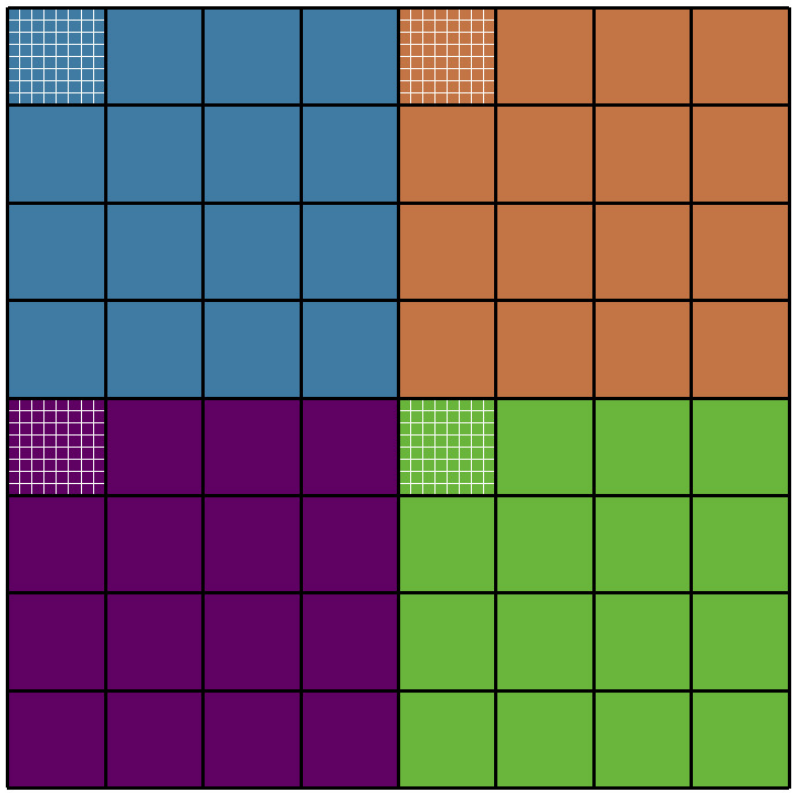
Pure MPI Implementation



Pure MPI Implementation



Pure MPI Implementation



Strong Scaling Setup

- CG-FEM Discretizations via DUNE-gdt with DUNE-pdelab backend
- YASPGRID (cubes) from DUNE-grid on coarse and fine scale
- 64^3 cubes on coarse scale with 8^3 fine cells each, 134.217.728 total
- Hardware: SuperMUC Phase 2, 256 to 8192 MPI ranks
- coarse system: DUNE-istl BiCGStab with ILUT preconditioning
- UMFPACK direct sparse solve for local problems

Testcase Setup

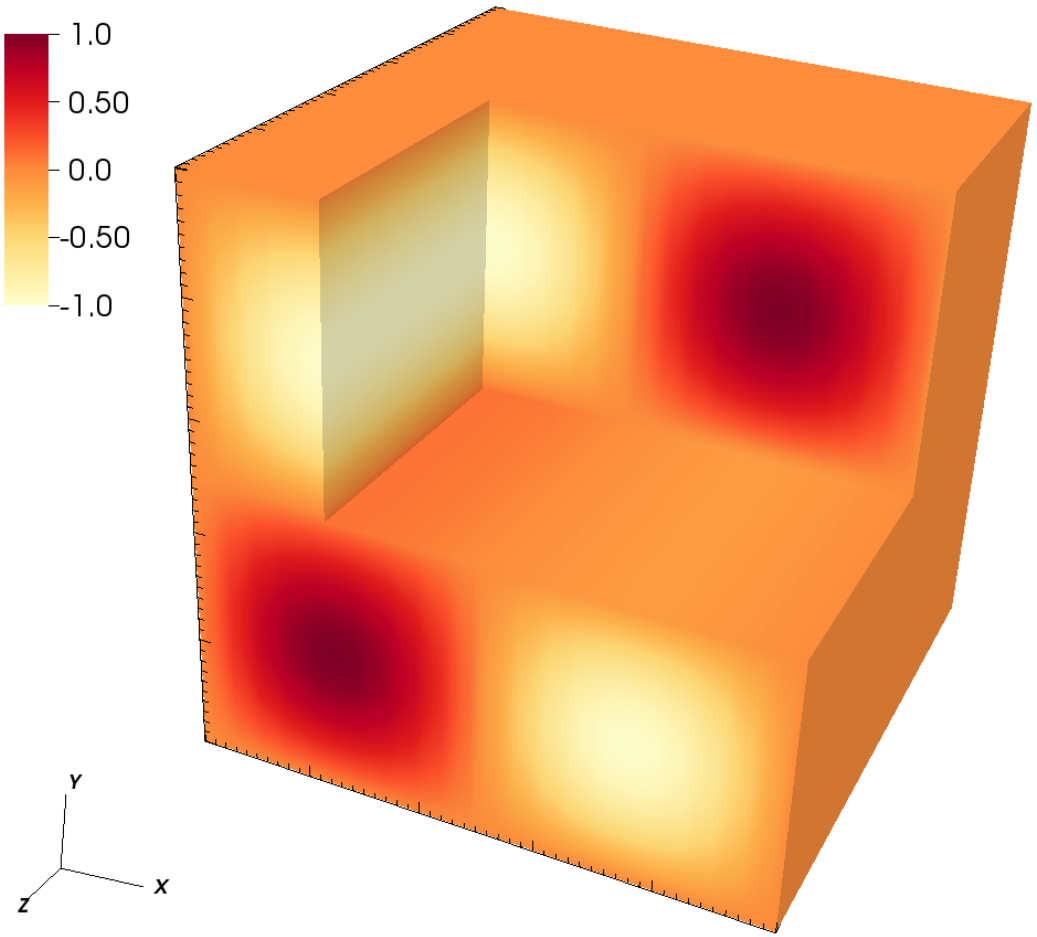
Standard diffusion problem in $\Omega = [0,1]^3$, with boundary conditions $u(x) = 0$ on $\partial\Omega \backslash x_3 \in \{0,1\}$ and Neumann-zero elsewhere. With diffusion A^ϵ and source f^ϵ given as:

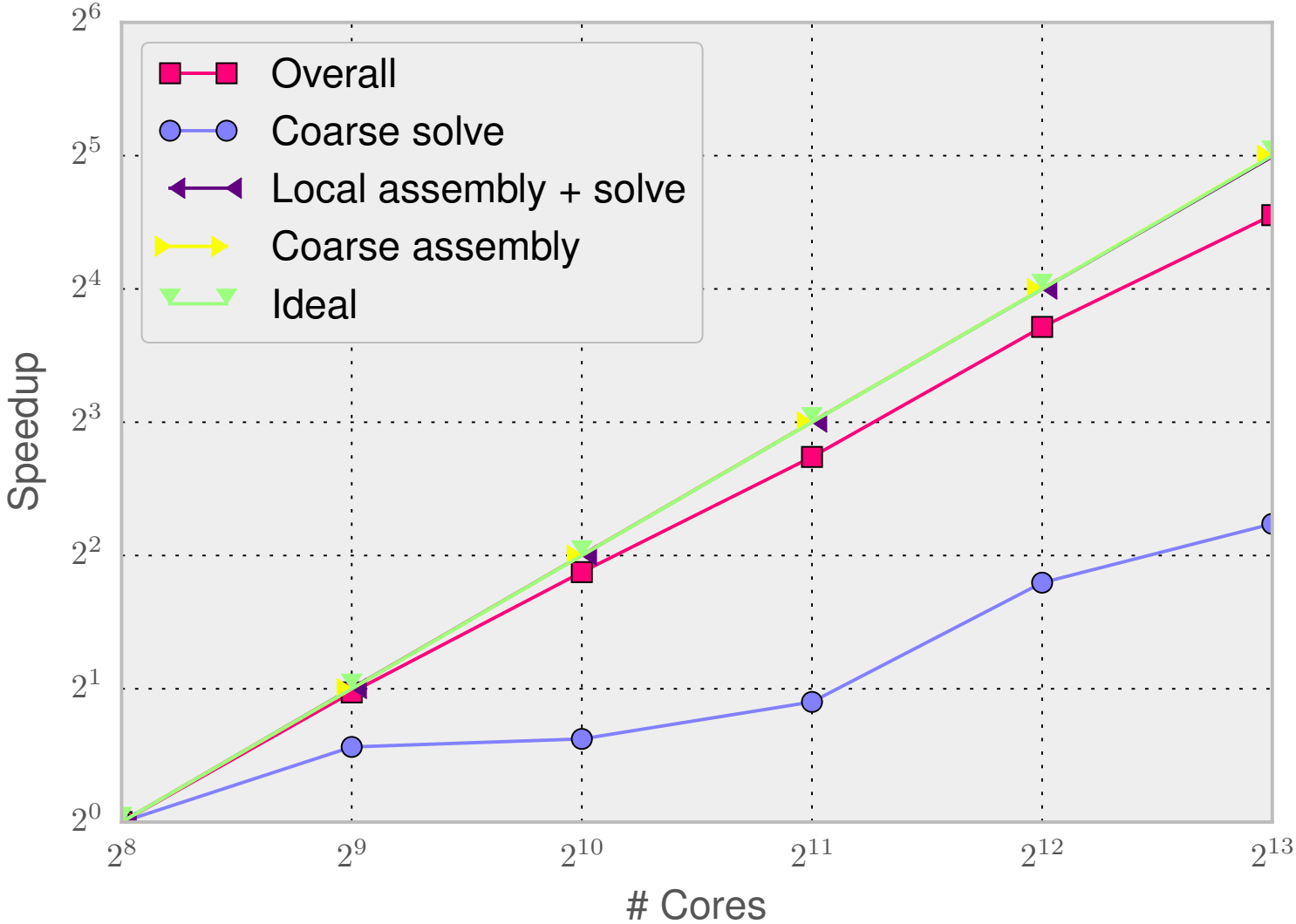
$$A^\epsilon(x_1,x_2,x_3) := \frac{1}{8\pi^2} \begin{pmatrix} 2\left(2+\cos\left(2\pi\frac{x_1}{\epsilon}\right)\right) & 0 & 0 \\ 0 & 1+\frac{1}{2}\cos\left(2\pi\frac{x_1}{\epsilon}\right) & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$f^\epsilon(x) := -\nabla\cdot\left(A^\epsilon(x)\nabla v^\epsilon(x)\right)$$

$$v^\epsilon(x_1,x_2,x_3) := \sin(2\pi x_1)\sin(2\pi x_2) + \frac{\epsilon}{2}\cos(2\pi x_1)\cos(2\pi x_2)\sin\left(2\pi\frac{x_1}{\epsilon}\right)$$

Analytical Solution





Problem: Coarse solve

- Very few degrees of freedom per MPI-rank. Strong Scaling ends with only 32 cells per rank.

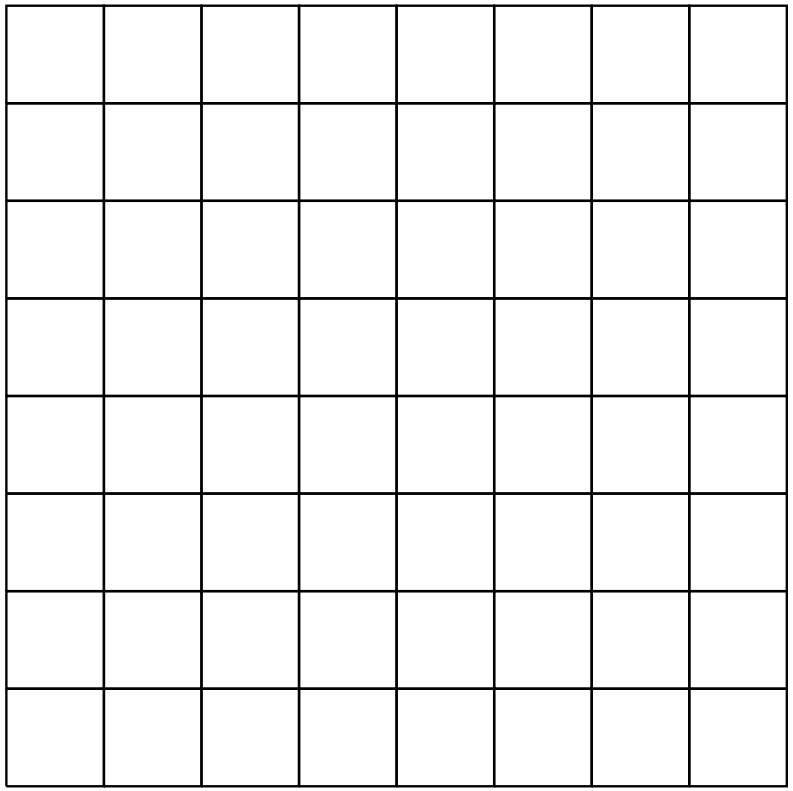
Problem: Coarse solve

- Very few degrees of freedom per MPI-rank. Strong Scaling ends with only 32 cells per rank.
- Decrease number of ranks while increasing amount of work per rank!

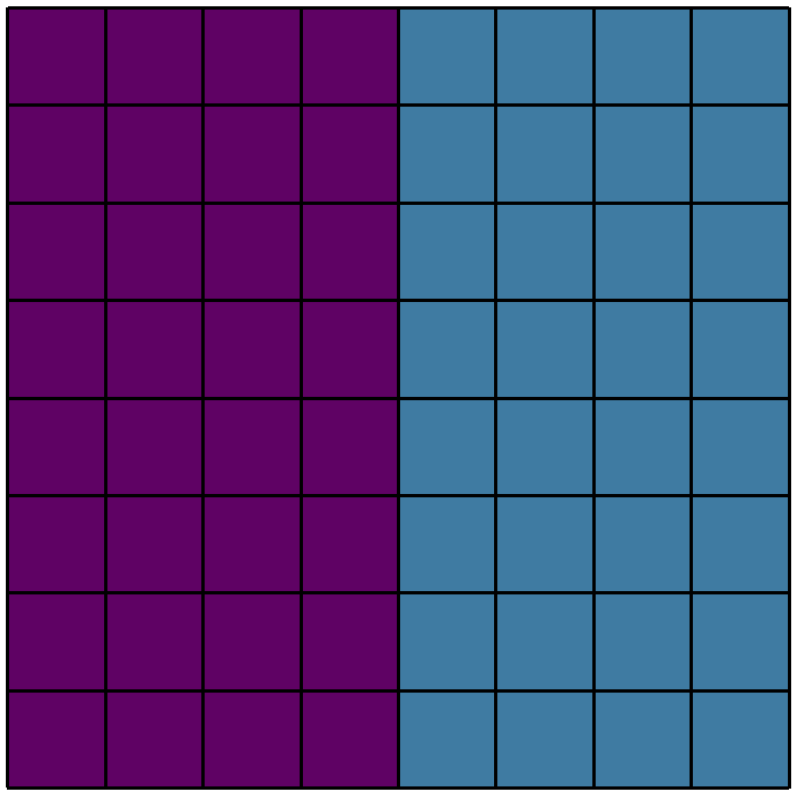
Problem: Coarse solve

- Very few degrees of freedom per MPI-rank. Strong Scaling ends with only 32 cells per rank.
- Decrease number of ranks while increasing amount of work per rank!
- Simple for local problems: they're already embarrassingly parallel

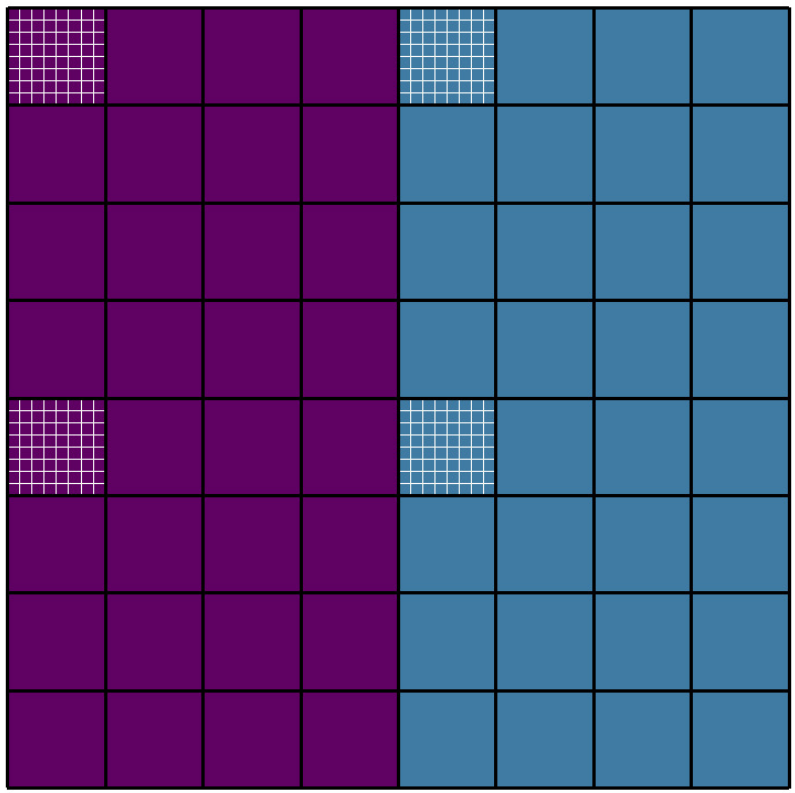
Hybrid MPI/Shared Memory Implementation



Hybrid MPI/Shared Memory Implementation

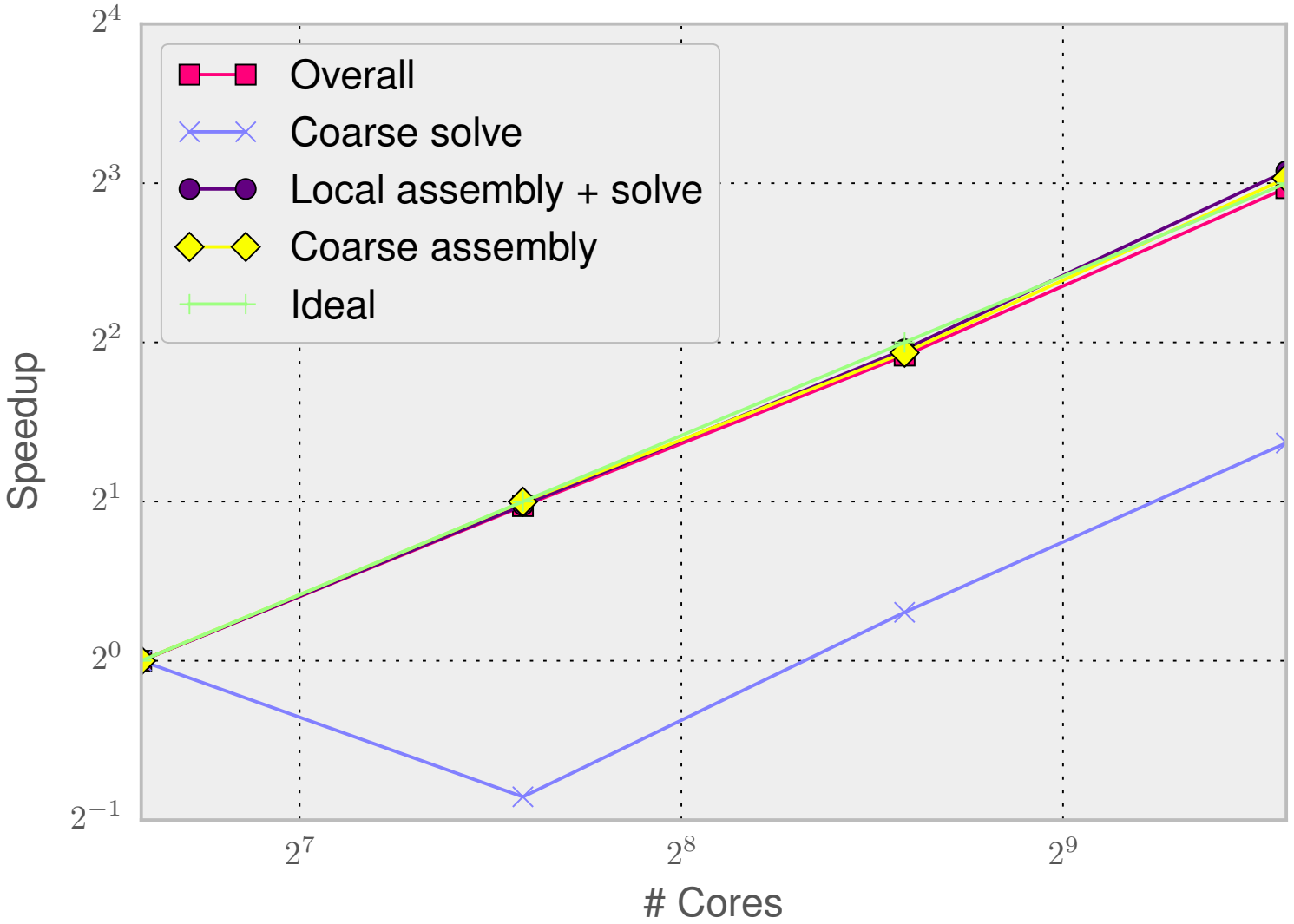


Hybrid MPI/Shared Memory Implementation



Hybrid Strong Scaling Setup

- 8^3 cubes on coarse scale with 32^3 fine cells each, 16.777.216 total
- Hardware: CHEOPS (RRZK Cologne)
- 16 to 128 MPI-ranks, one hexacore CPU per rank: 96 to 768 cores used



UQ and MC

- Consider ground water flow through some real world domain

UQ and MC

- Consider ground water flow through some real world domain
- Problem: full permeability field not accessible

UQ and MC

- Consider ground water flow through some real world domain
- Problem: full permeability field not accessible
- (With some assumptions) Possible to determine its many parameters from a number of measurements

UQ and MC

- Consider ground water flow through some real world domain
- Problem: full permeability field not accessible
- (With some assumptions) Possible to determine its many parameters from a number of measurements
- (Aggregate) Quantities of interest $Q(\omega)$ (ex. total flow trough domain) can only be characterised by stochastical distributions

UQ and MC

- Consider ground water flow through some real world domain
- Problem: full permeability field not accessible
- (With some assumptions) Possible to determine its many parameters from a number of measurements
- (Aggregate) Quantities of interest $Q(\omega)$ (ex. total flow trough domain) can only be characterised by stochastical distributions
- Conventionally this uncertainty is quantified by means of Monte Carlo methods

UQ and MC

- Consider ground water flow through some real world domain
- Problem: full permeability field not accessible
- (With some assumptions) Possible to determine its many parameters from a number of measurements
- (Aggregate) Quantities of interest $Q(\omega)$ (ex. total flow trough domain) can only be characterised by stochastical distributions
- Conventionally this uncertainty is quantified by means of Monte Carlo methods
- But: Notoriously bad convergence

MLMC Ideas

The main idea behind Multi Level Monte Carlo Method is to split

$$Q(\omega) = Q_0(\omega) + Q_1(\omega) - Q_0(\omega) + \cdots + Q_L(\omega) - Q_{L-1}(\omega) + Q(\omega) - Q_L(\omega)$$

with $L + 1$ increasingly accurate (+ time intensive) solvers. We reduce its variance accordingly, with many quick evaluations for the coarse solution and few slower solves for the differences. These solves are independent of one another $\rightarrow$ trivial parallelization. MLMC can greatly improve order of convergence compared with MC.

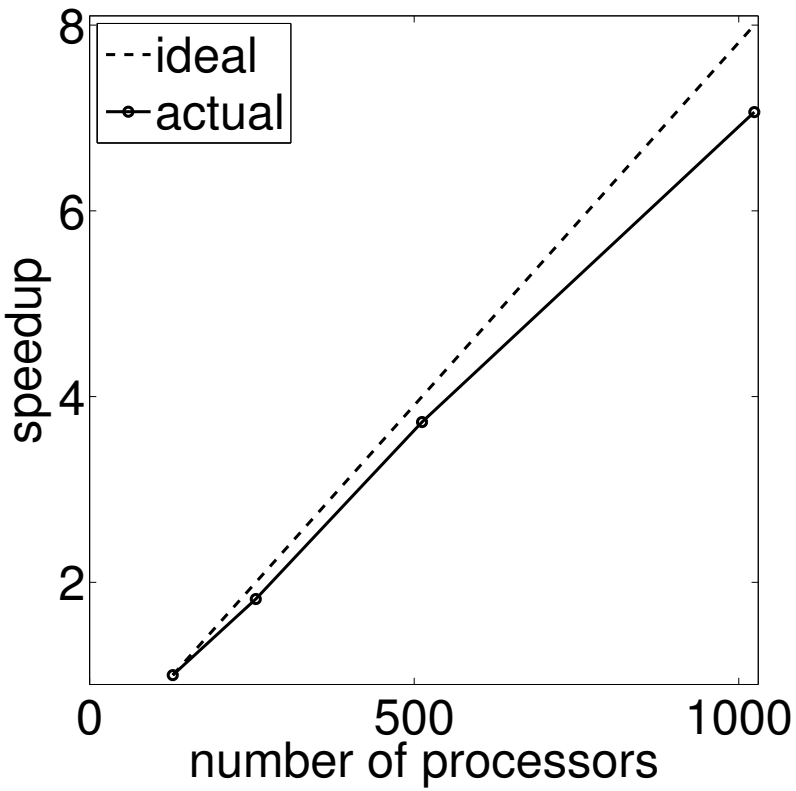
MLMC Implementation

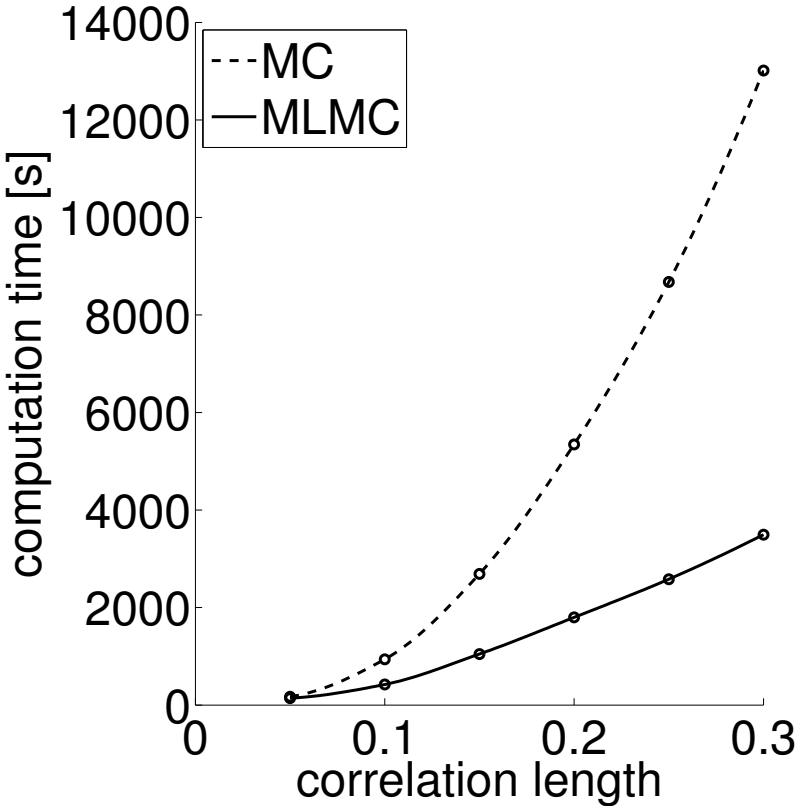
- DUNE module with very little requirements on pluggable solvers
- Optimizes selection of slow/fast solves to minimize time to solution
- Distributes work across groups of MPI-Ranks
- Tries to mimimize the communication overhead for estimating mean times and variances

MLMC Testcase setup

- stationary single phase flow through a unit cube
- random permeability field
- Boundary condition: constant pressure difference between left and right end faces
- Aggregate quantity $Q(\omega)$ sought is the total flow through the cube
- up to 1024 CPUs of the ITWM Cluster in Kaiserslautern

Scaling of a 2-level MsFEM/FEM and comparison with Standard MC





Thank you for your attention.

Get the code:

`https://github.com/wwu-numerik/dune-multiscale`

`https://github.com/wwu-numerik/dune-mlmc`