Large-Eddy Simulation Code for City Scale Environments

Vladimír Fuka, Zheng-Tong Xie

University of Southampton, Aerodynamics and Flight Mechanics

V.Fuka@soton.ac.uk, Vladimir.fuka@matfyz.cz
Z.Xie@soton.ac.uk

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk).

Outline

- Embedded CSE
- Motivation for the project
- Model ELMM
- Project targets / work packages
- Code optimizations
- One-way domain nesting
    - Turbulent inflow generation
- Two-way domain nesting
- Performance tests
- Conclusions

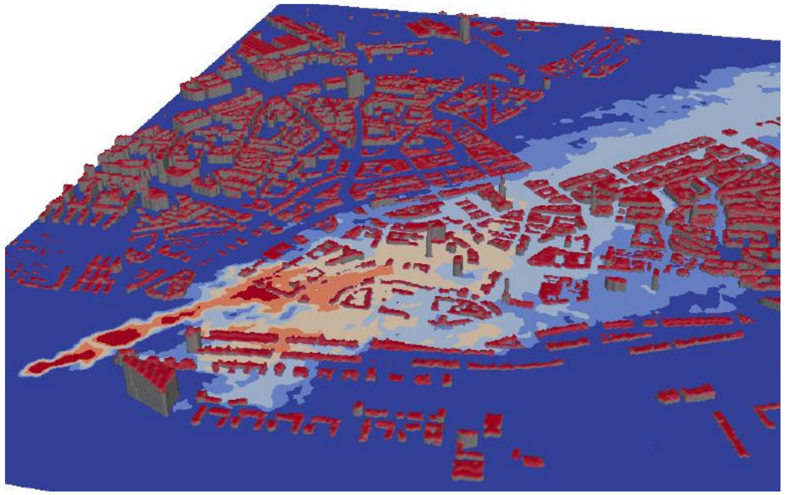
eCSE - Embedded CSE

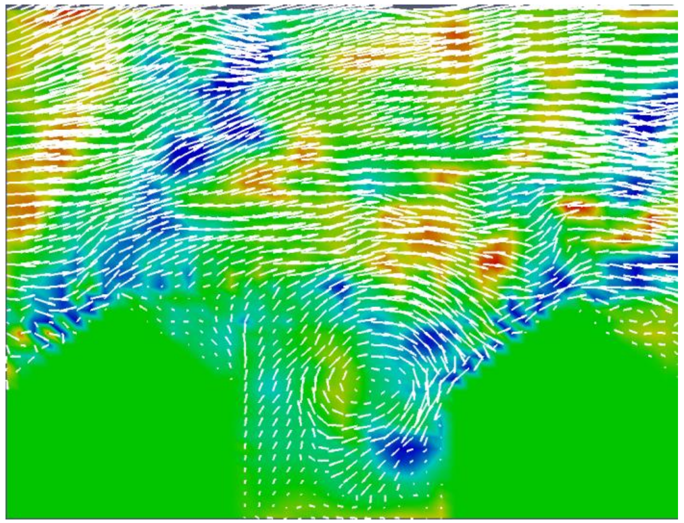
- Funding programme by ARCHER to develop software.
- Implementation of new algorithms.
- Improvements of scalability to high core counts.
- Improvements to enable new science.
- Porting existing codes to ARCHER

- This project was funded for 1 year.

Motivation for the project

- Coarse grid large eddy simulation (LES) on large areas
  - Complex terrain
  - Convective boundary layer structures
  - City scale (~10 km) effects
- Fine grid (1 m) LES in nested domains
  - Street canyons
  - Individual buildings
- Structured orthogonal grids
  - Simple implementation
  - Fast solution
  - High order discretizations





Model ELMM

- Extended Large-eddy Microscale Model (previously in-house code CLMM - Charles University LMM)
- Open source (GPL). Source code repository: https://bitbucket.org/LadaF/elmm
- Fortran 2008, parallel
- In the past mostly used for flow and pollution dispersion around buildings and in street canyons and street networks.
- Also used for simulations of the stable and convective boundary layer or the whole diurnal cycle of the atmospheric boundary layer.
- Used for simulations of a large city centre 2 km x 2 km within project COST ES1006, but in a coarse resolution.

Large Eddy Simulation

- Time-dependent incompressible Navier-Stokes equations filtered by a spatial filter

$$\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u}\boldsymbol{u}) = -\nabla p + \nabla \cdot \left((\nu + \nu_{\mathrm{sgs}}) \nabla \boldsymbol{u} \right),$$

$$\nabla \cdot \boldsymbol{u} = 0,$$

$\boldsymbol{u}$ velocity vector, p pressure, ν molecular viscosity of the air,

ν_{sgs} subgrid (eddy) viscocity — modelled by a subgrid stress model

- Only the scales describable by the computational grid (resolved scales) are simulated by the filtered N.-S. equations.
- The effect of the small scales approximated by the eddy viscosity. Not all models use eddy viscosity.
- In ELMM also fields of passive scalars, temperature, water vapour.

Numerical methods in ELMM

- Pressure-velocity coupling:
  - Projection (fractional step) method
- Temporal discretization:
  - 3rd order explicit Runge-Kutta
- Spatial discretization
  - 2nd order central differences
  - optionally 4th order central advection
- Subgrid modelling:
  - Smagorinsky model
  - Vreman model
  - σ model (this study)
  - mixed time scale model
- Immersed boundary method for obstacles on the uniform grid

Poisson solver

- Very important part of the projection method is the pressure correction (projection).
- Expensive step which enables the usage of the incompressible Navier-Stokes equations instead of very expensive compressible equations.
- Solution of Poisson equation

$$\nabla^2 \varphi = \nabla \cdot u$$

- Many iterative and direct methods available. On uniform grid the Fast Fourier transform method is especially effective.
- Custom open-source library PoisFFT. Separate repository, but it is a part of ELMM. https://github.com/LadaF/PoisFFT
- Computational complexity of FFT grows as O(*N* log(*N*)) — implications for weak scaling.

PoisFFT

- V. Fuka, 2015, *PoisFFT - a free parallel fast Poisson solver*, Applied Mathematics and Computation 267.
- Boundary conditions periodic, zero value Dirichlet or zero gradient Neumann. May differ in individual directions, but only certain combinations implemented.
- Written in Fortran 2008, bindings for C and C++.
- Depends on the common FFTW library (available on ARCHER and in Linux distributions).
- At the start of the project parallel using OpenMP or MPI, not both at the same time.
- MPI version requires PFFT, the 2D decomposition wrapper over FFTW. Available at https://github.com/mpip/pfft, PFFT was demonstrated to scale up to 262k cores on BlueGene/Q

Targets of the project

Main target:

Enabling simulations at city-scale — domains of size ~10 km with resolution down to 1 m.

Work packages:

1. WP1 Optimization and hybrid parallelization
   - serial performance
   - optimizations of MPI communications
   - hybrid OpenMP/MPI
2. WP2 Domain nesting
   - one-way nesting
   - two way nesting

Work package 1

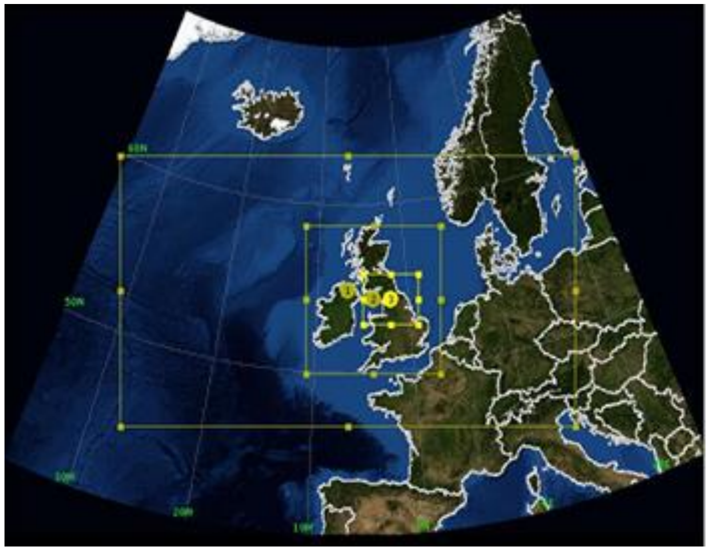
- Code of the most important subroutines in the dynamics core rewritten to avoid branch statements in the innermost loops.
- The `collapse()` OpenMP clause added where possible and where profiling identified scalability problems in the outermost index only.
- The MPI exchange of ghost-cell values between neighbouring processes rewritten to non-blocking MPI communication. This avoids the previous need for synchronization and splitting the communication in an odd-even pattern.
- Test shows single node speed-up around 20%.

Work package 1

- Existing OpenMP and MPI parallelization was combined as hybrid MPI/OpenMP paralelization.
- Hybrid parallelization reduces the number of MPI processes and thus also the large amount of network messages.
- OpenMP only FFT in FFTW much faster than the MPI one.
- Introduces one additional level of synchronization.
- MPI routines can be called only from one OpenMP thread. Synchronization necessary.

Work package 2

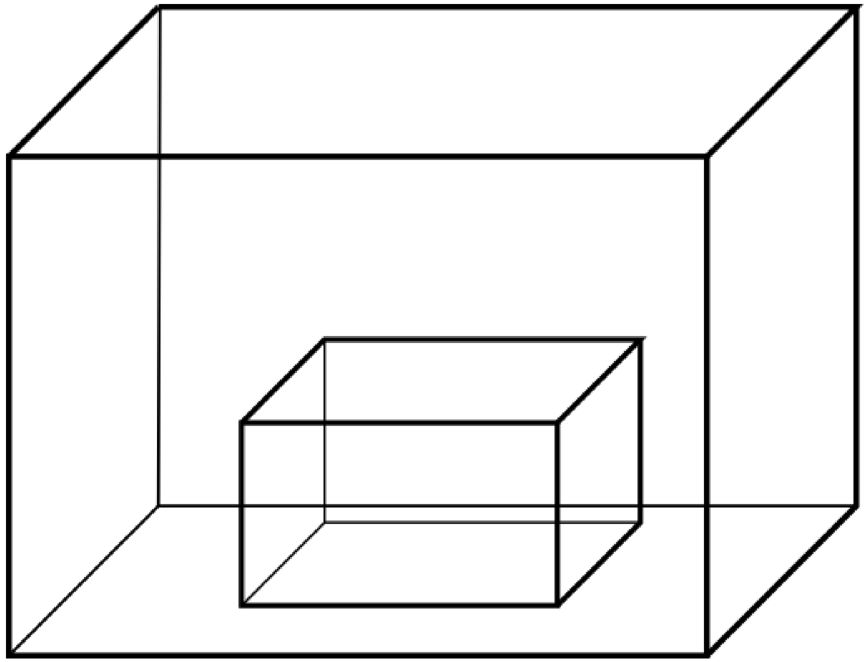
- Domain nesting (grid nesting) enables high resolution in selected areas
- Alternative to unstructured grid in general-purpose CFD codes.
- Common approach in weather prediction and climate modelling (but often in a simpler form), less common in CFD and in LES in particular.



Grid nesting in numerical weather prediction, http://www.hector.ac.uk/casestudies/wrf.php

One-way domain nesting

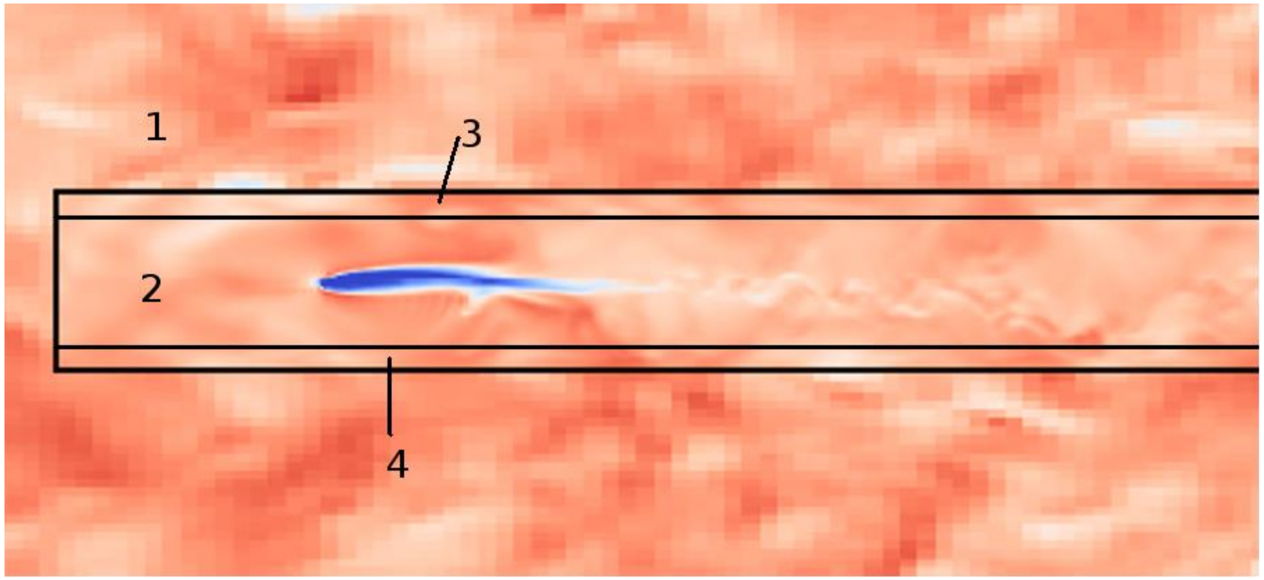
- Domains are separate model runs communicating with other domains. They have modified nesting boundary conditions.
- At the boundaries the outer domain fields and tendencies are interpolated on the nested grid.
- Buffer regions with nudging near boundaries.
- Refinement ratio arbitrary integer.



One-way domain nesting

- refinement ratio: nested domain has *n* x finer grid
- *n* time-steps in the nested domain per every time-step in the outer domain.
- Interpolations after every time-step of the outer domain.
- The domains are using separate MPI communicators (process sets) running on different CPUs.
- Flexibility – any domain can have an arbitrary number of inner (child) domains, each with different location, refinement ratio and other parameters.
- Can be also used to set-up the turbulent inlet generation with the flow recycling method.

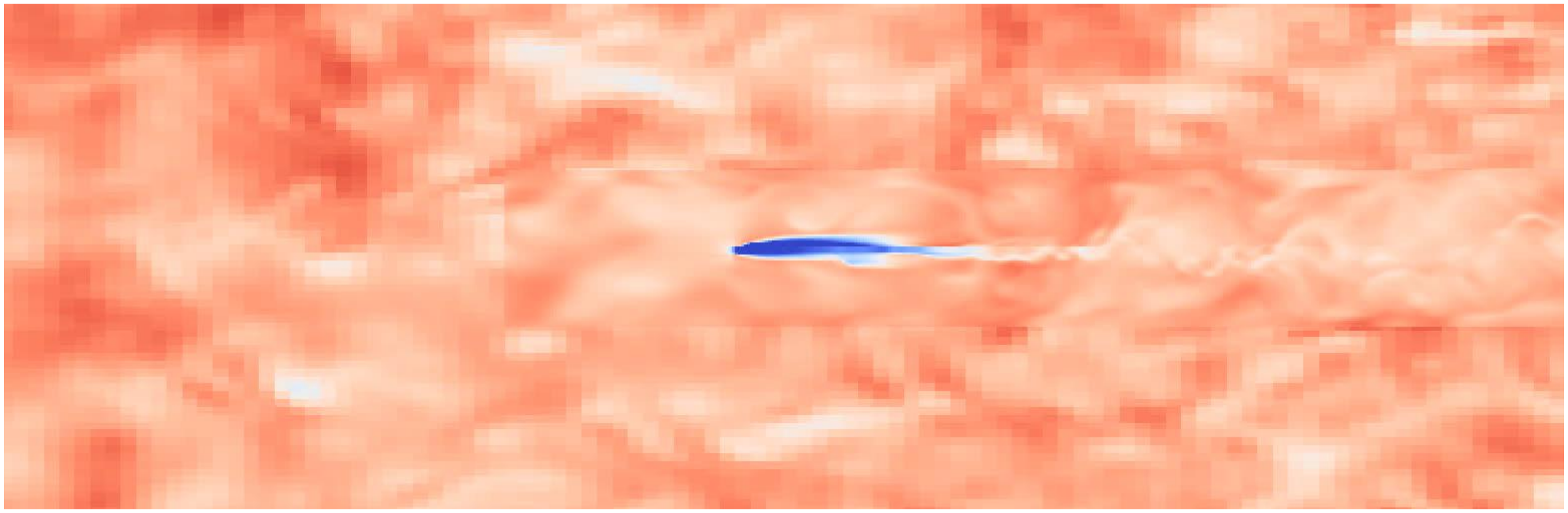
One-way domain nesting



1. Outer coarse-grid domain, only a small part shown. Turbulent inflow generation.

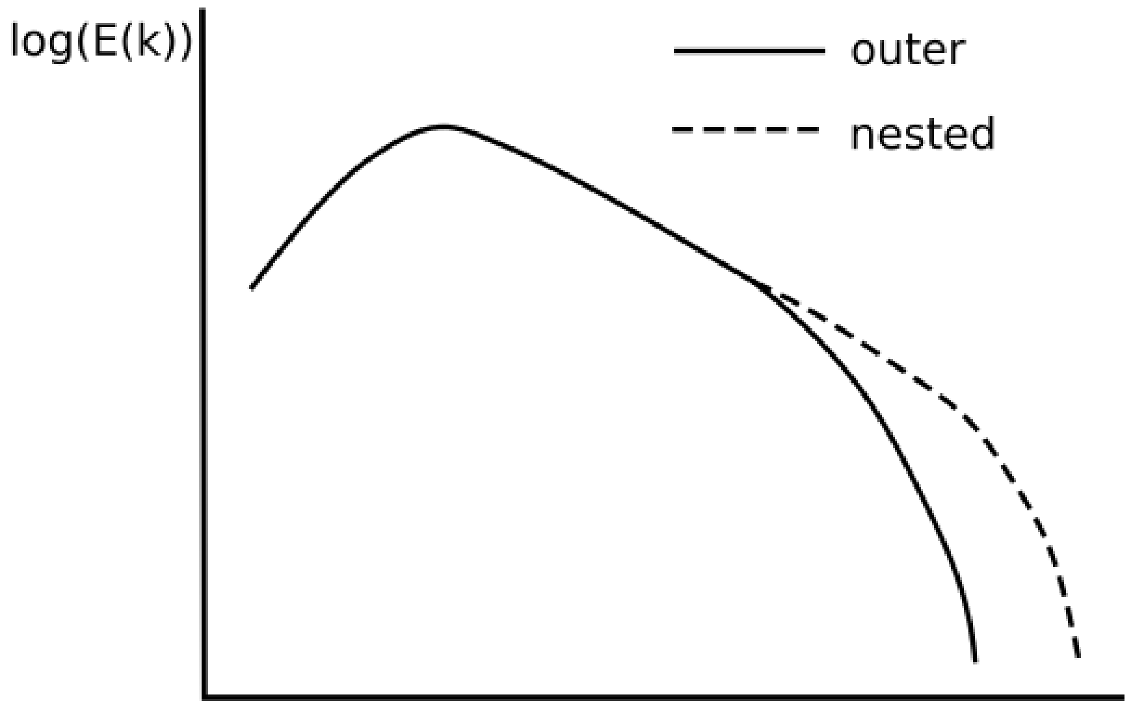
2. Inner fine-grid domain, refinement ratio 5.

3. and 4. Buffer zones where the inner grid is forced to be close to the outer grid solution. Necessary on boundaries where air can flow out of the domain.



Nesting turbulent flow

- Common approach - low grid resolution ratio (<=3) and straightforward interpolation
- Finer grid of the nested domain resolves eddies that were missing in the outer domain.
- The missing turbulent energy can be non-negligible and will affect the nested domain.



Inflow turbulence generation

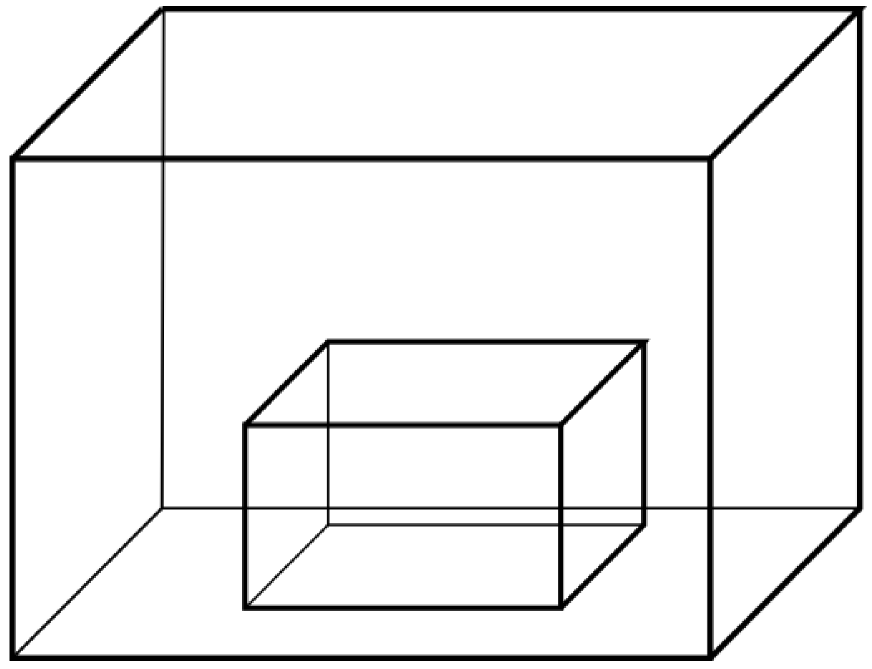
- Well established for generating synthetic turbulent inflow for LES (e.g. Xie and Castro 2006).
- Based on random numbers and filtering.
- Here only to generate the “missing” part of the spectrum
- “missing” TKE estimated from the outer resolved field (scale similarity):

$$k_{sgs} = (\hat{u} - u)^2 + (\hat{v} - v)^2 + (\hat{w} - w)^2$$

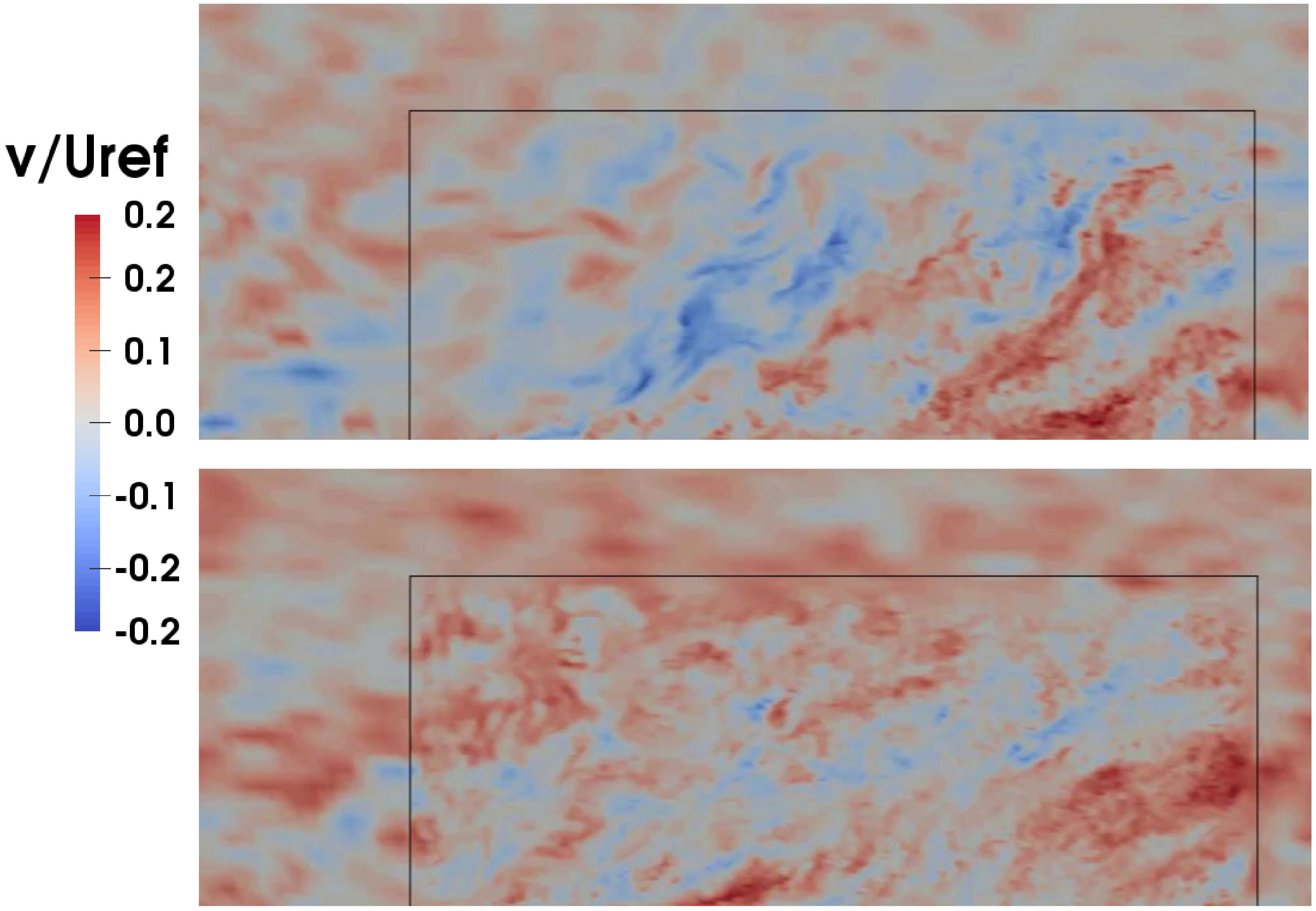
- ^ a test filter (Inagaki et al. 2003)

Test case – one way nesting with turbulent inflow generation

- neutral boundary layer over flat terrain (rough wall)
- periodic boundary conditions for the outer grid
- outer domain: 200 m x 100 m x 50 m
- nested domain: 50 m x 20 m x 20 m
- grid resolutions:
    - outer: $\Delta_o = 1$ m
    - nested: $\Delta_n = 0.2$ m
    - ratio: $\Delta_o/\Delta_n = 5$



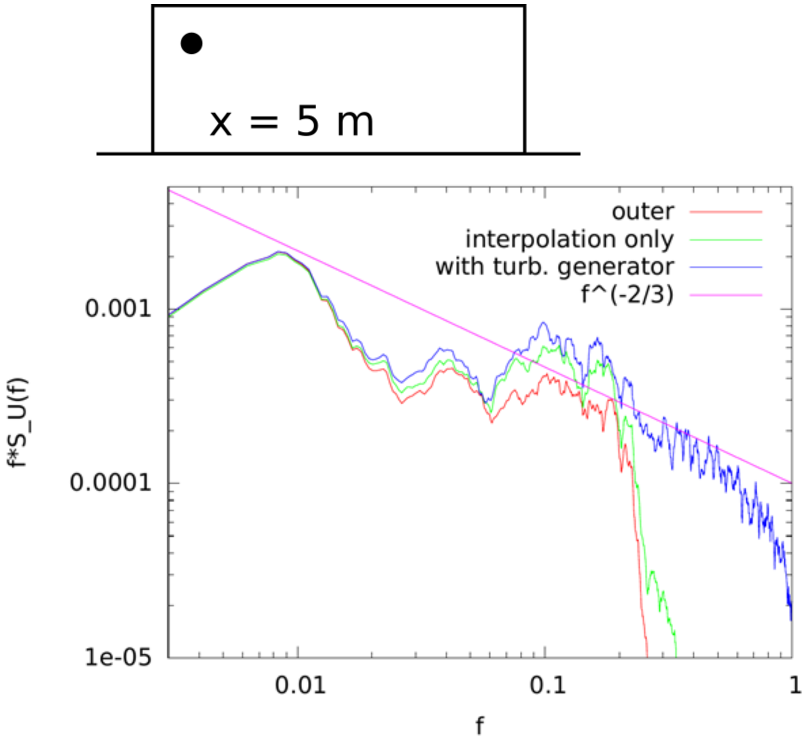
Spatial development

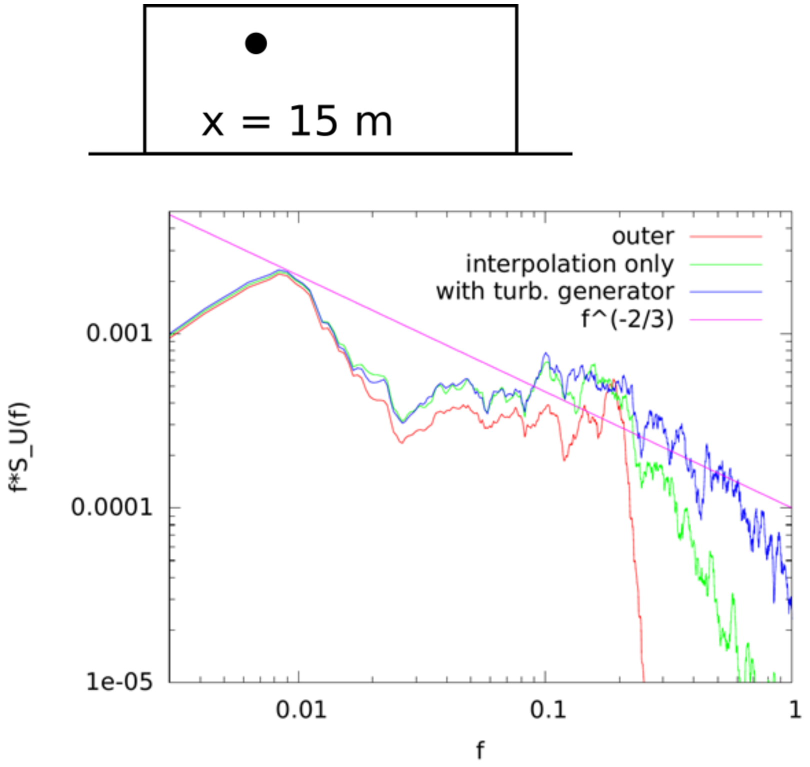


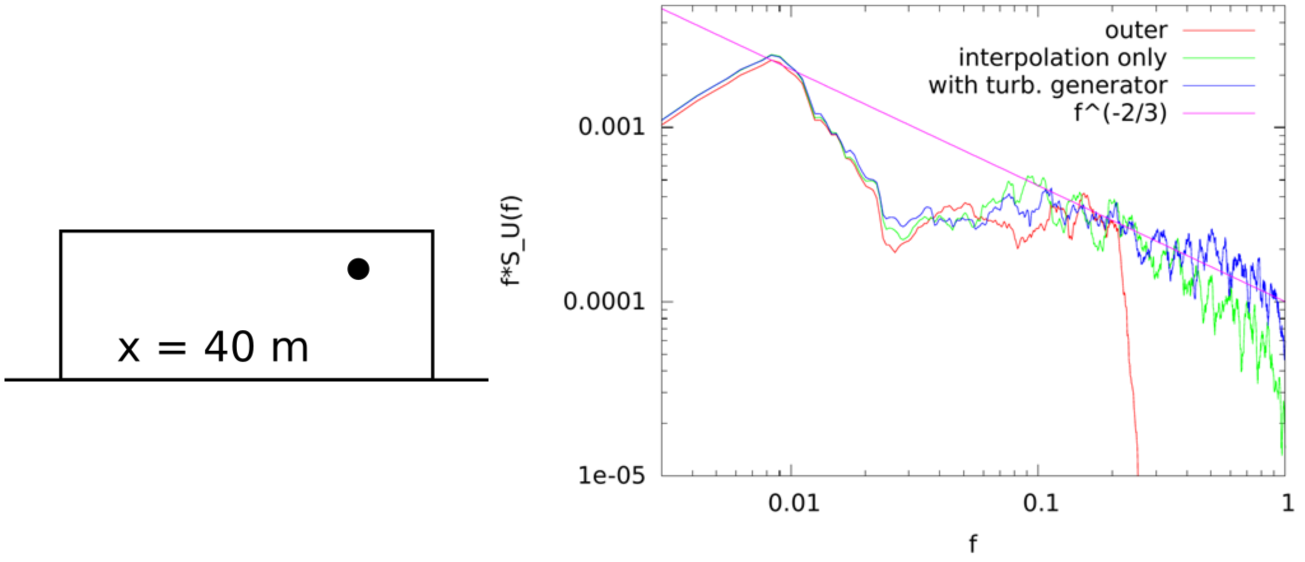
without turbulence generator

with turbulence generator

Test case – spectra







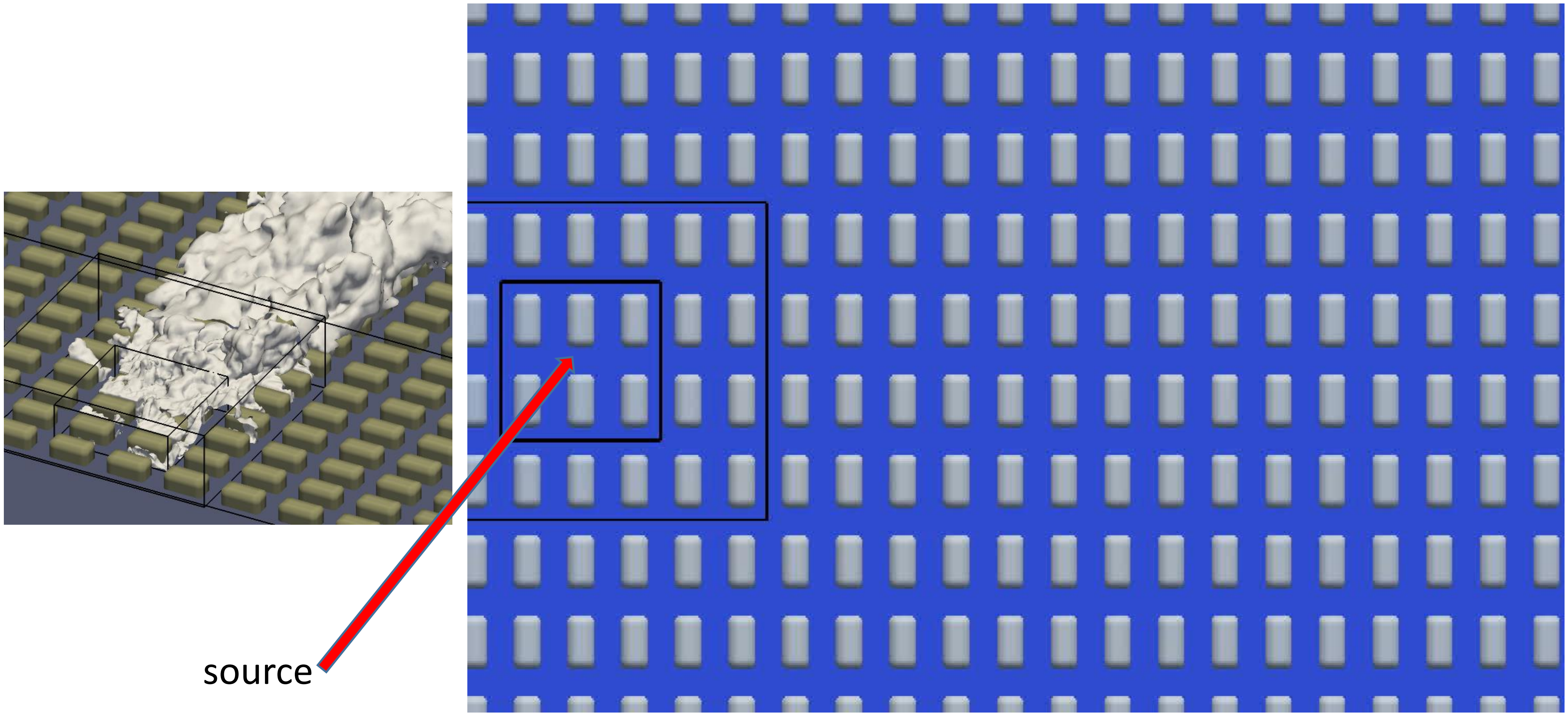
Two-way domain nesting

- Feed-back of the inner domain to the outer domain.
- Still area of active research.
- In ELMM especially useful when scalar source is in the inner domain.
- After the momentum equation is solved in both domains the solution inside the inner domain (but not in the buffer regions) is filtered onto the coarse grid and sent to the outer domain.
- The outer domain replaces the values in its grid with the received values and performs pressure correction.

Test case – pollution dispersion

- Dispersion from a small source of dangerous gas, high resolution necessary close to the source.
- Large neighbourhood can be hit, large area must be simulated.
- 3 domains: domain D2 immersed in D1 and D3 immersed in D2.
- Refinement ratio: 2
- Two-way nesting in both cases, D1$\leftrightarrow$D2 and D2$\leftrightarrow$D3
- Adapted from project DIPLOS (EPSRC, www.diplos.org)

Test case – pollution dispersion

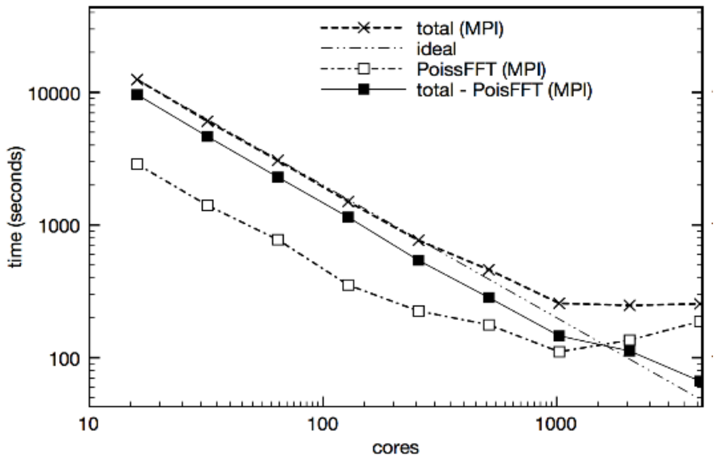


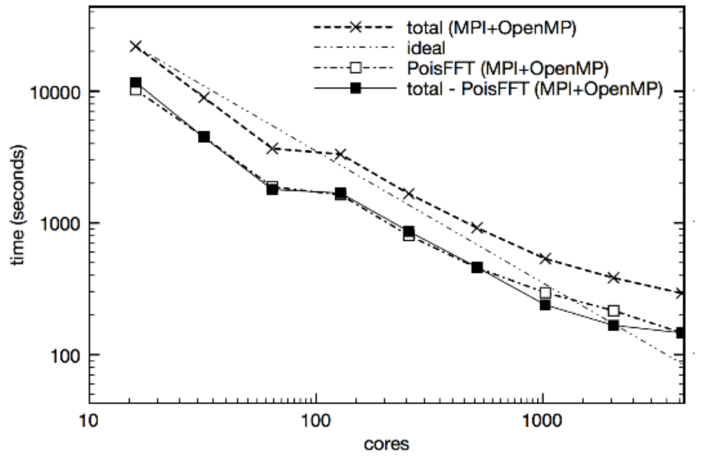
- Periodic boundary conditions for the flow
- Source in the centre of the innermost domain
- Animation of concentration on plane *z*=*h*/2, *h* is the building height
- The whole domain in high resolution would be 113 mil. cells, with nesting in total 2.3 mil. cells.

Performance tests – strong scaling

- Strong scaling: increasing number of processors and keeping the problem size. The time required for the solution should ideally decrease linearly with the number of processors.
- Depends on the problem size, larger problems should scale to larger numbers of CPUs.
- Chosen problem derived from the DIPLOS building array.
- Domain 48 *h* x 24 *h x 12 h*, grid 768 x 284 x 192 = 56.6 mil. cells. Moderate, city scale requires larger grids.
- Boundary conditions: turbulent inflow generation, free slip top and sides.

Performance tests – strong scaling



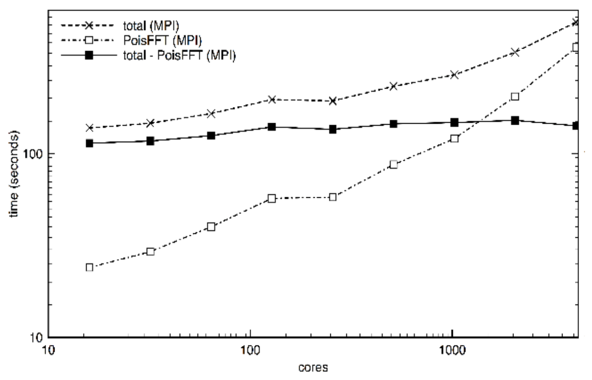


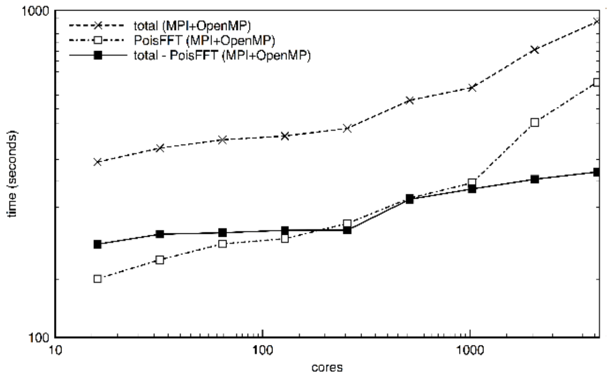
Strong scaling of ELMM performance: a) pure MPI, b) hybrid MPI/OpenMP

- Good scaling of pure MPI version up to 1024 cores.
- Hybrid version significantly slower at small CPU numbers due to the Poisson solver.
- Pure MPI Poisson slower doesn’t scale at all above 1024 cores.
- Hybrid OpenMP/MPI scales better at high CPU numbers.
- Most of the time spent in the 3rd party PFFT library. The author demonstrated scaling to much higher CPU numbers.

Performance tests – weak scaling

- Weak scaling: increasing number of processors and increasing the problem size so that each processor’s part of the grid has the same size in each run.
- The same boundary conditions as previously.
- 1 node domain: 6 h x 3 h x 12 h, grid: 96 x 48 x 192 = 885 736 cells.





Weak scaling of ELMM performance: a) pure MPI, b) hybrid MPI/OpenMP

- FFT complexity O(N log(N)), ideally should grow linearly on the plots.
- Again the bottle-neck is the Poisson solver, becomes the dominating part.
- ELMM-proper scales very well in pure MPI in the whole tested range.
- Possible mistake: not tested 1D MPI decomposition in the hybrid case. Allows avoiding a set off costly global transpositions.

Conclusions

- Capabilities of ELMM greatly enhanced.
- ELMM released as open source. https://bitbucket.org/LadaF/elmm
- Code optimized.
- One-way and two-way grid nesting allows very large domains in coarser resolution and high resolutions in important regions.
- Turbulent inflow generation proved to be important for correct inflow at the nesting boundaries.
- Problems with the performance of the Poisson solver at high numbers of CPUs. The underlying library previously successfully tested for much larger problems, we were not able to replicate this.
- ELMM is easy to compile on ARCHER. Build scripts aware of ARCHER specifics.