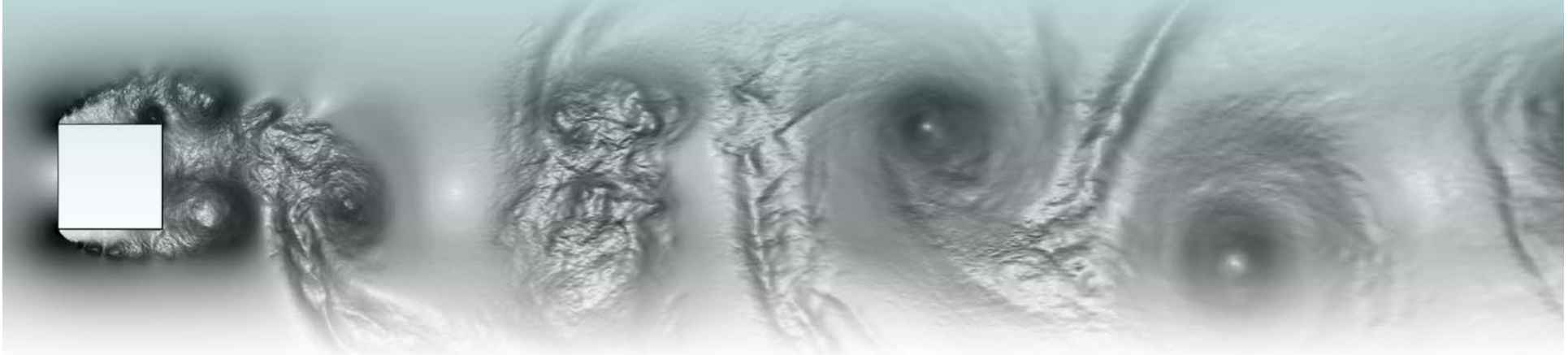


DIRECT NUMERICAL SIMULATION OF TURBULENT FLOWS WITH PARALLEL ALGORITHMS FOR VARIOUS COMPUTING ARCHITECTURES

A. V. Gorobets^{1,2}, F. X. Trias¹, R. Borrell³, G. Oyarzun¹, A. Oliva¹



¹ **Heat and Mass Transfer Technological Center**
Technical University of Catalonia, Barcelona, Spain



² **CAA LAB of KIAM RAS**, Moscow, Russia



³ **Termo Fluids S.L.**





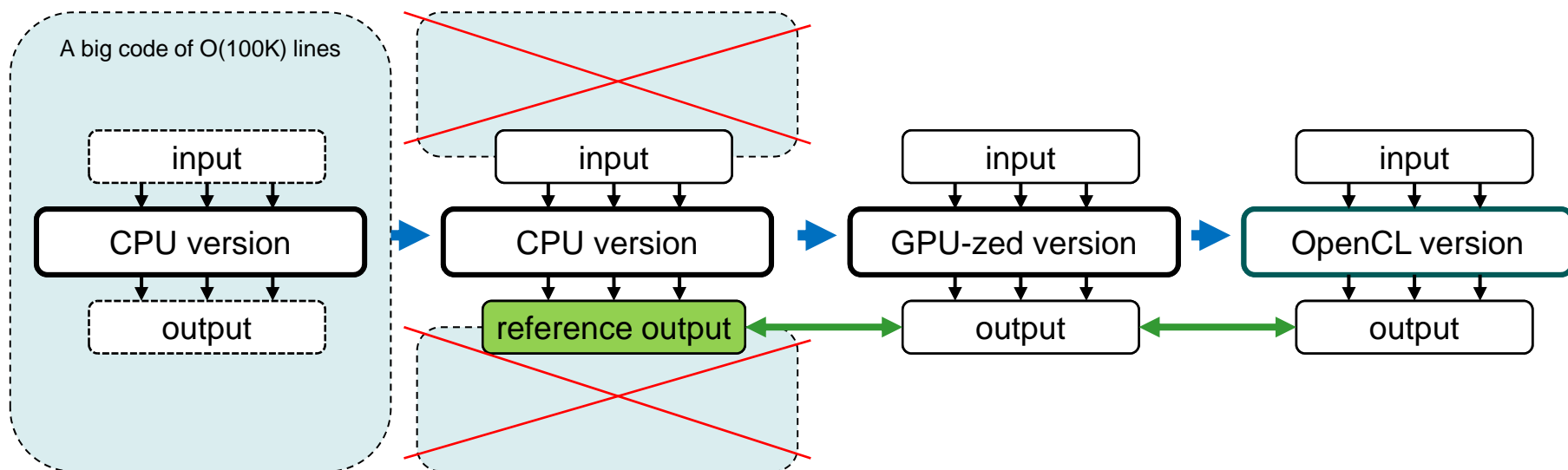
Parallel programming models and frameworks

	Cluster	CPU	NVIDIA GPU	AMD GPU	Intel Xeon Phi	ARM
MPI	+	+	-	-	±	-
OpenMP	-	+	-	-	+	-
OpenCL	-	+	+	+	+	+
CUDA	-	-	+	-	-	-
Distributed memory MIMD	+	+	-	-	-	-
Shared memory MIMD	-	+	-	-	+	-
Stream processing	+	+	+	+	+	+



Tactics: divide and conquer

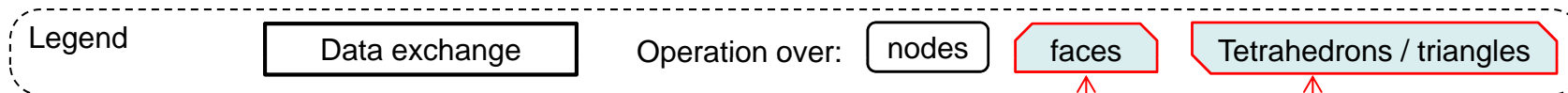
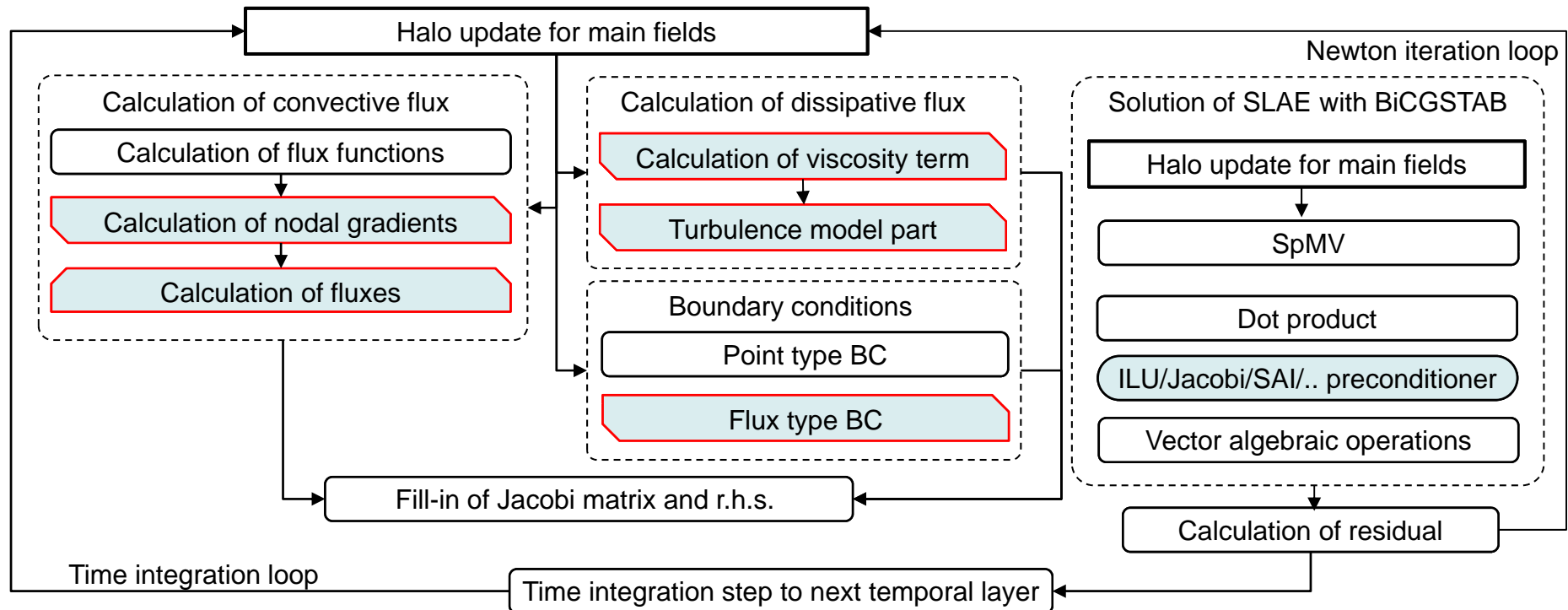
- Decomposition of algorithm into basic operations – as few as possible
- For each operation create standalone test with wrapping that includes profiling, inputs and reference outputs in files
- For each operation create “GPU-zed” version for CPU:
 - **adapt an operation to stream processing at lower parallelization level**
 - make alternative data structures that fit better accelerator architectures
 - mimic execution on accelerator by external parallel loop that iterates ranks of a workgroup
- For each operation on a base of GPUzed version create an OpenCL kernel and ensure correctness
- Optimize OpenCL version ensuring correctness every small step





Fit to stream processing or die

Outline of an edge-based vertex-centered algorithm

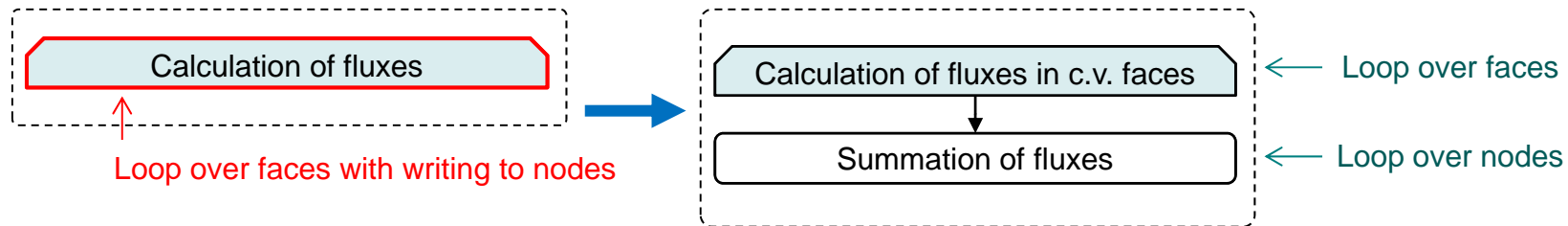


Data interdependency appears when modifying elements of one type in a loop over elements of another type

Loop tetrahedrons with writing to nodes
Loop over faces with writing to nodes

Fit to stream processing or die

1) Decomposition of operation into two operations over elements of different type:

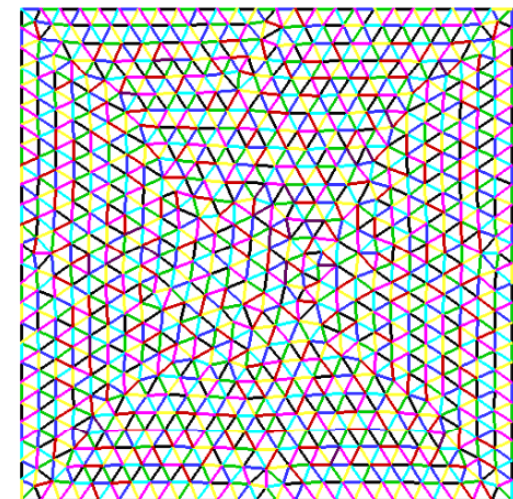
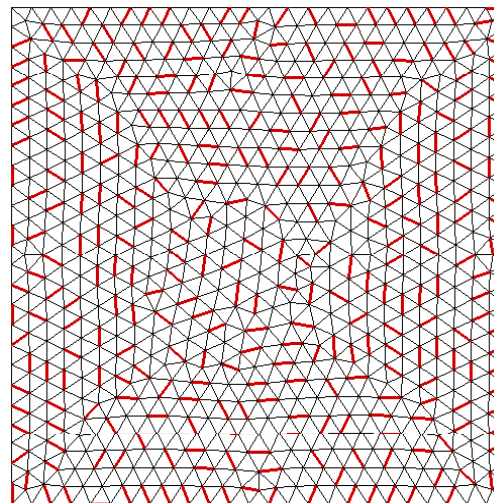


- Additional array over faces stores intermediate result
- Summation is in the loop over nodes using CSR-like inverse topology
- Needs additional memory and can't be used if intermediate storage is too big

2) Decomposition of operation into multiple operations over subsets:

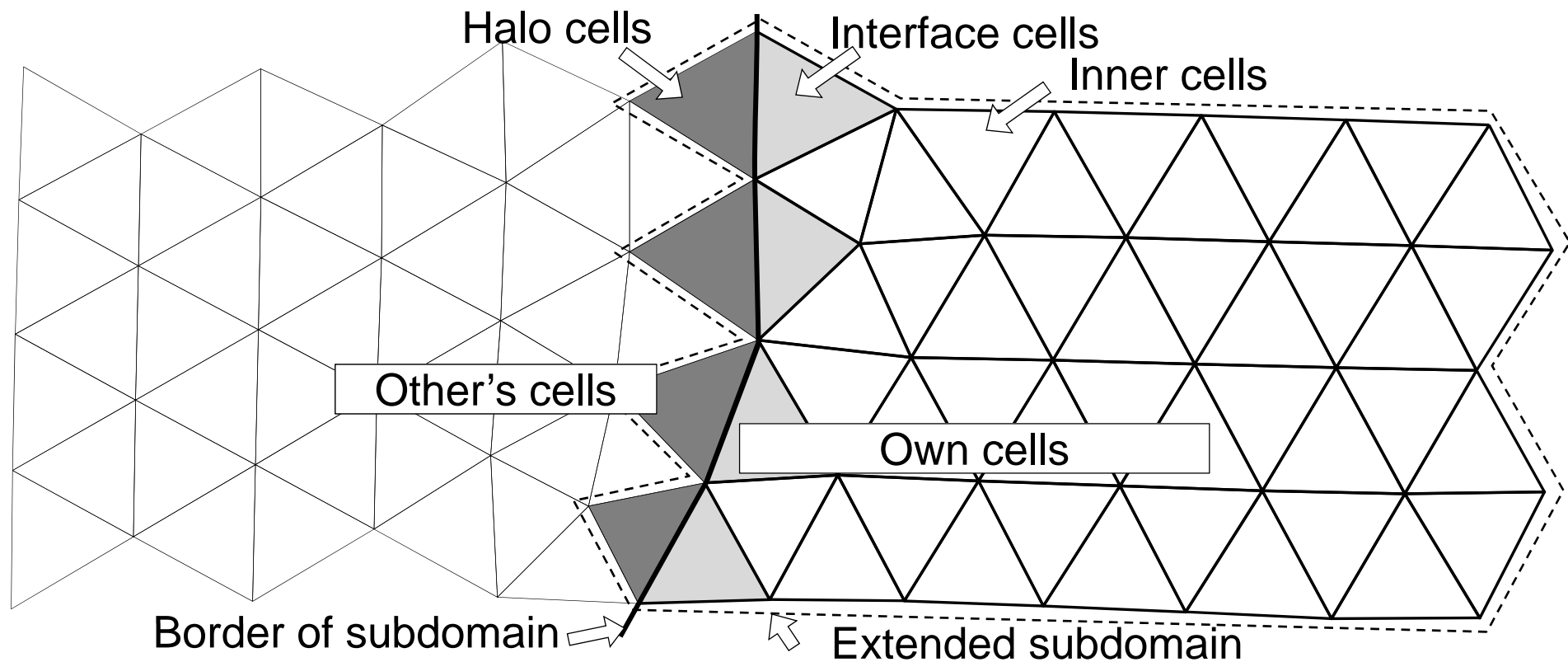
- Edges of the connectivity graph (dual mesh graph etc.) are colored in a way that no edges of one color share same node

- Needs multiple kernel executions
- Inefficient memory access:
no way for coalescing
- around 10-20% slower if 1) applicable



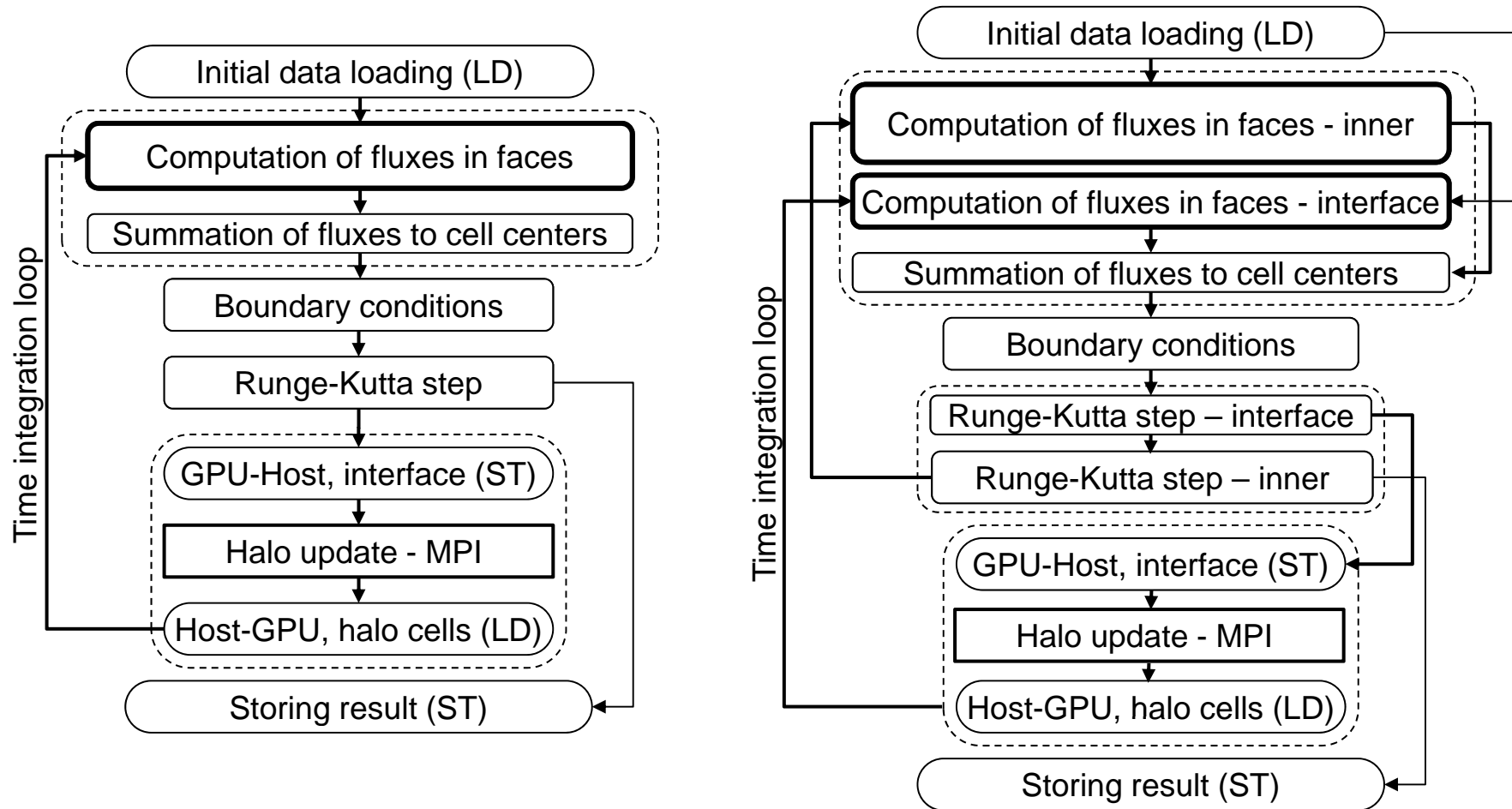
Sets of mesh elements in a domain decomposition

- A cell (or a mesh element) that belongs to a subdomain is its **own** cell
- An own cell that is coupled with a cell from another subdomain is an **interface** cell
- An own cell that is coupled only with own cells is an **inner** cell
- A cell from another subdomain that is coupled with an own cell is a **halo** cell





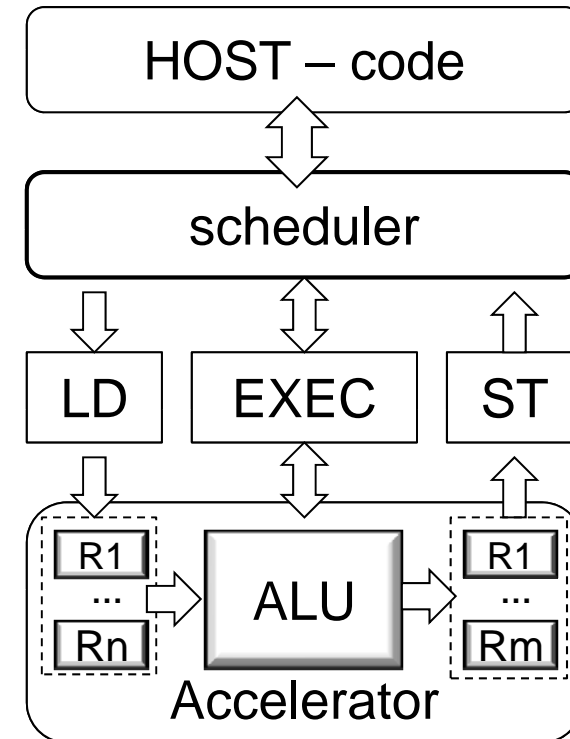
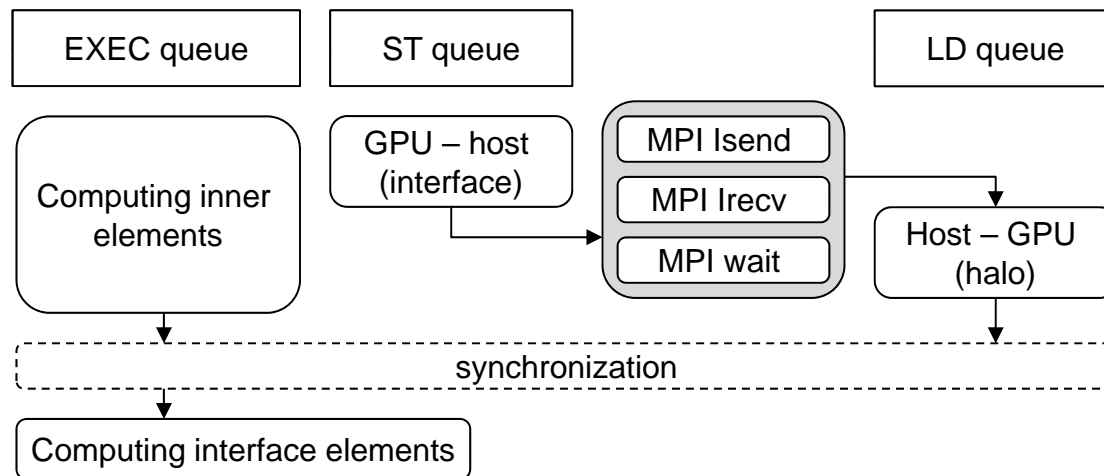
Graph of a finite-volume algorithm





Scheduling infrastructure

- **Register** is a region in device global memory, **instruction** is an OpenCL kernel for the device
- Scheduler has with 3 queues: **LD**, **ST**, **EXEC**
 - LD command: load from host to device
 - ST command: from device to host
 - EXEC command launches a kernel on a device
- Independent commands can run simultaneously

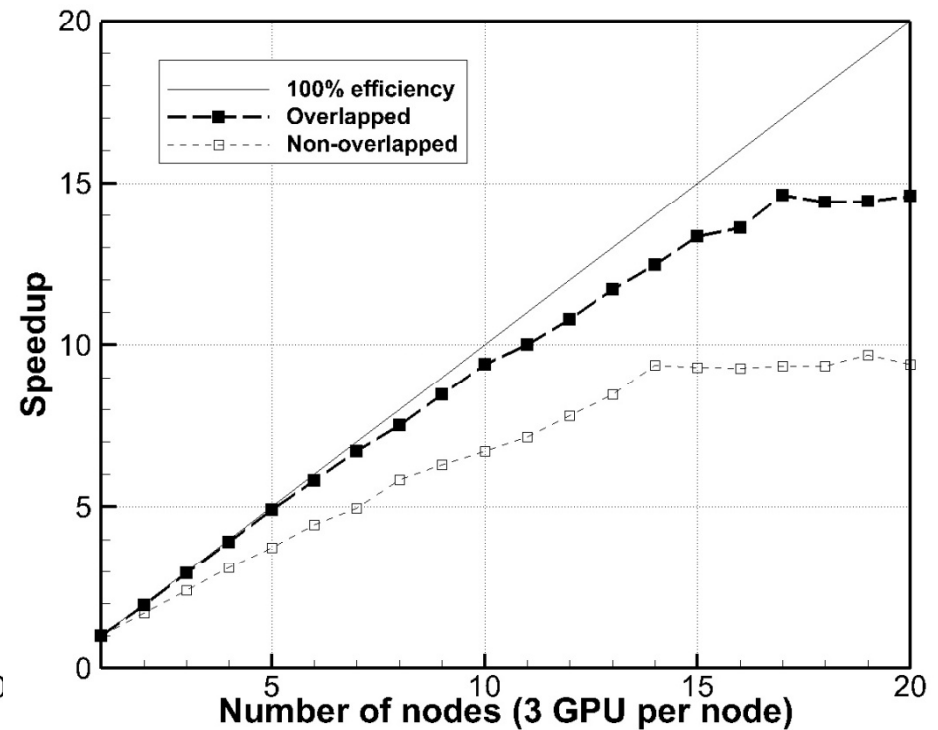
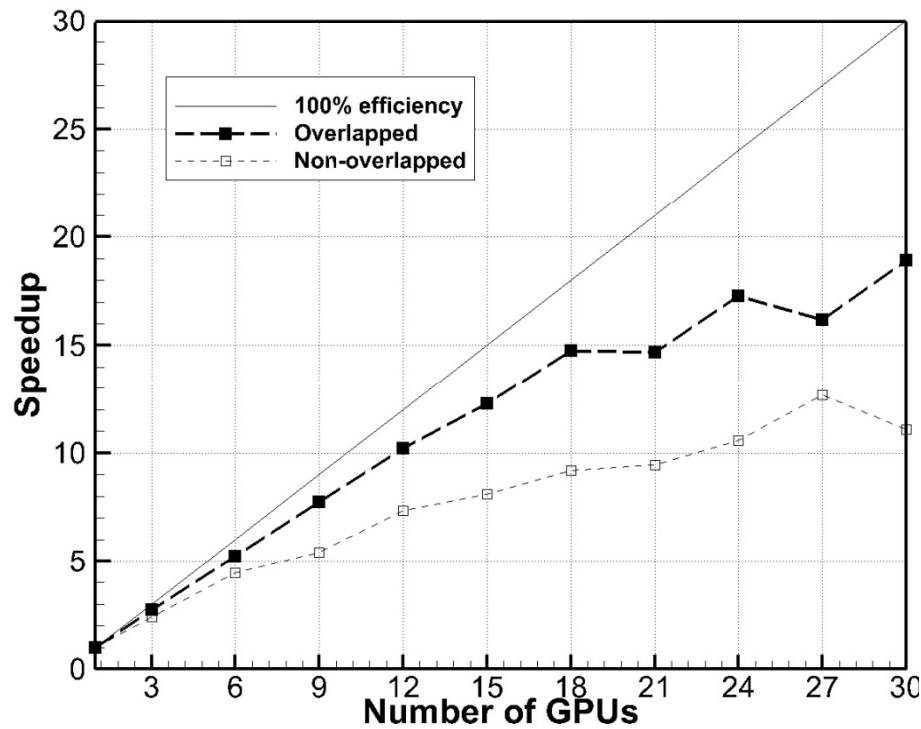


Bogdanov P. B., Efremov A. A. from Scientific Research Institute of System Development of RAS
 Programming infrastructure of heterogeneous computing based on OpenCL and its applications
 GPU Technology Conference GTC-2013, March 18-21, San Jose, California, USA.



Speedup on a hybrid supercomputer

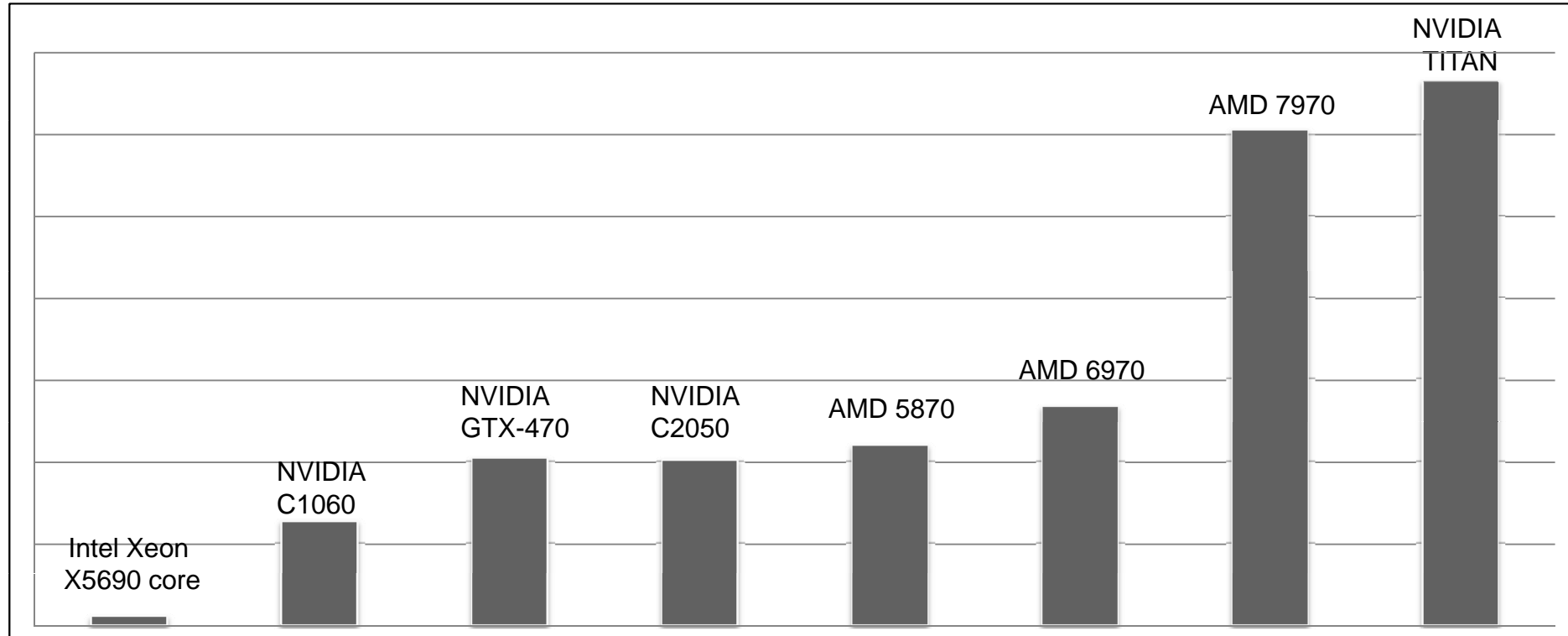
Speedups on K100 supercomputer for a mesh with 4 millions of cells (left) starting from one GPU and for a mesh with 16 millions of cells (right) starting from one computing node.





Comparison of performance

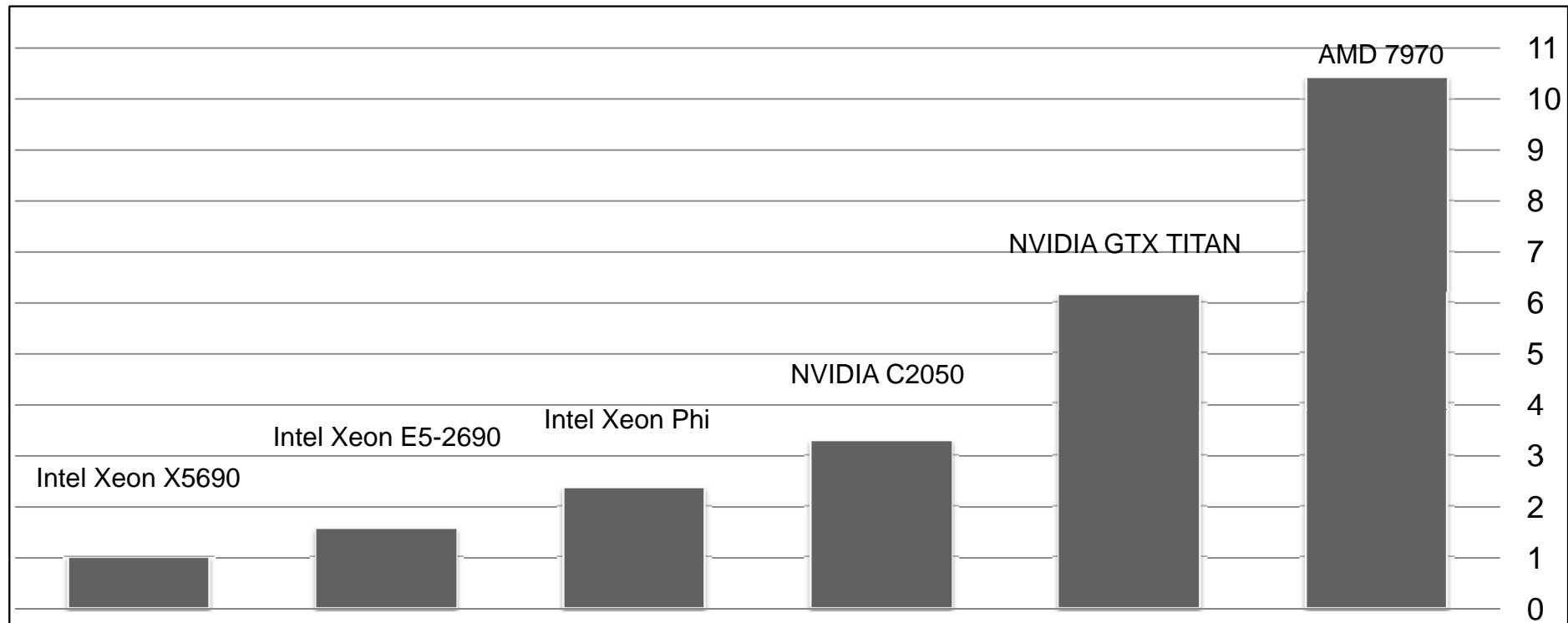
Comparison of computing devices on calculation of overall time step – 1st order scheme





Comparison of performance

Comparison of computing devices on calculation of overall time step – 2nd order scheme



An algorithm for incompressible flows with periodic direction

- Navier-Stokes system to solve:

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

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = \frac{\text{Pr}}{\sqrt{\text{Ra}}} \nabla^2 \mathbf{u} - \nabla p + \mathbf{f},$$

$$\frac{\partial T}{\partial t} + (\mathbf{u} \cdot \nabla) T = \frac{1}{\sqrt{\text{Ra}}} \nabla^2 T.$$

- Discrete system for pressure-velocity coupling:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \frac{3}{2} \mathbf{R}^n - \frac{1}{2} \mathbf{R}^{n-1} - Gp^{n+1},$$

$$M\mathbf{u}^{n+1} = 0,$$

$$\text{where } \mathbf{R}(\mathbf{u}) = -C(\mathbf{u})\mathbf{u} - D\mathbf{u} + f$$

- Fractional step projection method:

$$\text{Predictor velocity: } \mathbf{u}^p = \mathbf{u}^n + \Delta t \left(\frac{3}{2} \mathbf{R}^n - \frac{1}{2} \mathbf{R}^{n-1} \right)$$

$$\text{Unknown velocity: } \mathbf{u}^{n+1} = \mathbf{u}^p - G\tilde{p}, \text{ where } \tilde{p} = \Delta t p^{n+1}$$

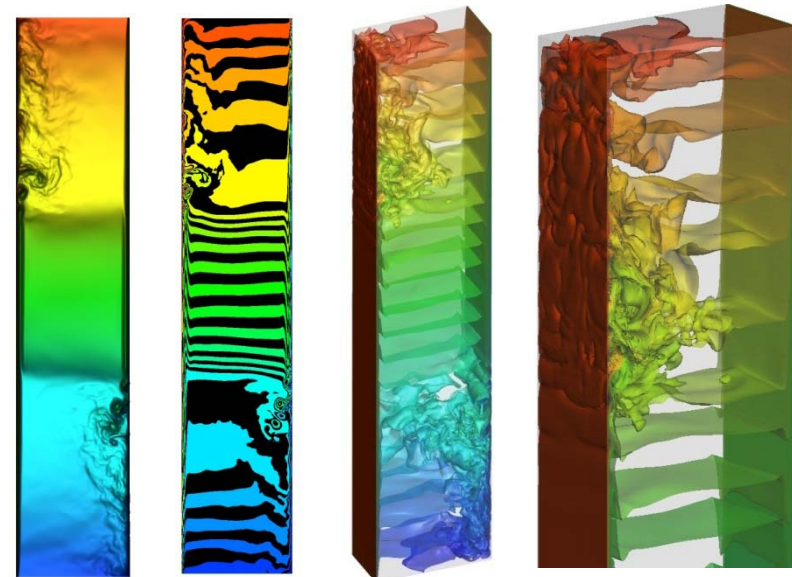
$$\text{Mass conservation equation: } M\mathbf{u}^{n+1} = M\mathbf{u}^p - GM\tilde{p} = 0$$

$$M\mathbf{u}^{n+1} = M\mathbf{u}^p - GM\tilde{p} = -M\Omega M^* \tilde{p} = \boxed{L\tilde{p} = M\mathbf{u}^p}$$

The Poisson equation

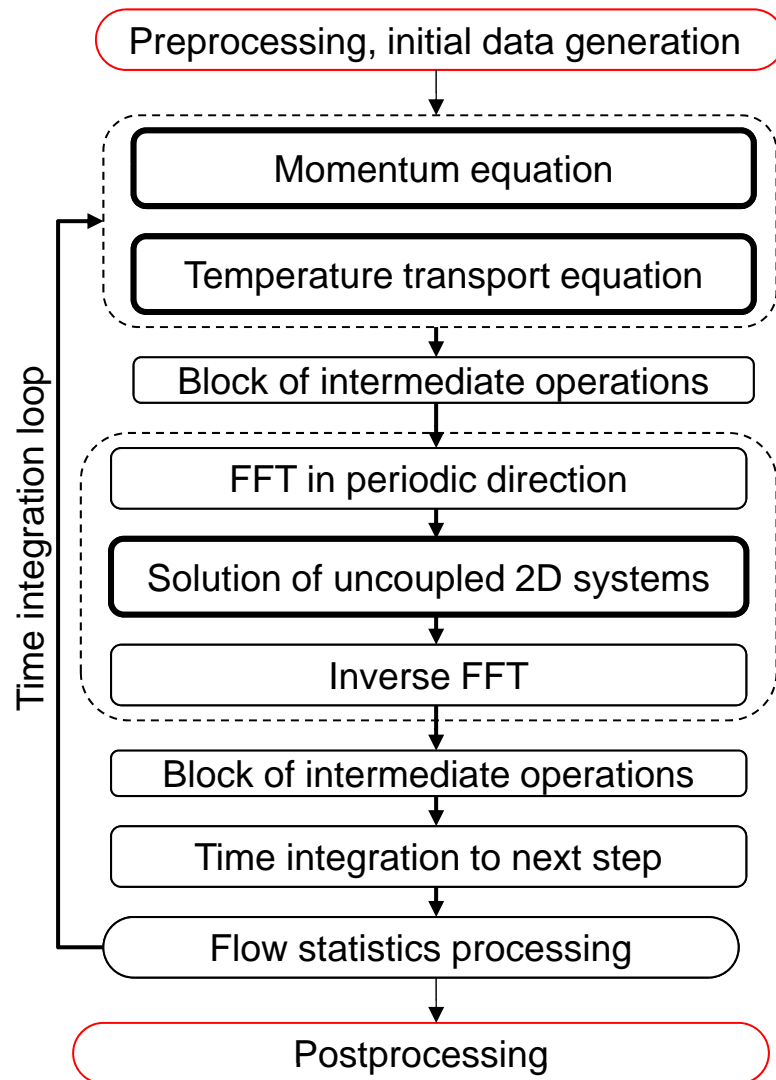
The algorithm of the time step

1. Predictor velocity field, \mathbf{u}^p is obtained explicitly
2. Temperature transport equation is solved explicitly
3. Correction, \tilde{p} , is obtained from the Poisson equation
4. Resulting velocity field, \mathbf{u}^{n+1} , is obtained





Outline of the algorithm



Basic operations

T1. Linear operator

T2. Nonlinear operator

T3. $y = \sum_{i=1}^n a_i x_i + c$

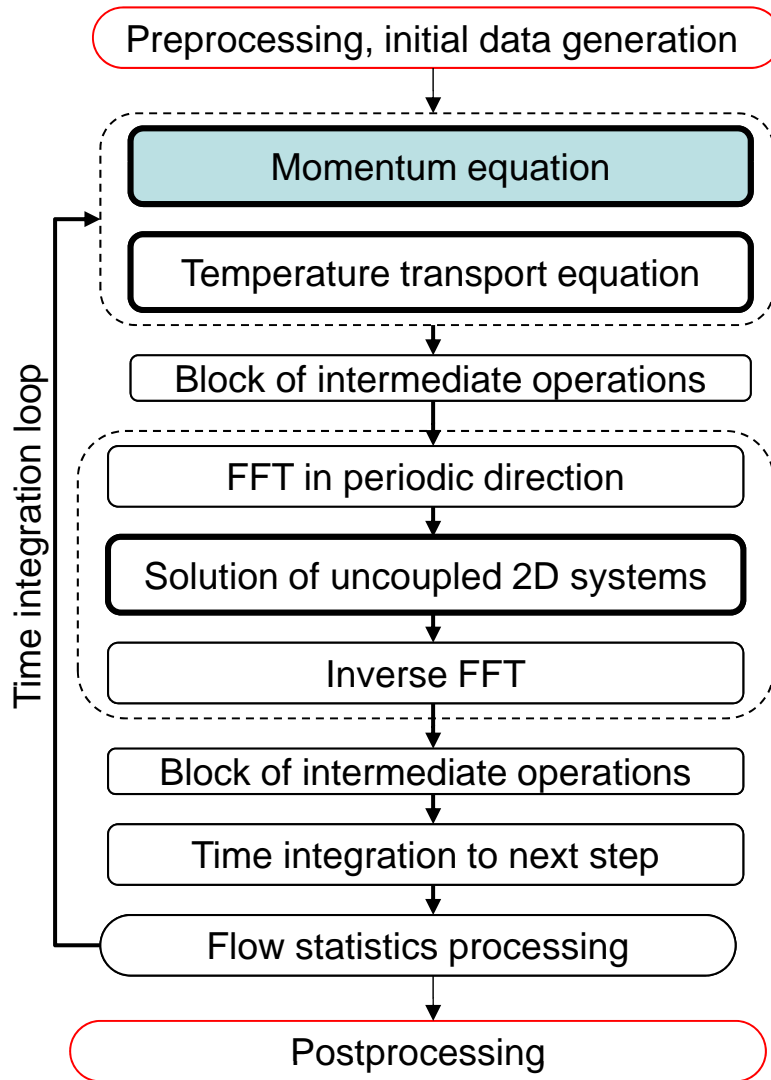
T4. FFT, inverse IFFT

T5. multi-SpMV

T6. reduction – norm, dot product

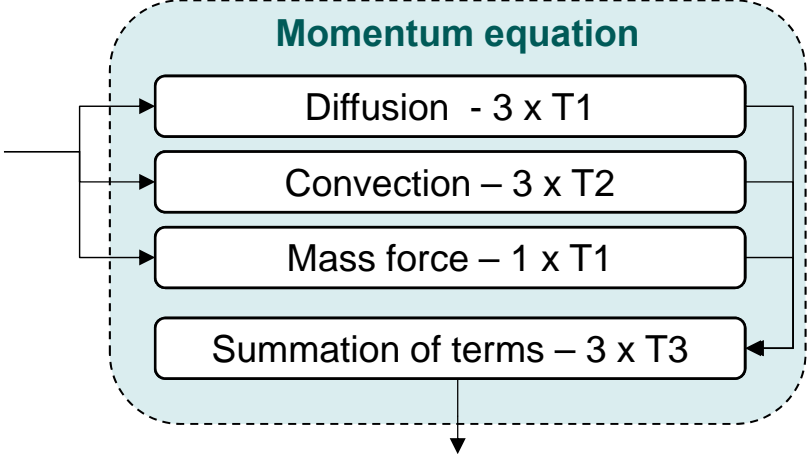


Outline of the algorithm



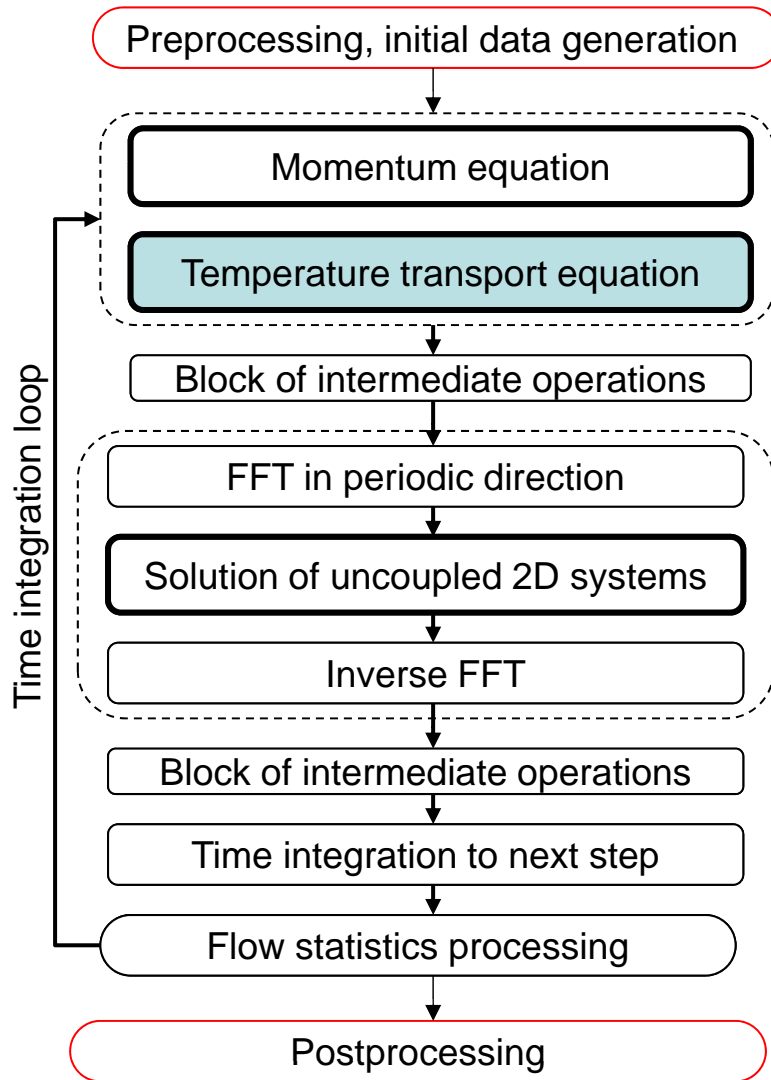
Basic operations

- T1. Linear operator
- T2. Nonlinear operator
- T3. $y = \sum_{i=1}^n a_i x_i + c$
- T4. FFT, inverse IFFT
- T5. multi-SpMV
- T6. reduction – norm, dot product



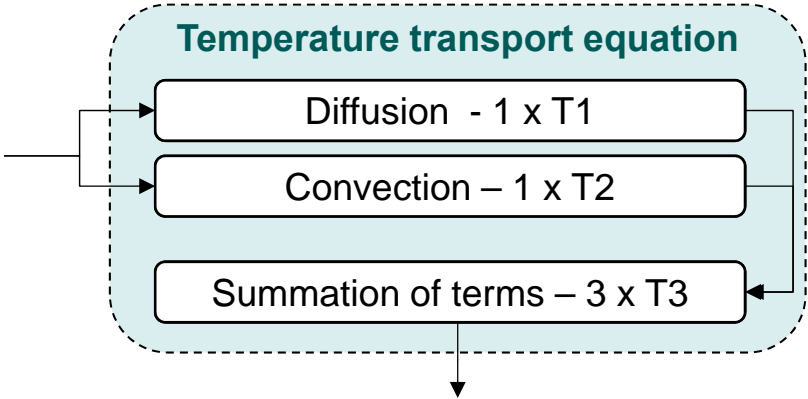


Outline of the algorithm



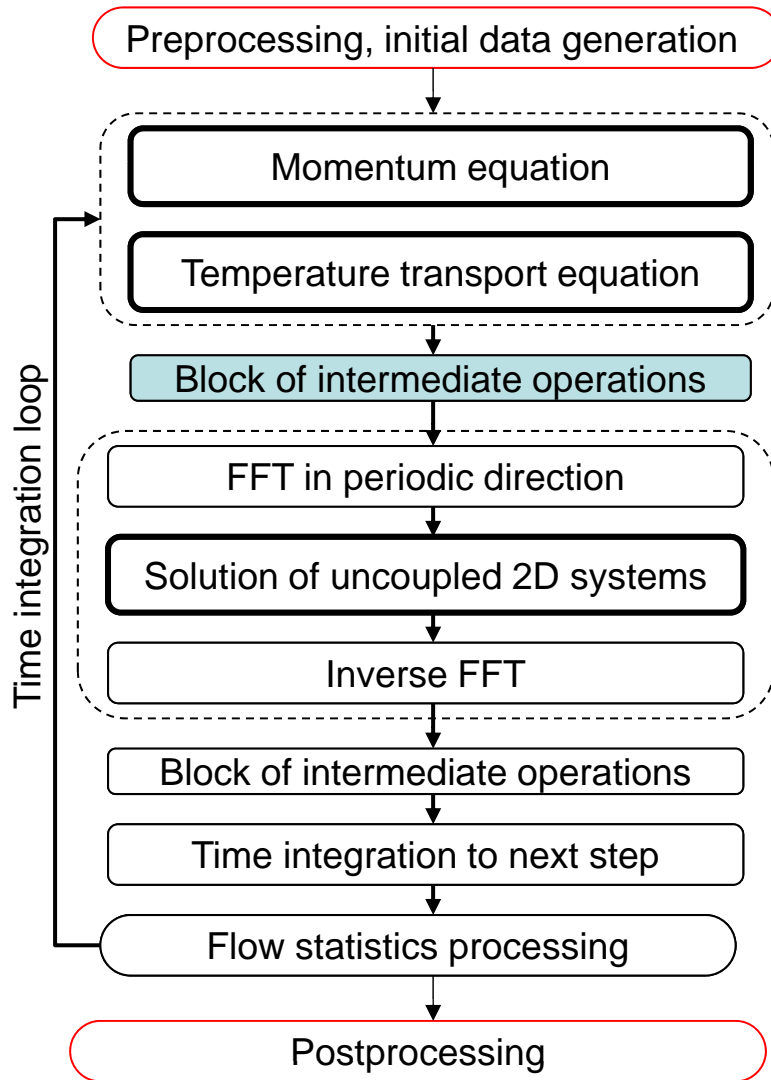
Basic operations

- T1. Linear operator
- T2. Nonlinear operator
- T3. $y = \sum_{i=1}^n a_i x_i + c$
- T4. FFT, inverse IFFT
- T5. multi-SpMV
- T6. reduction – norm, dot product



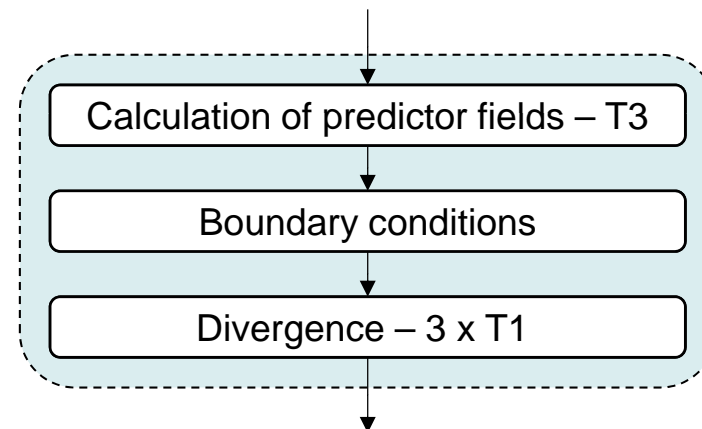


Outline of the algorithm

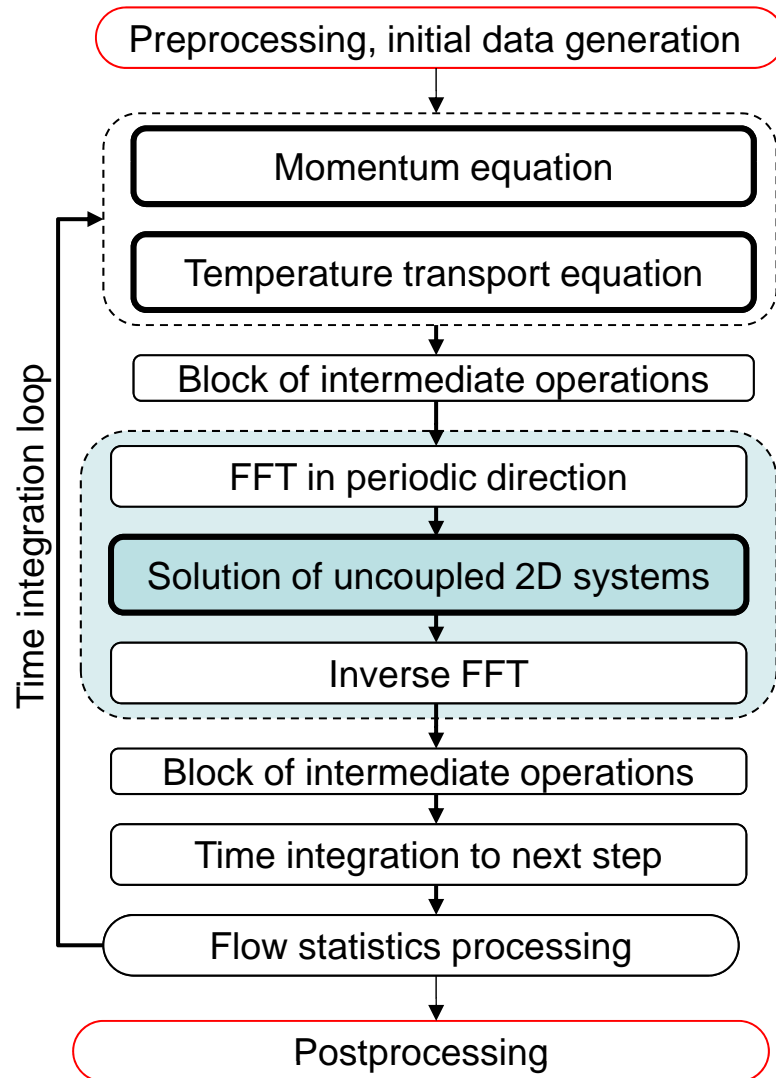


Basic operations

- T1. Linear operator
- T2. Nonlinear operator
- T3. $y = \sum_{i=1}^n a_i x_i + c$
- T4. FFT, inverse IFFT
- T5. multi-SpMV
- T6. reduction – norm, dot product

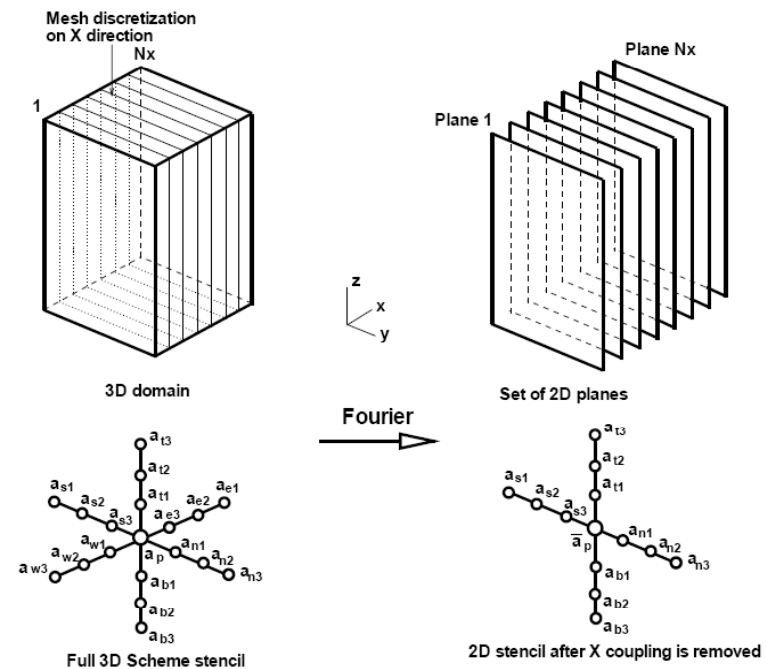


Outline of the algorithm



The algorithm of the solver

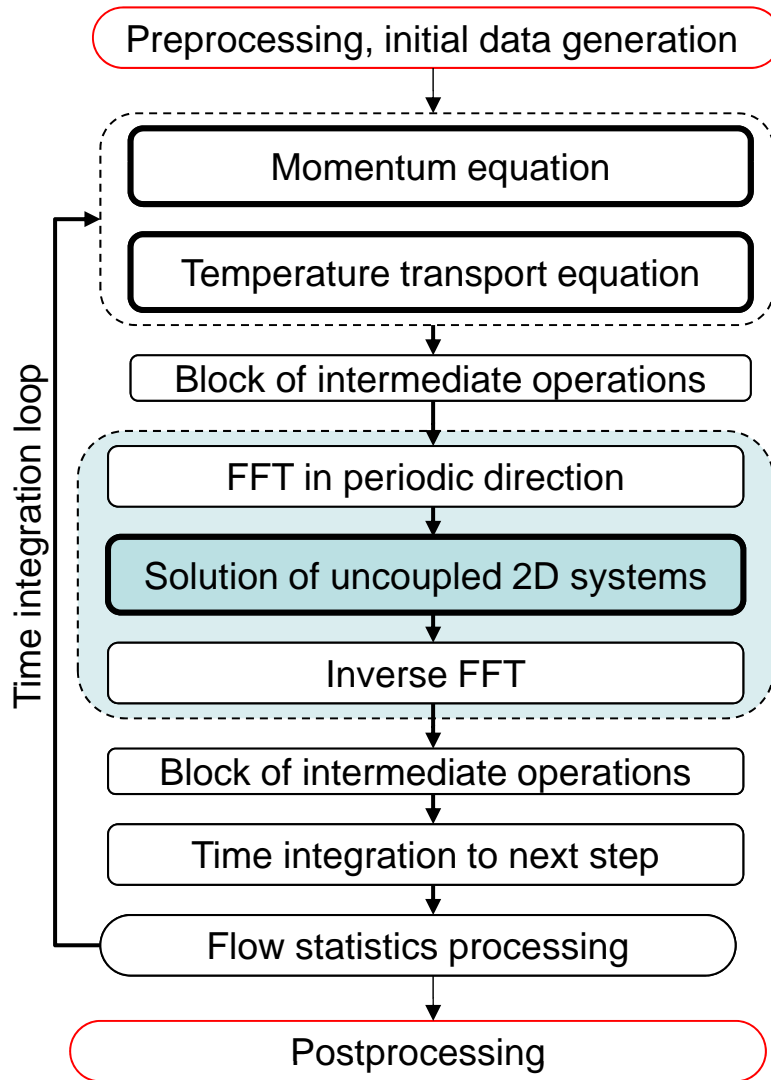
1. FFT diagonalization FFT uncouples 3D problem into set of independent 2D problems (planes)
2. The Schur complement based direct method is used to solve planes that correspond to lower Fourier frequencies
3. The preconditioned CG method is used to solve the remaining planes
4. Inverse FFT to restores solution of the 3D problem.



Uncoupling of a 3D domain using FFT

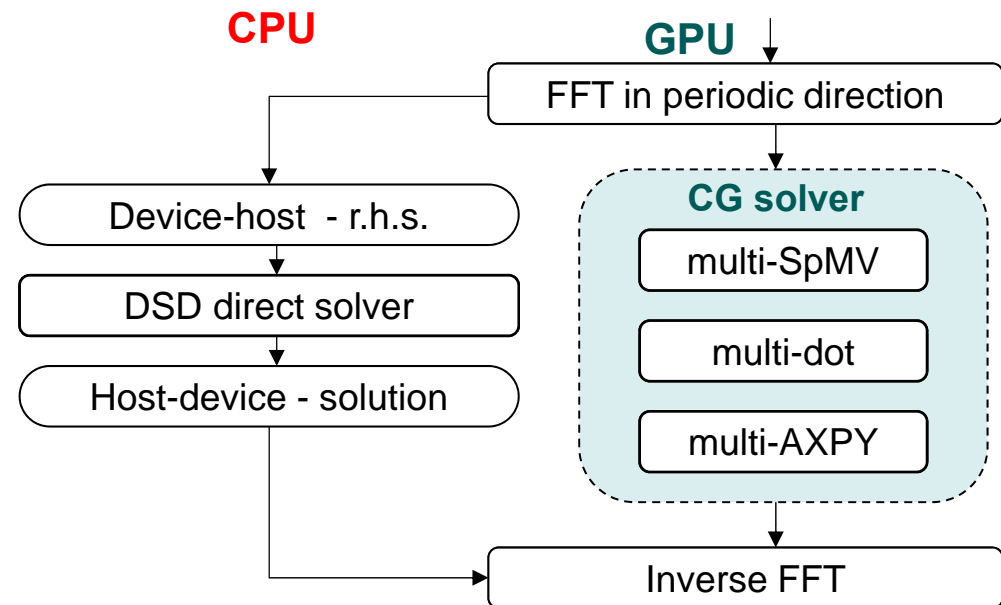


Outline of the algorithm



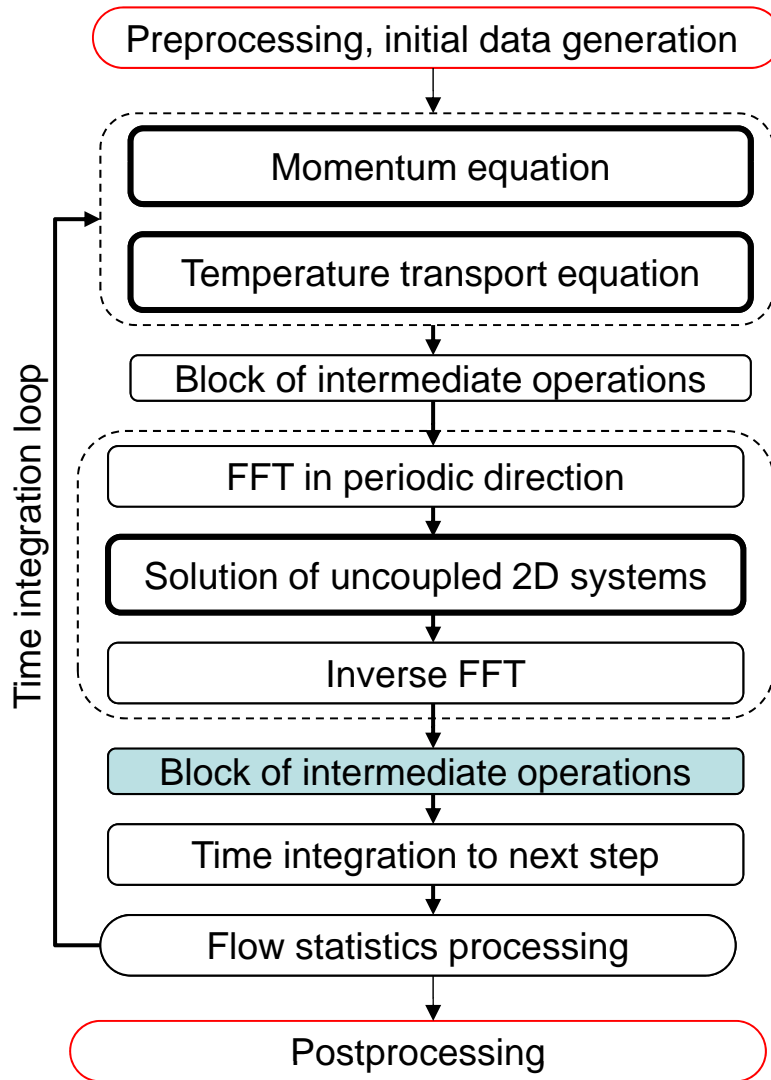
Basic operations

- T1. Linear operator
- T2. Nonlinear operator
- T3. $y = \sum_{i=1}^n a_i x_i + c$
- T4. FFT, inverse IFFT
- T5. multi-SpMV
- T6. reduction – norm, dot product





Outline of the algorithm



Basic operations

T1. Linear operator

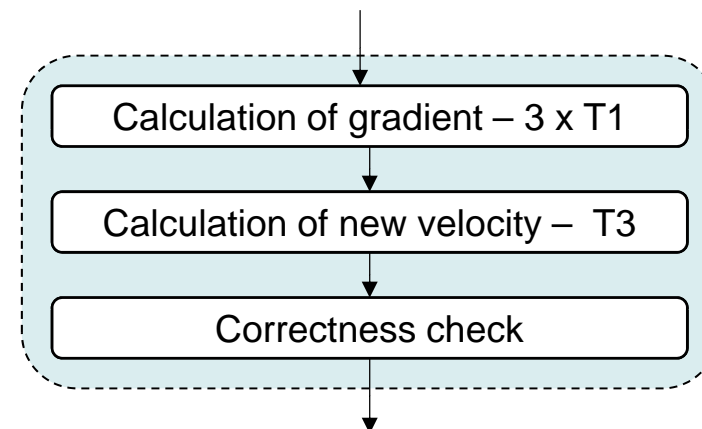
T2. Nonlinear operator

T3. $y = \sum_{i=1}^n a_i x_i + c$

T4. FFT, inverse IFFT

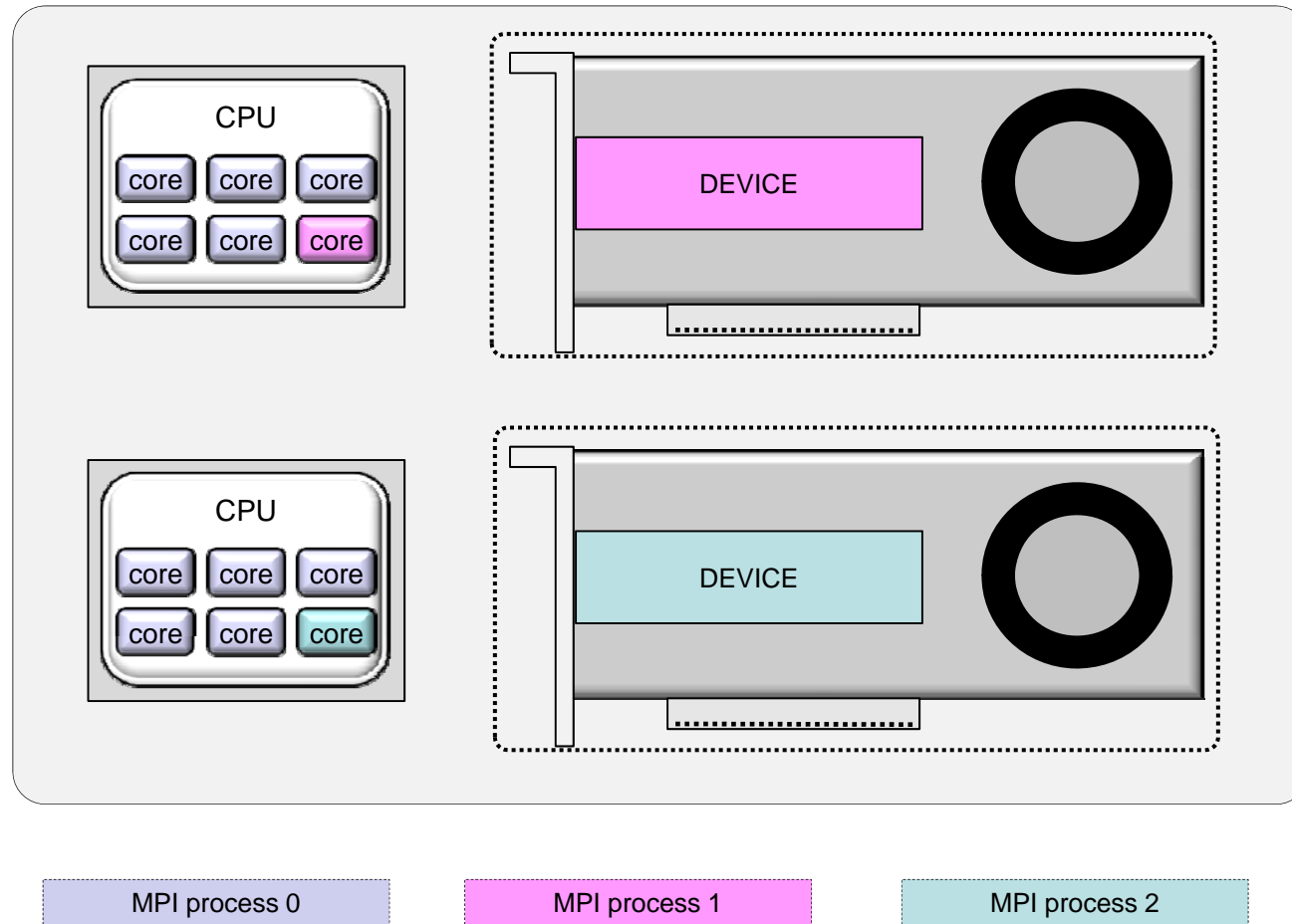
T5. multi-SpMV

T6. reduction – norm, dot product



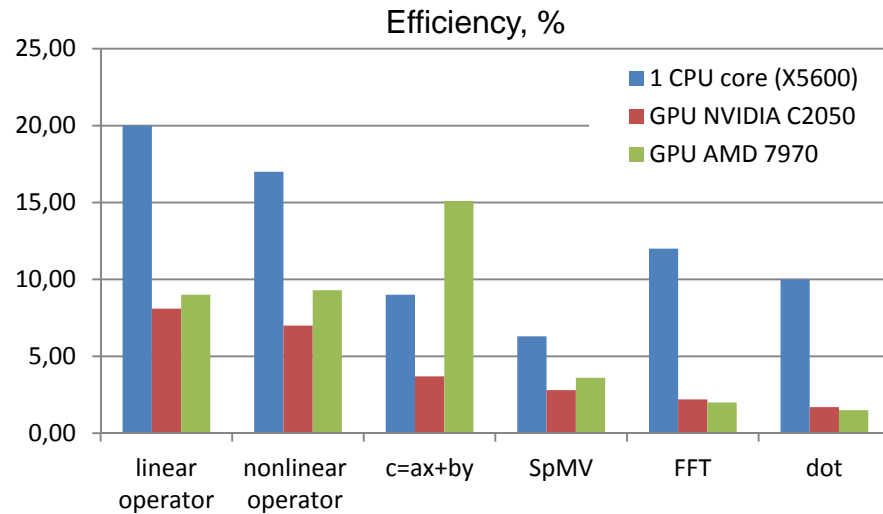


Loading heterogeneous node



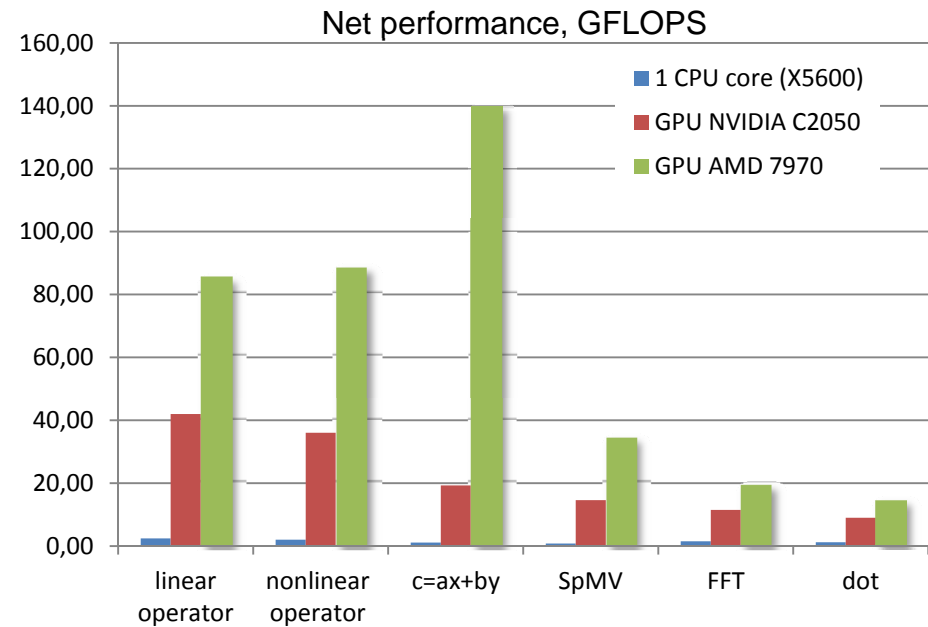
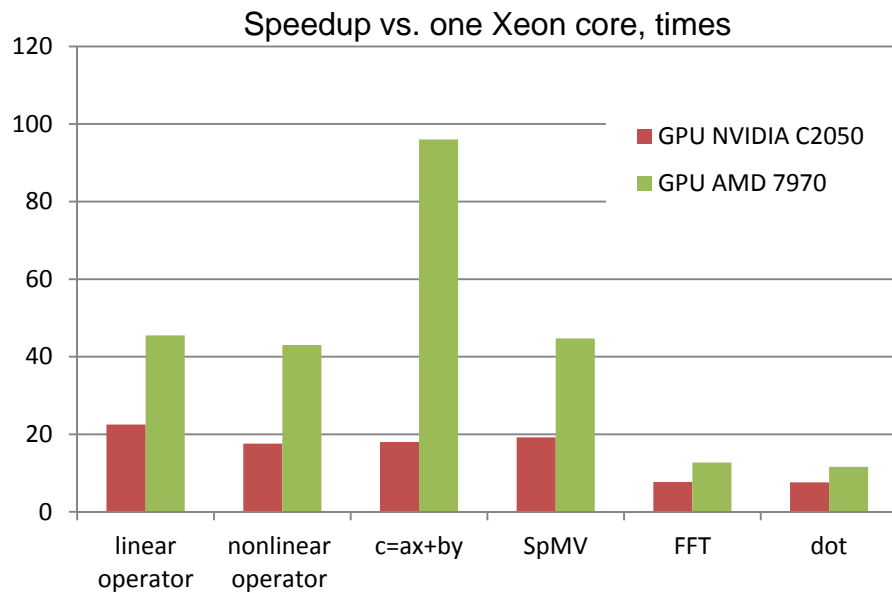


Performance of basic operations on GPU



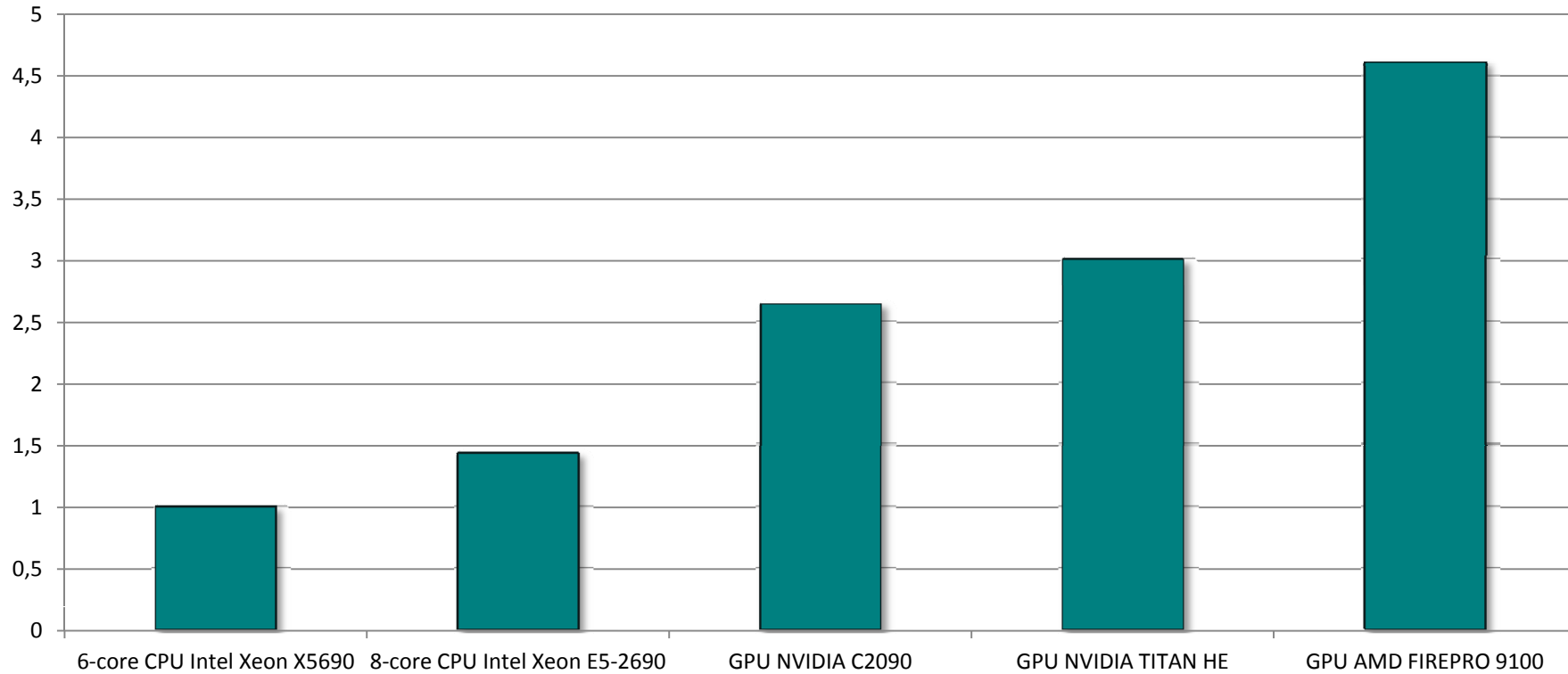
The test case

- DHC typical case is used
- 4-th order scheme
- Mesh size 1.2M nodes
- Imitates a typical computing load
- Reference: Intel Xeon X5690 core





Overlap performance on GPU





Termo Fluids: MPI-CUDA unstructured CFD code

The algorithm of the time step

Calculate predictor velocities $u^p = u^n - \Delta t (R(m_f, \bar{v}_f)u^n)$

Poisson equation $Lp^{n+1} = \frac{1}{\Delta t} (Mu^p)$

Correct velocities $g_p^{n+1} = Gp^{n+1}$
 $u^{n+1} = u^p - \Delta t(g_p^{n+1})$

Calculate the corrected mass flows

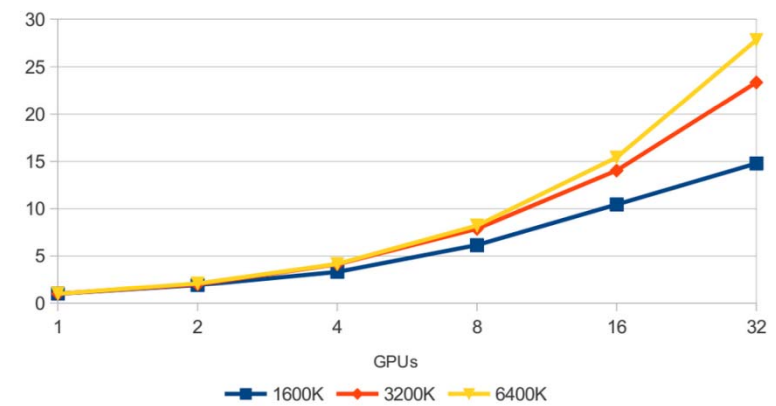
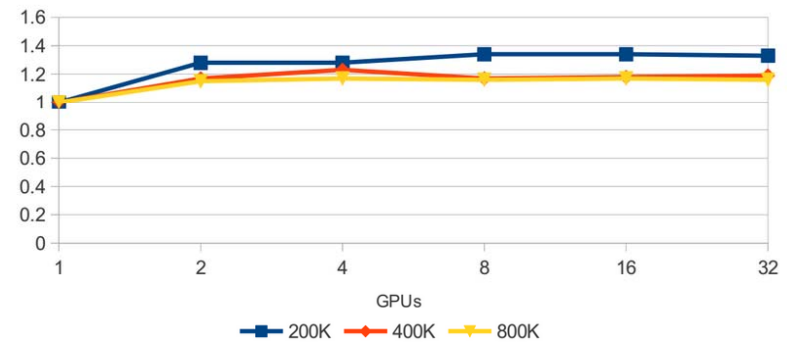
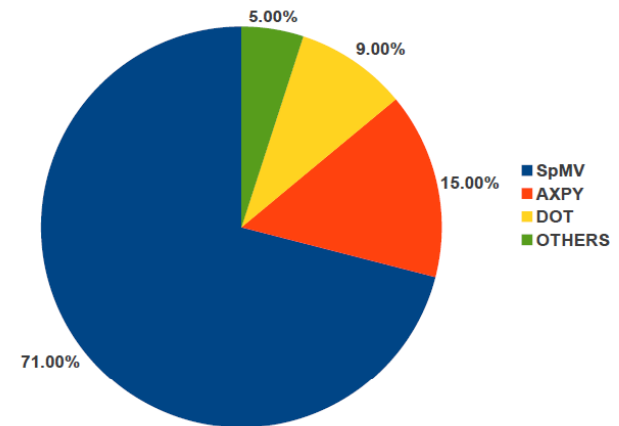
$$m_f = \rho(Eu^{n+1}) - \frac{1}{\sigma_1} ((Hp^{n+1}) - (E(g_p^{n+1})))$$

Evaluate LES model $g_u^{n+1} = Gu^{n+1}$

Ahmed car simulation

- CG solves the Poisson equation.
- 5.5M unstructured body-fitted mesh.
- MPI+CUDA outperforms in $\approx 8X$, in average, MPI-only.
- Energy saving rounds up $\approx 3.5X$

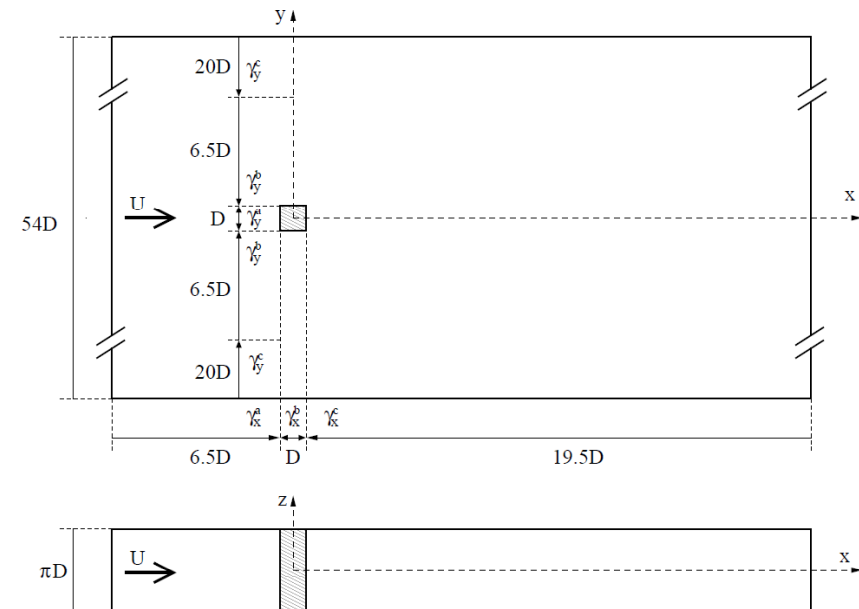
G. Oyarzun, R. Borrell, A. Gorobets, O. Lehmkuhl, A. Oliva, "Direct Numerical Simulation of Incompressible Flows on Unstructured Meshes Using Hybrid CPU/GPU Supercomputers", Procedia Engineering, 25-th PCFD, Volume 61, 2013, Pages 87-93.



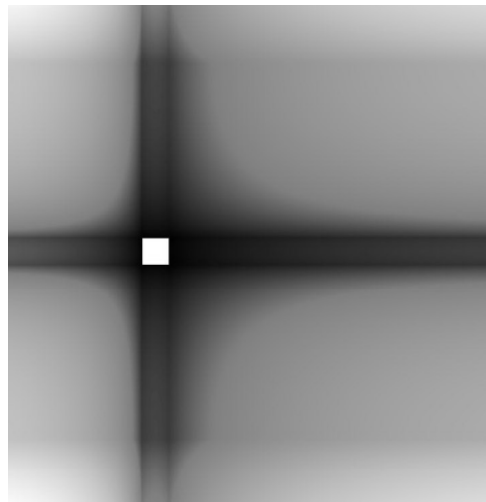


DNS of square cylinder

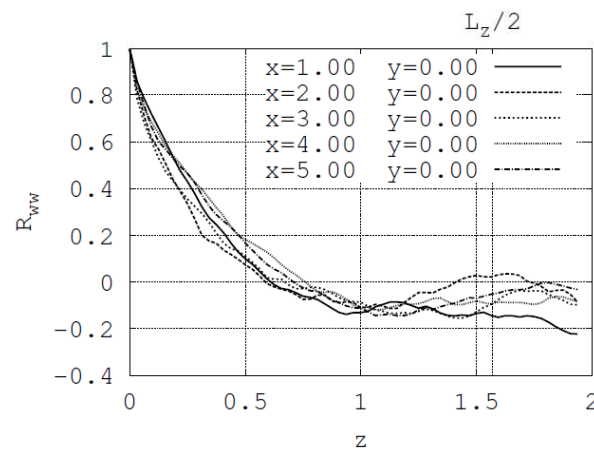
- Reynolds number 22000
- Prandtl number 0.71 (air)
- Mesh size 1174x1174x216 \approx 300M nodes
- up to 784 CPUs on MareNostrum supercomputer
- Overall computing price \sim 270K CPU hours



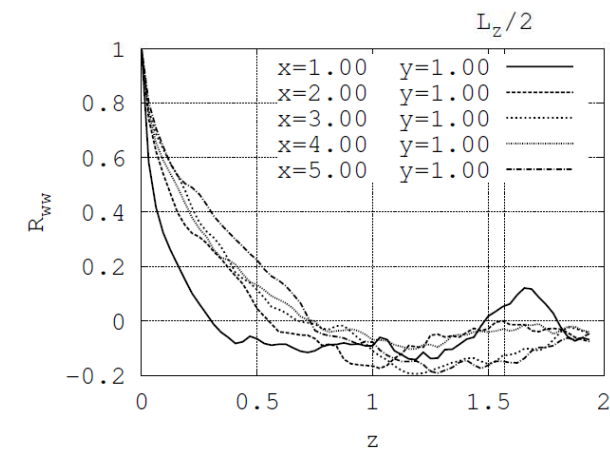
Computing domain



Mesh concentration



Two-point correlation of the span-wise velocity



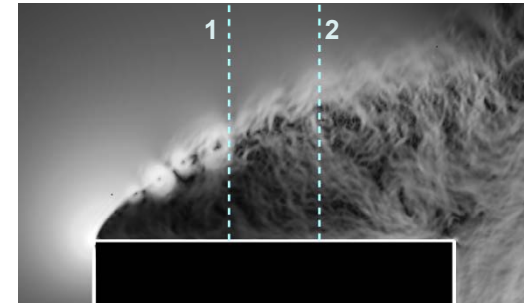


DNS of square cylinder

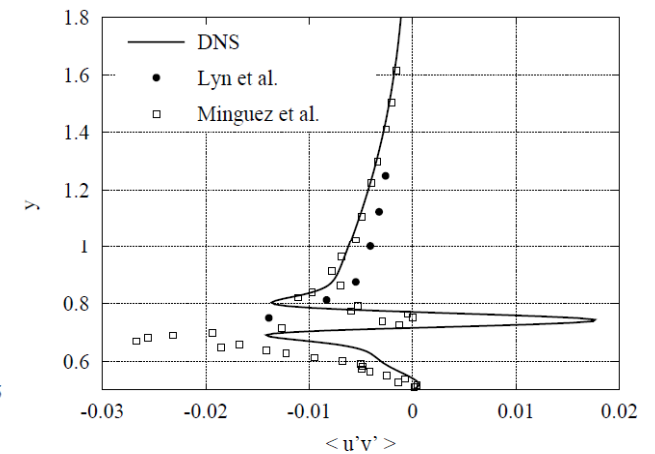
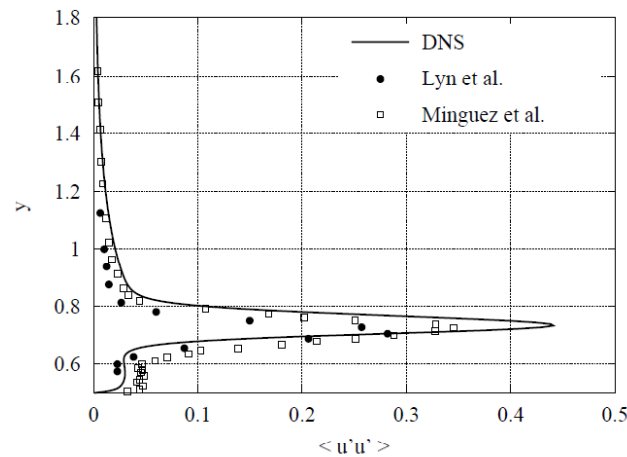
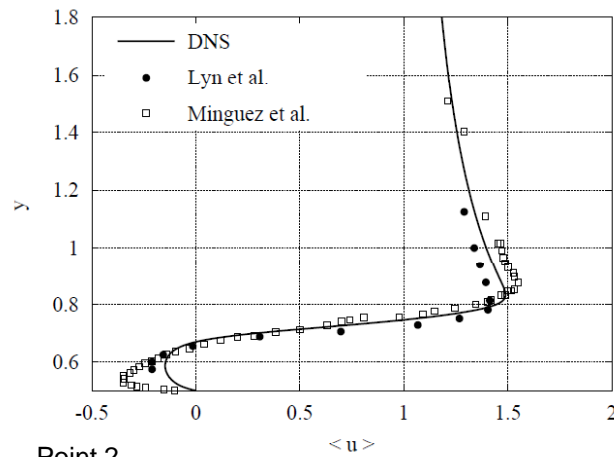
Case	St	$\langle C_D \rangle$	C_D^{rms}	C_L^{rms}	l_R
DNS	0.136	2.23	0.203	1.48	1.154
Minguez <i>et al.</i>	0.130	2.1	—	—	
Lyn <i>et al.</i>	0.133	2.1	—	—	
Lee	-	2.05	0.16 – 0.23	0.68 – 1.32	
Vickery	0.133	2.1	—	—	
Minguez <i>et al.</i>	0.141	2.2	—	—	1.28
Ochoa & Fueyo	0.139	2.01	0.22	1.4	—
Sohankar <i>et al.</i>	0.126 – 0.132	2.03 – 2.32	0.16 – 0.20	1.23 – 1.54	—
Verstappen & Veldman	0.133	2.09	0.178	1.45	—
Rodi	0.13	2.3	0.14	1.15	1.46
LES results	0.09 – 0.15	2.02 – 2.77	0.14 – 0.27	1.15 – 1.79	0.94 – 1.68
RANS results	0.134 – 0.159	1.64 – 2.43	≈ 0 – 0.27	0.31 – 1.49	0.98 – 2.80

DNS of square cylinder

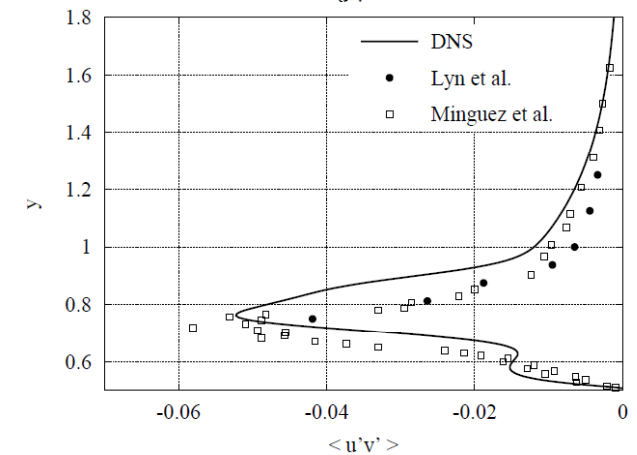
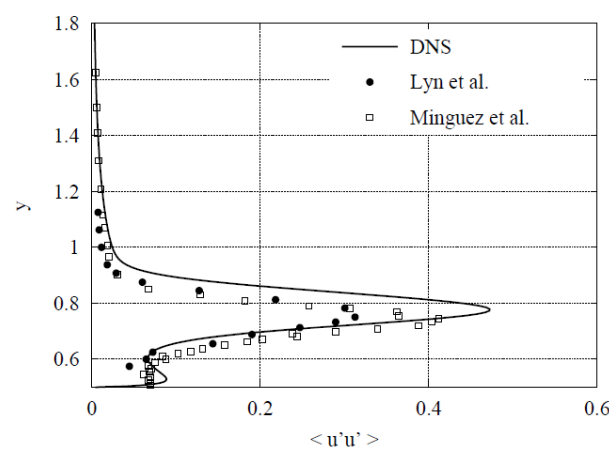
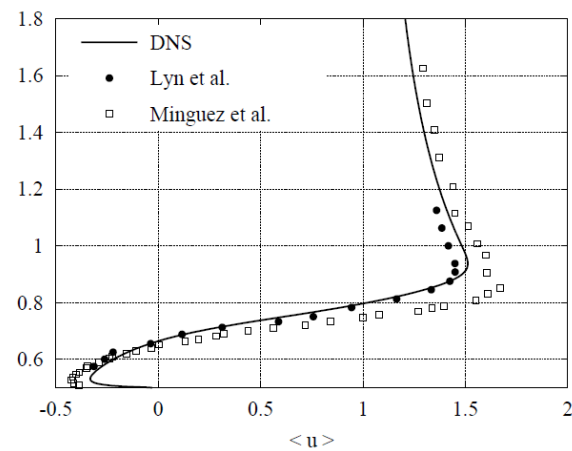
- Comparison of averaged profiles with experimental results



Point 1

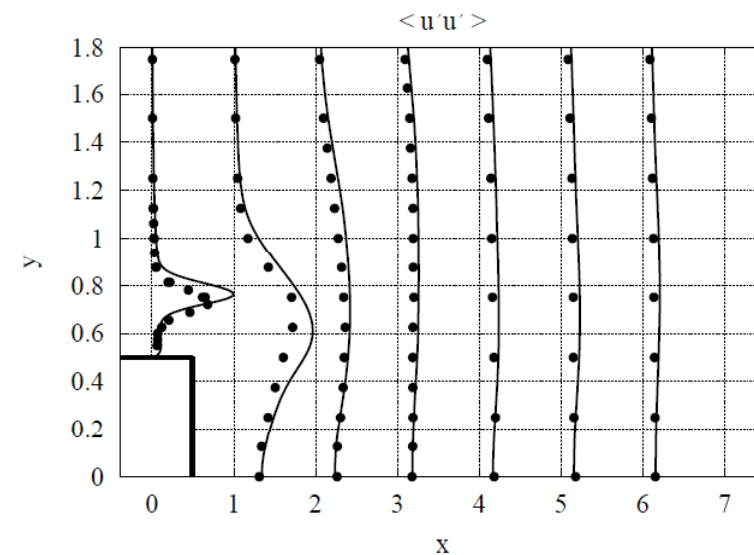
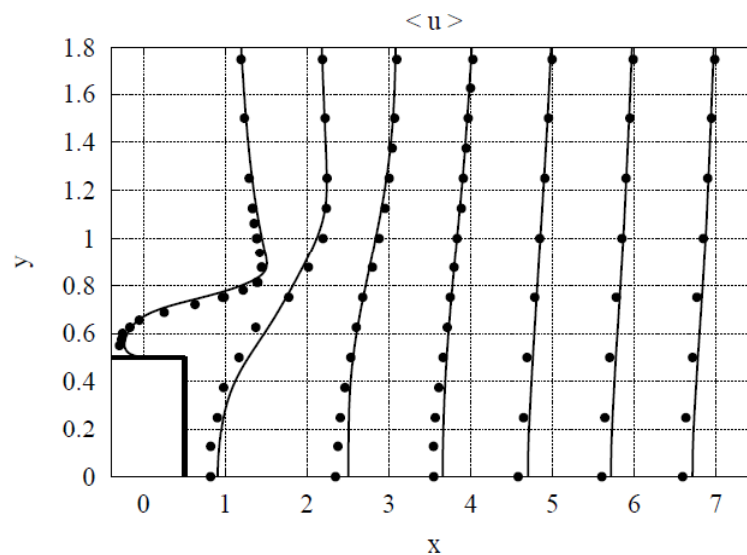
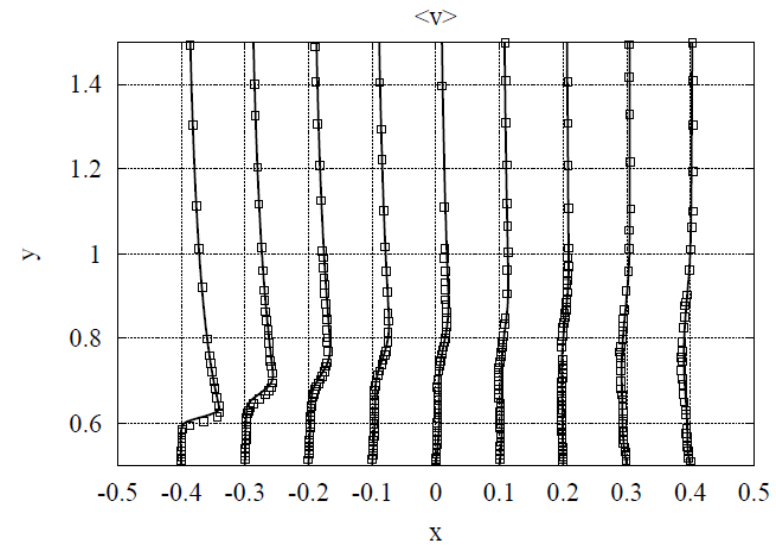
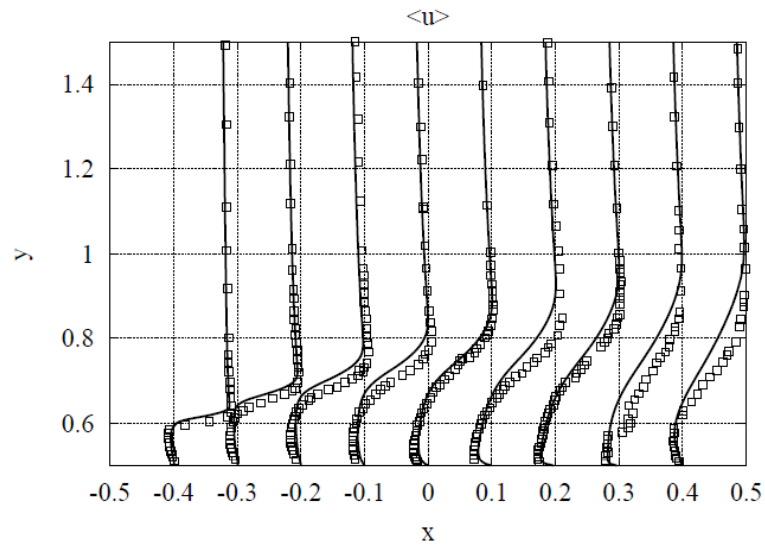


Point 2

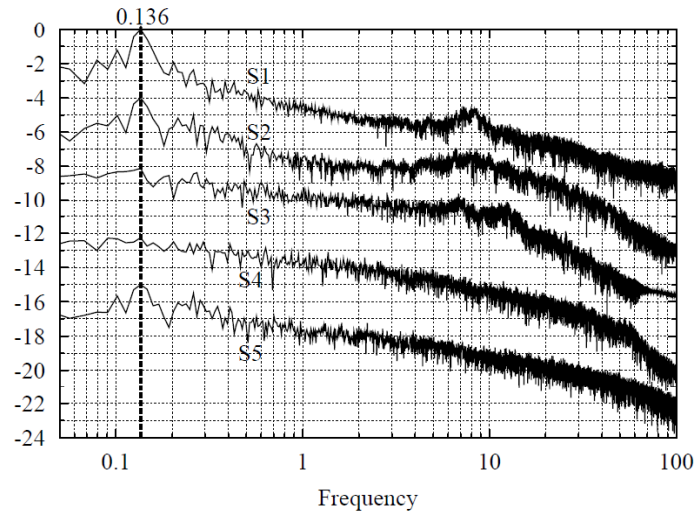


DNS of square cylinder

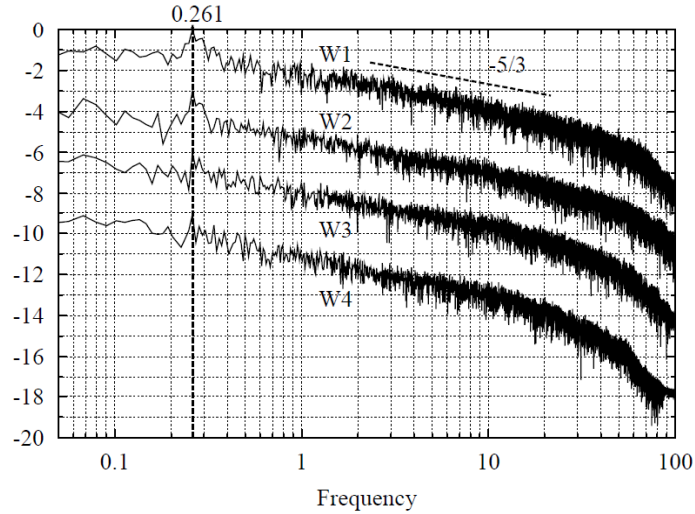
- Comparison of averaged profiles with experimental results (Minguez et al.)



DNS of square cylinder

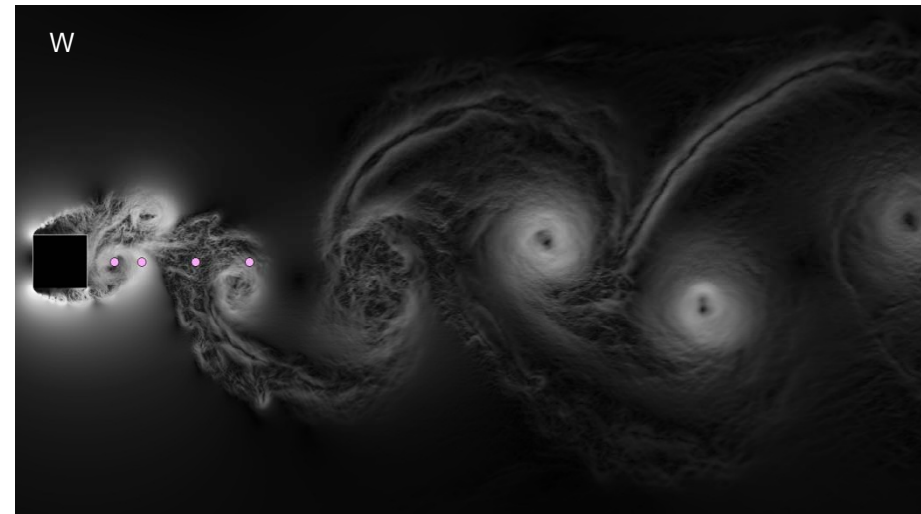
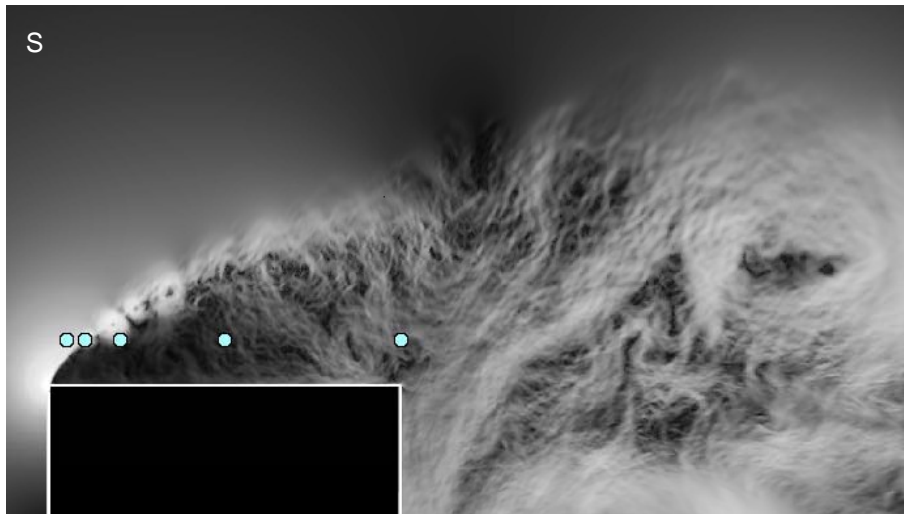


Normalized energy spectra of the stream-wise velocity at different locations



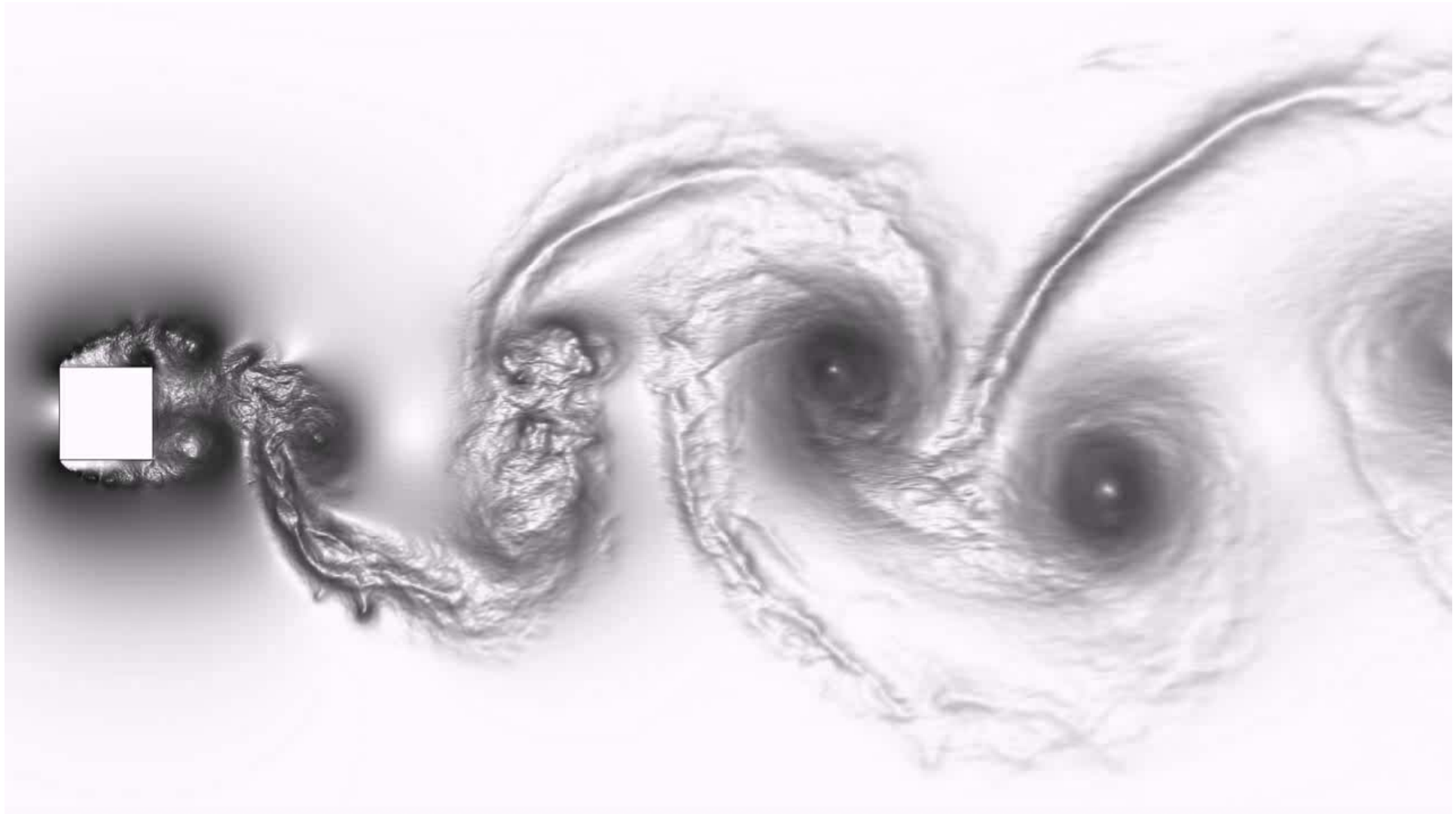
Point	x	y	Dominant frequency
S1	-0.45	0.63	0.136
S2	-0.40	0.63	0.136
S3	-0.30	0.63	0.136
S4	0.00	0.63	–
S5	0.50	0.63	0.136
W1	1.00	0.00	0.261
W2	1.50	0.00	0.261
W3	2.50	0.00	0.261
W4	3.50	0.00	0.261

List of monitoring locations



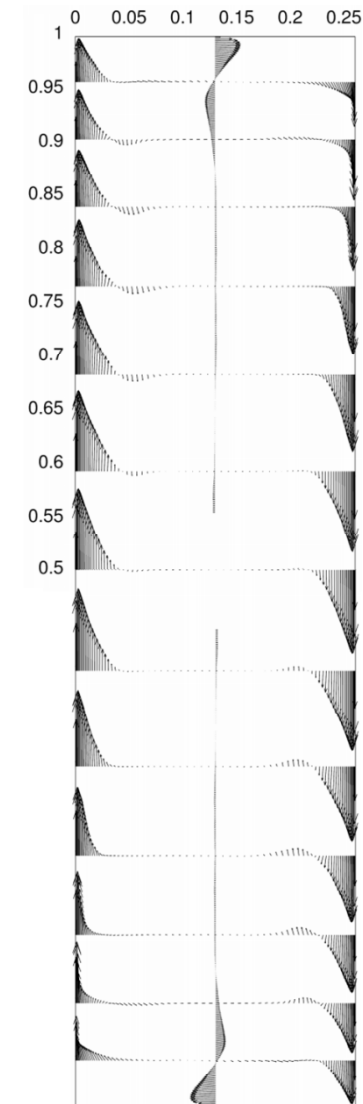
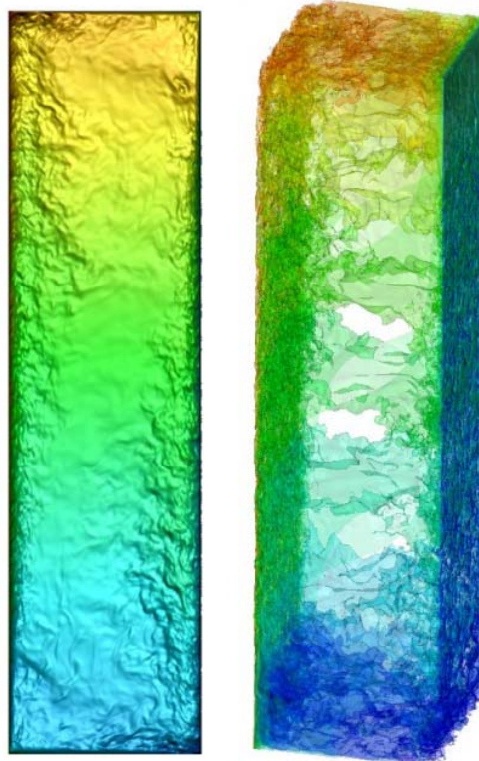
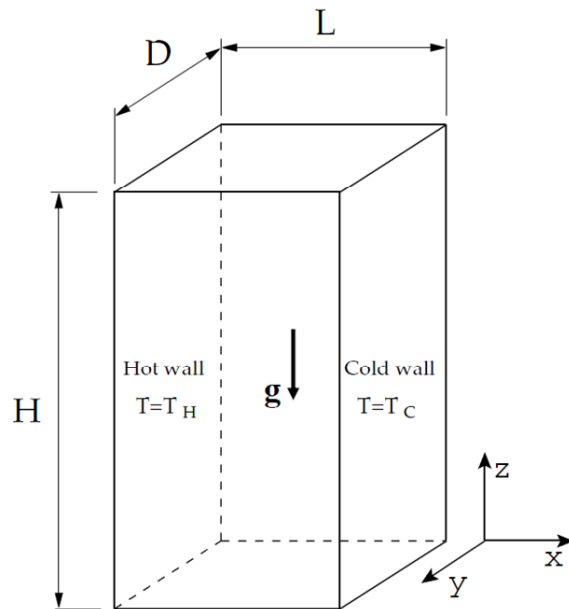


DNS of square cylinder



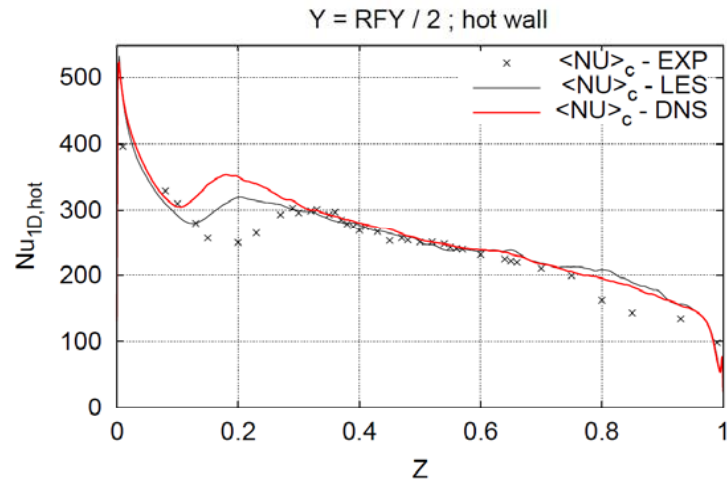
DNS of a 3D air-filled differentially heated cavity

- Rayleigh number 1.2×10^{11}
- Prandtl number 0.71 (air)
- Aspect ratio $1 \times 3.84 \times 0.86$
- Mesh size $400 \times 600 \times 200 = 48\text{M}$ nodes
- up to 400 CPUs on MVS10P supercomputer
- Overall computing price $\sim 160\text{K}$ CPU hours

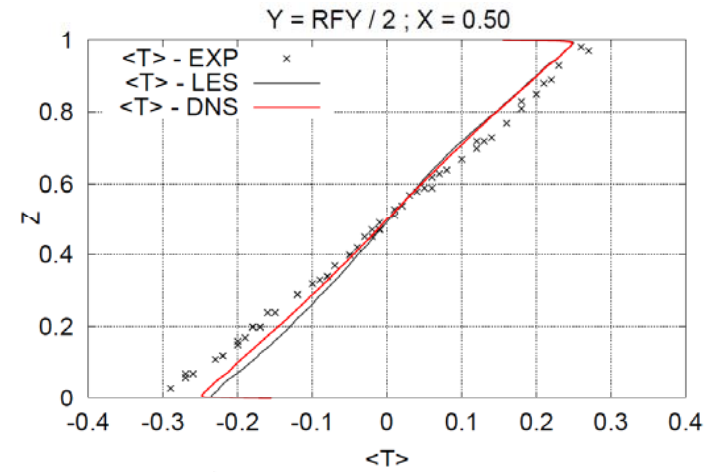




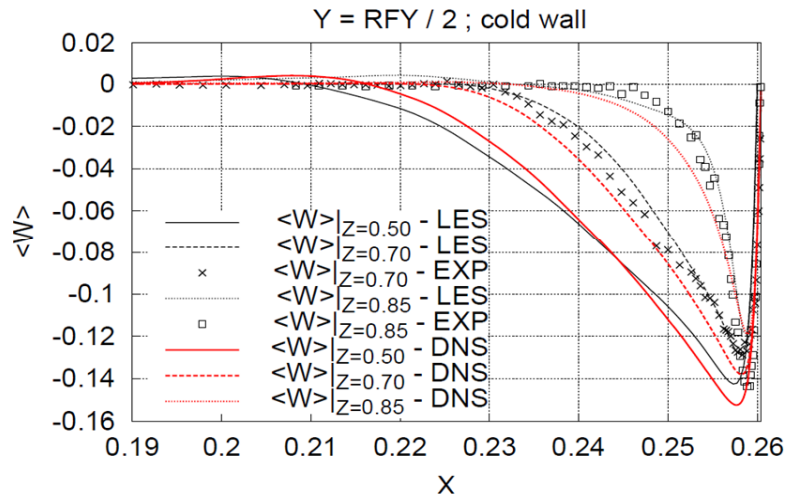
A 3D air-filled DHC: LES vs. DNS vs. experiment



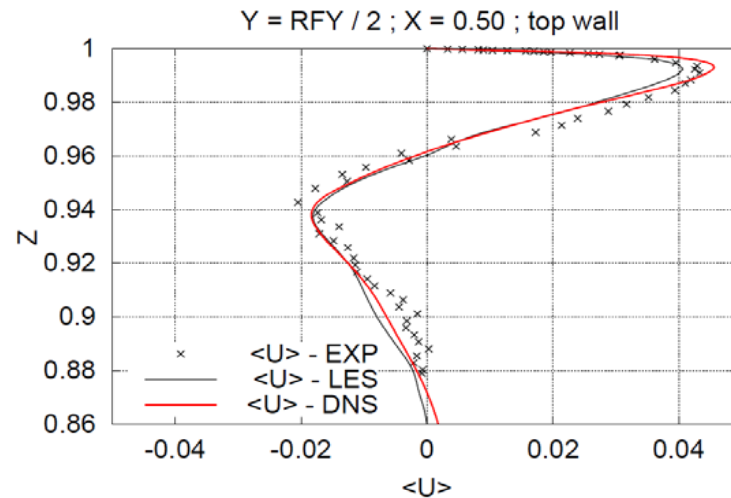
Nusselt profile at mid-depth on the hot wall.



Averaged temperature profile



horizontal profile at mid-depth



Vertical profile along the vertical centerline

Experiment

D. Saury, N. Rouger, F. Djanna, and F. Penot. Natural convection in an air-filled cavity: Experimental results at large Rayleigh numbers. *International Communications in Heat and Mass Transfer*, 38:679–687, 2011.

LES

A. Sergent, P. Joubert, S. Xin, and P. Le Quéré. Resolving the stratification discrepancy of turbulent natural convection in differentially heated air-filled cavities. Part II: End walls effects. *International Journal of Heat and Fluid Flow*, 39:15–27, 2013.



Conclusions

- We made several high-order F-V CFD codes for hybrid architectures:
compressible, unstructured hybrid mesh;
incompressible, unstructured hybrid mesh;
incompressible, structured mesh.
- Two new DNS have been computed. Results to be published soon.
- It is possible to obtain reasonable efficiency on modern hybrid systems with various finite-volume CFD algorithms...
- ...but it is such a torture!
- It is really a nightmare to develop, debug, maintain, modify a hybrid code
- CPU net efficiency 10-20%
 - GPU net efficiency 5-10%
 - Intel Xeon Phi net efficiency 2-3%

...

**Finally will appear ExaFLOP or WhateverFLOP supercomputer. Absolutely powerful.
That can compute absolutely nothing!**

LINPACK is the future of supercomputing...