Numerical simulation of atmosphere and ocean boundary layer turbulence on heterogeneous supercomputers

Mortikov E.V.[1,2], Debolskiy A.V.[1,3], Gashchuk E.M.[4,2], Glazunov A.V.[2]

[1]Research Computing Center, Lomonosov Moscow State University

[2]Marchuk Institute of Numerical Mathematics, Russian Academy of Science

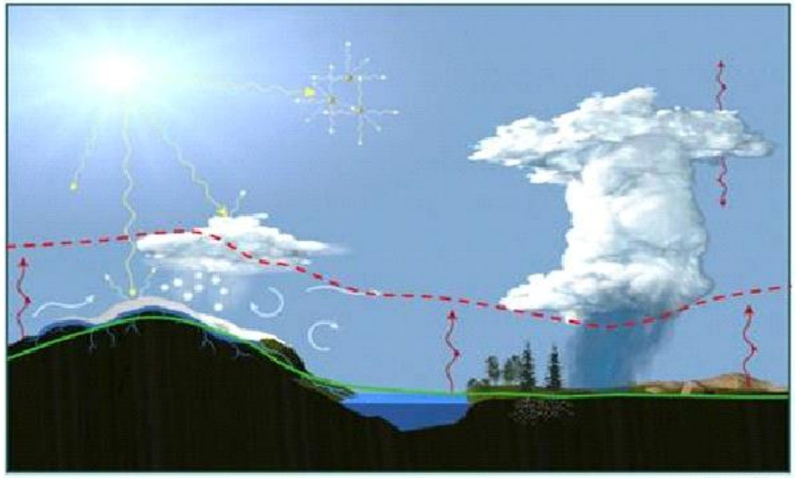
[3]A.M. Obukhov Institute of Atmospheric Physics, Russian Academy of Science

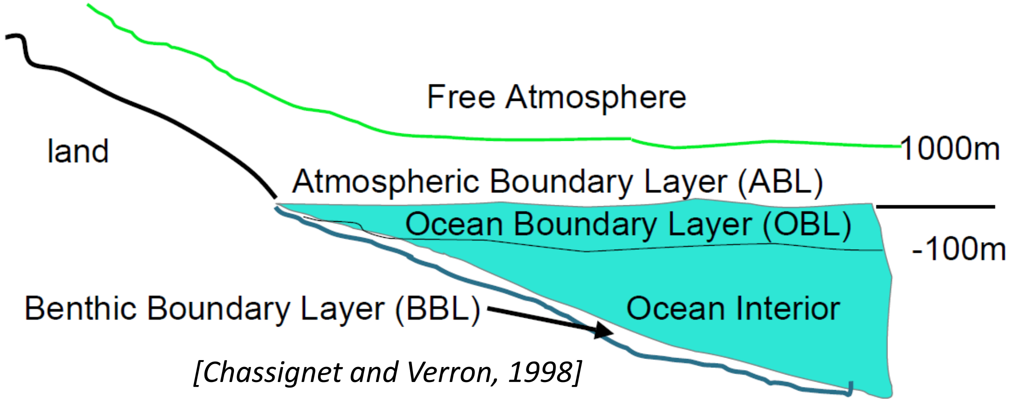
[4]Department of Mechanics and Mathematics, Lomonosov Moscow State University

Russian Supercomputing Days 2022

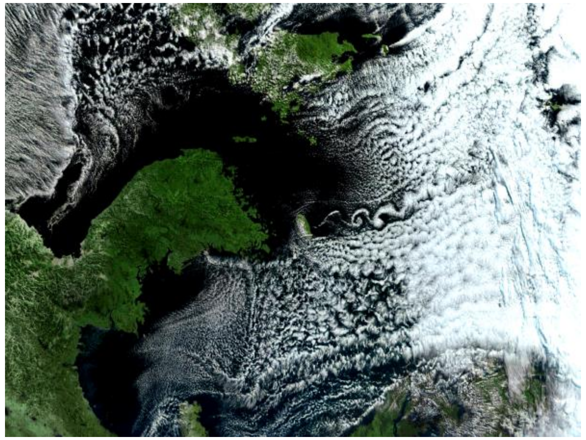
Atmosphere and ocean boundary layers

- **Atmospheric boundary layer**, $H_{ABL} \sim 10^2 - 10^3$ m
- **Oceanic boundary layer**, $H_{OBL} \sim 10^1 - 10^2$ m
- **Benthic boundary layer**, $H_{BBL} \sim 10^0 - 10^1$ m
- Free atmosphere and ocean interior connect through the OBL and ABL
- **Turbulence**, **stratification**, solar radiation, complex topography, clouds, surface waves, wave-turbulence interaction, Langmuir circulation etc.
- **Turbulence with very high Reynolds numbers**
  - ABL: Re ~ $\mathbf{10^9}$, OBL: Re ~ $\mathbf{10^6 - 10^7}$, BBL: Re ~ $\mathbf{10^5 - 10^6}$
- **Parameterizations for NWP and climate models**
  - **INMCM**, Institute of Numerical Mathematics climate model
  - **SL-AV**, Vorticity-divergence semi-Lagrangian global atmospheric model – NWP model used at Russian meteorological center





[Chassignet and Verron, 1998]



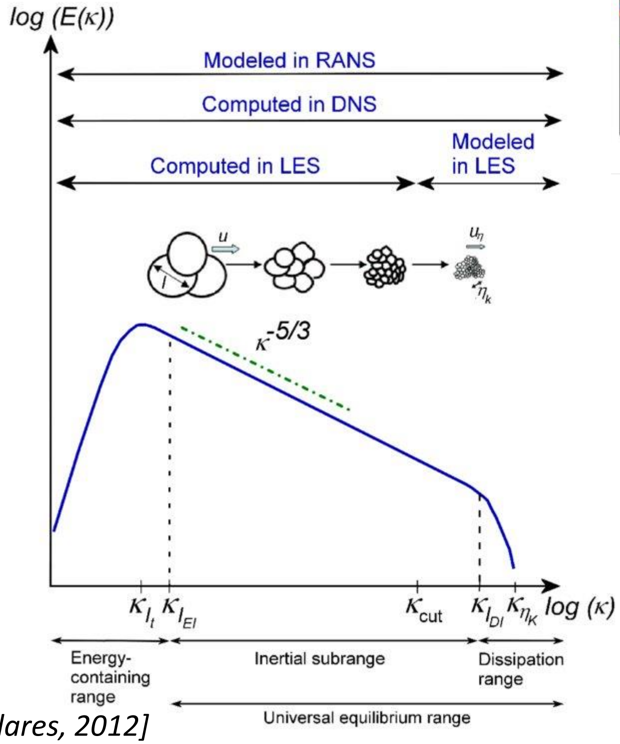
Numerical simulation of turbulent flows

- **DNS – Direct numerical simulation** – all scales explicitly resolved
- **LES – Large eddy simulation** – inertial range at least partially resolved on computational grid
- **RANS – Reynolds averaged Navier-Stokes** – fully modelled turbulence

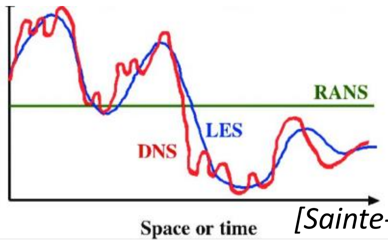
$$\frac{\partial u_i}{\partial t} = -\frac{\partial u_i u_j}{\partial x_j} - \frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}}\frac{\partial^2 u_i}{\partial x_i \partial x_j} + F_i^e$$

$$\frac{\partial u_i}{\partial x_i} = 0$$

$$\frac{\partial T}{\partial t} + \frac{\partial u_j T}{\partial x_j} = \frac{1}{\mathrm{PrRe}}\frac{\partial^2 T}{\partial x_j \partial x_j}$$



[Hilares, 2012]



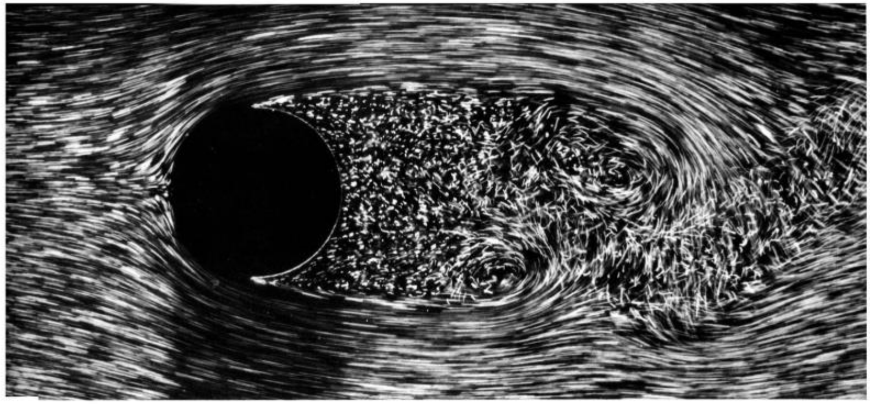
[Sainte-Rose, 2010]

$$\mathrm{Re} = \frac{UL}{(\nu/\rho_0)}$$

Reynolds number

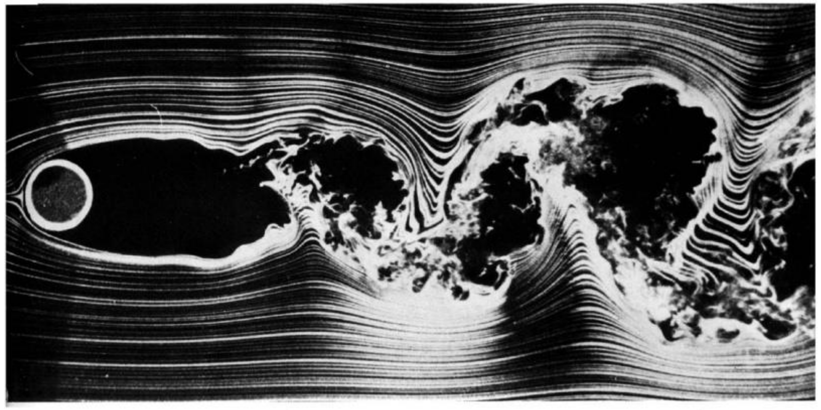
$$\tilde{\mu} = \mu/L$$
$$\tilde{\mu} = O(\mathrm{Re}^{-3/4})$$

Viscous length scale (size of the smallest eddies)



$\mathrm{Re} = 2000$ $\Delta \sim (1/300)L$

$N \sim 10^8$



$\mathrm{Re} = 10000$ $\Delta \sim (1/1000)L$

$N \sim 10^9$

Unified DNS/LES/RANS numerical model

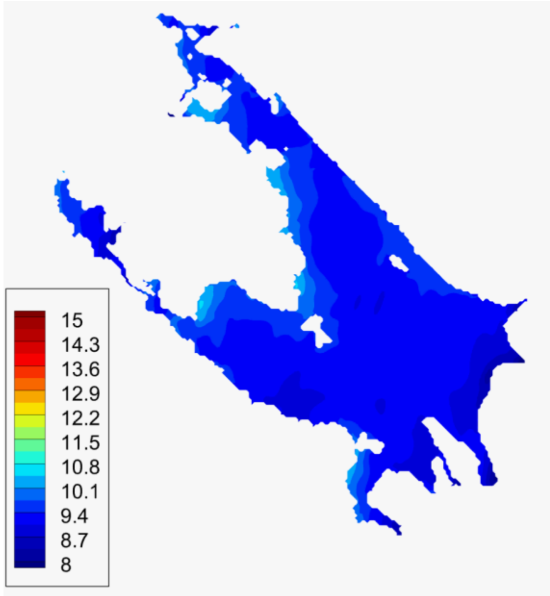
- **Unified DNS-, LES-, RANS- code developed at RCC MSU & INM RAS**
- **DNS**
  - Navier-Stokes equations for viscous incompressible fluid
  - Boussinesq approximation for stratified flows
- **LES**
  - Filtered Navier-Stokes eq.
  - Smagorinsky model, SSM, AMD …
  - Dynamic procedure
- **RANS**
  - Non-hydrostatic urban environment model
  - Hydrostatic lake model + biochemistry model (with IAP RAS)
  - Two-equation turbulence models
- **Multiphase flow simulations** (CLSVOF)
- **Lagrangian particles**

$$\frac{\partial \overline{u}_i}{\partial t} = -\frac{\partial \overline{u}_i \overline{u}_j}{\partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} - \frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 \overline{u}_i}{\partial x_i \partial x_j} + \overline{F}_i^e$$

$$\frac{\partial \overline{u}_i}{\partial x_i} = 0$$

$\tau_{ij} = \overline{u_i u_j} - \overline{u}_i \overline{u}_j$ – **closure needed**

$$\boxed{\tau_{ij}^{smag} = -2\left(C_s \overline{\Delta}\right)^2 \left|\overline{S}\right| \overline{S}_{ij}}$$



Rybinsk Reservoir, surface temperature, may-june

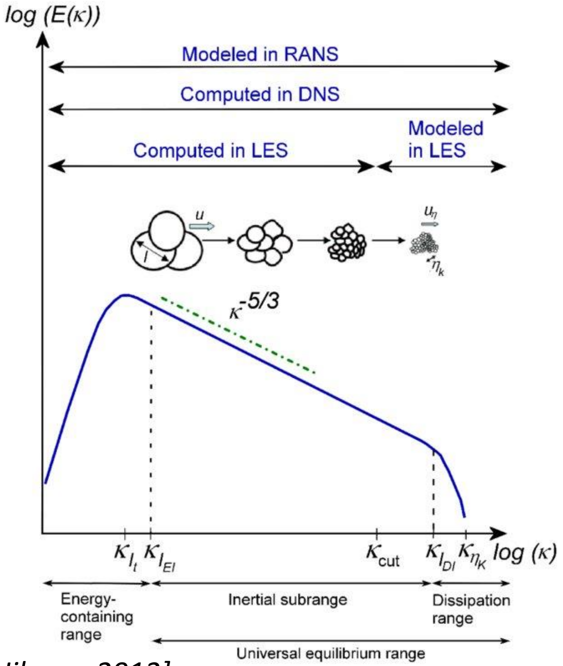
Direct numerical simulation

- Numerical solution of Navier-Stokes equations – ***no turbulence model!***

$$\frac{\partial u_i}{\partial t} = -\frac{\partial u_i u_j}{\partial x_j} - \frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}}\frac{\partial^2 u_i}{\partial x_j \partial x_j} + F_i^e$$

$$\frac{\partial u_i}{\partial x_i} = 0$$

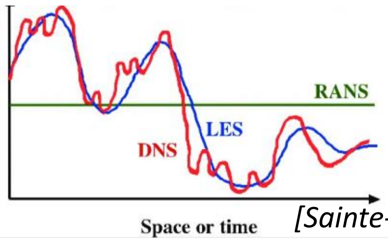
$$\frac{\partial T}{\partial t} + \frac{\partial u_j T}{\partial x_j} = \frac{1}{\mathrm{PrRe}}\frac{\partial^2 T}{\partial x_j \partial x_j}$$



[Hilares, 2012]

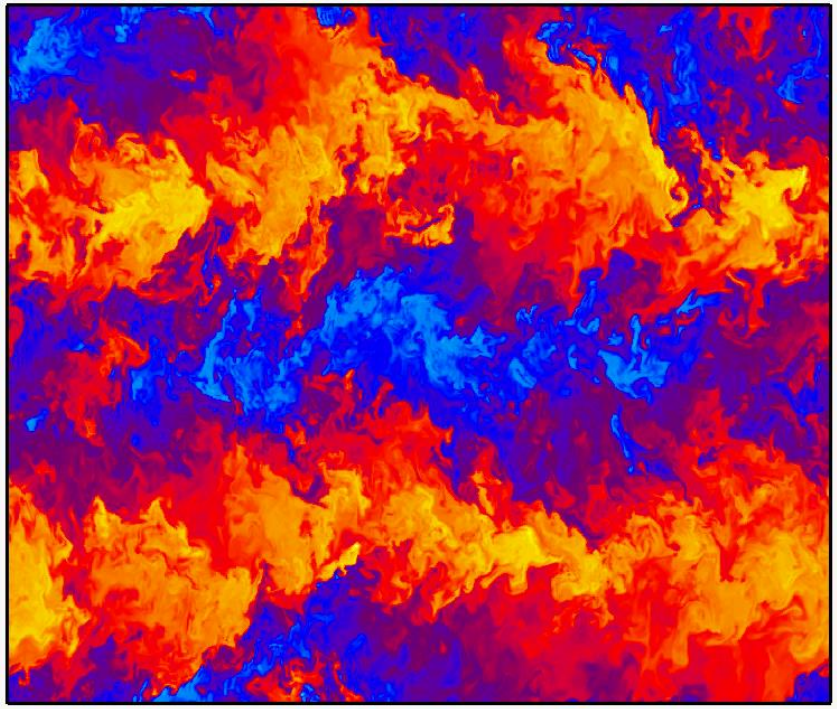
$$\mathrm{Re} = \frac{UL}{\nu}$$ **– Reynolds number**

$$\mathrm{Pr} = \frac{\nu}{\chi}$$ **– Prandtl number**



[Sainte-Rose, 2010]

Couette flow neutral stratification



[Mortikov et al., 2019]

$$\tilde{\mu} = \mu / L$$

$$\tilde{\mu} = O(\mathrm{Re}^{-3/4})$$

– viscous length scale (size of the smallest eddies)

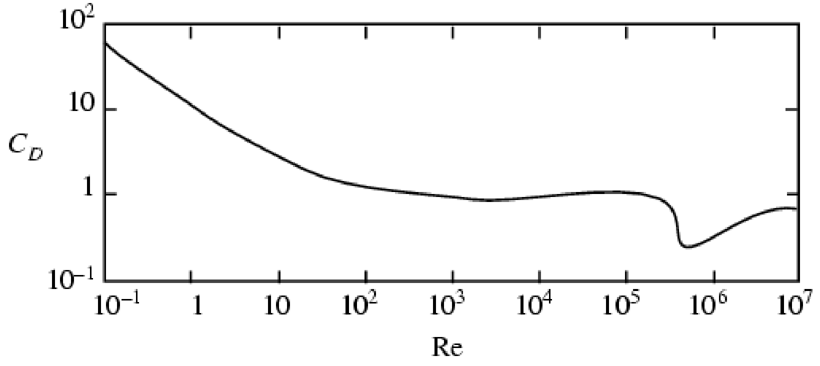
<u>**High grid resolution is necessary**</u>

$$N \sim O(10^8) - O(10^9)$$

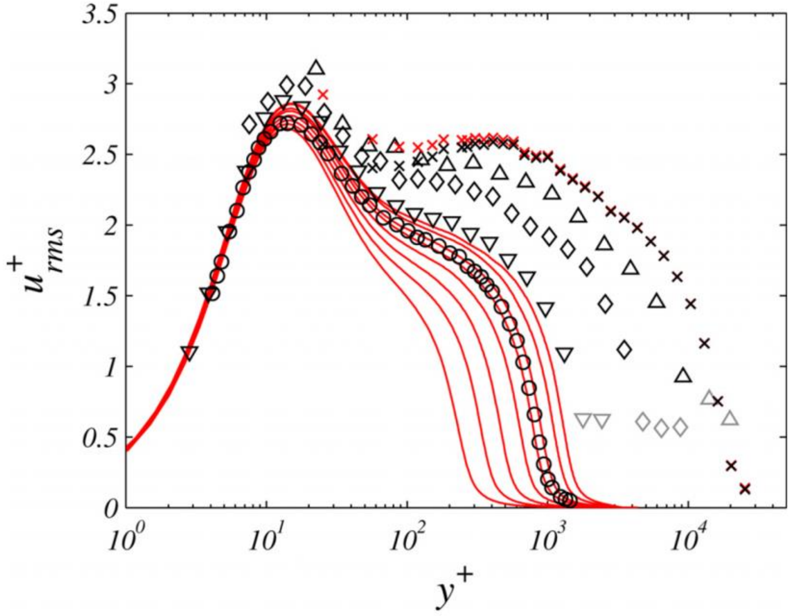
Why DNS?

- Numerical solution of Navier-Stokes equations – ***no turbulence model!***
- **Physics of fluids & turbulence research**
- **Development and verification of subgrid parameterizations for LES/RANS models**
- DNS of turbulent Couette flow **Re=120 000**: **10^8** grid cells, **1000** CPU cores, **72** hours of computations
  - **Re=500 000** when using all CPU cores of “Lomonosov-2” for a week
  - **Re=1 000 000** when using all CPU cores of “Tianhe-2” for a week
  - **Can we increase Re when using coprocessors, e.g. GPUs/Intel Xeon Phi?**
- **How large Re is large enough?**
  - Maximum Re values obtained in DNS **comparable with OBL**, but still **lower by a couple of orders of magnitude compared with ABL turbulence**

Drag coefficient vs Re for flow over circular cylinder

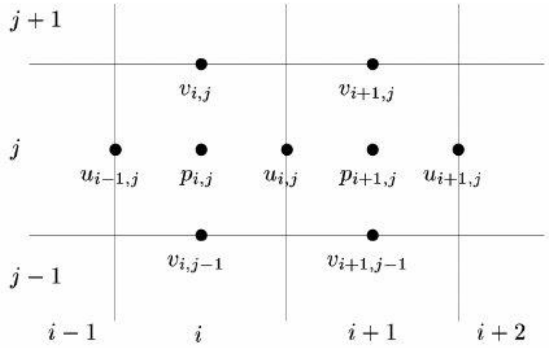


Streamwise velocity RMS channel flow: DNS & experiments (Alfredsson et al., 2011)



Numerical model

- **Finite-difference** 2nd and 4th order schemes on rectangular staggered grids
  - Conservation of momentum and energy *[Morinishi et al., 1998; Vasilyev, 2000]*
  - Finer grid resolution in **near-wall regions**



staggered grid alignment

- **Fractional step method**

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n}}{\Delta t}+(\mathbf{u}\cdot\nabla\mathbf{u})^{n+\frac{1}{2}}=\frac{1}{\mathrm{Re}}\left(\nabla^2\mathbf{u}\right)^{n+\frac{1}{2}}-\nabla p^{n+1}+(\mathbf{F}^e)^{n+\frac{1}{2}}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0$$

$$p^{n+1}=p^{n}+\phi^{n+1}$$

$$\frac{\tilde{\mathbf{u}}-\mathbf{u}^{n}}{\Delta t}+(\mathbf{u}\cdot\nabla\mathbf{u})^{n+\frac{1}{2}}=\frac{1}{\mathrm{Re}}\left(\nabla^2\mathbf{u}\right)^{n+\frac{1}{2}}-\nabla p^{n}+(\mathbf{F}^e)^{n+\frac{1}{2}}$$

$$\frac{\mathbf{u}^{n+1}-\tilde{\mathbf{u}}}{\Delta t}=-\nabla\phi^{n+1}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0$$

$$\nabla\cdot\nabla\phi^{n+1}=\frac{\nabla\cdot\tilde{\mathbf{u}}}{\Delta t}$$

$$\mathbf{u}^{n+1}=\tilde{\mathbf{u}}-\Delta t\cdot\nabla\phi^{n+1}$$

$$p^{n+1}=p^{n}+\phi^{n+1}$$

- **Using explicit approximation for nonlinear terms** – Adams-Bashforth methods
- **Multigrid method** for solution of system of linear equations ~ **O(N) complexity**
- **FD & immersed boundary methods** for handling complex (& moving) geometry

Direct numerical simulation

- Numerical solution of Navier-Stokes equations

$$\frac{\partial u_i}{\partial t} = -\frac{\partial u_i u_j}{\partial x_j} - \frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j \partial x_j} + F_i^e$$

$$\frac{\partial u_i}{\partial x_i} = 0$$

$$\frac{\partial T}{\partial t} + \frac{\partial u_j T}{\partial x_j} = \frac{1}{\mathrm{PrRe}} \frac{\partial^2 T}{\partial x_j \partial x_j}$$

Buoyancy (stratified turbulence) and **Coriolis** terms

Drag force:

Rayleigh friction &
Forchheimer drag – model surface roughness elements, e.g. canopy

$$\mathbf{F}_d = C_d a(z) |\overline{\mathbf{u}}| \overline{\mathbf{u}}$$

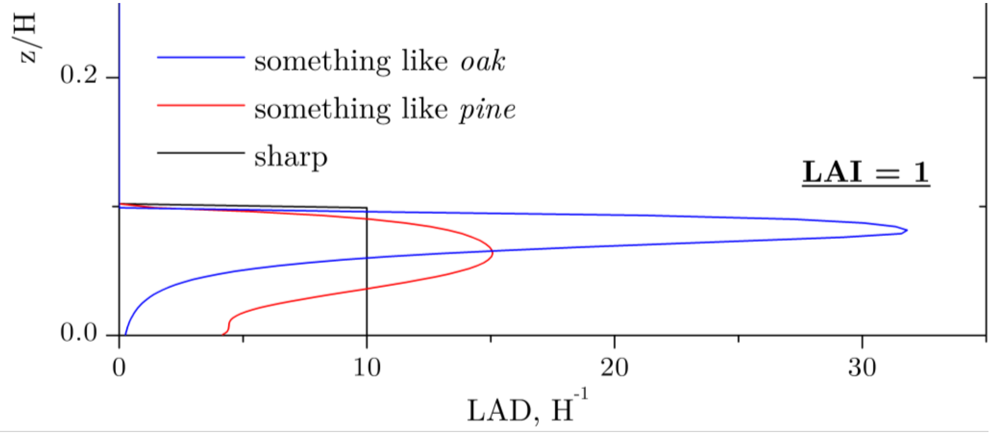
– different drag definitions, see [Bhattacharjee et al., 2022]

- **Passive tracers transport**

$$\frac{\partial C_k}{\partial t} + \frac{\partial u_j C_k}{\partial x_j} + u_j \left(\frac{\partial C_k}{\partial x_j} \right)_0 = \frac{1}{\mathrm{ScRe}} \frac{\partial^2 C_k}{\partial x_j \partial x_j} - T_l^{-1} C_k$$

External scalar gradient

Decay with fixed life-time



- Dirichlet boundary conditions or prescribed flux
- **Dynamics CPU time ~ 10 species transport** *[Mortikov & Debolskiy, 2021]*

Direct numerical simulation

- **Lagrangian particle transport**

$$\frac{d\mathbf{x}_i(t)}{dt} = \mathbf{v}_i$$

$$m_i \frac{d\mathbf{v}_i(t)}{dt} = \mathbf{f}_B + \mathbf{f}_D + \ldots$$

passive particles:

$$\mathbf{v}_i = \mathbf{u}(\mathbf{x}_i(t), t)$$

drag force:

$$\mathbf{f}_D = m_i \frac{(\mathbf{u}(\mathbf{x}_i(t), t) - \mathbf{v}_i)}{\tau_i} g(\mathrm{Re}_i)$$

$$\tau_i = \frac{\rho_i d_i^2}{18\nu}, \mathrm{Re}_i = \frac{\rho_f d_i |\mathbf{u}(\mathbf{x}_i(t), t) - \mathbf{v}_i|}{\nu}$$

buoyancy force:

$$\mathbf{f}_B = (\rho_i - \rho_f)\, V_i g$$

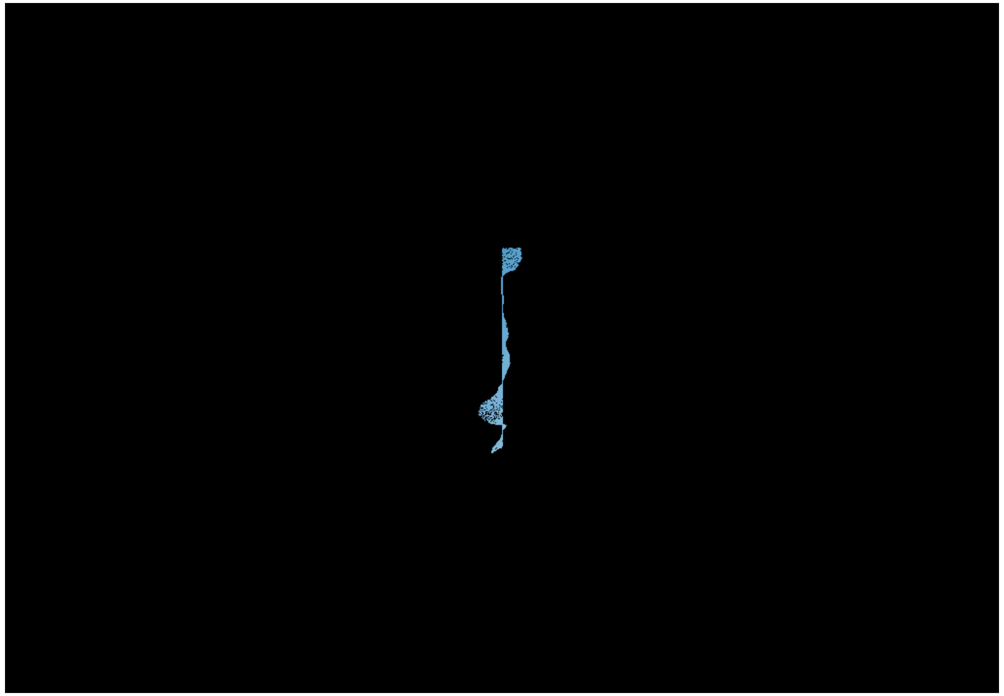
  - Trilinear interpolation
  - Handling multiple sources/sinks & *particle groups*
  - Particle decay with fixed life-time
  - Elastic collisions with walls

- **Calculate particles trajectories**

- **Two-way disperse phase/fluid coupling**

$$\mathbf{F}_k^e = (1/\rho_f) \sum_{p \in k} -\mathbf{f}_D \delta_k(\mathbf{x}_p)$$

- **Coupling with external particle library** *[Varentsov, 2022]*

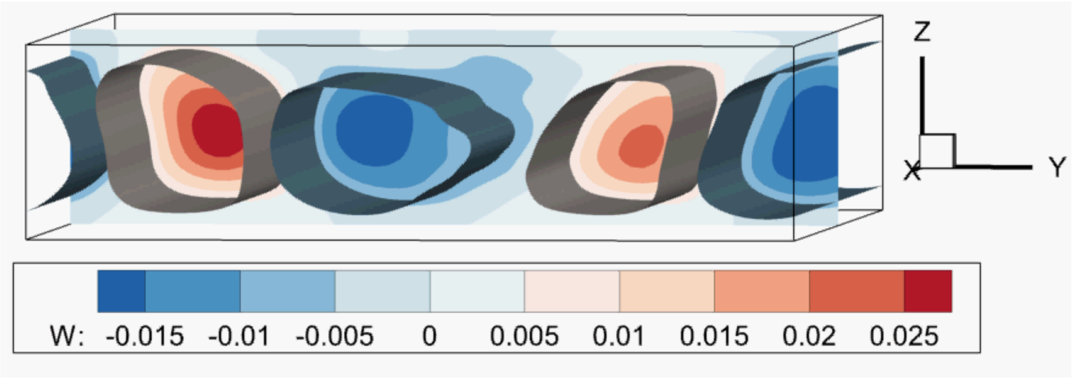


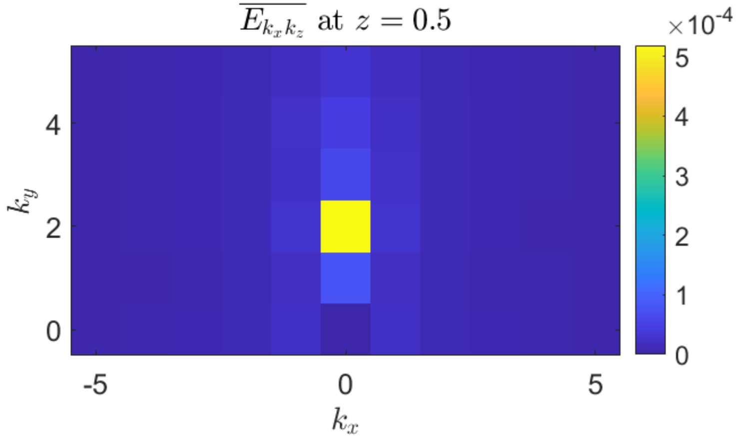
DNS as a research tool

- **Large data-sets**, single snapshot of velocity & pressure: **O(1)-O(10) GB**
- **On the fly flow analysis**
  - **Different modes of data output:** 3D, 2D, 1D, point-like measurements & integral flow characteristics
  - **Statistics calculation:** first-order moments & up to budget eqs. for second-order moments

$$\frac{D\langle u_i u_k\rangle}{Dt} = \left(-\langle u_i u_j\rangle\frac{\partial U_k}{\partial x_j} - \langle u_k u_j\rangle\frac{\partial U_i}{\partial x_j}\right) - \frac{\partial\langle u_i u_j u_k\rangle}{\partial x_j} - \left\langle u_k\frac{\partial p}{\partial x_i} + u_i\frac{\partial p}{\partial x_k}\right\rangle + \frac{g}{\theta_0}\left(\delta_{k3}\langle u_i\theta\rangle + \delta_{i3}\langle u_k\theta\rangle\right) + \nu\left\langle u_k\frac{\partial^2 u_i}{\partial x_j\partial x_j} + u_i\frac{\partial^2 u_k}{\partial x_j\partial x_j}\right\rangle$$

  - **Spectrum analysis:** 1D & 2D spectra, spectral energy density time series
  - Single variable **probability density functions** & **joint p.d.f.**
  - **Flow manipulation at run-time**, e.g. filtration

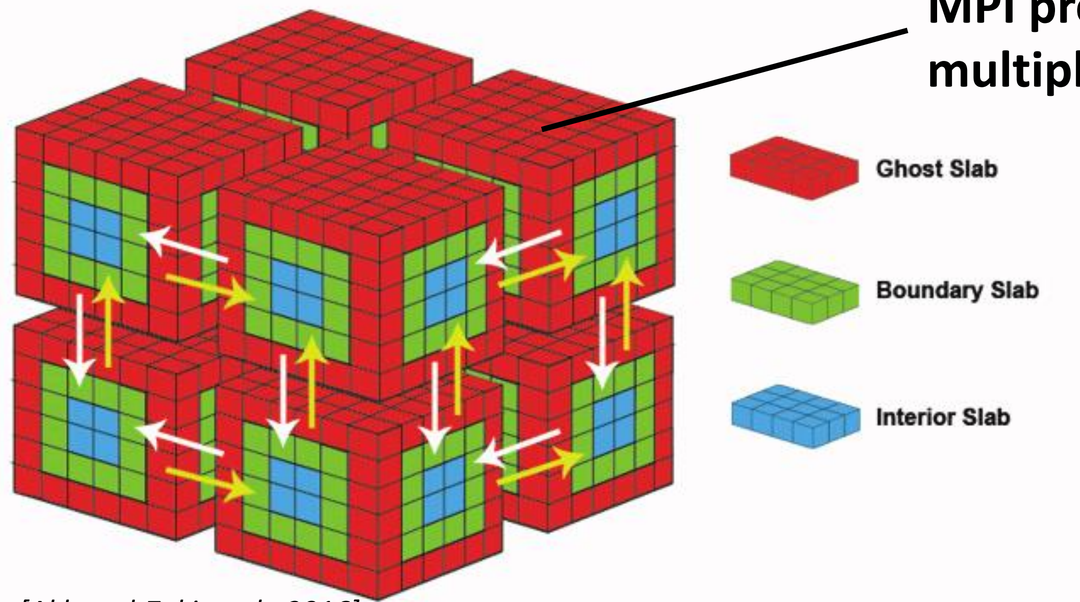




Large-scale structures (rolls) in neutral Couette flow

Parallel implementation

- **C/C++ code**
- **MPI domain decomposition**



MPI process with single or multiple OpenMP threads

[Akhmed-Zaki et al., 2016]

Lomonosov-2 supercomputer, ©MSU, T-platforms

Intel Xeon CPU
Nvidia GPU

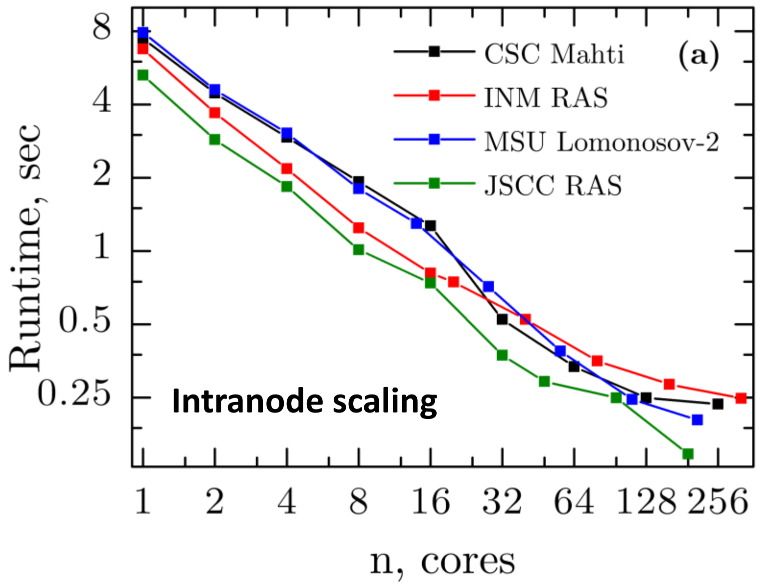
MVS-10P, -10Q supercomputers, ©RAS, RSC Group

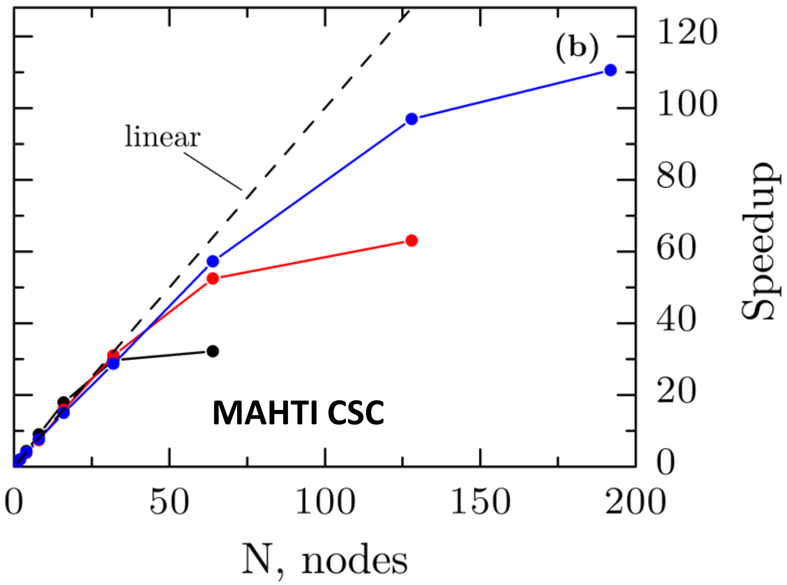
Intel Xeon CPU
Intel Xeon Phi

- **Using OpenMP on multicore processors**
  - Overlap MPI communications with computations
  - Cache-aware algorithms/thread synchronization become more important
- **Each MPI process works with particles only inside the grid block it holds – particles move from one MPI process to another**

Parallel implementation

- **MPI-OpenMP CPU scaling**





Scaling up to 25000 cores on CSC Mahti supercomputer (AMD EPYC)

- **Code ported on Intel Xeon Phi architecture**
- **Running on ARM-based CPUs (Kunpeng 920 processors)**

Argonne Theta Supercomputer (4096 Intel Xeon Phi cards)

Table 1. LR case run-time, in seconds per 1000 time steps

	AMD Rome 7H12	Intel Xeon Gold 6140	Kunpeng 920
single core, x2 and (x4)	39.94 (54.58)	48.53 (59.84)	123.70 (166.28)
max cores, x2 and (x4)	2.16 (2.89)	5.94 (7.94)	2.12 (2.90)

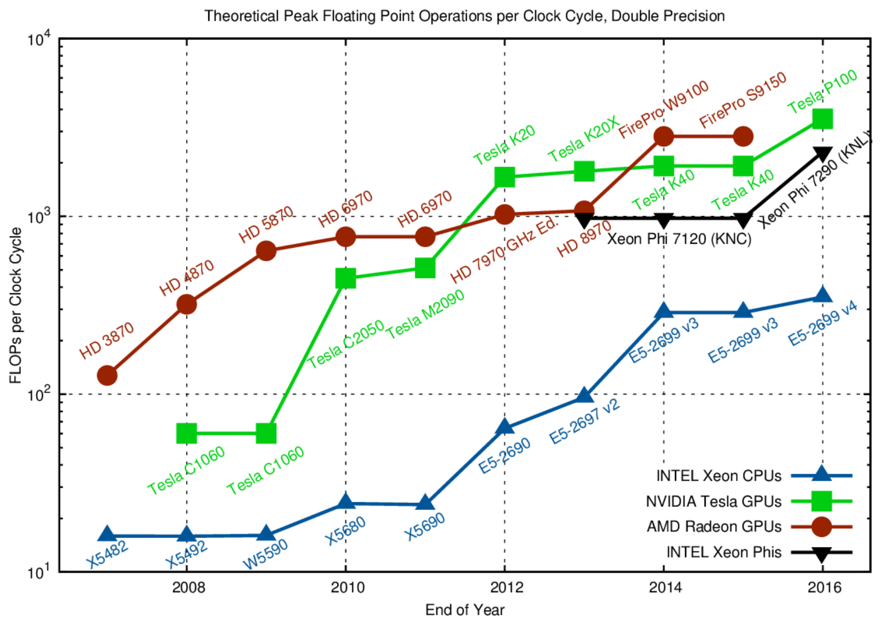
Table 2. HR case run-time, in seconds per 1000 time steps

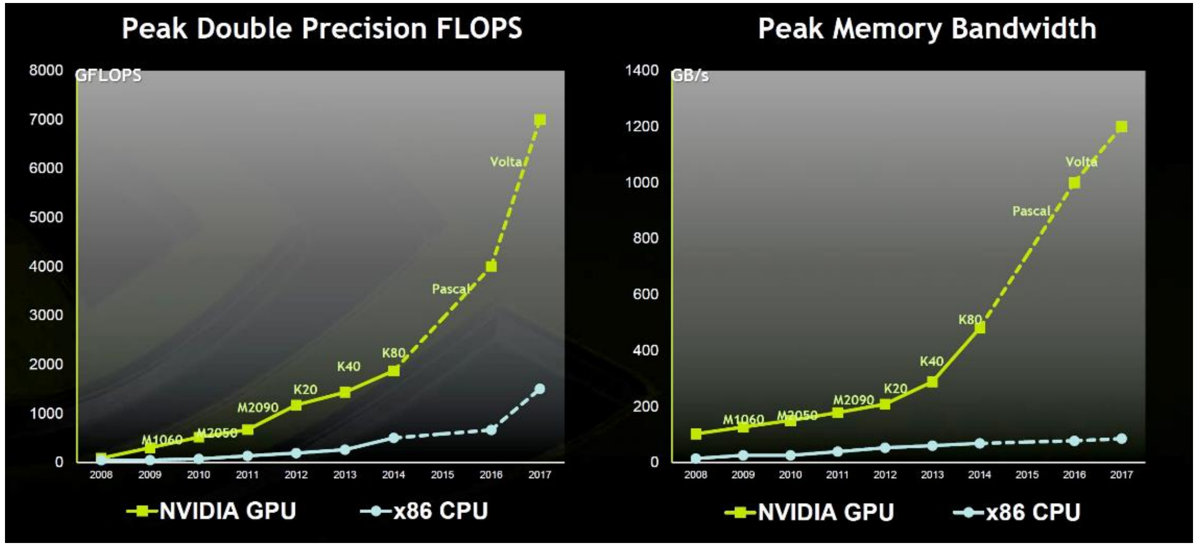
	AMD Rome 7H12	Intel Xeon Gold 6140	Kunpeng 920
single core, x2 and (x4)	285.23 (372.19)	391.11 (458.09)	1018.81 (1511.12)
max cores, x2 and (x4)	10.23 (13.49)	26.83 (32.81)	18.27 (25.02)

[Mortikov and Debolskiy, 2021]

Why GPUs?

- **Graphics Processing Units** (GPUs) – energy efficiency, cheap ($/FLOPs) & high performance ***for some problems***
- Increase in performance of supercomputers in the last 10 years **in large part due to the advent of coprocessors**: **GPUs** (*Lomonosov-2, Summit*) or **Intel Xeon Phi** (*Tianhe-2*)
- Speed-up of hydrodynamic models when ported to GPUs:
  - **x20-x40** compared with CPU core
  - **x2-x4** compared with CPU node
- Speed-up of molecular dynamics when ported to GPUs:
  - **x500-x1000** compared with CPU core
- Adapt models & algorithms to new *Frontiers*: ***exascale*** and ***post-exascale*** systems

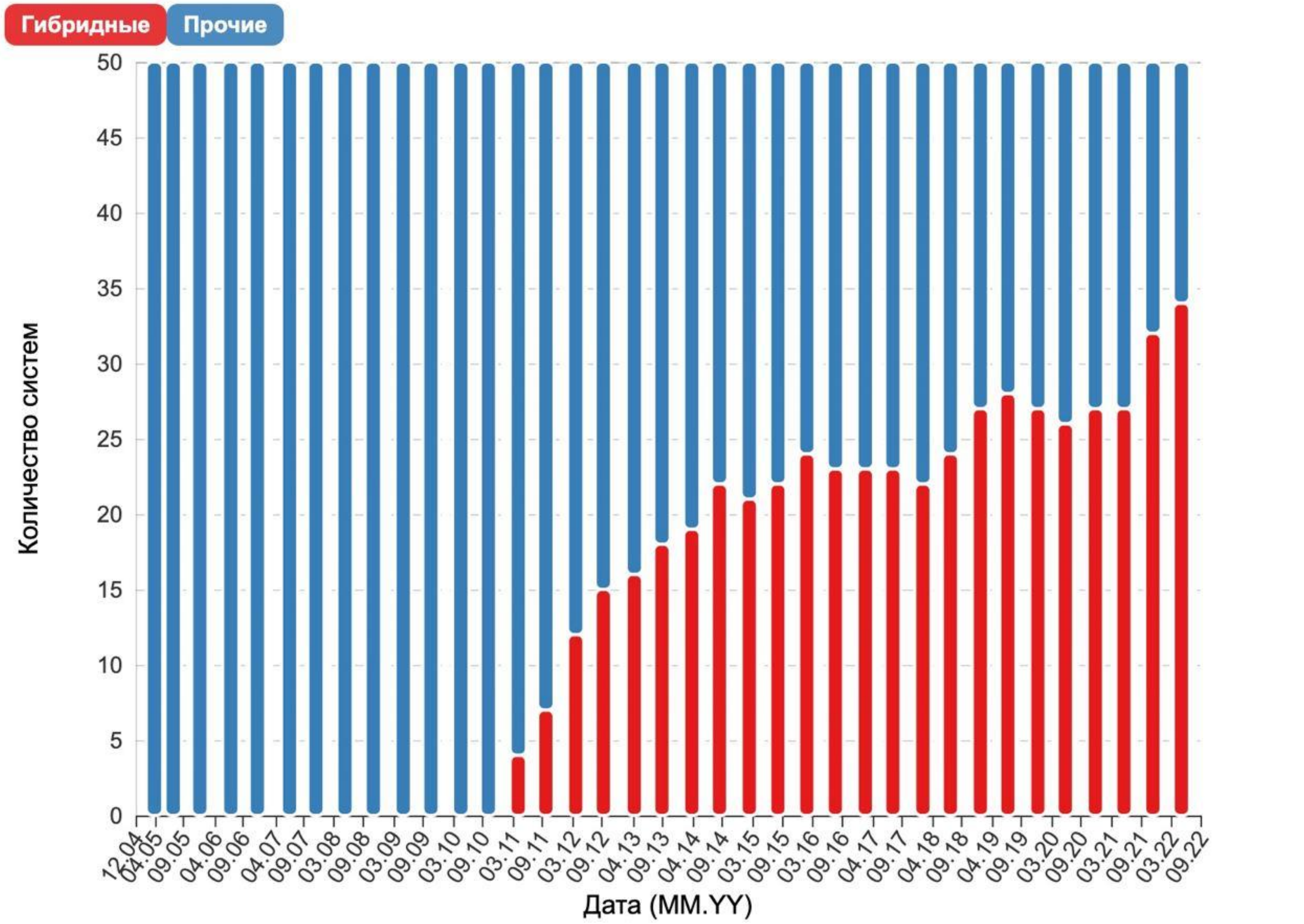




Why GPUs?

1	«Червоненкис» Яндекс, Москва	199 398 1592	199: CPU: 2x AMD EPYC 7702 , 1024 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet
2	«Галушкин» Яндекс, Москва	136 272 1088	136: CPU: 2x AMD EPYC 7702 , 1024 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet
3	«Ляпунов» Яндекс, Москва	137 274 1096	137: CPU: 2x AMD Epyc 7662, 512 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet
4	«Кристофари Нео» SberCloud (ООО «Облачные технологии») , СберБанк, Москва	99 198 792	99: CPU: 2x AMD EPYC 7742, 2048 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / 10 Gigabit Ethernet / 200 Gigabit Ethernet
5	«Кристофари» SberCloud (ООО «Облачные технологии») , СберБанк, Москва	75 150 1200	75: NVIDIA DGX-2 CPU: 2x Intel Xeon Platinum 8168 24C 2.7GHz, 1536 GB RAM Acc: 16x NVIDIA Tesla V100 EDR Infiniband / 100 Gigabit Ethernet / 10 Gigabit Ethernet
6	«Ломоносов-2» Московский государственный университет имени М.В.Ломоносова, Москва	1696 1696 1856	1536: CPU: 1x Intel Xeon E5-2697v3, 64 GB RAM Acc: 1x NVIDIA Tesla K40M 160: CPU: 1x Intel Xeon Gold 6126, 96 GB RAM Acc: 2x NVIDIA Tesla P100 FDR Infiniband / Gigabit Ethernet / FDR Infiniband

Количество гибридных систем (по наличию ускорителей) в редакциях Top50

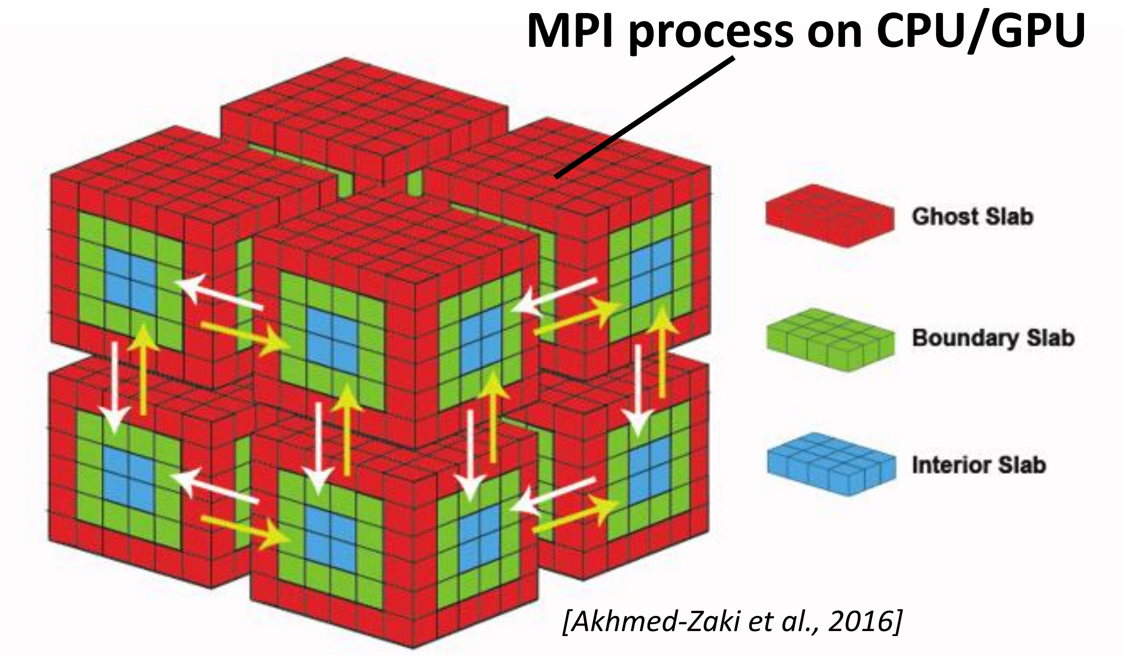


DNS on CPU/GPU systems

- **DNS model fully ported on hybrid CPU/GPU systems**
  - Includes dynamics, Lagrangian particles transport & run-time flow processing support on GPUs
  - Using C/C++ & MPI/OpenMP/CUDA [only Nvidia GPUs]
- **Just compile & run** – single executable:

```
./exe -arch cpu   →  all MPI processes on CPU
./exe -arch gpu   →  all MPI processes on GPU
./exe -arch mix   →  MPI processes both on CPU/GPU
```

- **Same setup using configuration files when running in CPU/GPU or mixed modes**
- **Almost no differences in high-level code**, e.g. in implementation of time integration of NS eqs.
- **<u>Still – need support for two (CPU & GPU) low-level versions of the code</u>**, e.g. stencil operators
- MATLAB & Python suites for data post-processing and visualization

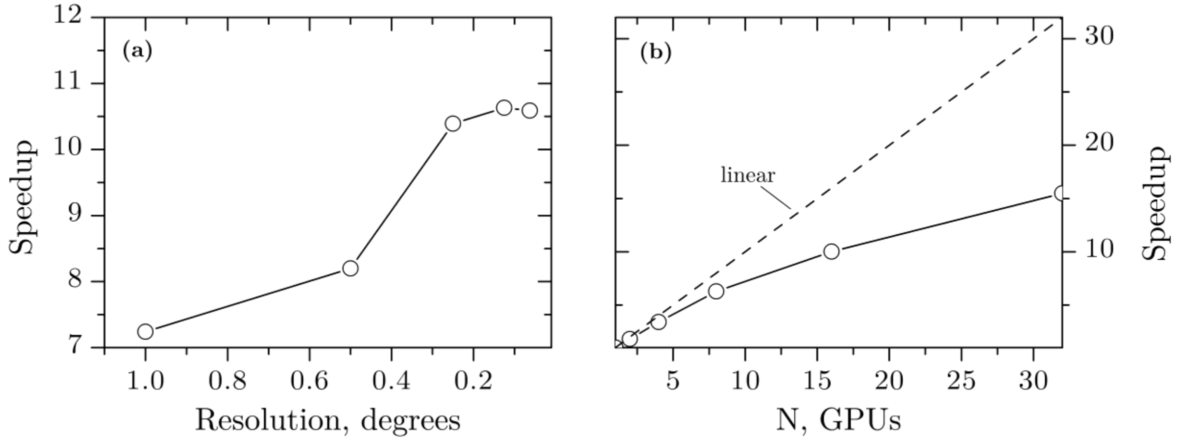


[Akhmed-Zaki et al., 2016]

DNS on CPU/GPU systems

- **Tracer/particle transport good speed-up (x10/x50) on GPUs compared with single CPU node**
- MPI data communications involve CPU-GPU memory transfers – **a scaling bottleneck**
- **Dynamics considerably slower** – multigrid method results in non-efficient GPU usage
- **Need large grids to gain good performance improvement on GPUs**
- Mix CPU/GPU mode – needs load balancing**, BUT – allows ensemble simulations using both node CPU & GPU**

GPU vs CPU speedup for scalar transport on sphere



CPU core, sec	CPU node/MPI, sec	CPU node/OpenMP (sec)	GPU Kepler (sec)	GPU Pascal (sec)	GPU Volta (sec)
164.1	28.6	27.3	35.9	13.1	8.4
648.2	108.4	110.2	83.16	28.1	13.7
-	548.8		-	141.4	67.25

DNS: starting from 400 000 grid nodes (and x8 in next rows)
CPU: Intel Xeon E5-2697 v3 2.60GHz

DNS with offload on GPU

- **Offload parts of computations on GPU**
  - MPI process runs the code on CPU **except** the offloaded modules on GPU
  - **Tracers & particles transport** are good candidates for offloading – more efficient (in terms of both performance and scaling) on GPUs compared with dynamics module

Lagrangian particles transport:

$$\frac{d\mathbf{x}_i(t)}{dt} = \mathbf{v}_i, m_i \frac{d\mathbf{v}_i(t)}{dt} = \mathbf{f}_B + \mathbf{f}_D + \ldots$$

Passive tracers transport:

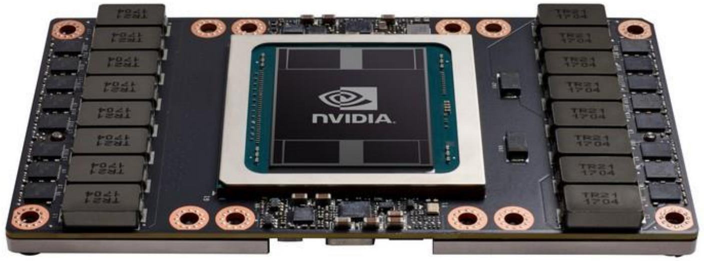
$$\frac{\partial C_k}{\partial t} + \frac{\partial u_j C_k}{\partial x_j} = \frac{1}{\mathrm{ScRe}} \frac{\partial^2 C_k}{\partial x_j \partial x_j} + F_k$$

Dynamics & data processing:

$$\frac{\partial u_i}{\partial t} = -\frac{\partial u_i u_j}{\partial x_j} - \frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j \partial x_j} + F_i^e$$

$$\frac{\partial u_i}{\partial x_i} = 0$$

$$\frac{\partial T}{\partial t} + \frac{\partial u_j T}{\partial x_j} = \frac{1}{\mathrm{PrRe}} \frac{\partial^2 T}{\partial x_j \partial x_j}$$



CPU & GPU memory transfer each time step – may be overlapped with computations

Conclusions

- **DNS/LES/RANS models and codes have to take into account modern-day HPC zoo**
  - Enable a lot of data processing at run-time
- **Optimizations tuned for specific architectures**
  - ARM/Intel/AMD CPUs require different code optimizations
  - Using CUDA shared-memory gives benefits not on all GPU architectures in use & more
- ***Unified codes*** – much less code to code☺
- **Adapting to novel GPU architectures**
  - Tensor cores, half precision …
  - Running on both Nvidia & AMD GPUs
  - ML-based algorithms
- **Using half precision** – expecting **x2 improvement** in memory transfer & computations
  - Is the precision enough for at least some of the model blocks?
  - Enabling mixed precision algorithms

Code available per request at RCC GitLab:
http://tesla.parallel.ru/

Thank you for your attention!

Email: evgeny.mortikov@gmail.com

pixels.com:
Whitney Knapp Bowditch

Conclusions

- **DNS/LES/RANS models and codes have to take into account modern-day HPC zoo**
  - Enable a lot of data processing at run-time
- **Optimizations tuned for specific architectures**
  - ARM/Intel/AMD CPUs require different code optimizations
  - Using CUDA shared-memory gives benefits not on all GPU architectures in use & more
- ***Unified codes*** – much less code to code☺
- **Adapting to novel GPU architectures**
  - Tensor cores, half precision …
  - Running on both Nvidia & AMD GPUs
  - ML-based algorithms
- **Using half precision** – expecting **x2 improvement** in memory transfer & computations
  - Is the precision enough for at least some of the model blocks?
  - Enabling mixed precision algorithms

Code available per request at RCC GitLab server: http://tesla.parallel.ru/