JARA|HPC

Direct Numerical Simulation of Fluid Turbulence at Extreme Scale with psOpen

Jens Henrik Göbbert[1,2], Michael Gauding[3], Cedrick Ansorge[4], Bernd Hentschel[1,2], Torsten Kuhlen[2,5], Heinz Pitsch[6]

[1] Jülich Aachen Research Alliance, JARA-HPC, Germany
[2] Virtual Reality Group, RWTH Aachen University, Germany
[3] Chair for Numerical Thermo-Fluid Dynamics, Technische University Freiberg, Germany
[4] Max Planck Institute for Meteorolgy, Hamburg, Germany
[5] Jülich Supercomputing Centre – Forschungszentrum Jülich GmbH, Germany
[6] Institut for Combustion Technology (ITV), RWTH Aachen University,, Germany

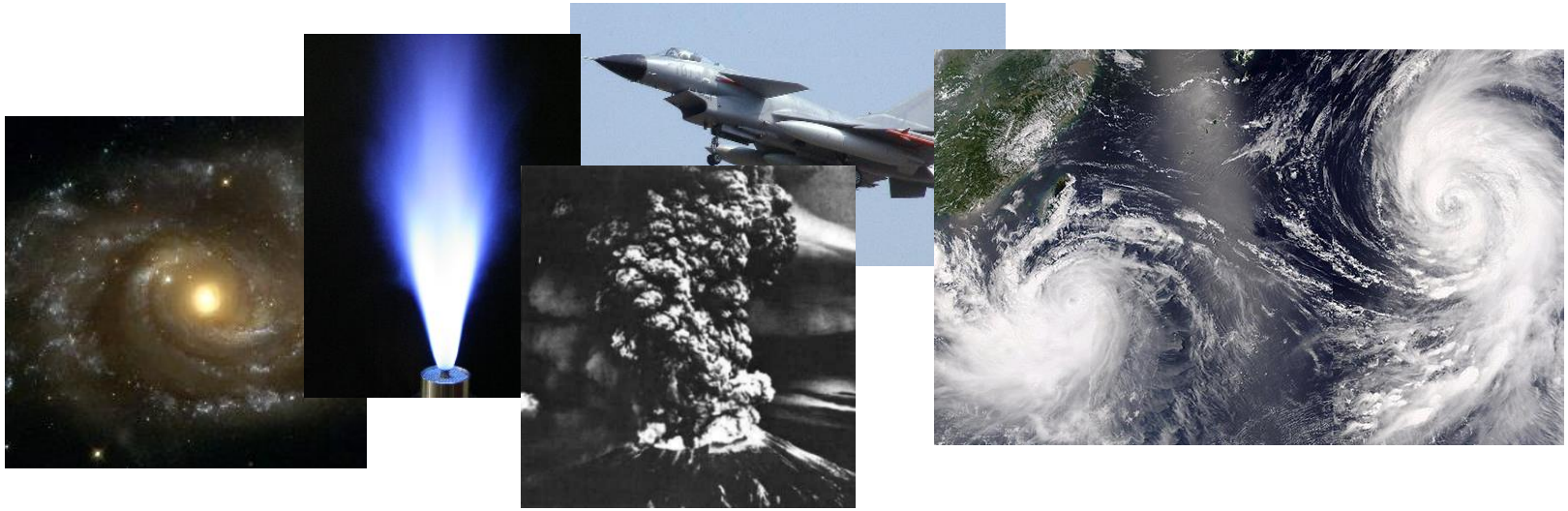
RWTH AACHEN UNIVERSITY JÜLICH FORSCHUNGSZENTRUM

Motivation

Turbulence

phenomenologically a fluid regime characterized by

chaotic and **stochastic** property changes



Motion of turbulent flows is

one of the unsolved problems of classical physics

Outline

- introduction to pseudo-spectral DNS
- 3D Fast Fourier Transformations
  - implicit filtering
  - overlapping communication and computation
- scaling
- conclusion

Direct Numerical Simulations (DNS)

- exact solution of Navier-Stokes equations $\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{\omega} \times \boldsymbol{u} = -\nabla \left(p + \frac{1}{2}\boldsymbol{u}^2 \right) + \nu \nabla^2 \boldsymbol{u}$
  - partial differential equation
  - with non-linear term $\nabla \cdot \boldsymbol{u} = 0\,,$

Computational challenges in DNS

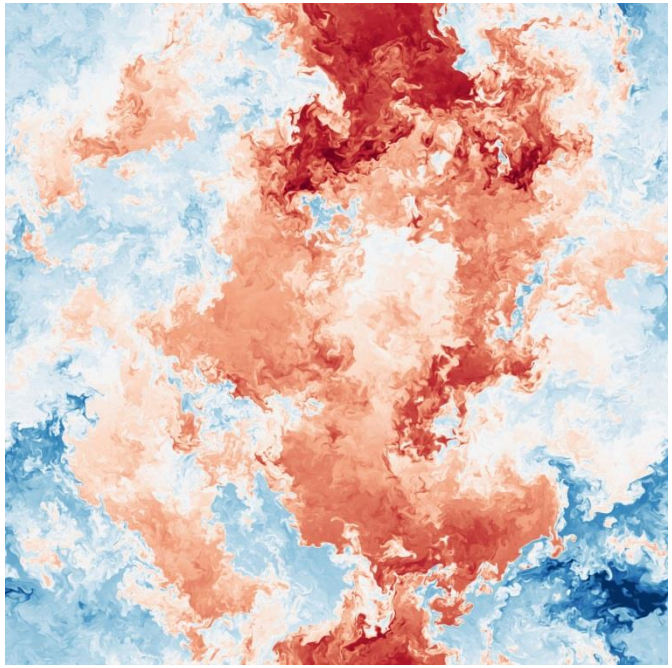
- smallest structures of the flow have to be resolved by the computational grid
- ratio of largest to smallest structures increases with the Reynolds number
- the total number of grid points is $\boldsymbol{N^3 \propto Re^{9/4}}$

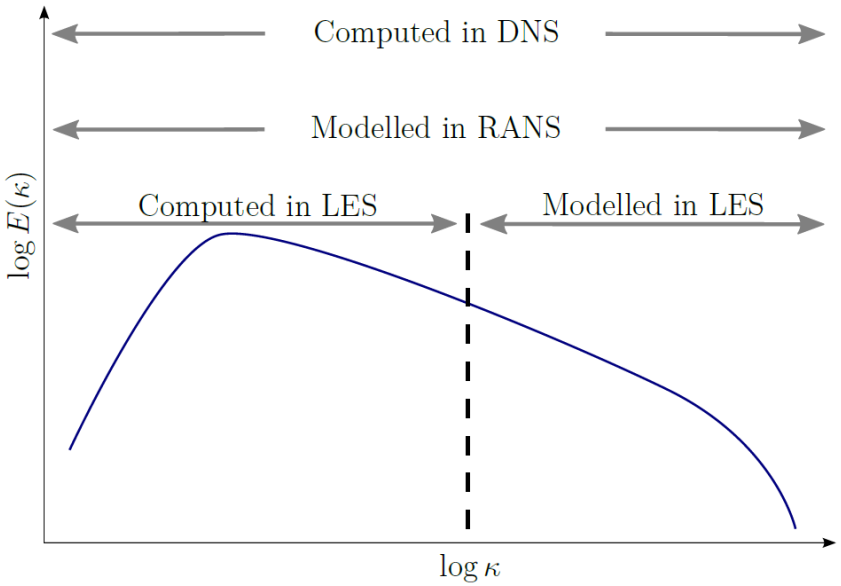
→ DNS of high Reynolds number flows are computational very expensive

Resolution Requirements

Turbulence Research

resolving smallest scales

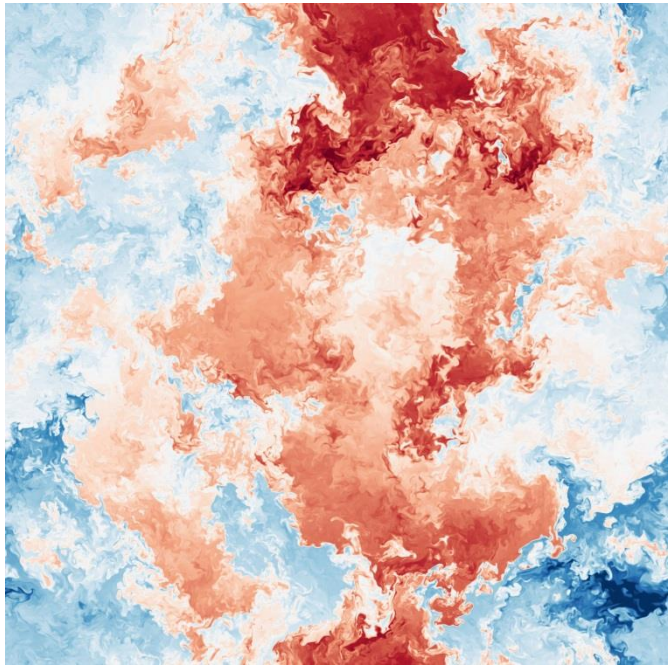


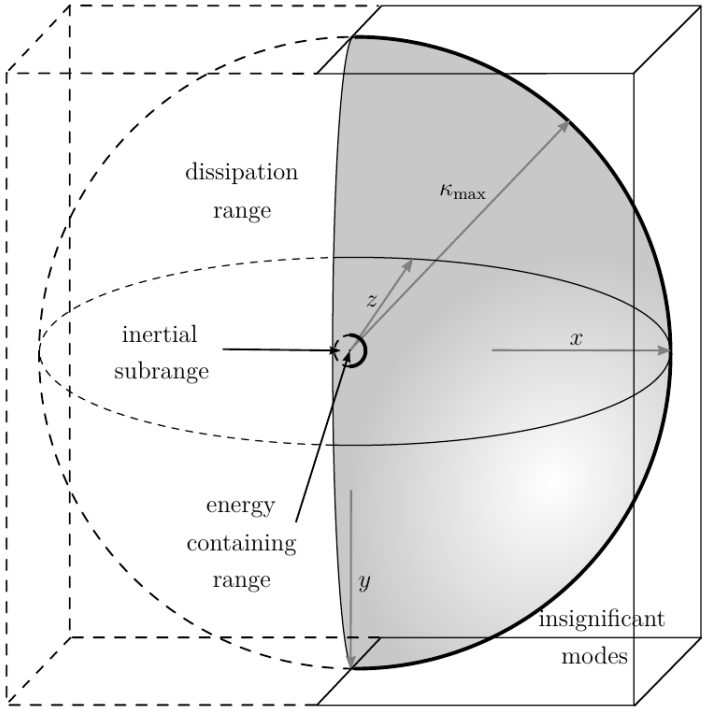


Resolution Requirements

Turbulence Research

resolving smallest scales





Pseudo-Spectral Method

Navier-Stokes equation is efficiently solved by pseudo-spectral method

- spatial highly accurate due to utilization of Fourier transform
- derivatives are point operations in spectral space

$$\mathcal{F}\left[\frac{\partial}{\partial x_i}\phi(\boldsymbol{x})\right] = i\kappa_i\hat{\phi}(\boldsymbol{\kappa}) \qquad \mathcal{F}\left[\frac{\partial^2}{\partial x_i^2}\phi(\boldsymbol{x})\right] = -\kappa_i^2\hat{\phi}(\boldsymbol{\kappa})$$

but

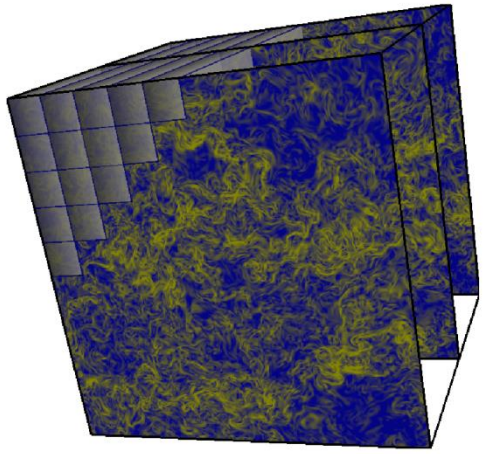
- multiplications within non-linear terms are computed in real space

→ 3d Fast Fourier Transformations (3d-FFT) required

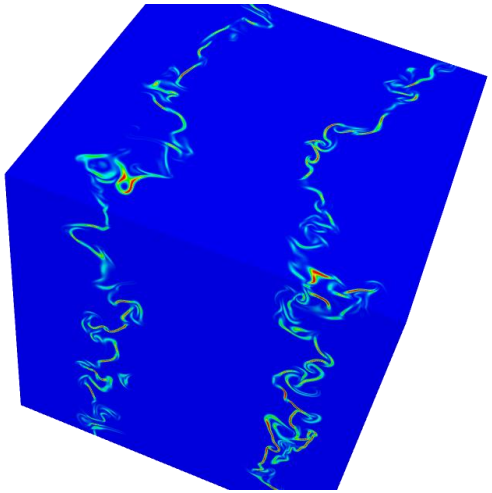
Pseudo-Spectral Application

psOpen – DNS by a pseudo-spectral approach

- isotropic homogeneous decaying/forced turbulence
- turbulent channel flow
- turbulent premixed flames
- scalar mixing



domain decompositioning

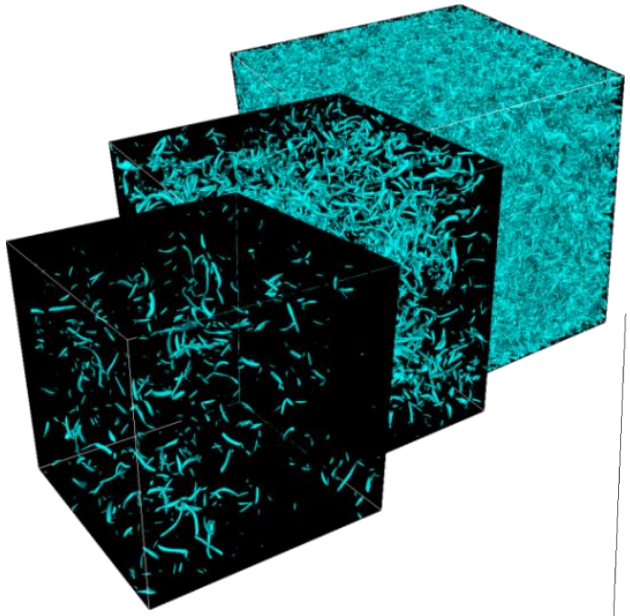


scalar mixing

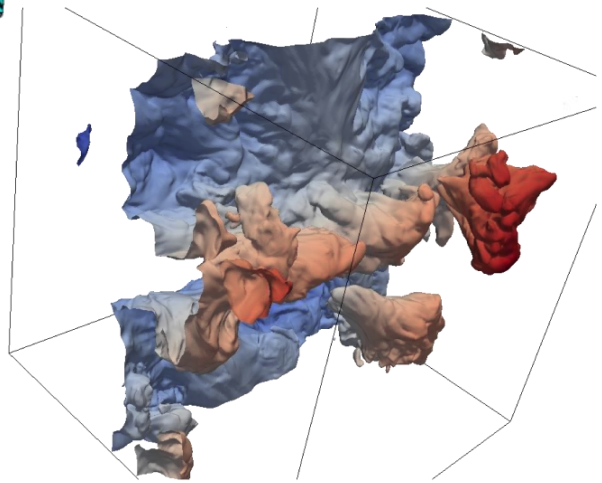
- Fortran 95
- MPI+OpenMP , single/double precision
- I/O using HDF5
- developed with the help of JSC 2007-2015

Pseudo-Spectral Application

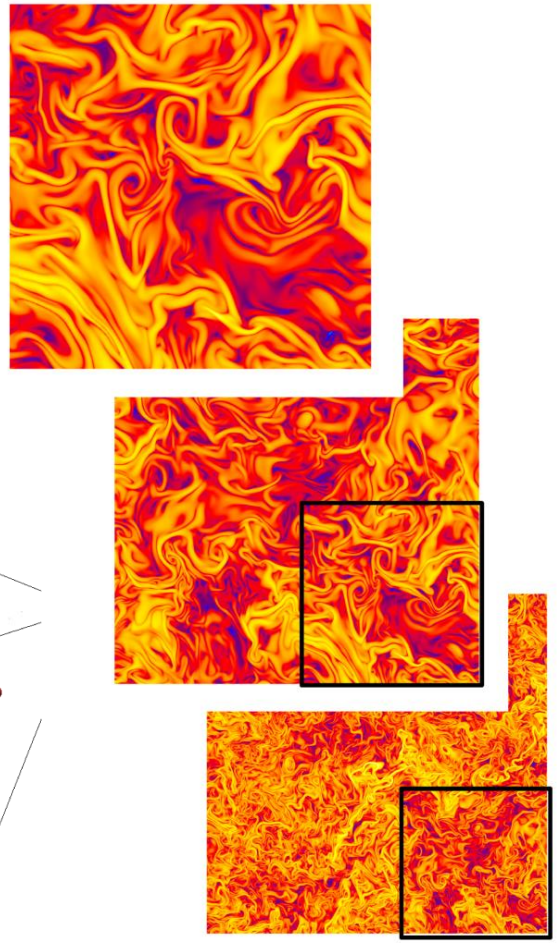
psOpen – DNS by a pseudo-spectral approach



vortex structure in
decaying turbulence



highly resolved structure of turbulent premixed flame

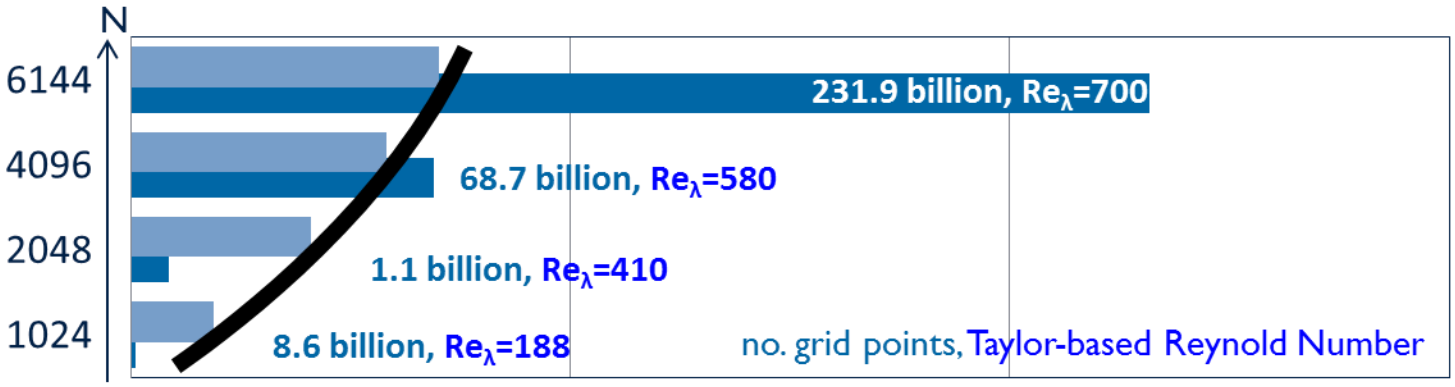


scalar dissipation

Data Base

current largest simulation – isotropic homogeneous forced turbulence

- $N^3 = 6144^3 = 231.9$ billion grid cells in physical space
- production runs on 8 192 - 16 384 nodes
- 262 144 – 524 288 MPI threads
- 32 MPI processes x 2 OpenMP threads per node
- input/output file up to 4084 GB



comparing setups of different computations on JUQUEEN

Data Base

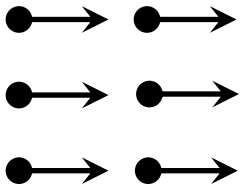
	A0	A1	A2	A3	A4	A5	A6
N^3	512^3	1024^3	1024^3	2048^3	2048^3	4096^3	4096^3
Re_λ	88	119	184	215	331	529	754
file size (GB)	8	64	64	512	512	4096	4096
M	180	60	60	10	10	10	10
data size (TB)	1.44	3.81	3.81	5	5	22	22

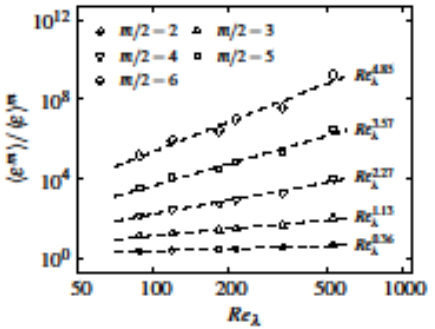
	B0	B1	B2	B3	B4	B5	B6
N^3	720^3	1440^3	1440^3	2816^3	2816^3	5632^3	6144^3
Re_λ	84	115	173	207	297	529	770
file size (GB)	22	177	177	1331	1331	5324	6912
file size compressed (GB)	6.6	52.6	52.6	393.2	393.2	1572.8	2041.9
M	40	20	20	10	10	5	5
data size (TB)	0.88	3.54	3.54	13.3	13.3	22.4	34.5
data size compressed (TB)	0.26	1.05	1.05	3.9	3.9	6.6	10.3

Turbulence Research

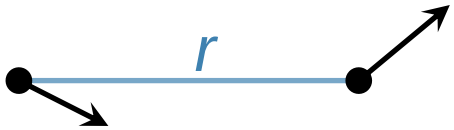
Data Analysis

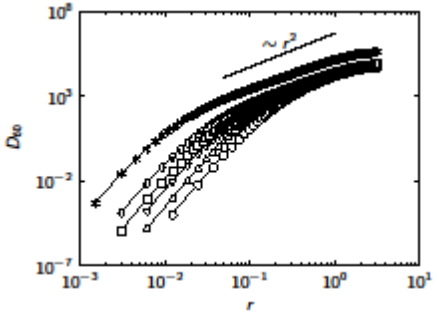
1-point statistics



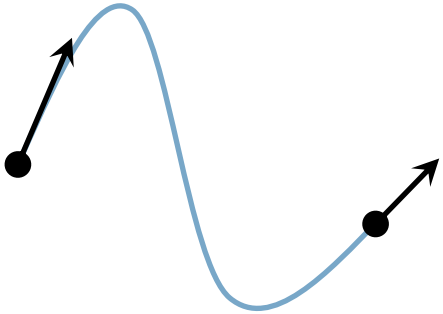


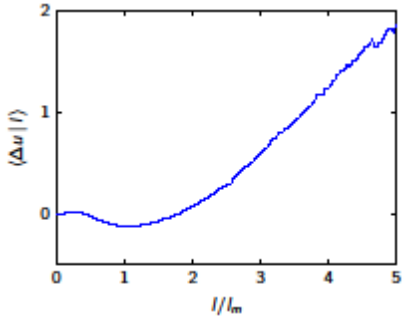
2-point statistics





statistics along path lines

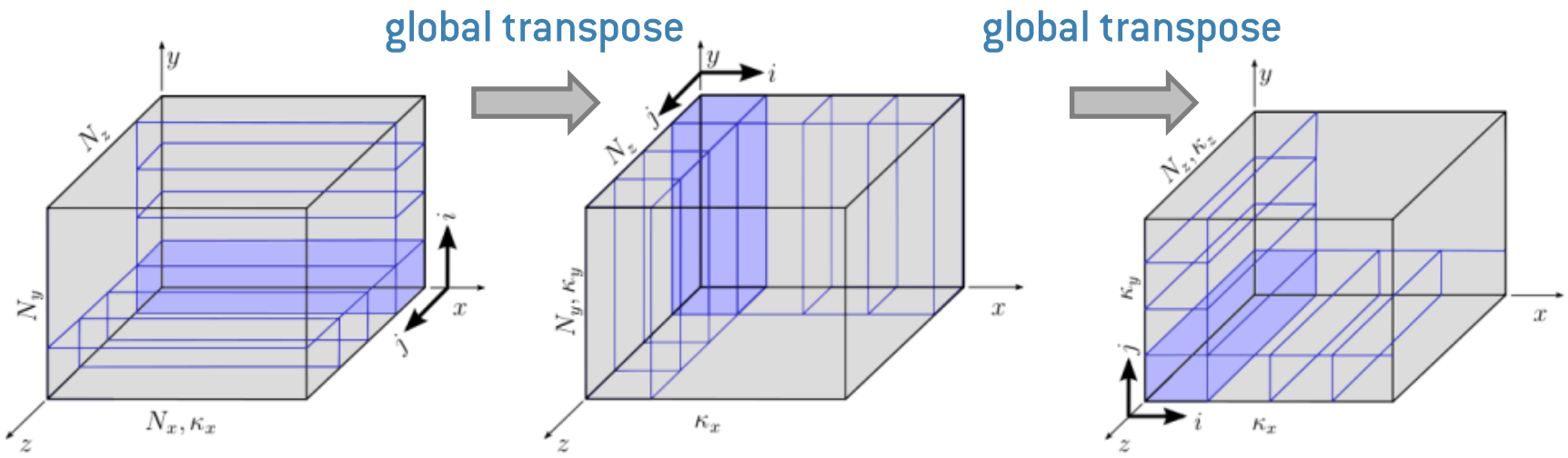




3D Fast Fourier Transformations

3D-FFTs on distributed memory systems

- distribute computational domain over all processors (1D, 2D)
- global transpose between computation of 1D-FFTs



1D-FFT in x

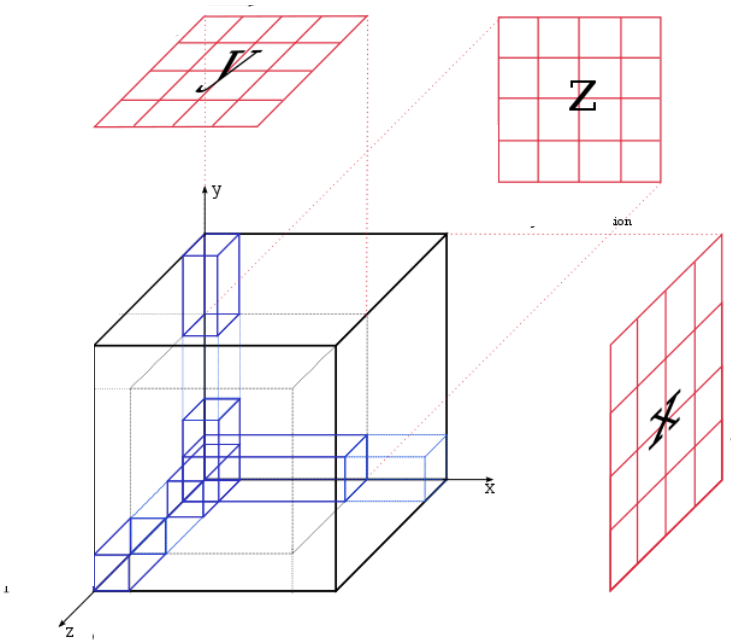
1D-FFT in y

1D-FFT in z

3D Fast Fourier Transformations

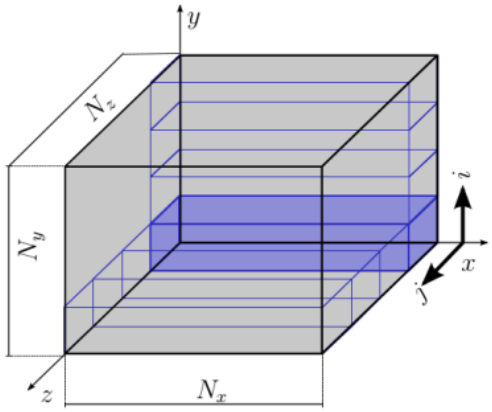
3D-FFTs on distributed memory systems

- 80% of the compute time required by 3d-FFTs
  - all-to-all communication
  - 2x data redistribution among processes

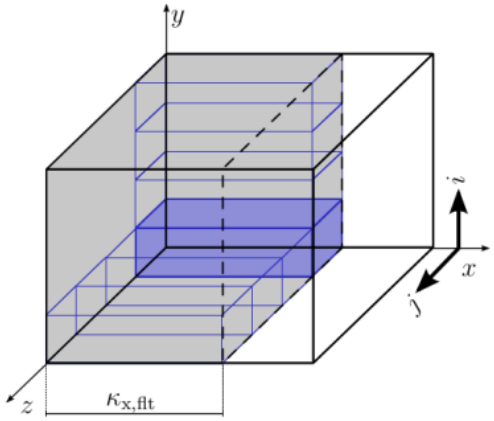


- development of an improved 3d-FFT library ‘nb3dfft’
  - data reduction including the filter process
  - overlapping communication and computation

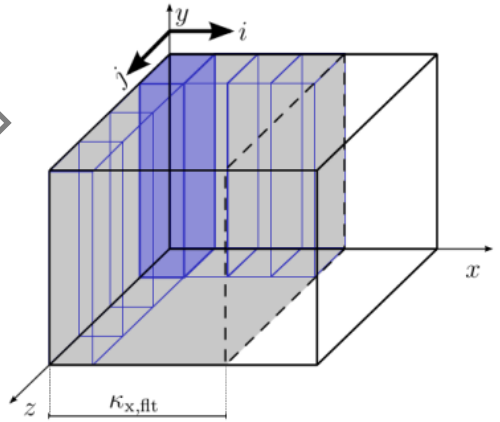
Implicit Filtering



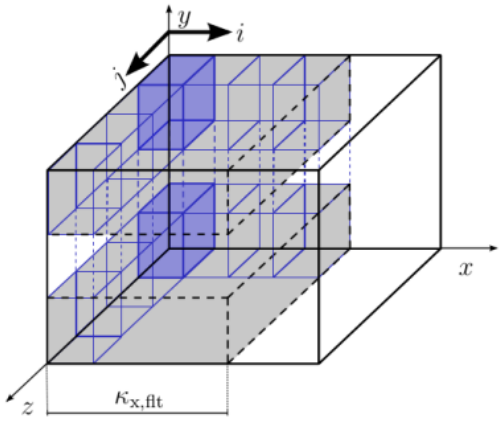
(a) 1d-FFTs applied in x direction



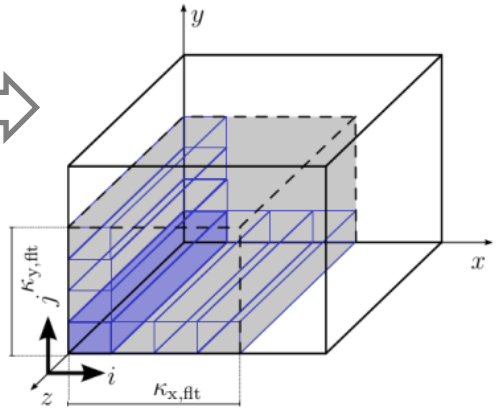
(b) cut-off filtering in x while packing for the transpose



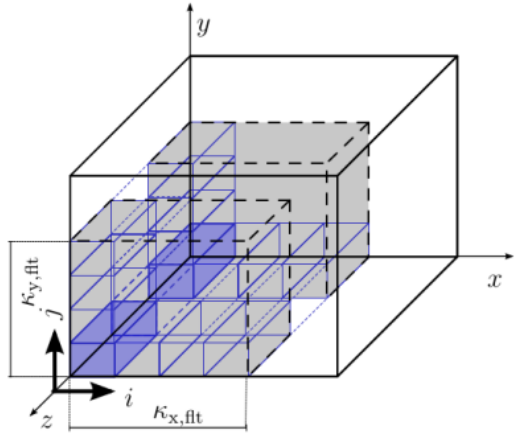
(c) unpacking and applying 1d-FFT in y direction



(d) cut-off filtering in y while packing for the transpose

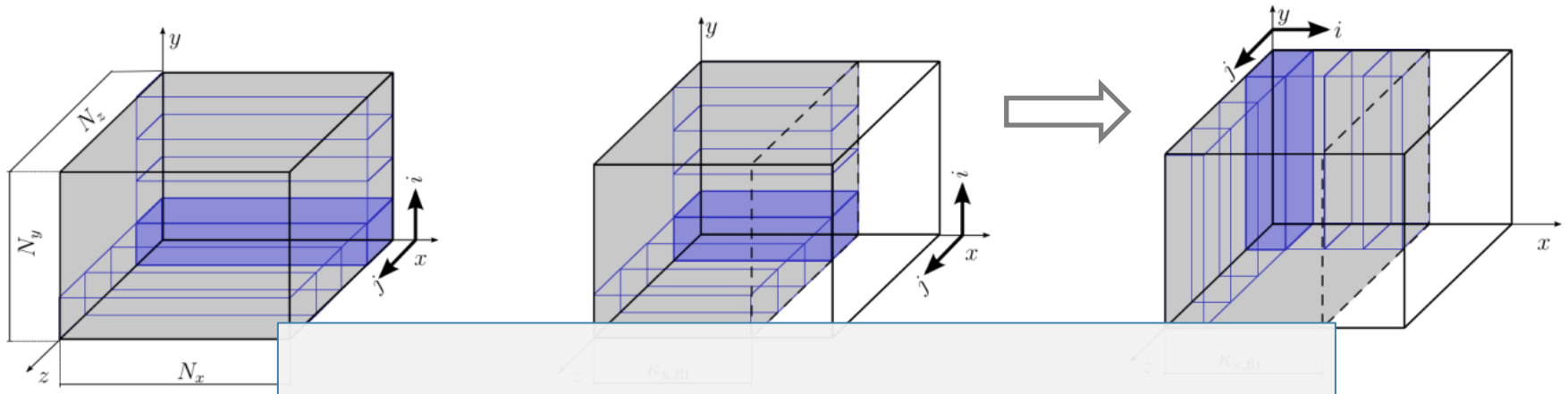


(e) unpacking and applying 1d-FFT in z direction



(f) cut-off filtering in z considering access patterns

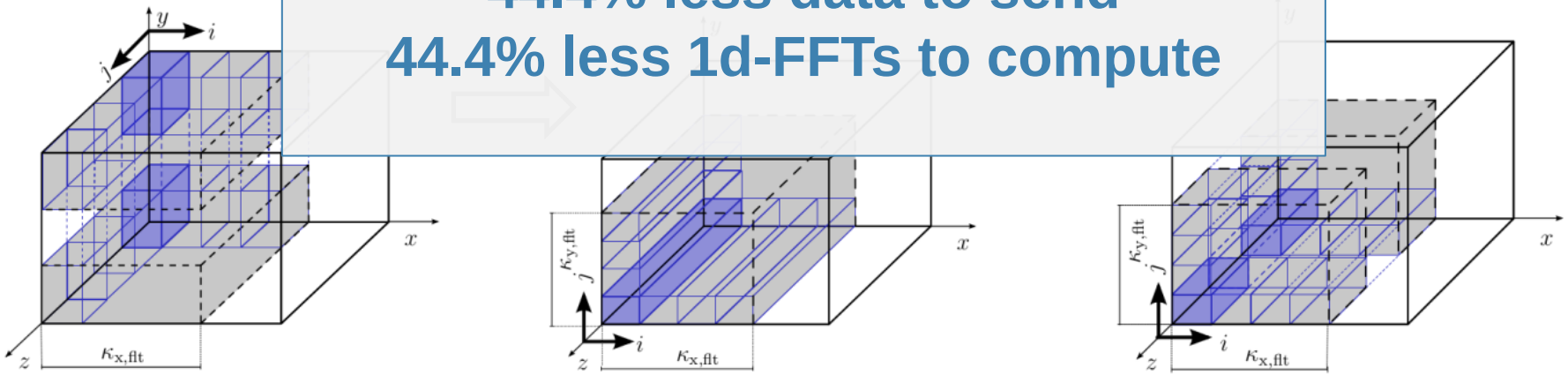
Implicit Filtering



(a) 1d-FFTs applie
in x direction

(c) unpacking and applying
1d-FFT in y direction

70.4% data reduction
44.4% less data to send
44.4% less 1d-FFTs to compute



(d) cut-off filtering in y
while packing for the transpose

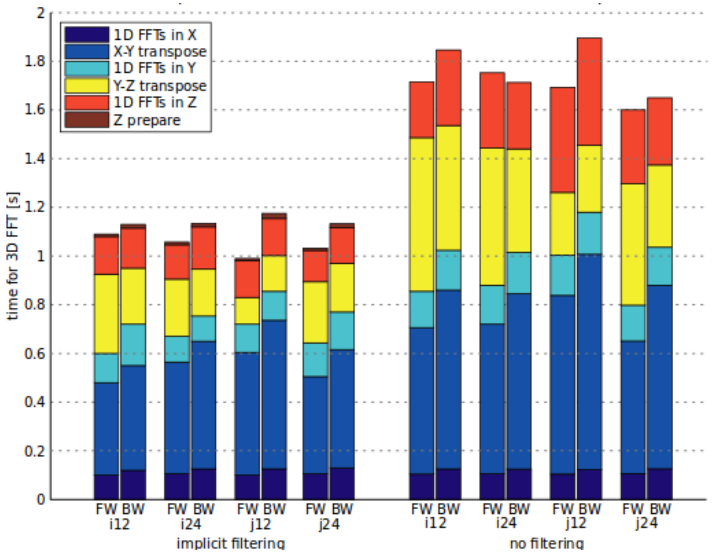
(e) unpacking and applying
1d-FFT in z direction

(f) cut-off filtering in z
considering access patterns

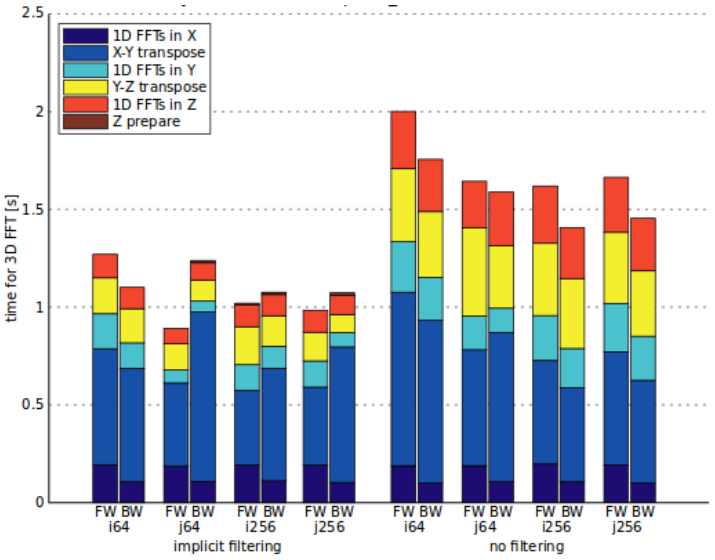
Implicit Filtering

Performance Measurements

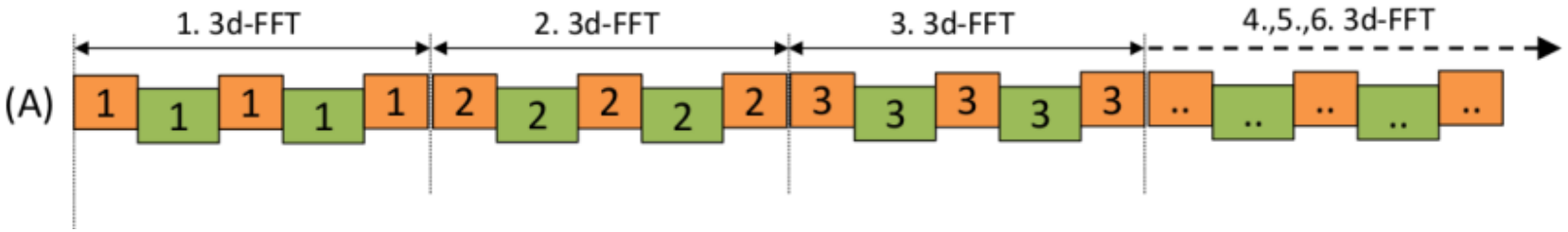
- RWTH Compute Cluster
  - 672 MPI processes on 56 nodes
  - domain size 2048^3



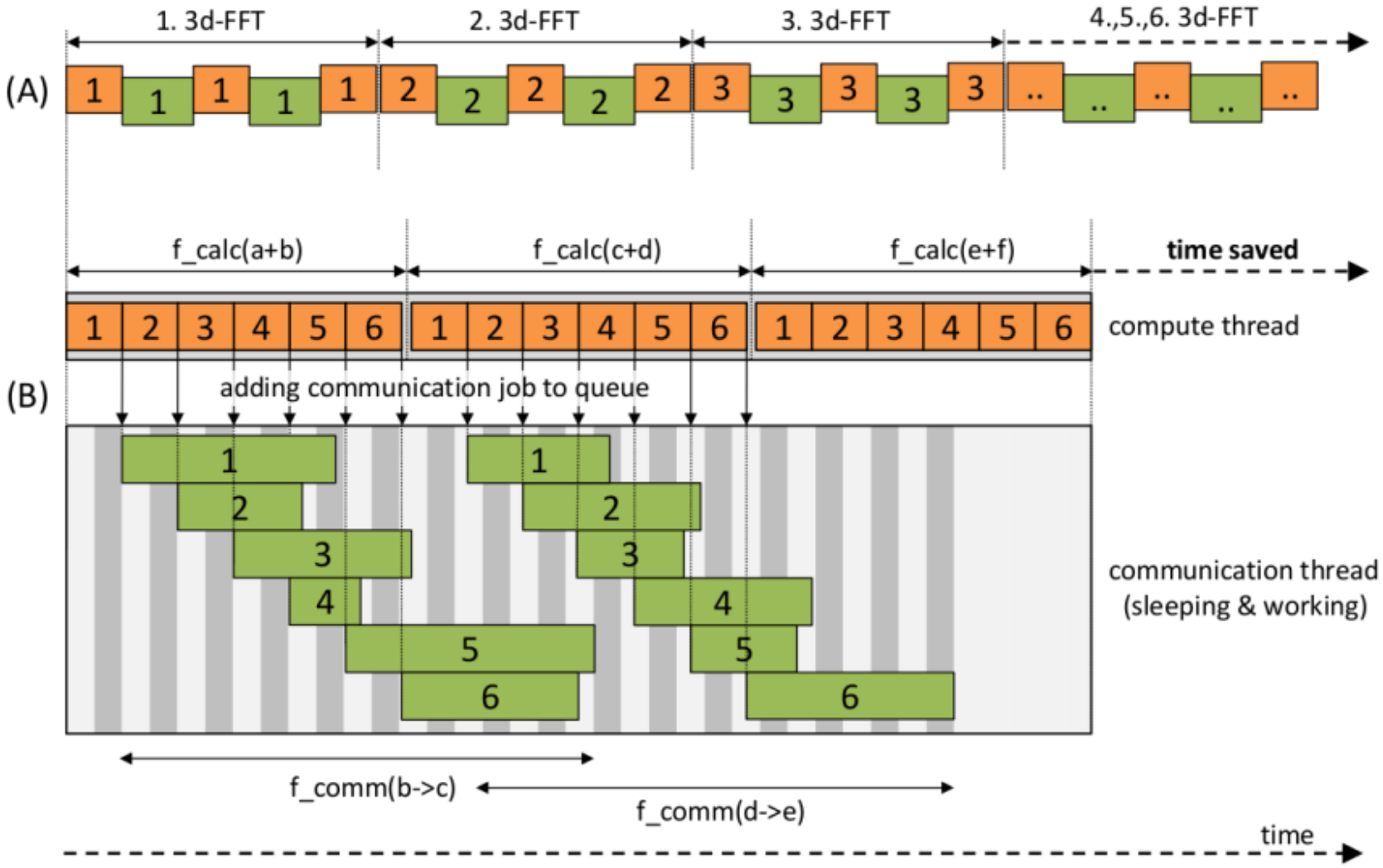
- JUQUEEN
  - 16.384 MPI processes on 8192 cores
  - domain size 4096^3



Overlapping Communication & Computation



Overlapping Communication & Computation



Overlapping Communication & Computation

procedure ITERATE VELOCITY $(n \to n+1)$
- **input:** $\hat{\boldsymbol{u}}^n_{1|2|3},\ \widehat{\boldsymbol{G}}^{n-1}_{4|5|6},\ \hat{f}^{n-1}_{\boldsymbol{u}}$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ compute vorticity
  - $\hat{\omega}^n_{x,10} \leftarrow \kappa_z \hat{u}^n_{y,2} - \kappa_y \hat{u}^n_{z,3}$
  - $\hat{\omega}^n_{y,11} \leftarrow \kappa_x \hat{u}^n_{z,3} - \kappa_z \hat{u}^n_{x,1}$
  - $\hat{\omega}^n_{z,12} \leftarrow \kappa_y \hat{u}^n_{x,1} - \kappa_x \hat{u}^n_{y,2}$
- $\boldsymbol{\omega}^n_{10|11|12} \xleftarrow{\mathcal{F}} \hat{\boldsymbol{\omega}}^n_{10|11|12}$

▷ 3x in-place backward 3d-FFT
- $\boldsymbol{u}^n_{7|8|9} \xleftarrow{\mathcal{F}} \hat{\boldsymbol{u}}^n_{1|2|3}$

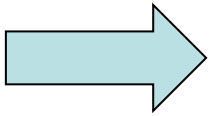
▷ 3x out-of-place backward 3d-FFT
- **for all** grid points $(\boldsymbol{N})$ **do:** ▷ compute $\boldsymbol{G}^n = \boldsymbol{\omega}^n \times \boldsymbol{u}^n$
  - $G^n_{x,13} \leftarrow u^n_{y,8} \cdot \omega^n_{z,12} - u^n_{z,9} \cdot \omega^n_{y,11}$
  - $G^n_{y,14} \leftarrow u^n_{z,9} \cdot \omega^n_{x,10} - u^n_{x,7} \cdot \omega^n_{z,12}$
  - $G^n_{z,15} \leftarrow u^n_{x,7} \cdot \omega^n_{y,11} - u^n_{y,8} \cdot \omega^n_{x,10}$
- $\widehat{\boldsymbol{G}}^n_{13|14|15} \xleftarrow{\mathcal{F}} \boldsymbol{G}^n_{13|14|15}$

▷ 3x in-place forward 3d-FFT
- $\widehat{\boldsymbol{G}}^n_{13|14|15} \xleftarrow{filter} \widehat{\boldsymbol{G}}^n_{13|14|15}$ ▷ dealiasing
- $\widehat{\boldsymbol{G}}^n_{13|14|15} \overset{zero\ copy}{\rightleftharpoons} \widehat{\boldsymbol{G}}^{n-1}_{4|5|6}$ ▷ swap memory
- $\hat{u}^n_{16} \overset{zero\ copy}{\rightleftharpoons} \hat{u}^n_2$ ▷ swap memory
- $\hat{f}^n_{\boldsymbol{u}} \leftarrow \mathrm{F}(\hat{f}^{n-1}_{\boldsymbol{u}})$ ▷ compute forcing energy
- **for** wave numbers $(\kappa_{ijk} \mid 0 \le i,j,k \le 2)$ **do:** ▷ add forcing energy
  - $\hat{\boldsymbol{u}}^{\text{force}}_{1|2|3} \leftarrow \mathrm{F}_{\text{force}}(\hat{\boldsymbol{u}}^n_{1|16|3},\ \hat{f}^n_{\boldsymbol{u}})$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ advance velocity
  - $\hat{\boldsymbol{u}}^{\text{step1}}_{1|2|3} \leftarrow \mathrm{F}_{\text{step1}}(\hat{\boldsymbol{u}}^{\text{force}}_{1|2|3},\ \widehat{\boldsymbol{G}}^n_{4|5|6},\ \widehat{\boldsymbol{G}}^{n-1}_{13|14|15})$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ apply projection tensor
  - $\hat{\boldsymbol{u}}^{n+1}_{1|2|3} \leftarrow \mathrm{F}_{\text{step2}}(\hat{\boldsymbol{u}}^{\text{step1}}_{1|2|3})$
- $\langle \hat{\boldsymbol{u}}^{n+1}_{1|2|3} \rangle = 0$ ▷ set mean velocity to zero
- **output:** $\hat{\boldsymbol{u}}^{n+1}_{1|2|3},\ \widehat{\boldsymbol{G}}^n_{4|5|6},\ \boldsymbol{u}^n_{7|8|9},\ \hat{u}^n_{16}$

procedure ITERATE PASSIVE SCALAR $(n \to n+1)$
- **input:** $\hat{\phi}^n_{17},\ \hat{J}^{n-1}_{18},\ \boldsymbol{u}^n_{7|8|9},\ \hat{u}^n_{16}$
- $\widehat{\boldsymbol{\nabla}\boldsymbol{\phi}}^n_{10|11|12} \leftarrow \hat{\phi}^n_{17}$ ▷ 3x compute derivatives
- $\nabla\boldsymbol{\phi}^n_{10|11|12} \xleftarrow{\mathcal{F}} \widehat{\boldsymbol{\nabla}\boldsymbol{\phi}}^n_{10|11|12}$

▷ 3x in-place backward 3d-FFT
- $J^n_{10} \leftarrow \boldsymbol{u}^n_{7|8|9} \cdot \nabla\boldsymbol{\phi}^n_{10|11|12}$ ▷ compute convective term
- $\hat{J}^n_{10} \xleftarrow{\mathcal{F}} J^n_{10}$

▷ 1x in-place forward 3d-FFT
- $\hat{J}^n_{10} \xleftarrow{filter} \hat{J}^n_{10}$ ▷ dealiasing
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ advance passive scalar
  - $\hat{\phi}^{n+1}_{17} \leftarrow \mathrm{F}_{\text{adv}}(\hat{\phi}^n_{17},\ \hat{J}^{n-1}_{18},\ \hat{J}^n_{10},\ \hat{u}^n_{16})$
- $\hat{J}^n_{10} \overset{zero\ copy}{\rightleftharpoons} \hat{J}^{n-1}_{18}$ ▷ swap memory
- **output:** $\hat{\phi}^{n+1}_{17},\ \hat{J}^n_{18}$



procedure ITERATE VELOCITY $(n \to [n+1])$, PASSIVE SCALAR $([n-1] \to n)$
- **input (velocity):** $\hat{\boldsymbol{u}}^n_{1|2|3},\ \widehat{\boldsymbol{G}}^{n-1}_{4|5|6},\ \hat{f}^{n-1}_{\boldsymbol{u}}$
- **input (passive scalar):** $\hat{\phi}^{n-1}_{17},\ J^{n-1}_{19},\ \hat{J}^{n-2}_{18},\ \hat{u}^{n-1}_{16}$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ compute vorticity
  - $\hat{\omega}^n_{x,10} \leftarrow \kappa_z \hat{u}^n_{y,2} - \kappa_y \hat{u}^n_{z,3}$
  - $\hat{\omega}^n_{y,11} \leftarrow \kappa_x \hat{u}^n_{z,3} - \kappa_z \hat{u}^n_{x,1}$
  - $\hat{\omega}^n_{z,12} \leftarrow \kappa_y \hat{u}^n_{x,1} - \kappa_x \hat{u}^n_{y,2}$
- $\boldsymbol{\omega}^n_{10|11|12} \xleftarrow{\mathcal{F}+3/2filter} \hat{\boldsymbol{\omega}}^n_{10|11|12}$ ▷ 3x in-place backward 3d-FFT
- $\boldsymbol{u}^n_{7|8|9} \xleftarrow{\mathcal{F}+3/2filter} \hat{\boldsymbol{u}}^n_{1|2|3}$ ▷ 3x out-of-place backward 3d-FFT
- $\hat{J}^{n-1}_{19} \xleftarrow{\mathcal{F}+3/2filter} J^{n-1}_{19}$ ▷ 1x in-place forward 3d-FFT
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ advance passive scalar
  - $\hat{\phi}^n_{17} \leftarrow \mathrm{F}_{\text{adv}}(\hat{\phi}^{n-1}_{17},\ \hat{J}^{n-1}_{19},\ \hat{J}^{n-2}_{18},\ \hat{u}^{n-1}_{16})$
- $\hat{J}^{n-1}_{19} \overset{zero\ copy}{\rightleftharpoons} \hat{J}^{n-2}_{18}$ ▷ swap memory
- **for all** grid points $(\boldsymbol{N})$ **do:** ▷ compute $\boldsymbol{G}^n = \boldsymbol{\omega}^n \times \boldsymbol{u}^n$
  - $G^n_{x,13} \leftarrow u^n_{y,8} \cdot \omega^n_{z,12} - u^n_{z,9} \cdot \omega^n_{y,11}$
  - $G^n_{y,14} \leftarrow u^n_{z,9} \cdot \omega^n_{x,10} - u^n_{x,7} \cdot \omega^n_{z,12}$
  - $G^n_{z,15} \leftarrow u^n_{x,7} \cdot \omega^n_{y,11} - u^n_{y,8} \cdot \omega^n_{x,10}$
- $\widehat{\boldsymbol{\nabla}\boldsymbol{\phi}}^n_{10|11|12} \leftarrow \hat{\phi}^n_{17}$ ▷ 3x compute derivatives
- $\widehat{\boldsymbol{G}}^n_{13|14|15} \xleftarrow{\mathcal{F}+3/2filter} \boldsymbol{G}^n_{13|14|15}$ ▷ 3x in-place forward 3d-FFT
- $\nabla\boldsymbol{\phi}^n_{20|21|22} \xleftarrow{\mathcal{F}+3/2filter} \widehat{\boldsymbol{\nabla}\boldsymbol{\phi}}^n_{20|21|22}$ ▷ 3x in-place backward 3d-FFT
- $J^n_{19} \leftarrow \boldsymbol{u}^n_{7|8|9} \cdot \nabla\boldsymbol{\phi}^n_{20|21|22}$ ▷ compute convective term
- $\widehat{\boldsymbol{G}}^n_{13|14|15} \overset{zero\ copy}{\rightleftharpoons} \widehat{\boldsymbol{G}}^{n-1}_{4|5|6}$ ▷ swap memory
- $\hat{u}^n_{16} \overset{zero\ copy}{\rightleftharpoons} \hat{u}^n_2$ ▷ swap memory
- $\hat{f}^n_{\boldsymbol{u}} \leftarrow \mathrm{F}(\hat{f}^{n-1}_{\boldsymbol{u}})$ ▷ compute forcing energy
- **for** wave numbers $(\kappa_{ijk} \mid 0 \le i,j,k \le 2)$ **do:** ▷ add forcing energy
  - $\hat{\boldsymbol{u}}^{\text{force}}_{1|2|3} \leftarrow \mathrm{F}_{\text{force}}(\hat{\boldsymbol{u}}^n_{1|16|3},\ \hat{f}^n_{\boldsymbol{u}})$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ advance velocity
  - $\hat{\boldsymbol{u}}^{\text{step1}}_{1|2|3} \leftarrow \mathrm{F}_{\text{step1}}(\hat{\boldsymbol{u}}^{\text{force}}_{1|2|3},\ \widehat{\boldsymbol{G}}^n_{4|5|6},\ \widehat{\boldsymbol{G}}^{n-1}_{13|14|15})$
- **for all** wave numbers $(\boldsymbol{\kappa})$ **do:** ▷ apply projection tensor
  - $\hat{\boldsymbol{u}}^{n+1}_{1|2|3} \leftarrow \mathrm{F}_{\text{step2}}(\hat{\boldsymbol{u}}^{\text{step1}}_{1|2|3})$

Overlapping Communication & Computation

- message sizes stay large
- number of messages do not change
- 2 synchronization points instead of 2x13

Performance increase by overlapping comm & comp

	1024^3	2048^3	4096^3	6144^3
1024 nodes	15.50%	21.60%	-	-
2048 nodes	-	26.63%	29.63%	-
4096 nodes	-	17.70%	19.78%	-
8192 nodes	-	11.64%	22.69%	17.53%

Scaling

Results from Extreme Scaling Workshop 2015

Memory requirements:

- 20 datasets

float	6144^3	8192^3	12288^3
4	34.6 TB	81.9 TB	276.5 TB ?
8	69.1 TB	163.8 TB	---

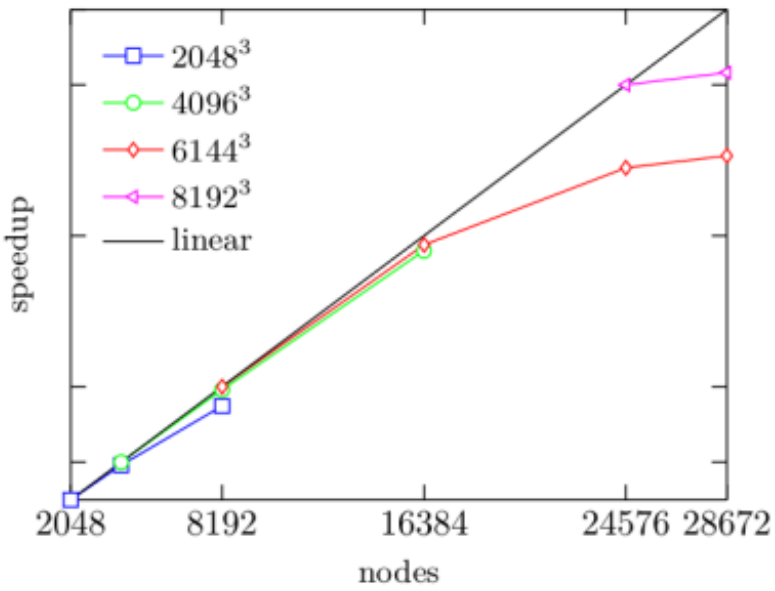


Table 1.2: Strong scaling tests of psOpen for various grid sizes.

bg_size	rpn	ranks	tpp	threads	time per iteration in seconds 2048^3	4096^3	6144^3	8192^3
2048	32	65536	2	131072	0.6295	3.0785	-	-
4096	32	131072	2	262144	0.3281	1.8815	-	-
8192	32	262144	2	524288	0.1805	0.9606	2.3276	-
16384	32	524288	2	1835008	-	0.4950	1.2132	-
24576	32	786432	2	1572864	-	-	0.9530	3.7811
28672	32	917504	2	1835008	-	-	0.9278	3.6801

Conclusion

- making pseudo-spectral codes for DNS scale on large systems is challenging
  - 3d-FFTs require lots of MPI_alltoall() calls, which cannot be avoided
- for best performance key functions have to rewritten

- 3d-FFT can be optimized, if it is not thought as a single function call, but as part of the algorithm

  - Moving the required filter operation of the main algorithm into the 3d-FFT reduces the data to be send by 44.4% and the 1d-FFTs by 44.4%

  - Calling multiple 3d-FFT for different fields at the same time allows overlapping of communication and computation -> ~20% performance gain

Thank you for your attention.