In-situ analysis and visualization of massively parallel computations of transitional and turbulent flows

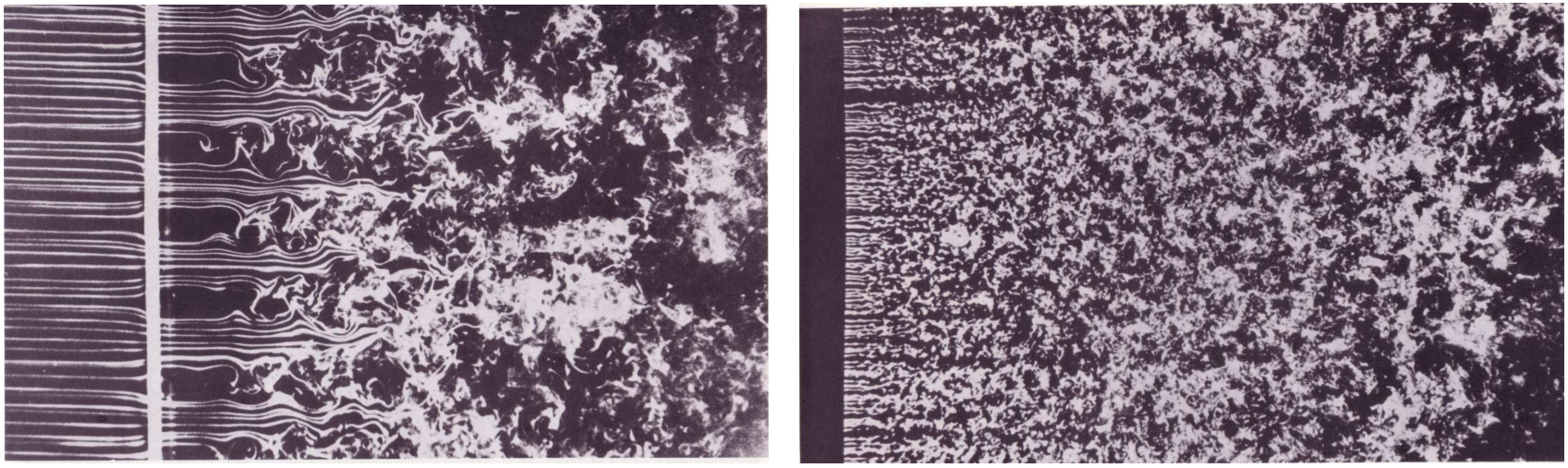
Anne Cadiou, Marc Buffat, Christophe Pera*
Bastien Di Pierro, Frédéric Alizard, Lionel Le Penven

Laboratoire de Mécanique des Fluides et d'Acoustique
*Département de Mécanique de L'Université Lyon 1
CNRS, Université Lyon I, École Centrale de Lyon, INSA de Lyon

Workshop TDMF, Orsay
Thursday 30th november 2017

Outline

Turbulent flows



Generation of turbulence behind a grid, T. Corke and H. Nagib in M. Van Dyke, 1982

Fluctuations over a wide range of non-linearly interacting scales

$\Downarrow$

Understanding the physics of turbulence
has very early involved direct numerical simulations

Direct Numerical Simulations (DNS)

$\Rightarrow$ Resolve all length and time scales

Navier-Stokes equations

Conservation of mass and momentum

$$\partial_t \boldsymbol{U} + \boldsymbol{U}.\nabla \boldsymbol{U} = -1/\rho \ \nabla p + \nu \Delta \boldsymbol{U}$$
$$\nabla . \boldsymbol{U} = 0$$

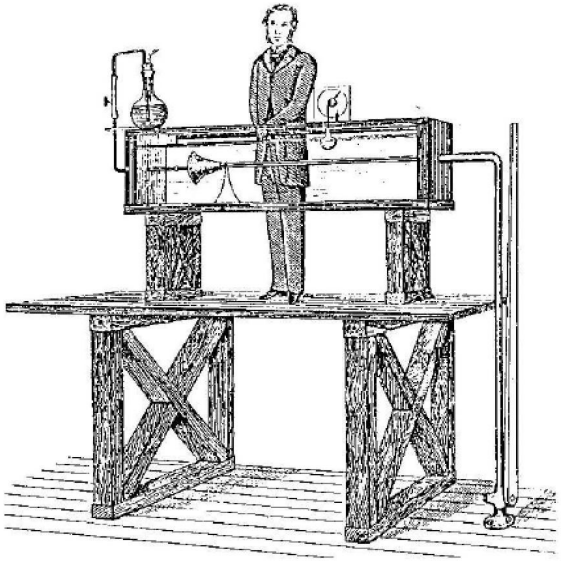
(velocity $\boldsymbol{U}$, pressure p, density ρ, viscosity ν)

$+$ initial and boundary conditions

$\Downarrow$

- Gives access to detailed physical quantities (beyond experiments)
- Computationally intensive

How do flows become turbulent?



The general results were as follows :—

(1) When the velocities were sufficiently low, the streak of colour extended in a beautiful straight line through the tube, Fig. 3.

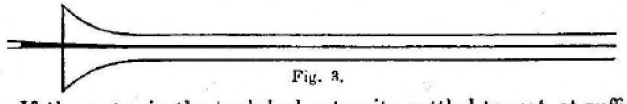


Fig. 3.

(2) If the water in the tank had not quite settled to rest, at sufficiently low velocities, the streak would shift about the tube, but there was no appearance of sinuosity.

(3) As the velocity was increased by small stages, at some point in the tube, always at a considerable distance from the trumpet or intake, the

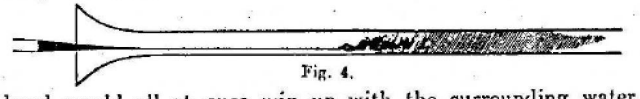


Fig. 4.

colour band would all at once mix up with the surrounding water, and fill the rest of the tube with a mass of coloured water, as in Fig. 4.

Any increase in the velocity caused the point of break down to approach the trumpet, but with no velocities that were tried did it reach this.

On viewing the tube by the light of an electric spark, the mass of colour resolved itself into a mass of more or less distinct curls, showing eddies, as in Fig. 5.

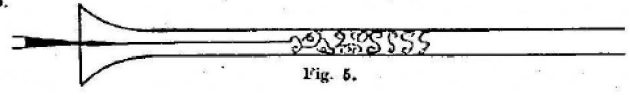


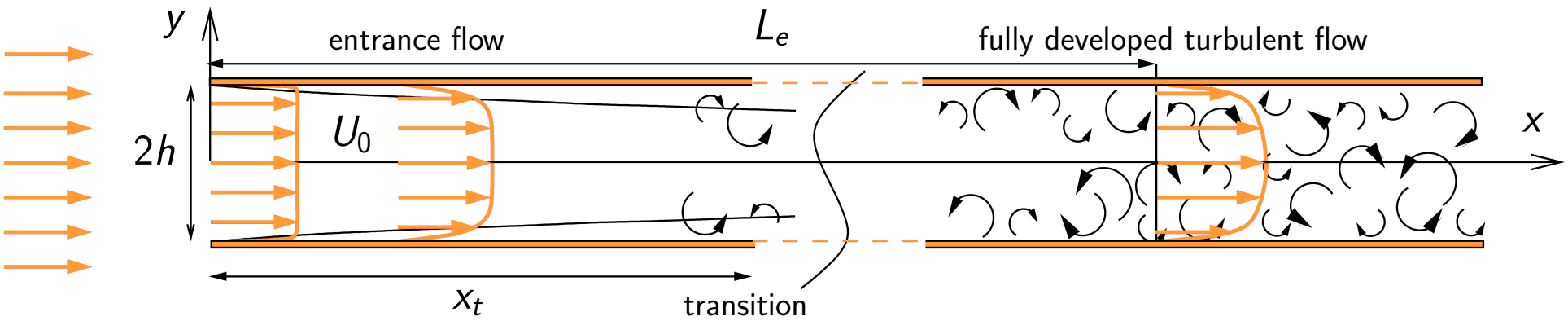
Fig. 5.

O. Reynolds' pipe flow experiment (1883)

Observation of the laminar, transitional and turbulent flow regimes

DNS of transitional and turbulent flows

Viscous laminar boundary layers behave as a selective disturbance amplifier

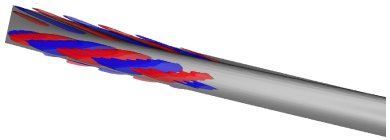


- selection of **primary instabilities**
  (such as Tollmien-Schlichting waves or Klebanov modes)
  are well predicted by **linear stability theory**
- transition result from **secondary instabilities**
  even if all primary modes are asymptotically stable (streaks)

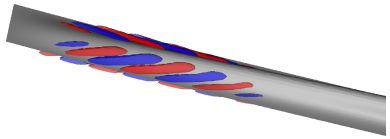
In spatially evolving flows, primary modes interact **nonlinearly** with the base flow and **breakdown** is always a nonlinear processus

$\Rightarrow$ linear theory no longer applies

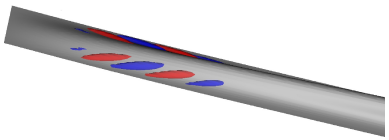
Global optimization and space-time nonlinear dynamic



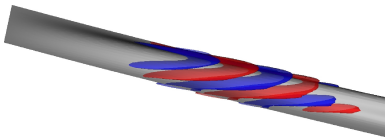
(a) $t = 0.$ ω_x.



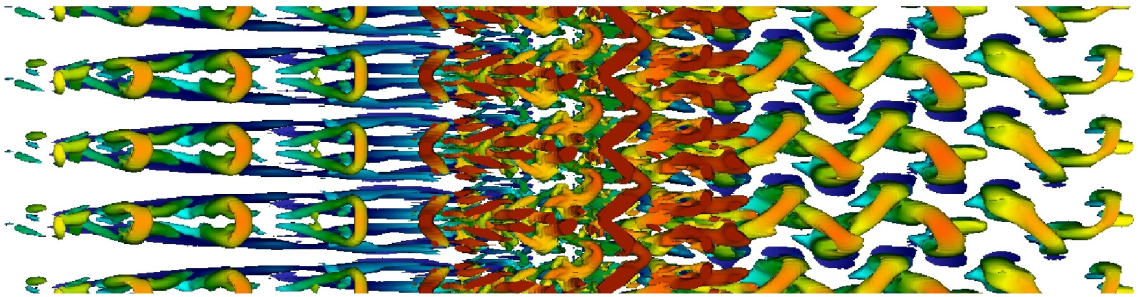
(b) $t = 1.$ ω_x.



(c) $t = 1.$ u' ↗



(d) $t = 3.$ u' ↗

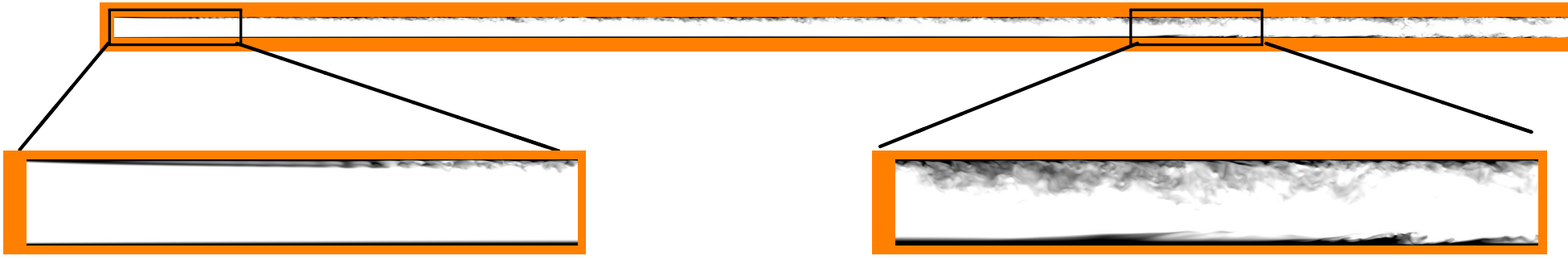


A. Cadiou et al., *Linear and nonlinear space-time dynamics of optimal wavepackets for streaks in a channel entrance flow* ,
EUROMECH Colloquium 591, Bari, September 2017

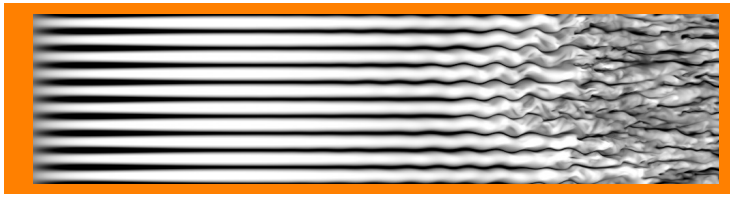
Boundary layers interaction and receptivity

Top view of the lower wall :

Side view of the channel flow :



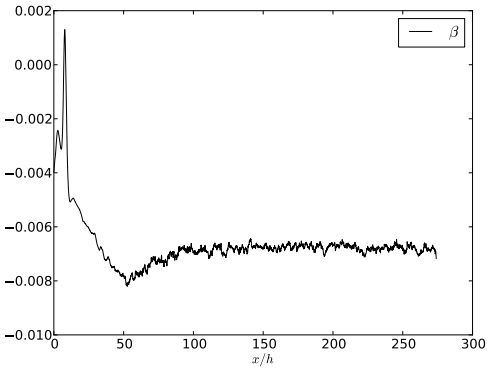
Bottom view of the upper wall :

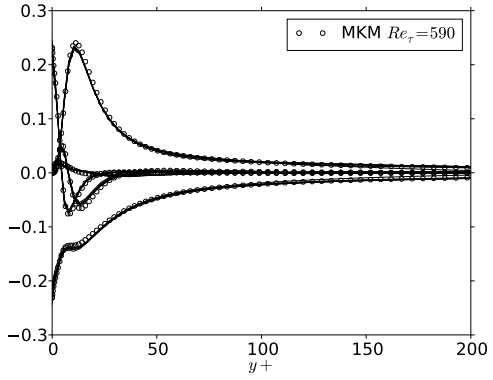


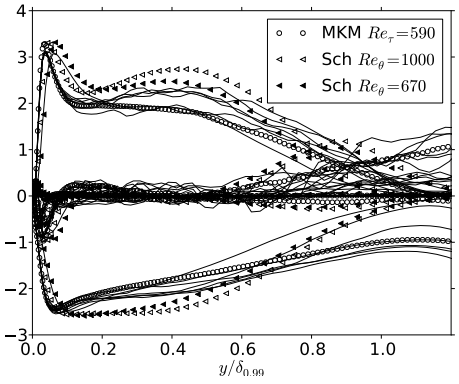
Top view of the lower wall :

A. Cadiou et al., *DNS of turbulent by-pass transition at the entrance of a plane channel*, Progress in Turbulence V, 2013.

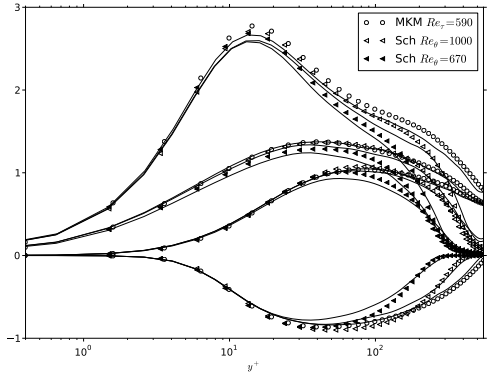
Towards fully developed turbulent channel flow





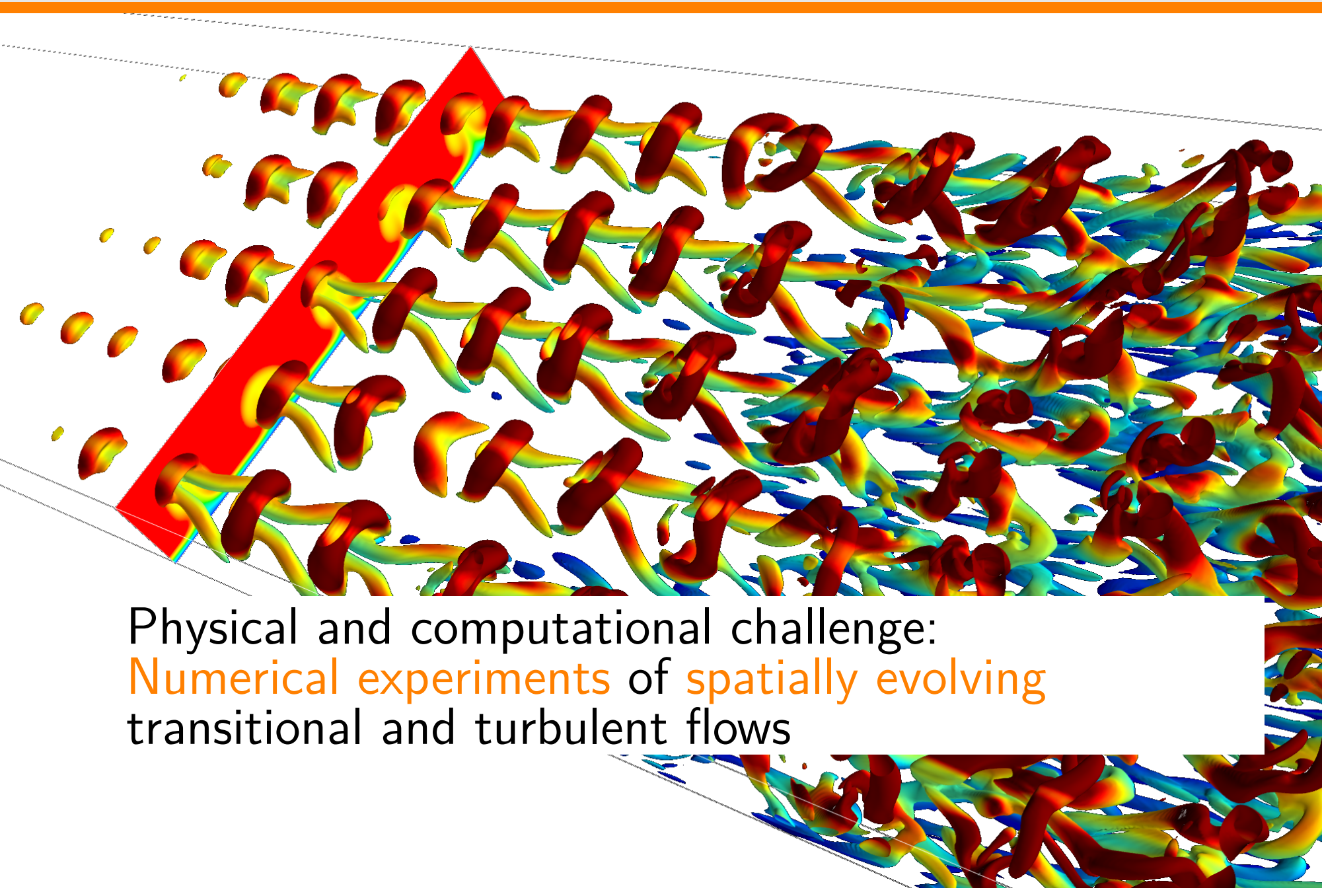


at $Re_h = 10000$
Laminar, $L_e/h \sim 1200$
Turbulent, $L_e/h \sim 110$
Fully developed turbulence, $Re_\tau = 567$



Time-averaged second-order velocity correlations in wall distance at 3 sections

M. Capuano et al., *DNS of the turbulent flow evolving in a plane channel from the entry to the fully developed state*, Progress in Turbulence VI, 2015.



Physical and computational challenge:
Numerical experiments of spatially evolving
transitional and turbulent flows

Outline

HPC ?

"High-Performance Computing is the **use of super computers** and **parallel processing** techniques for solving complex computational problems." *(from Techopedia)*

Very elongated (and large) geometry

- Numerical experiments require spectral accuracy
- $L_x/h \times L_y/h \times L_z/h = 280 \times 2 \times 9.4$
- $34560 \times 192 \times 768$ modes (~ 5 billions)

Periodic turbulent box ($Re_\tau = 590$), *Moser, Kim, Mansour, 1999*

- $L_x/h \times L_y/h \times L_z/h = 6.4 \times 2 \times 3.2$
- $384 \times 256 \times 384$ modes (~ 38 millions)

Requires from 100 to 10000 cores

Large configuration in space and time

- $34560 \times 192 \times 768$ modes ($\sim$ 5. billions of modes)
- travel 1 length with it=600000 iterations.

Memory constraint

- $N = N_x \times N_y \times N_z$, with N very large
- large memory requirement (executable $\sim 2To$)
- BlueGene/P 0.5 Go per core $\Rightarrow \sim$ 4000 cores needed

Wall clock time constraint

- CPU time $150h \sim$ 6 days on $\sim$ 16000 cores
- with 100 cores (if possible), 160 times slower, $24000h \sim$ 3 years

Big Data?

"Big data is a blanket term for any collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.
The challenges include capture, **storage**, search, sharing, **transfer**, **analysis** and **visualization**. "
(from Wikipedia)

- An old (and recurrent) problem of fluid mechanics simulations
- But storage, network flow rate and connectivity grow more slowly than computation

$\Rightarrow$ Exponential production of data

$\Rightarrow$ Revisit traditional usage

Data I/O and management

Large amount of highly partitioned files

- Large data
  - case $34560 \times 192 \times 768$: one velocity field ~ 120 Go
    statistics ~ 1 To
- Large amount of files, could rapidly exceeds inode or quota limit
  - statistics on ~ 2000 processes, $\sim 16\,000$ files
  - write ~ 140 time step during travel length ($L_x = 280$)
    (disk quota ~ 16 To)

Outline

Spectral approximation

Spectral coefficients with $N_x \times N_y \times N_z$ modes

$$U(x,y,z,t) = \sum_{m=-N_x/2}^{N_x/2} \sum_{p=-N_z/2}^{N_z/2} \left[\sum_{n=0}^{N_y-1} \alpha_{OS,n}^{mp} \hat{U}_{OS,n}^{mp} + \sum_{n=0}^{N_y-1} \alpha_{SQ,n}^{mp} \hat{U}_{SQ,n}^{mp} \right]$$

- Optimal representation of a solenoidal velocity field
- Elimination of the pressure

Spectral approximation

- Fourier-Chebyshev approximation with a Galerkin formulation
- Time integration with Crank Nicolson / Adams Bashforth scheme (2nd order) or implicit Runge-Kutta (3rd order)

Resolution of coupled systems for nonlinear advective terms

At each time step, $N_x \times N_z$ linear systems of dimension $N_y - 3$ are solved

$$A_{OS}^{mp} \alpha_{OS}^{mp} = b_{OS}^{mp}$$

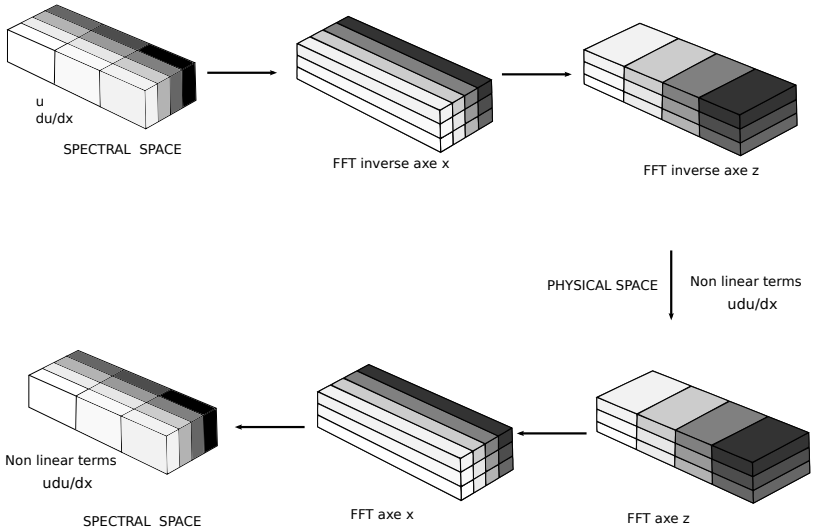
$$A_{SQ}^{mp} \alpha_{SQ}^{mp} = b_{SQ}^{mp}$$

A_{OS}^{mp} and A_{SQ}^{mp} are sparse matrices (resp. 7D and 5D)
$b^{mp} = b^{mp}(\alpha_{SQ}^{mp}, \alpha_{OS}^{mp})$
contains non-linear terms
(convolution products coupling every α_n^{mp})

$\Rightarrow$ b is calculated in physical space
$\Rightarrow$ must perform FFTs in each direction

Per iteration, i.e. at each time step,
27 FFT (direct or inverse) are performed ($\sim$ 16 millions of FFT)

2D domain decomposition



- Chebyshev between walls (y direction, $N_y + 1$ modes)
- 2D FFT in periodical directions (x direction and z direction)
- Transpose from $y-$pencil to $x-$pencil, $x-$pencil to $z-$pencil and back

Increase the number of MPI processes and reduce wall clock time

- 1D decomposition: MPI $\leq N_y$

  $34560 \times 192 \times 768 \rightarrow$ max. of MPI processes: nproc=192
- 2D decomposition: MPI $\leq N_y \times N_z$

  $34560 \times 192 \times 768 \rightarrow$ max. of MPI processes: nproc=147 456
- Perform data communications and remapping
- Choose data rearrangement to limit the increase in communications

Illustration

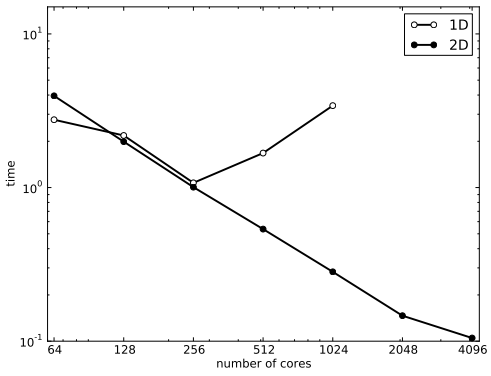


Figure: Time per iteration for a $1024 \times 256 \times 256$ case.

- improve the maximum of MPI processes
- could be limited by memory availability

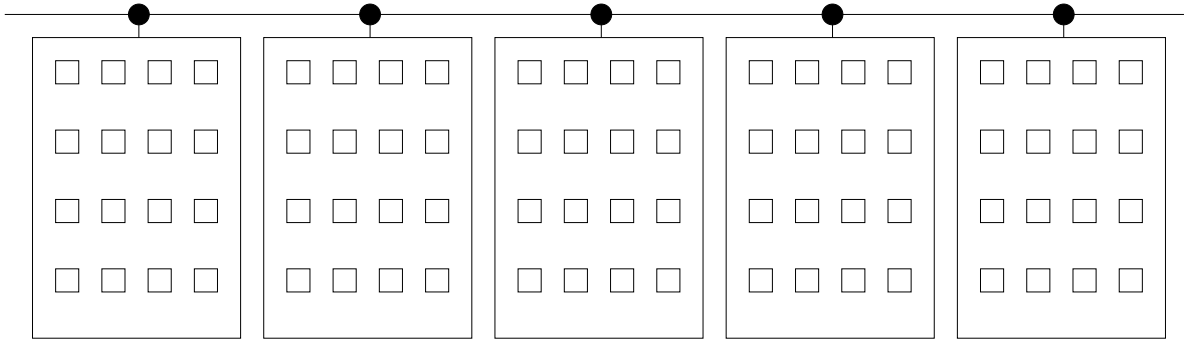
Constraints related to modern many-cores platforms

Tendancy towards many-cores platforms

- Limited number of nodes
- Increase of cores per node (BlueGene/P $= 4$ - SuperMUC $= 16$)

Increase MPI processes

- allow larger number of modes within the same wall clock time
- limit the memory available per processus



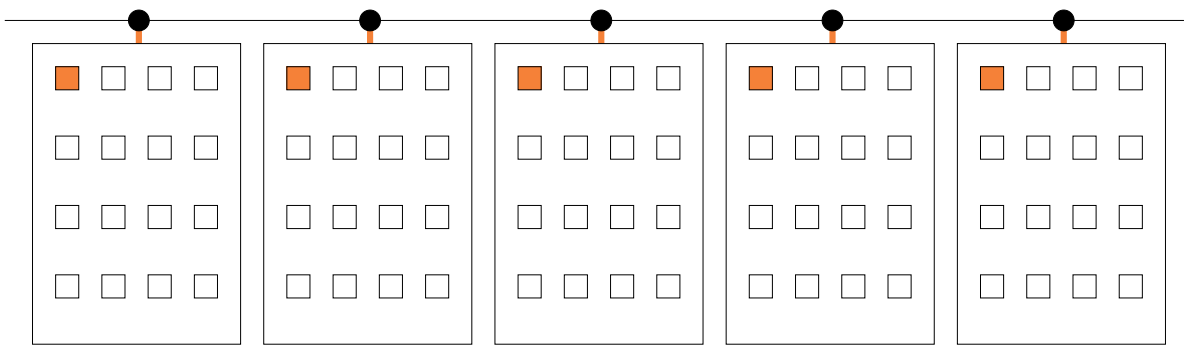
Constraints related to modern many-cores platforms

Tendancy towards many-cores platforms

- Limited number of nodes
- Increase of cores per node (BlueGene/P $= 4$ - SuperMUC $= 16$)

Increase MPI processes

- allow larger number of modes within the same wall clock time
- limit the memory available per processus



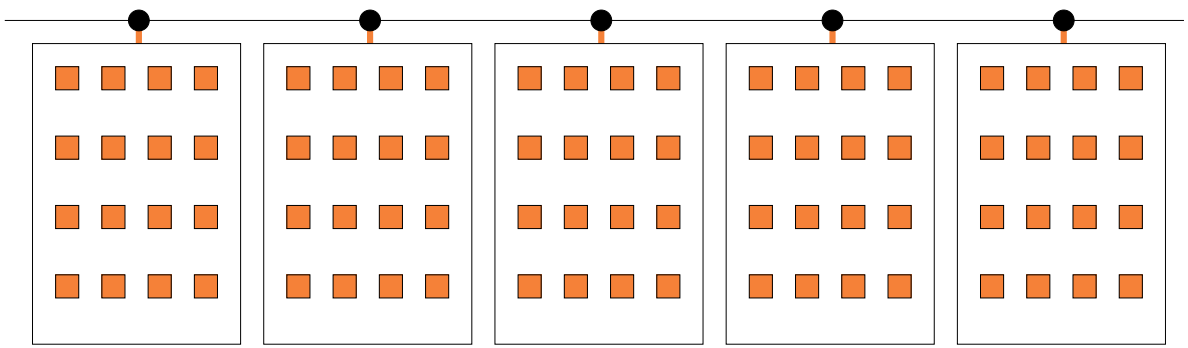
Constraints related to modern many-cores platforms

Tendancy towards many-cores platforms

- Limited number of nodes
- Increase of cores per node (BlueGene/P $= 4$ - SuperMUC $= 16$)

Increase MPI processes

- allow larger number of modes within the same wall clock time
- limit the memory available per processus



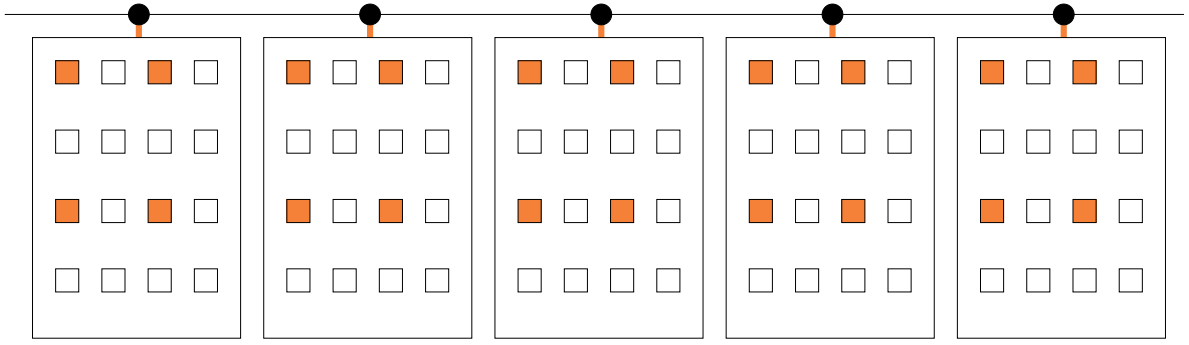
Constraints related to modern many-cores platforms

Tendancy towards many-cores platforms

- Limited number of nodes
- Increase of cores per node (BlueGene/P $= 4$ - SuperMUC $= 16$)

Increase MPI processes

- allow larger number of modes within the same wall clock time
- limit the memory available per processus



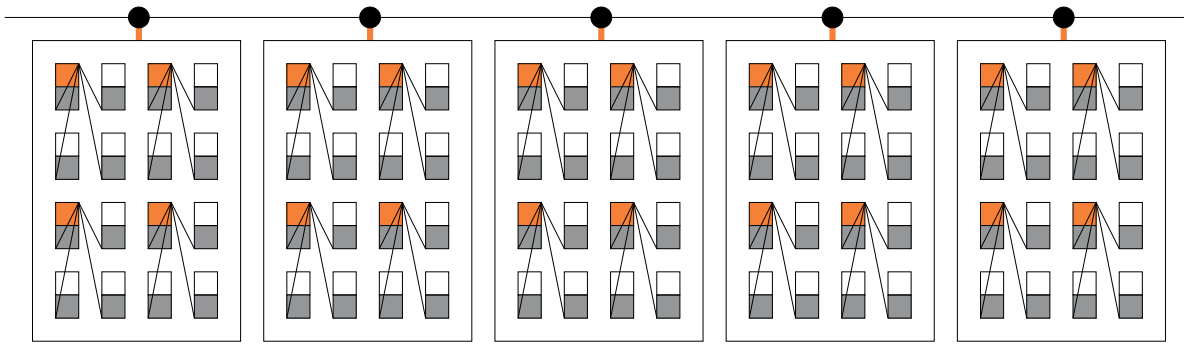
Constraints related to modern many-cores platforms

Tendancy towards many-cores platforms

- Limited number of nodes
- Increase of cores per node (BlueGene/P $= 4$ - SuperMUC $= 16$)

Increase MPI processes

- allow larger number of modes within the same wall clock time
- limit the memory available per processus



Hybrid OpenMP/MPI

Suitable for recent many-core platforms

- Reduces the number of MPI processes
  - Reduces the number of communications
  - Increases the available memory size per node

Modification for many threads

- Time of thread creation exceeds inner loop time execution
- Implementation of explicit creation of threads
- Recover full MPI performance and allow further improvment.

Illustration

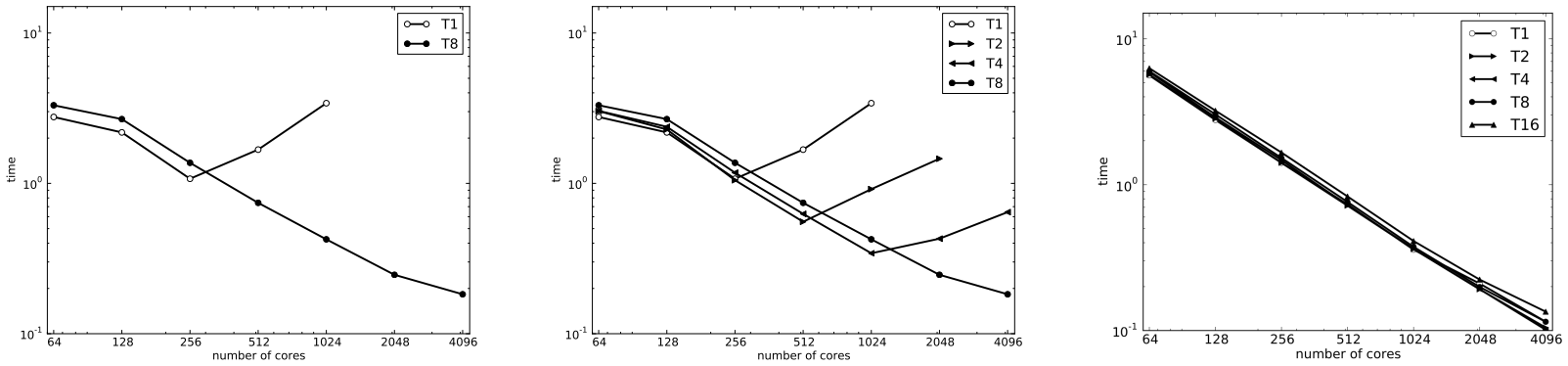


Figure: Time per iteration for a $1024 \times 256 \times 256$ case.

Suitable for recent many-core platforms

- Reduces the number of MPI processes
  - Reduces the number of communications
  - Increases the available memory size per node
- Implementation of explicit creation of threads
  - Coarse grained OpenMP needed for fast inner loop
  - Define a new synchronization barrier

Measure

MPI proc./node	threads per node	nodes	cores	time per it. (s)	gain
16	1	16	256	1.46	
8	1	32	512	1.47	
4	1	64	1024	1.43	
2	1	128	2048	1.44	
1	1	256	4096	1.44	1.00
1	2	256	4096	0.74	1.95
1	4	256	4096	0.38	3.79
1	8	256	4096	0.21	6.86
1	16	256	4096	0.14	10.28
16	1	256	4096	0.11	12.45
8	1	256	2048	0.20	6.85
4	1	256	1024	0.35	3.91
2	1	256	512	0.71	1.93
1	1	256	256	1.37	1.00

Time per iteration for the $1024 \times 256 \times 256$ case.

More than domain decomposition ...

Tasks parallelization : overlap communication by computation

- reduces by 20% time per iteration

Placement of processus

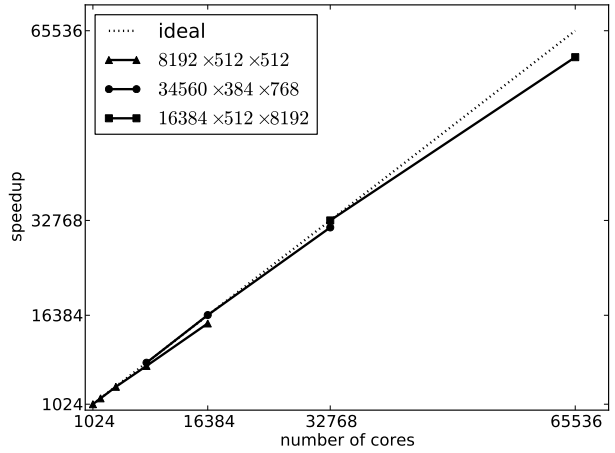
- specific on each platform, optimize interconnection communications
- avoid threads to migrate from one core to another

  example: TORUS versus MESH in BlueGene/P platform - 50% faster

Illustration

$N_x \times N_y \times N_z$	cores	map.	comm.(%)		time per iteration (s)	
			Mesh	Torus	Mesh	Torus
$1024 \times 256 \times 256$	512	$16(\times 32)$	16.2	-	0.95	-
	1024	$32(\times 32)$	15.8	-	0.52	-
	2048	$32(\times 64)$	15.2	12.0	0.28	0.23
$4096 \times 512 \times 512$	2048	$32(\times 64)$	19.9	7.8	4.55	3.96
	4096	$64(\times 64)$	30.8	10.2	4.29	1.98
	8192	$64(\times 128)$	39.2	12.7	2.25	1.09

Efficiency



- Fairly portable on HPC
- Reasonable efficiency on $O(10^5)$ cores
- Small time spent waiting for communications $\sim 12\%$
- Fast wall-clock time for a global numerical method (1.3 s/it on BlueGene/P - 0.2 s/it on SuperMUC for 1 billions of modes)

J. Montagnier et al., *Towards petascale spectral simulations for transition analysis in wall bounded flow*, Int. J. for Numerical Methods in Fluids, 2012

HPC implementation

Parallelization

- 2D domain decomposition with MPI and FFT 3D
- Optimal data rearrangement to limit communications

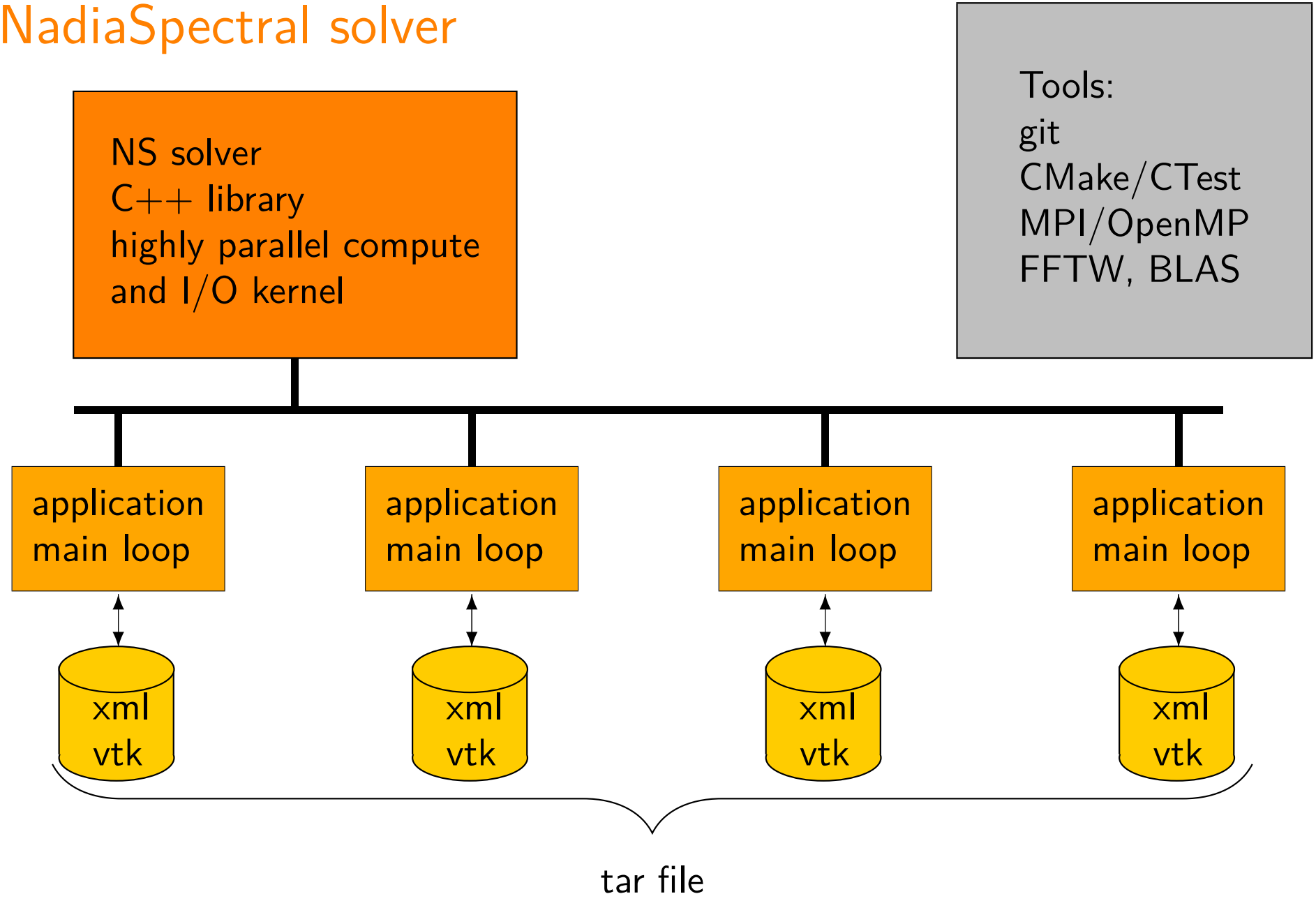
Hybrid MPI/OpenMP on recent many-cores HPC platforms

- implementation of explicit threads creation
- tasks parallelization (masks communications)

Parallel Input/Output for check-pointing and Data management

- Fast parallel I/O using standard XML/VTK format
- Unix I/O faster than MPI I/O (2x)
- I/O files embedded in a tar file (pvd + parallel vtr)
- perform FPZIP compression if needed (lossless or lossy (48bits))

NadiaSpectral solver



Outline

Bottleneck: data manipulation

Simulation (multi-run batch) on LRZ SuperMUC

- $\sim$ 5 billions of modes $34560 \times 192 \times 768$
- run with $\sim 1s/\Delta t$ on 16 384 cores 2048 partitions
- Large data $\sim$ 120 Go/Δt, statistics $\sim$ 1 To

Manipulation of very large and highly partitioned data

- Data manipulation during simulation (checkpoint data)
- Data manipulation for analysis, post-treatment and visualization

$\Rightarrow$ **parallel strategy mandatory**

Data manipulation during simulation

Data Input/Output and storage

- Large data
  - case $34560 \times 192 \times 768$: one velocity field ~ 120 Go
    statistics ~ 1 To

$\Rightarrow$ Use parallel IO (each processes writes its own data)

- Large amount of file, could rapidly exceeds inode or quota limit
  - statistics on ~ 2000 processes, $\sim 16\,000$ files
  - write ~ 140 time step during travel length ($L_x = 280$)
    (disk quota ~ 16 To)
- Manage the large amount of data generated

$\Rightarrow$ Use of predefined parallel format (VTK, HDF5, NetCFD, ...)

beware not to add useless complexity for regular structured data

$\Rightarrow$ wrap in tar archive file or separated directory

$\Rightarrow$ Optimize data transfert between platform

$\Rightarrow$ or perform co-analysis of the flow without writting flow fields

Data manipulation after simulation

Data processing

- Part of the analysis is performed during simulation
- Part of it is explored afterwards

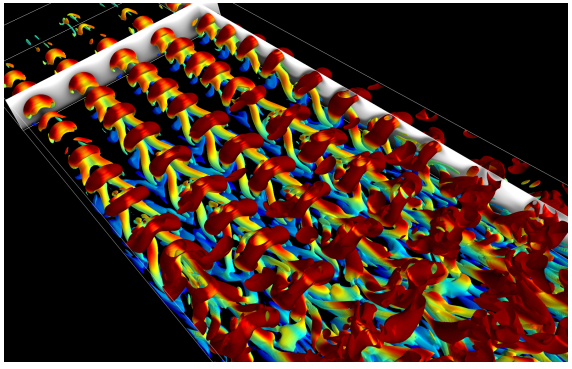
3D visualization

- Cannot be performed directly on HPC platforms

Requirements and constraints

- Entails spatial derivation, eigenvalues evaluation ...
- Preserve accuracy of the simulation
- Should be interactive and when ready on batch mode

Traditional usage for 3D visualization



Workflow

- Computation on remote platform
- Write data result on disk during computation
- Transfer data to local server
- Use open-source softwares for analysis

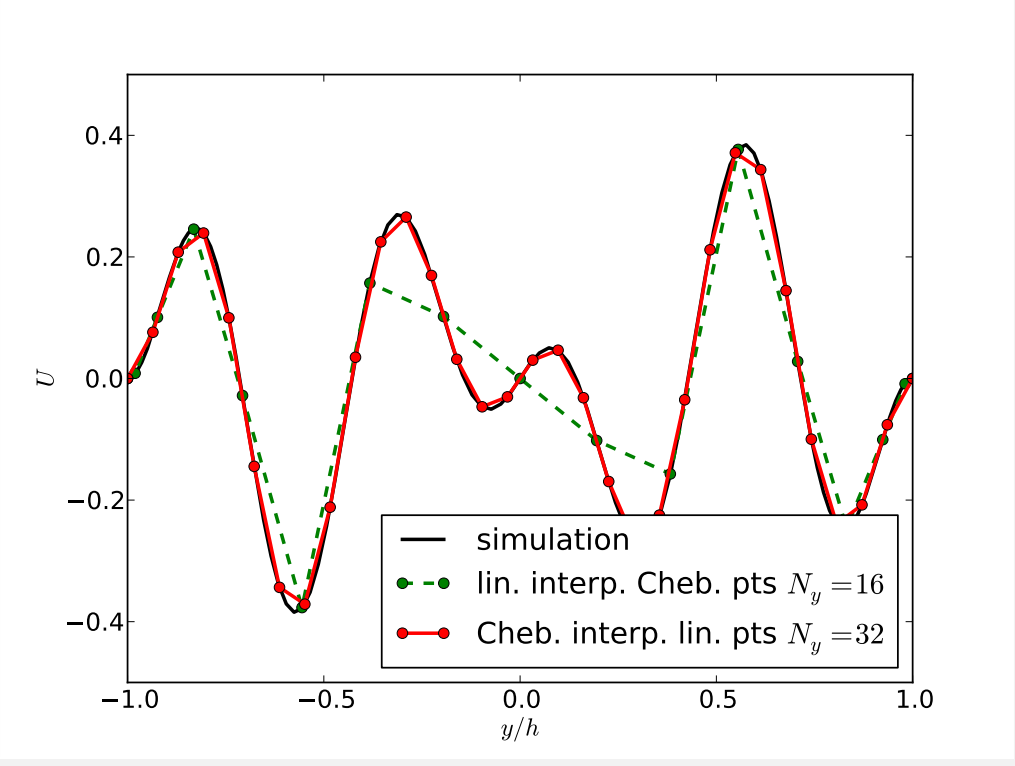
Open-source softwares

- **VisIt** : parallel general interactive tools (with our own DB reader plugin)
- **ParaView** : (idem)
- **Mayavi** : Python VTK interface
- **Python + matplotlib** : 1D , 2D + some 3D

Limitation for spectral approximation

Linear interpolation between collocation points

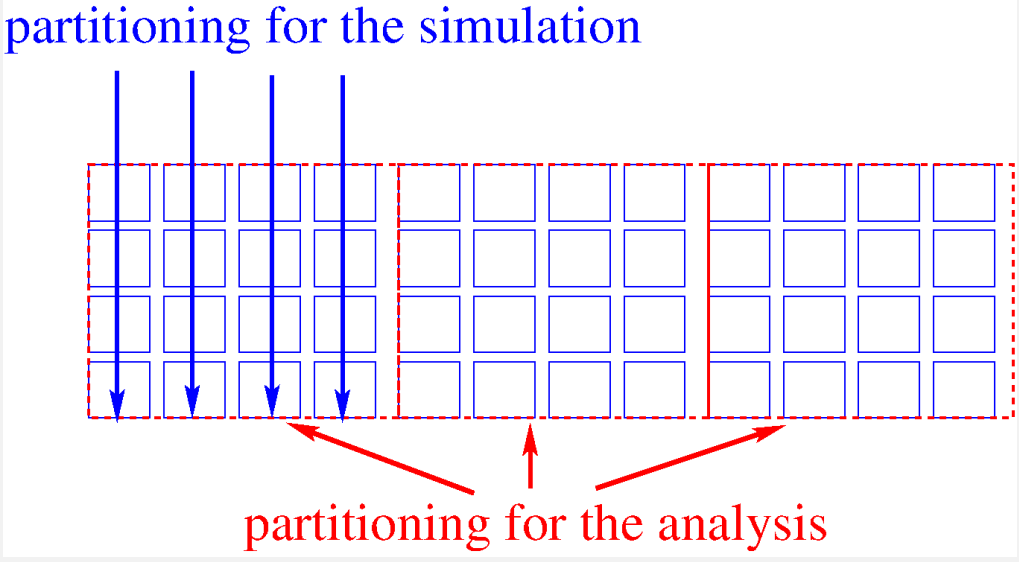
ă



- Missing information, yet contained in the original spectral 3D field
- Rendering and interactivity slow down on non regular grid

Domain decomposition

Slices between partitions of the domain are ignored



Simulation uses non-overlapping domain decomposition
Visualization requires overlapping domain decomposition

Workflow

HPC platform (Tier-0, Tier-1)

parallel transfer (GridFTP, ...)

data server

NFS

HPC cluster (Tier-2)

NFS

or

ssh tunneling

user

NX (x2go)/vnc

graphic stations (Tier-2)

Analysis and visualization of stored data

Analysis a posteriori (in parallel)

- Python script with mpi4py
- parallel client server with 2D/3D (matplotlib, mayavi)
- interface with C++ lib using swig
- manipulate tar data file
- analyze with simpler parallel partitioning (1D)
- preserve the same accuracy in the compute and analyze steps

- Still requires disk I/O, data transfert and storage
- Data storage and post-treatment identified as a major challenge

  S. Requena, Big Data and HPC, 2013

In situ (real computational time) visualization

Remote co-processing during simulation without stored data

Open-source software

- **VisIt**
- **ParaView**

Limitations

- run with the same granularity as the simulation
- affect speed of computation

Requirements

- Preserve spectral accuracy
- Computation of quantities from simulations variables
- Fast enough
- Act on simulation parameters (like in experiment)

Requirements for embedded analysis

Code instrumentation

- add parallel analysis code as independant MPI processes
  - use its own time-step
  - interact with the simulation every $\sim 100\Delta t$
  - can use dedicated nodes
  - use a coarse and simpler domain decomposition (operators)
  - interpolated on finer regular overlapping grids (visualization)
  - can modify the parameters of the simulation (control)

Interface with parallel analysis and visualisation

- Python and matplotlib
- VisIt (libsim)
- allow interactivity and scripting

Embedded (in-situ) analysis using Python

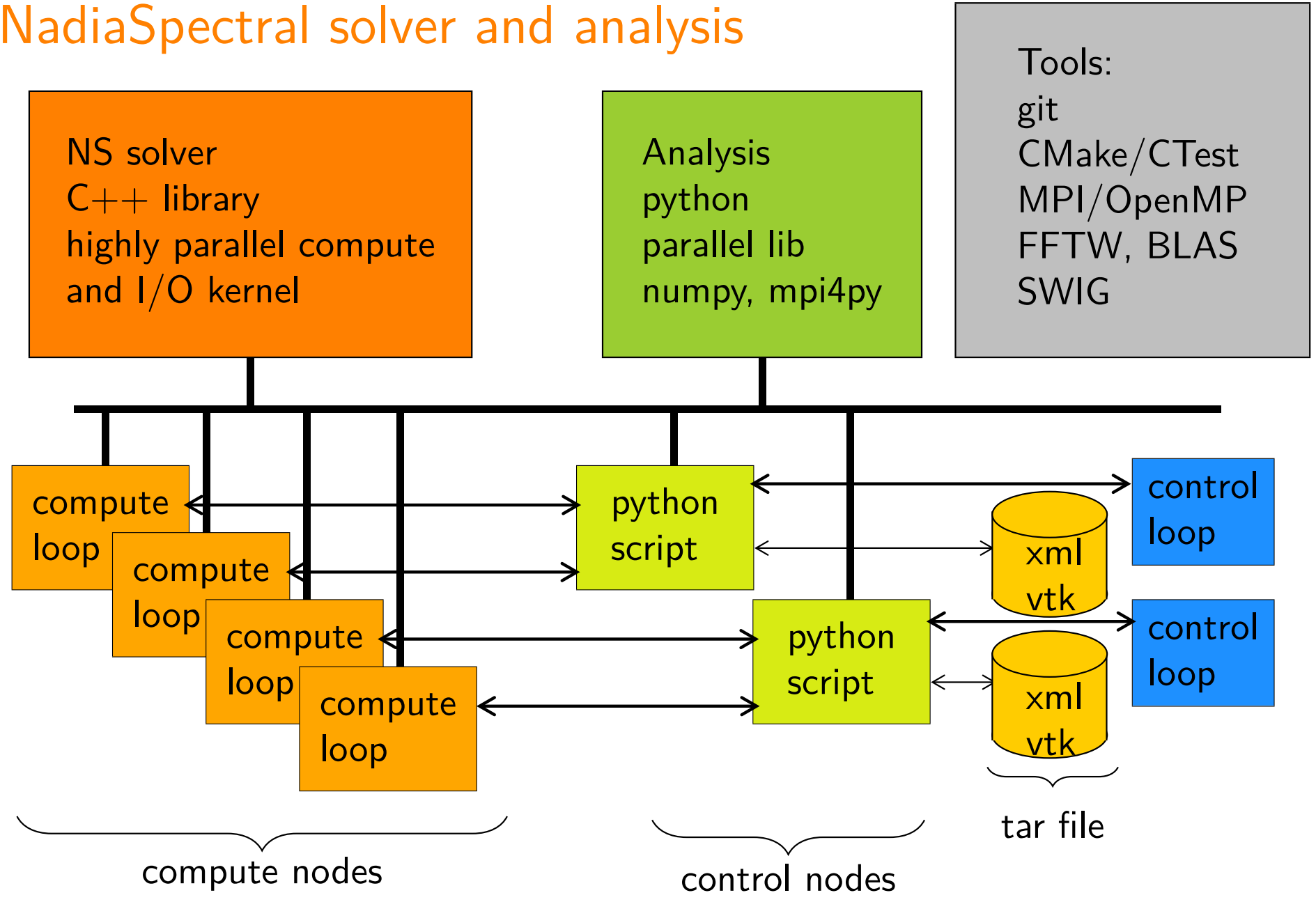
Common usage of Python for HPC

- simulation driven by Python script
- HPC Python code using mpi4py

Our approach

- No performance degradation of the computation kernel
- Embed a Python interpreter in C++
- Only used for the analysis and control of the simulation
  - Simpler domain decomposition for the analysis
  - Require 2 groups of MPI processes
    - N_k nodes for computation kernel (in C++)
    - N_a analysis nodes (with Python): $N_a \ll N_k$

NadiaSpectral solver and analysis



Follow time evolution of flow structures

Explore time evolution at $Re_h = 2500$

$5760 \times 192 \times 512$ modes ($\sim$ 566 millions of modes)

In-situ analysis (embedded to the simulation)

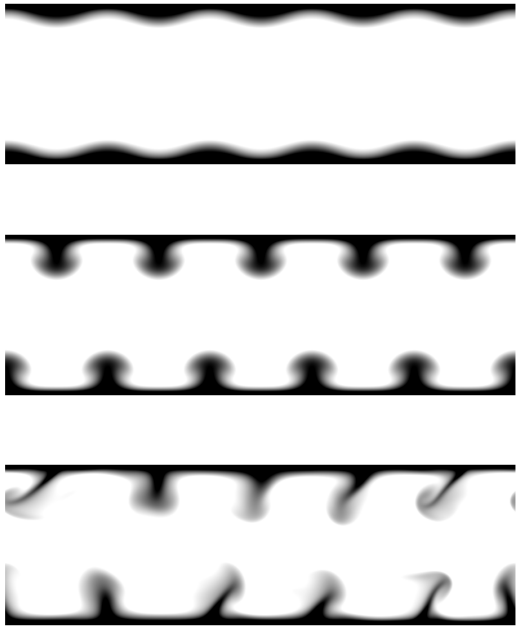
- Run simulation on 144 nodes ($128 + 16$ nodes)
  - 512 MPI procs, 4 MPI processes per node, 4 threads
  - 2048 cores ($\sim$ 128 thin nodes)
  - 16 MPI procs, 1 MPI processus per node
  - 16 cores ($\sim$ 16 fat nodes)
- Analyze every 50 time steps
- Computation of Λ_2 criteria during 10 time steps

$\Rightarrow$ **does not affect global CPU time**

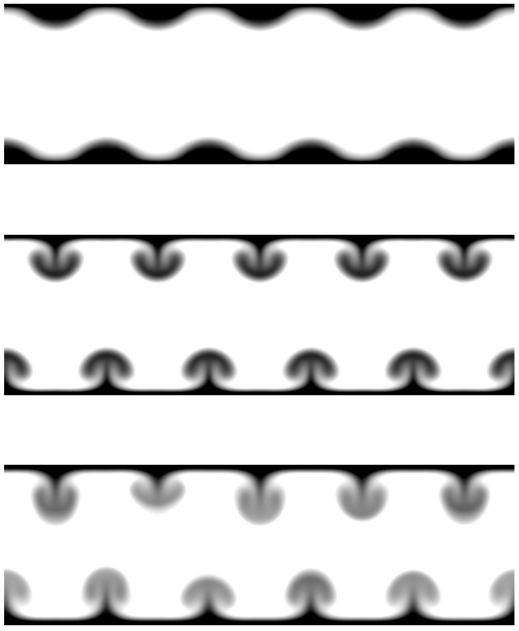
Generate an evolution in time with more than 3000 images
... and interact with the code !

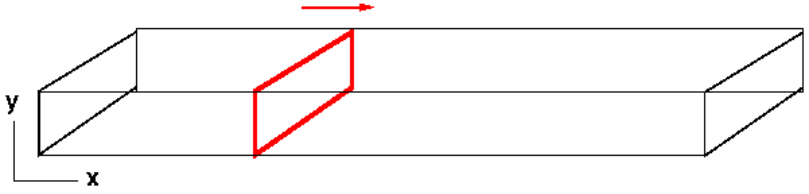
Sinuous and varicous instabilities

$A^{opt} = 2\%$



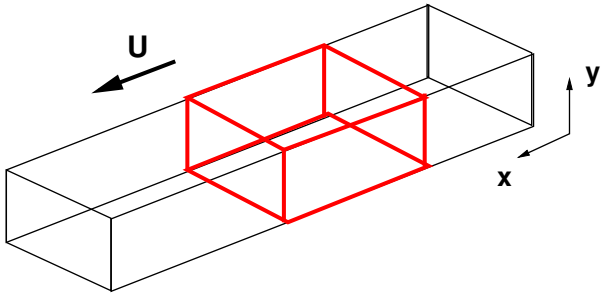
$A^{opt} = 4\%$

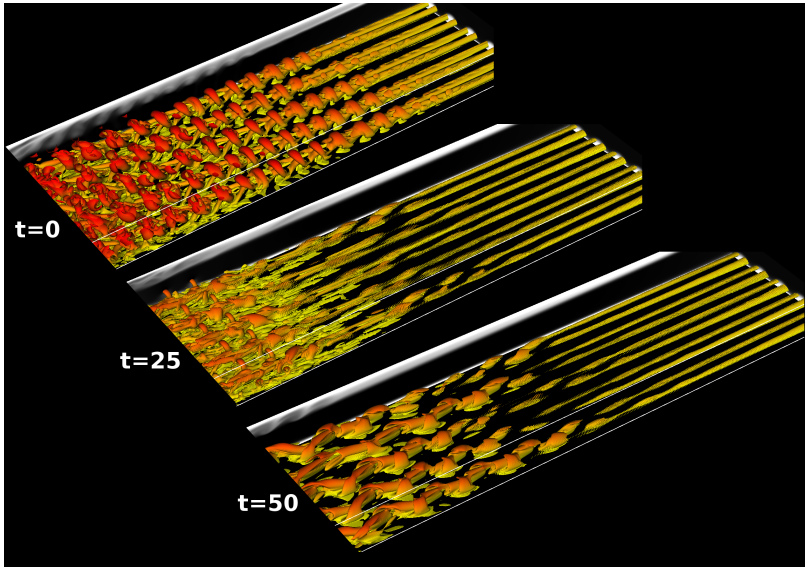




at fixed time t

Visualization of 3D temporal evolution



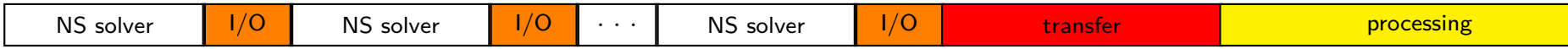


Present workflow versus traditional

Comparison of strategies for the simulation $5760 \times 192 \times 768$ of modes on 2 084 cores for 512 MPI partitions.

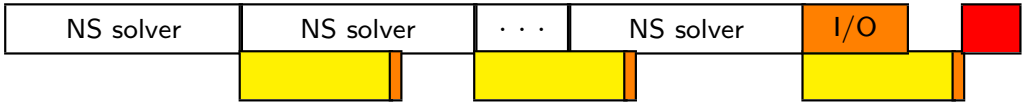
10 analysis at the same physical time interval Δt.

Traditional usage, 2 048 cores for 512 MPI partitions, 1.4 To of data,



or when sharing cores with analysis,

Revisited in-situ co-processing analysis, solver on 2 048 cores for 512 MPI partition, 16 cores for co-processing, 28 Go output data



Benefits of the new implementation

- Simplify the analysis
- Get faster developments and tests
- Make overlap I/O and computation, asynchoneous execution and communications
- Allow in-situ visualization
- but need a more complex environment !
  - coupling between C++/python/external tools (VisIt)
  - asynchroneous communications
  - depend on a large number of external libraries (compatibility)
  - make the porting and tuning on HPC platforms more complex limitation of the module system

Outline

What was achieved for HPC simulations

A suitable development and software environment

- code C++
- BLAS, GSL
- MPI/OpenMP - optimized libraries (e.g. FFTW, MKL)
- cmake, git
  - swig interface Python and a C++ library derived from the code
  - python, mpi4py, numpy, matplotlib, mayavi, visit ...

Development of a parallel strategy for the code

- revisit parallel strategy of the code
- revisit strategy of data transfer and storage
- revisit strategy for the analysis and visualization

To read mode ...

1. *M. Buffat and A. Cadiou and L. Le Penven and Ch. Pera,*
   In-situ analysis and visualization of massiely parallel computations, (2015), Int. J. of High Perf. Computing Appl.
2. *M. Capuano and A. Cadiou and M. Buffat and L. Le Penven,*
   DNS of the turbulent flow evolving in a plane channel from the entry to the fully developed state, (2015), Progress in Turbulence VI.
3. *A. Cadiou, M. Buffat, L. Le Penven*
   Bypass transition at the entrance of a plane channel, (2014), EFMC10, Copenhagen.
4. *A. Cadiou, M. Buffat, L. Le Penven, and J. Montagnier,*
   DNS of turbulent by-pass transition at the entrance of a plane channel. (2013), Progress in Turbulence V, p59-64.
5. *M. Buffat, L. Le Penven., A. Cadiou, and J. Montagnier,*
   DNS of bypass transition in entrance channel flow induced by boundary layer interaction. (2013), European Journal of Mechanics
6. *J. Montagnier, A. Cadiou, M. Buffat, L. Le Penven,*
   Towards petascale spectral simulations for transition analysis in wall bounded flow, (2012), Int. Journal for Numerical Methods in Fluids
7. *M. Buffat, L. Le Penven, and A. Cadiou,*
   An efficient spectral method based on an orthogonal decomposition of the velocity for transition analysis in wall bounded flow. (2011), Comput. Fluids.