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

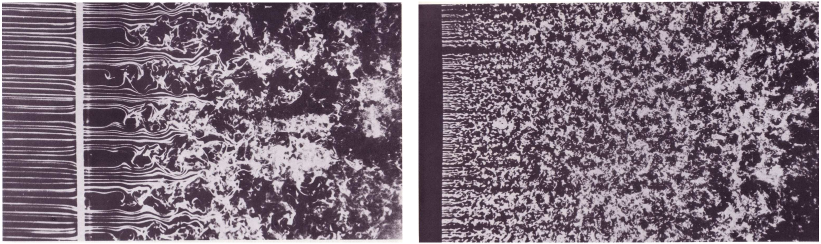
Anne Cadiou, Marc Buffat, Christophe Pera*
Bastien Di Pierro, Frédéric Alizard, Lionel Le Penven

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

19th International Workshop on
Advanced Computing and Analysis Techniques in Physics Research
Saas Fee, March 11th, 2019

Fluid Mechanics

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)

⇒ Resolve all length and time scales

Navier-Stokes equations

Conservation of mass and momentum

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

$$\nabla.\boldsymbol{U} = 0$$

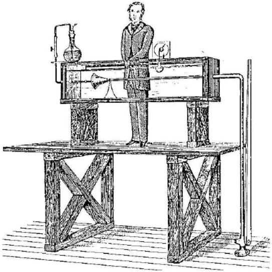
(velocity $\boldsymbol{U}$, pressure p, density ρ, viscosity ν)

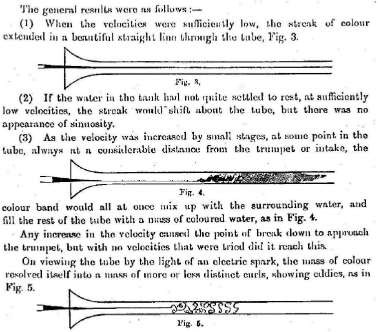
+ initial and boundary conditions

⇓

- 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.

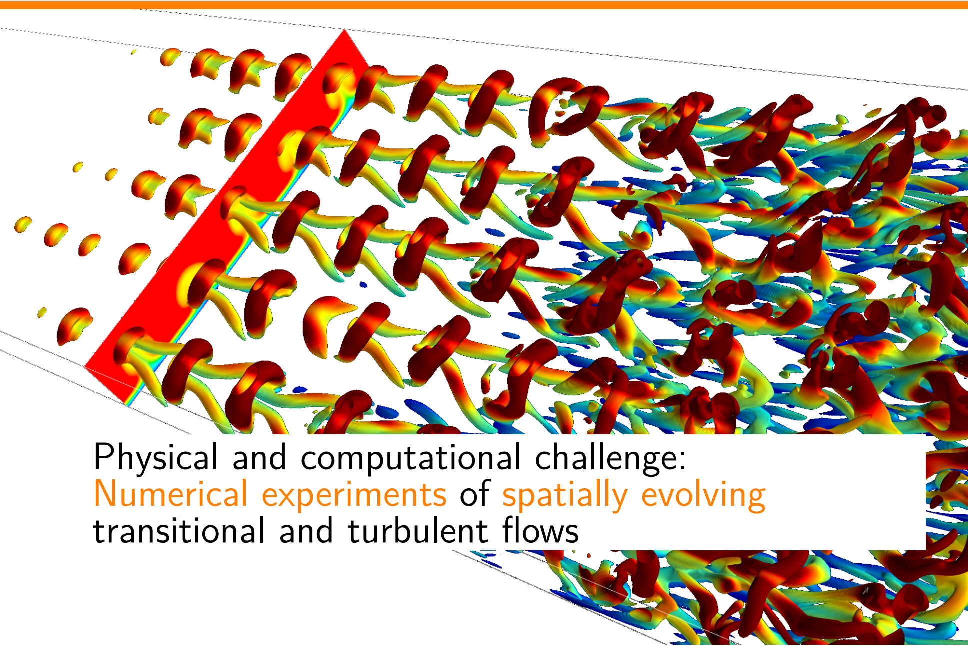
Any increase in the velocity caused the point of break down to approach the trumpet, but with no velocities that were tried did it reach this.

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



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

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 ($\sim$ 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 ($\sim$ 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 $\sim 2\,To$)
  - 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

$\Rightarrow$ Revisit traditional usage

Manage very large and highly partitioned files

Large data

- $34560 \times 192 \times 768$ modes

  one velocity field occupies ~ 120 Go

  statistics ~ 1 To

Large amount of files, could rapidly exceeds inode or quota limit

- statistics on ~ 2000 processes, $\sim 16\,000$ files
- need to write ~ 140 time step during travel length ($L_x = 280$) (disk quota ~ 16 To)

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

$\Rightarrow$ **parallel strategy mandatory**

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 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

$$A^{mp}_{OS}\alpha^{mp}_{OS} = b^{mp}_{OS}$$

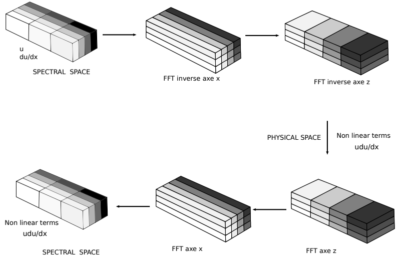
$$A^{mp}_{SQ}\alpha^{mp}_{SQ} = b^{mp}_{SQ}$$

A^{mp}_{OS} and A^{mp}_{SQ} are sparse matrices (resp. 7D and 5D)
$b^{mp} = b^{mp}(\alpha^{mp}_{SQ}, \alpha^{mp}_{OS})$
contains non-linear terms
(convolution products coupling every α^{mp}_n)

⇒ b is calculated in physical space (pseudospectral method)
⇒ 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

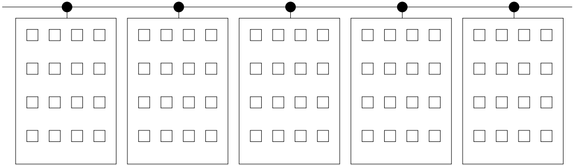
Constraints related to modern many-cores platforms

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



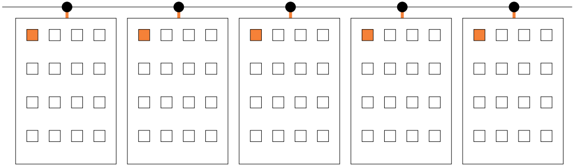
Constraints related to modern many-cores platforms

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



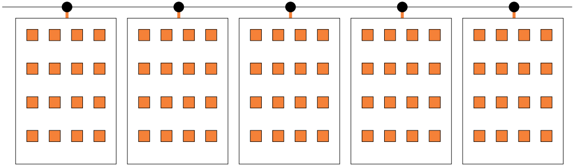
Constraints related to modern many-cores platforms

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



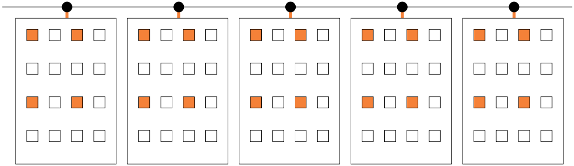
Constraints related to modern many-cores platforms

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



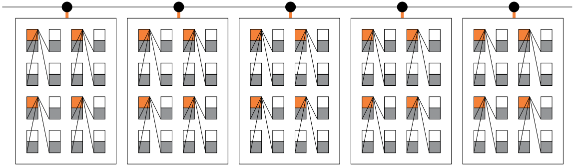
Constraints related to modern many-cores platforms

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.

More than domain decomposition ...

Tasks parallelization : overlap communication by computation

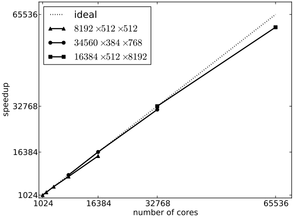
- reduces by 20% time per iteration

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

Speedup, efficiency and wall-clock time per time step



speedup on JUQUEEN (PRACE)

- 88% of efficiency on 16384 cores
- 0.2 s/dt sur SuperMUC for 10^9 modes on 16384 cores

Data manipulation during simulation

I/O and storage

- Fast parallel I/O using standard XML/VTK or HDF5 format

  beware not to add useless complexity for regular structured data
- Unix I/O faster than MPI I/O (2x)

  in our experience on large HPC platforms
- I/O files embedded in a tar file (pvd + parallel vtr) or in a separate directory

  to avoid exceeding inodes number and fasten transfer
- perform FPZIP compression if needed (lossless or lossy (48bits))

⇒ Optimize data transfert between platform

or perform co-analysis of the flow without writting raw data

Data manipulation after simulation

Data processing

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

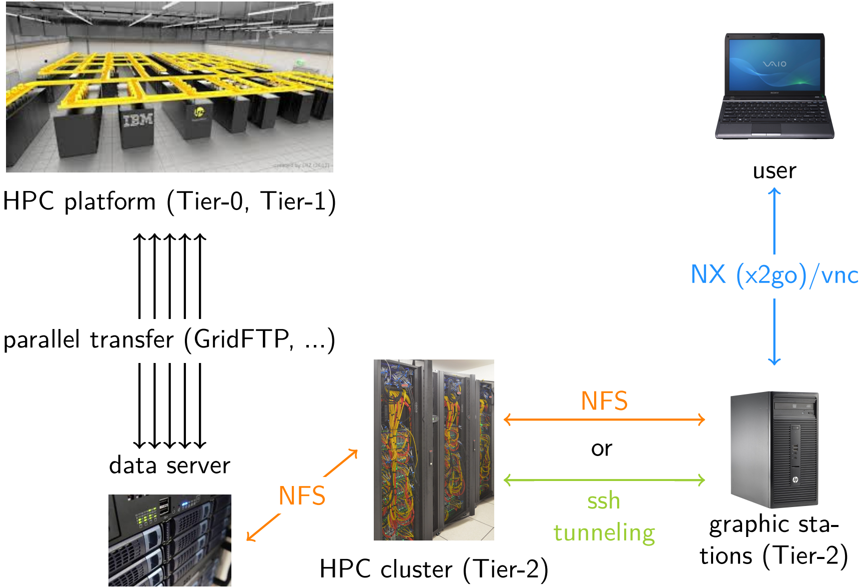
3D visualization

- 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

Client/server workflow



Analysis and visualization of stored data

A posteriori on raw data (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) analysis and visualization

Remote co-processing during simulation without I/O of raw data

Analysis

Compute density energy spectra, statistics, etc. at physical relevant stride

3D visualization

Open-source software

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

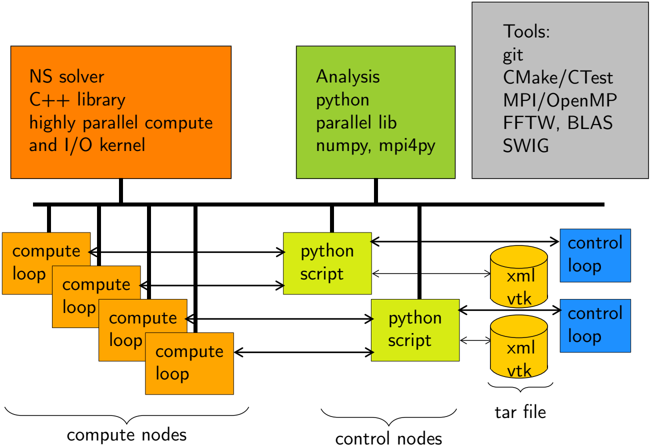
Limitations

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

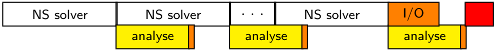
Requirements

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

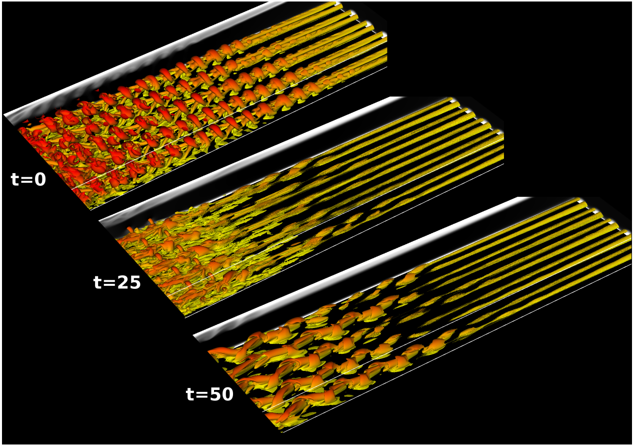
Present in situ workflow



Ansynchoneous analysis



Example : visualization of sinuous/varicose instabilities



Control of the numerical experiment :
modification of the amplitude of the perturbation at the channel entry
while the simulation is running

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

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