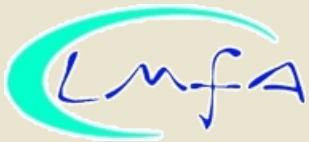
High Performance computing and Big Data for turbulent transition analysis

Marc Buffat, Lionel Le Penven, Anne Cadiou

LMFA, UCB Lyon 1, ECL, INSA, CNRS

CCDSC September 2014



Outline

HPC and Fluid Mechanics

HPC supports scientific research

- **Challenge** : gain a better understanding of turbulence
- Increases numerical model accuracy
- Enables to explore multi-physics and multi-scales effects
- Helps to quantify prediction uncertainties and errors

Consequences

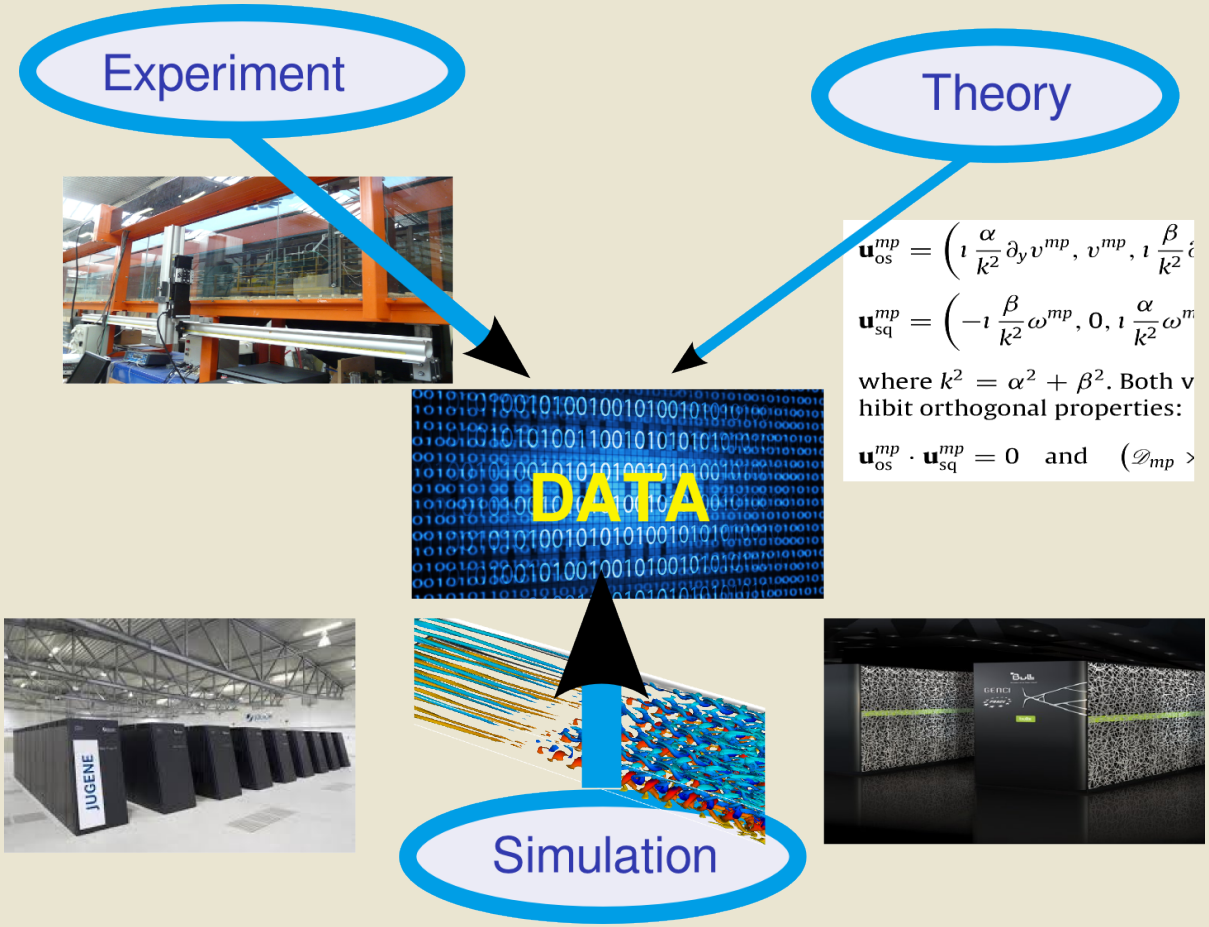
- CFD is a large consumer of HPC
- $\rightsquigarrow$**Generates an increasing amount of large data**

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, curation, **storage**, search, sharing, **transfer**, **analysis** and **visualization**."
(WIKIPEDIA)

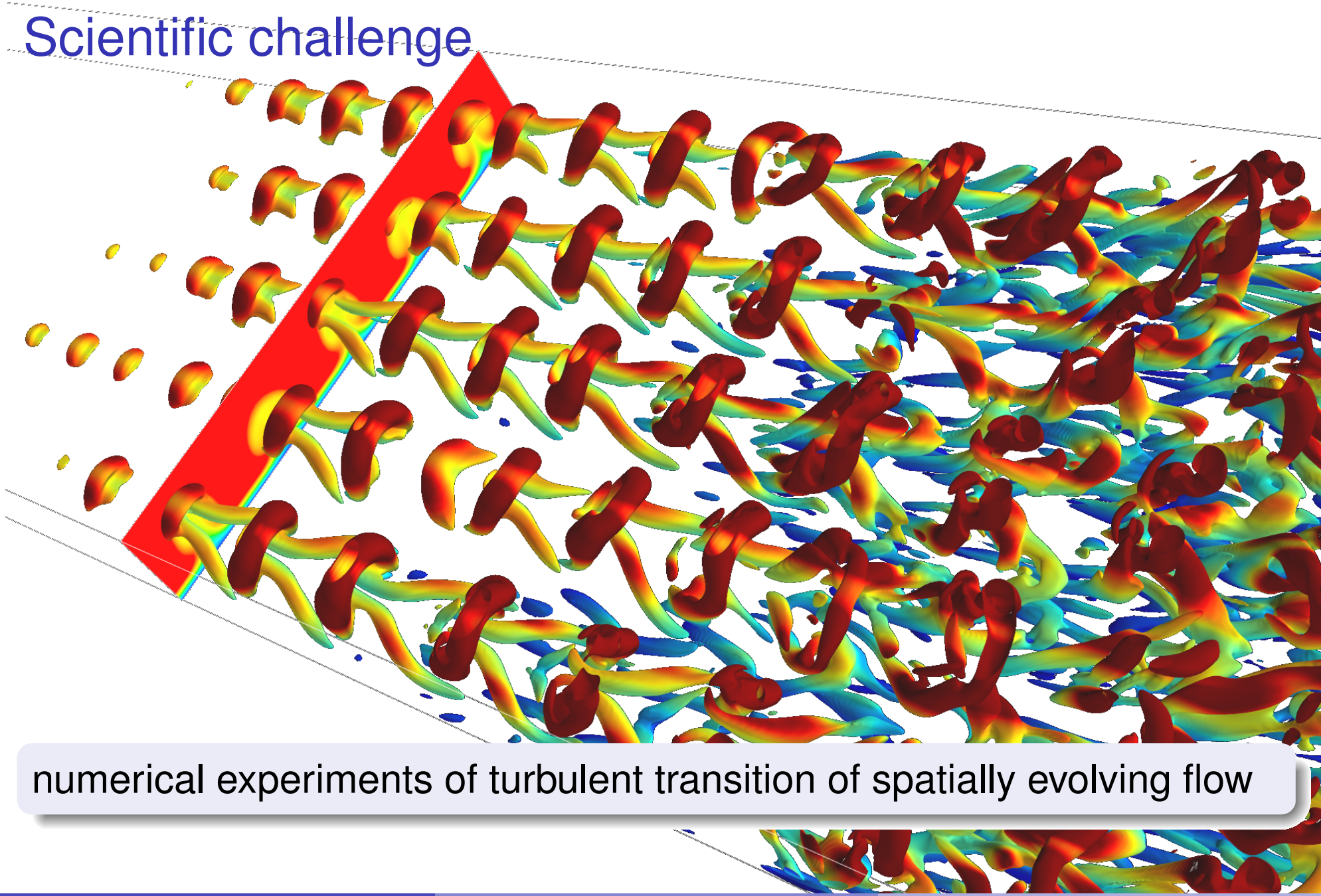
- An old (and recurrent) problem in CFD
- But storage, network flow rate and connectivity growth less than computing power
  - $\Longrightarrow$Exponential production of data
  - $\Longrightarrow$**Revisit traditional usage**

The fourth paradigm: "*data-intensive scientific discovery*" *(Kristin Tolle, Tony Hey, Stewart Tansley, 2009)*



$\Longrightarrow$Revisit work-flow analysis to get closer to num. experiments

Scientific challenge



numerical experiments of turbulent transition of spatially evolving flow

Stability of entrance and developing channel flow

Transition at the entrance of the channel flow at sub-critical Reynolds number

- Development length and evolution towards a developed flow
- Stability of the developing entry flow
- Boundary layer interaction
- Evolution of turbulence properties in the developing flow

Very elongated geometry

- Transition and Turbulence numerical experiments require spectral accuracy
- Geometry size implies large - and anisotropic - number of modes

Buffat et al., Non modal sub-critical transition of channel entry flow, ETC14, Sep. 2013

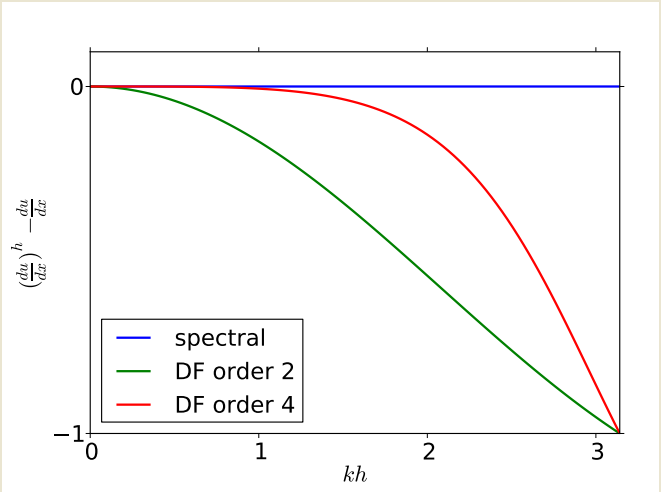
Numerical challenge

Wide range of non-linearly interacting scales

- Numerical experiment of turbulent transition:
  $\Longrightarrow$need to resolve the flow at all scales
- Scale separation $R_\lambda \sim Re_t^{0.5}$
  spatial resolution $N^3 \sim Re_t^{9/4}$, time frequency $\tau \sim Re_t^{11/4}$

Spectral methods are attractive, due to their high spatial accuracy

- Spatial derivatives are **exact**
- **Exponential** convergence



Since the 70's, extensively applied to simulation of turbulent flows but, their **implementation on new HPC must be carefully considered.**

NadiaSpectral code

DNS solver for the Navier-Stokes equations

- Spectral approximation: Fourier Chebyshev
- Galerkin formulation using an orthogonal decomposition of $\overrightarrow{\mathbf{U}}$
- Optimal representation of the solenoidal velocity field (2 scalars)
- Time integration with Crank Nicholson/ Adams Bashforth
- initially parallelize on $O(100-1000)$ processors

Numerical bottleneck on new HPC

- FFTs in each direction (FFT3D)
- per iteration (time-step) $27\times$ FFT3D (direct & inverse)
- global operation (in each direction)
- difficult to parallelize efficiently on new HPC

HPC Implementation

- NadiaSpectral solver written in C++, using FFTW or Intel/IBM.
- Used since 10 years with strong validation (using git)
- Fairly portable on HPC (using cmake)

Parallelization using MPI

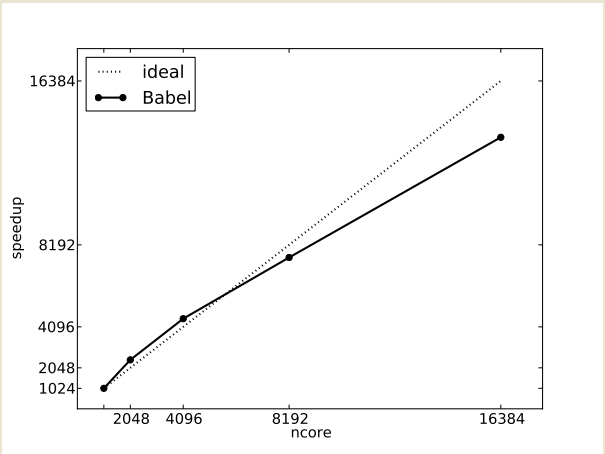
- 2D domain decomposition using MPI
- FFT3D using 3 $\neq$ 2D domain decompositions
- choose data rearrangement to limit communication

Hybrid MPI/OpenMP on recent many-core HPC

- implementation of explicit creation of threads
- task parallelization (mask communication)

`http://ufrmeca.univ-lyon1.fr/~buffat/NadiaSpectral`

HPC Efficiency



- Fairly portable on HPC (BlueGene, Curie, Linux Cluster, ..)
- Reasonable efficiency on $O(10^4 - 10^5)$ cores
- Small time spent waiting for communications $\sim 10\%$
- Fast wall clock time for a global numerical method (1.3s/it on BlueGene/P - 0.2s/it on SuperMUC for $\sim$ billions of modes)

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

Bottleneck of large and massively parallel data

- Simulation (multi-run batch) on LRZ SUPERMUC
  PRACE project

- ~ 5 billions modes $34560 \times 192 \times 768$
- run with $\sim 1s/\Delta t$ on 16384 cores 2048 partitions
- Large data sets: $\overrightarrow{\mathbf{U}} \sim 120\,Go/\Delta t$, statistic $\sim 1\,To$

- Manipulation of very large and highly partitioned data
- Data manipulation during simulation (checkpoint data)
- Data manipulation for analysis, post-treatment and visualization
- **Parallel strategy mandatory**

Data manipulation during simulation

Data Input/Output and storage

- Large data sets: $\sim 0.2\,To\,/\Delta t$ (checkpoint data), $1\,To$ statistic:
  $\Longrightarrow$ parallel IO
- Manage the large amount of data generated (keep it simple)
  - Use of predefined parallel format (VTK) wrap in tar file

$\Longrightarrow$Optimize data transfer between platform (gridFTP)
$\Longrightarrow$Or perform co-analysis of the flow without writing flow fields

Data manipulation after simulation

Data processing

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

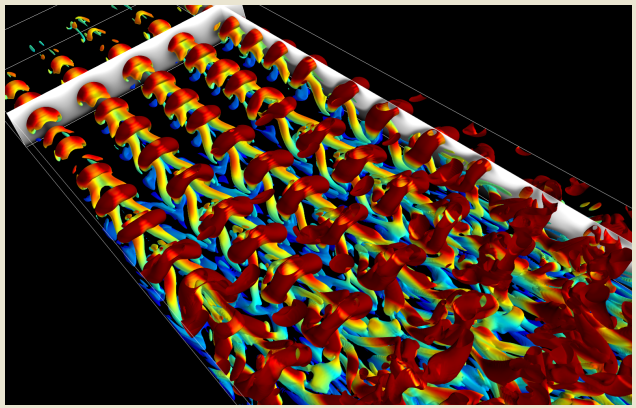
3D visualization

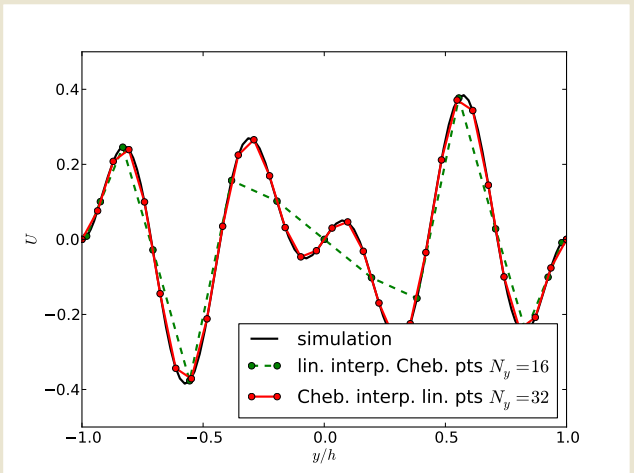
- Cannot be performed directly (or difficult) on HPC platforms

Requirements and constraints

- Entails spatial derivation, eigenvalues evaluation ...
- Preserve accuracy of the simulation
- Should be interactive and when ready on batch mode
- Must be parallel, but on a smaller scale

Need to revisit traditional usage





Work-flow with visualization tools

- Computation on remote platform
- Write data result on disk during computation
- Transfer data to local server

Limitation of current visualization

- linear interpolation between collocations points
- loose of information for problem with non-overlapping partitions
- rendering slow on non regular grid

Parallel client-server analysis tools

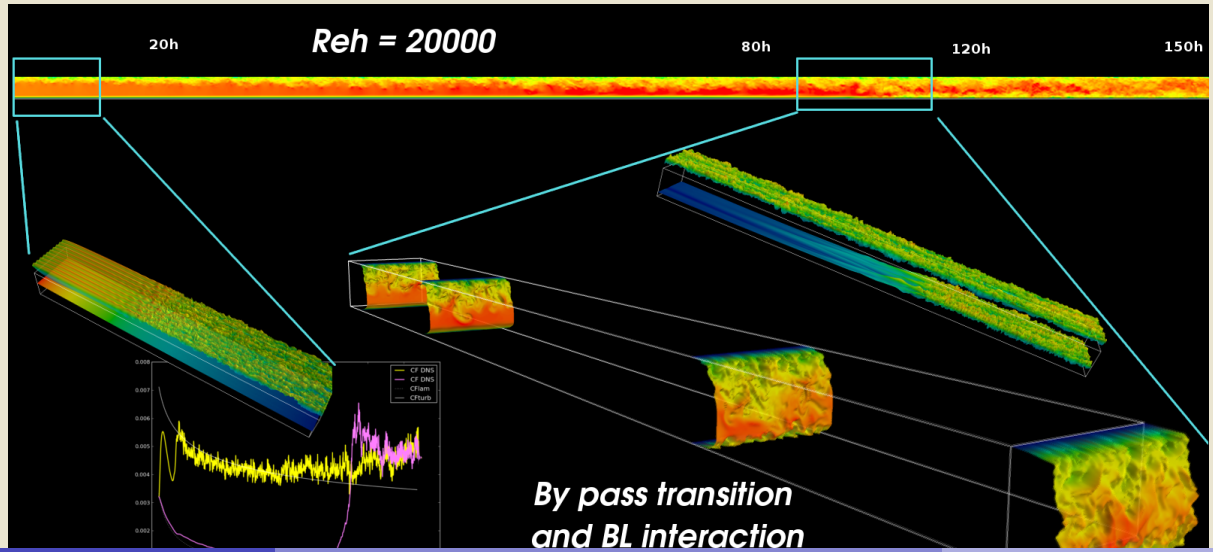
Parallel server

- automatic repartitioning
- re-sampling of the data
- spectral interpolation
- Python + NUMPY + MPI4PY + SWIG
- Python UDF

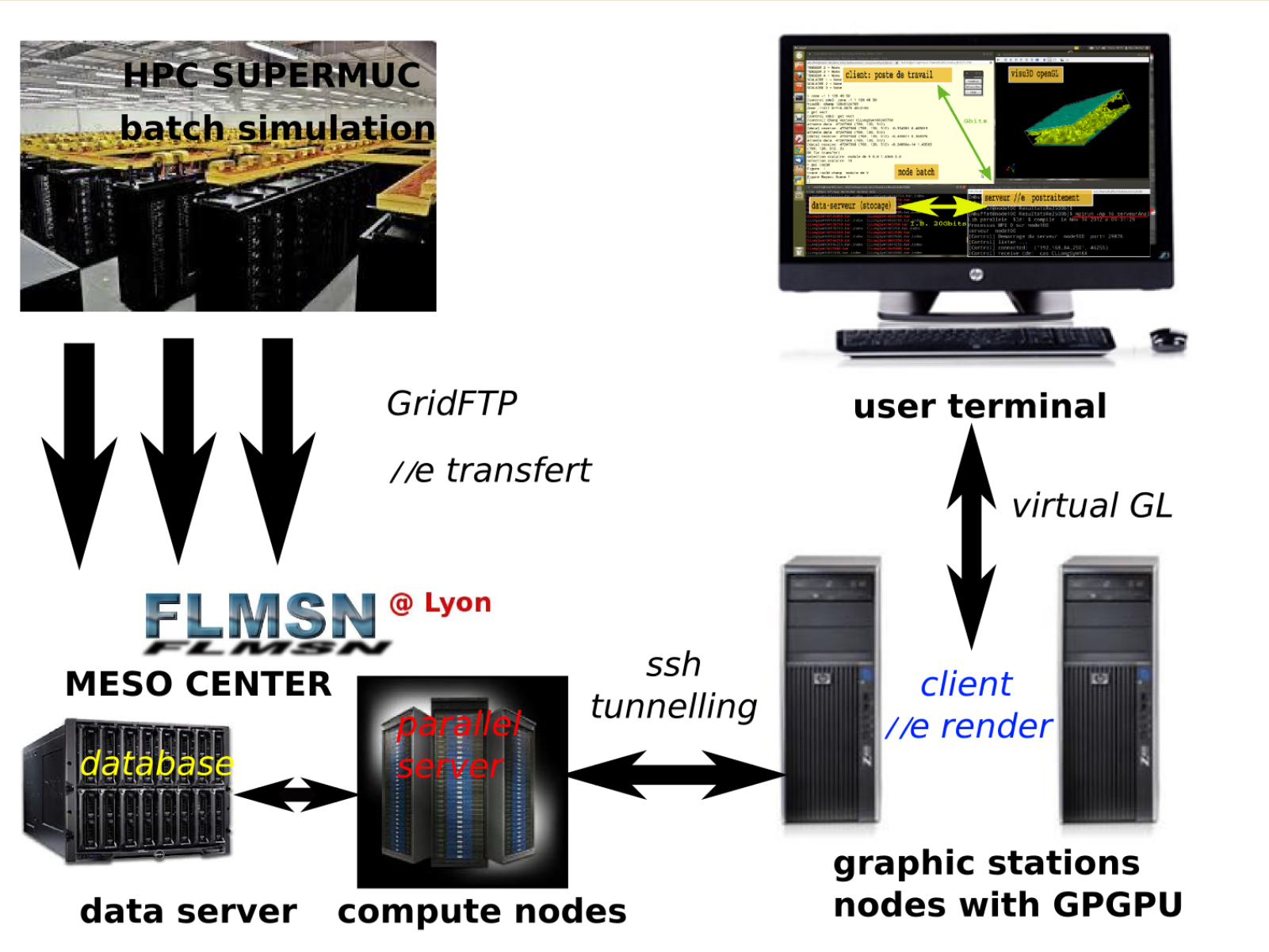
Multiple clients

1. matplotlib 1D + 2D
2. mayavi lib 3D visualization
3. VisIt 3D //e visualization

- Python + UDF + script



Work-flow for the analysis



In-situ (real time) analysis

Remote co-processing during simulation without stored data

Requirements

- Preserve spectral accuracy
- Analysis at a lower parallel scale than the simulation
- Computation of quantities from simulations variables
- Fast enough
- Act on simulation parameters (like in experiment)

Existing solution (tight coupling)

1. **VisIt (libsim)**
2. ParaView
3. Danaris (INRIA)

Limitations

- run with the same granularity as the simulation
- affect speed of computation
- assume data ready for the visualization

Hybrid in-situ analysis

code instrumentation

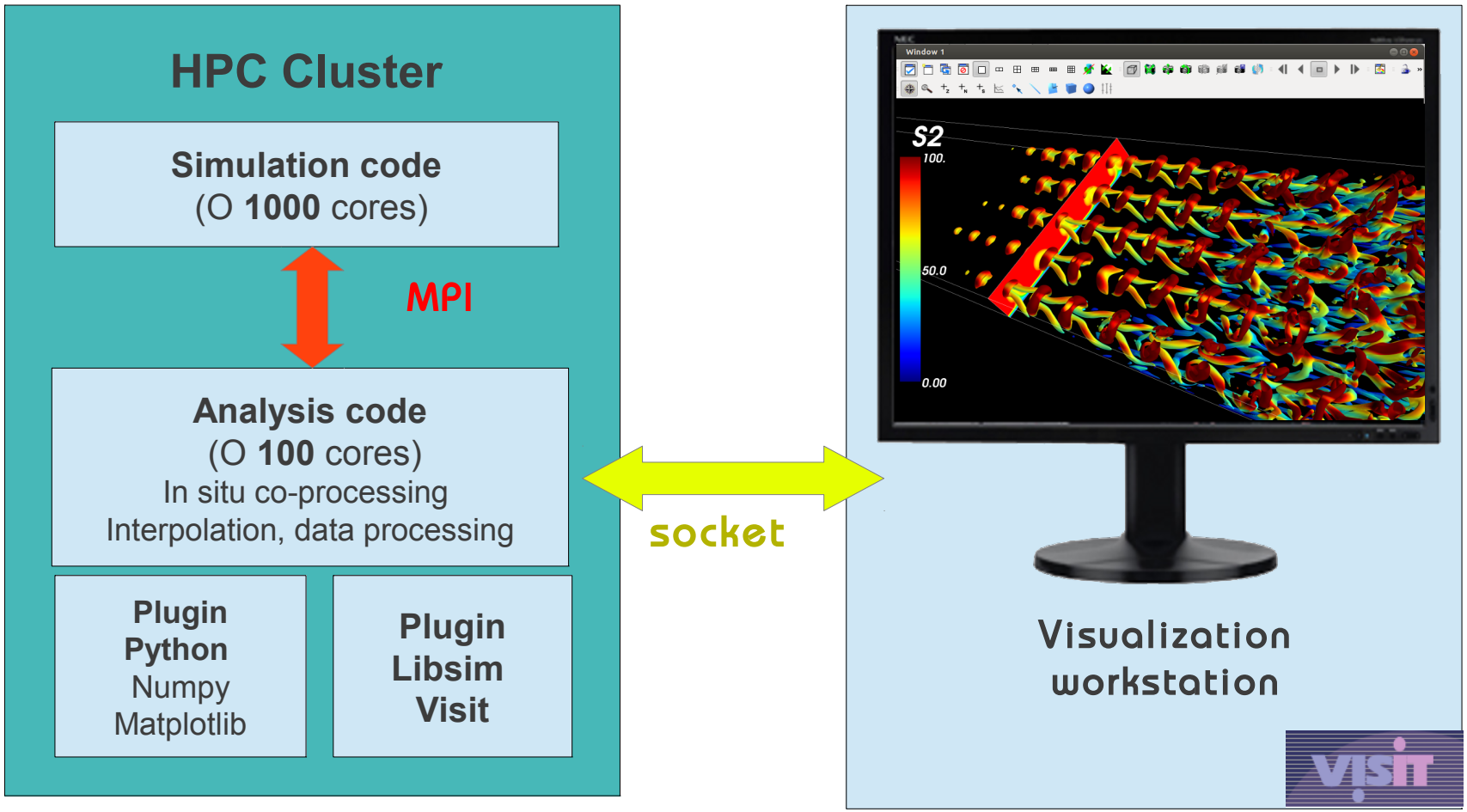
- add parallel analysis code as independant MPI process
  - use it own time step
  - interact with the simulation every 10-100 Δt
  - can use dedicated nodes
  - use a coarse and simpler domain decomposition
  - interpolatate on finer regular overlapping grid
  - can change the parameter of the simulations (control)

Interface with parallel analysis and visualisation

- python + matplolib
- Visit (libsim)
- allow interactivity and scripting

Hybrid parallel in situ analysis work-flow

In-situ visualization



HPC analyze : follow time evolution of flow structures

Explore time evolution at $Re_h = 25000$

- $5760 \times 128 \times 512$ modes ($\sim$ 380 millions of modes)
- ($L_x = 75$)

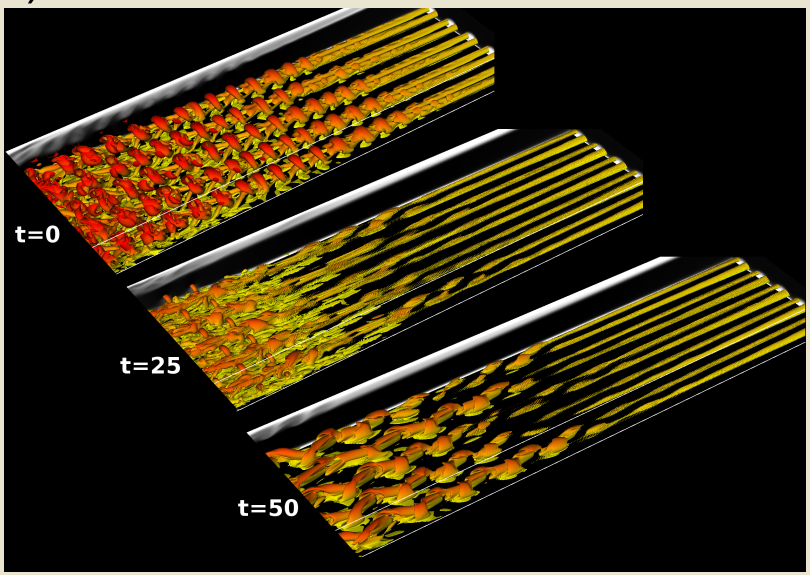
In situ analysis (embedded to the simulation)

- Run simulation on 160 nodes (128+ 32 nodes)
  - 512 MPI procs, 4 MPI processes per node + 4 threads
  - 2048 cores ($\sim$ 128 thin nodes)
  - 64 MPI procs, 2 MPI processes per node
  - 512 cores ($\sim$ 32 fat nodes)
- Analyze every 25 time steps
- Computation of Λ_2 criteria during 10 time steps
  $\Longrightarrow$ does not affect global CPU time

Generates an evolution in time with more than 3000 images with code interaction!

Visu In-situ: results and demonstration

1. Temporal evolution of turbulent transition at $Re_h = 2500$ (entrance channel flow)



2. Demonstration: visu-insitu from home

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

Thank you for your attention