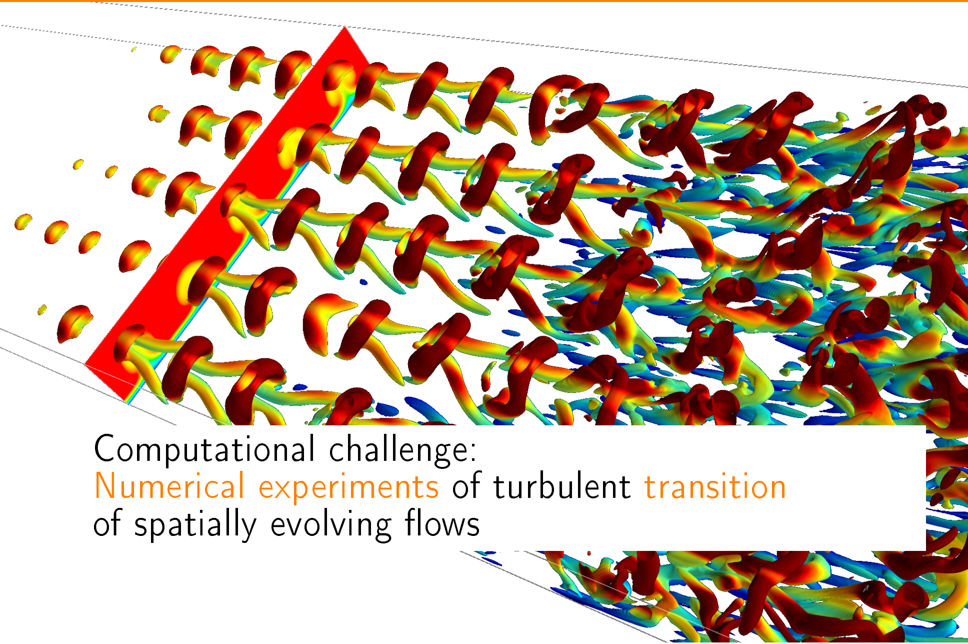
Passage à l'échelle Tier1 -> Tier0 d'une application de CFD par hybridation MPI / OpenMP

A. Cadiou, M. Buffat, L. Le Penven, J. Montagnier

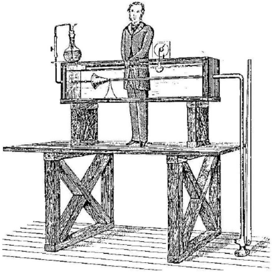
Laboratoire de Mécanique des Fluides et d'Acoustique
CNRS, Université Lyon I, École Centrale de Lyon, INSA de Lyon

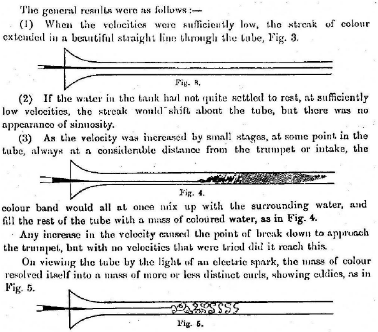
Equip@Meso, Rouen 2013
Mécanique des fluides numérique intensive : méthodes et nouvelles applications



Computational challenge:
Numerical experiments of turbulent transition
of spatially evolving flows

O. Reynolds' pipe flow experiment (1883)





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.

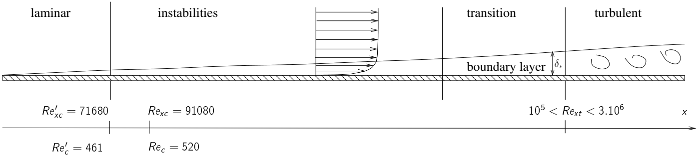
colour band would all at once mix up with the surrounding water, and fill the rest of the tube with a mass of coloured water, as in Fig. 4.

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.

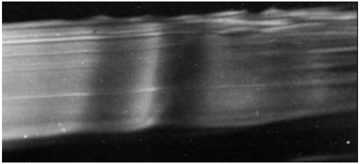
Fig. 5.

Transition in boundary layers



$$Re_x = \frac{U_\infty x}{\nu}$$

$$Re = \frac{U_\infty \delta_*}{\nu}$$



H. Werlé

- low level of perturbation ($< 1\%$)
- Tollmien-Schlichting waves (2D)

→ transition

M. Matsubara and P.H. Alfredsson

- moderate level of perturbation
- streaks (3D), Klebanoff modes

→ by-pass transition (lower Re_x)

Stability of entrance and developing channel flow

Transition at the entrance of the channel flow at high Reynolds number

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

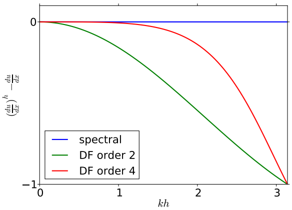
Very elongated geometry

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

Spectral approximation

- Numerical experiment: need to resolve the flow at all scales .
- As $\left(\frac{L}{\eta}\right)^3 \sim Re^{9/4} \nearrow$, increasingly stringent condition for turbulence.
- Spectral methods are attractive, due to their high spatial accuracy.

- Spatial derivatives are calculated exactly.
- Exponential convergence for smooth solutions (faster than FE, FD ...).



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

Incompressible Navier-Stokes equations

Governing equations

$$\begin{aligned} \frac{\partial U}{\partial t} + U.\nabla U &= -\nabla p + \frac{1}{Re}\Delta U \\ \nabla.U &= 0 \\ & U(t=0) = U_0 \\ & U|_{\partial\Omega} \end{aligned}$$

Galerkin formulation using an orthogonal decomposition of the velocity

$$U = U_{OS}(U.e_y) + U_{SQ}((\nabla \times U).e_y)$$

spectral approximation

$$U(t,x,y,z) = \sum_i \hat{U}_i(t)\alpha_i(x,y,z)$$

Numerical method

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

Resolution of coupled systems for non-linear 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

Challenge: from 100 to 10000 cores

Example of configuration: computational domain size $280 \times 2 \times 6.4$

- $34560 \times 192 \times 768$ modes ($\sim$ 5. billions of modes)
- travel 1 length with it=600000 iterations. ($\sim$ 16 millions of FFT)

Memory constraint

- $N = N_x \times N_y \times N_z$, with N very large
- large memory requirement ($\sim 2\,To$)
- BlueGene/P 0.5 Go per core $\Rightarrow$ $\sim$ 4000 cores needed
- $N_x >> N_y, N_z$, elongated in one direction
- 1D domain decomposition $\Rightarrow$ limited to $\sim$ 100 cores
- can only simulate a 40 times shorter channel length

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

Outline

Implementation on HPC platforms

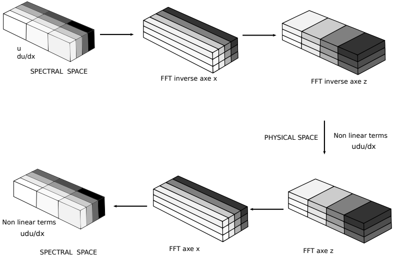
- MPI strategy to scale from $O(100)$ to $O(10\,000)$ core
- Hybrid strategy to migrate on many-core platform
- Additional constraint for optimization
- Data manipulation during simulation
- Data manipulation for analysis and post-treatment

Outline

Implementation on HPC platforms

- MPI strategy to scale from $O(100)$ to $O(10\,000)$ core
- Hybrid strategy to migrate on many-core platform
- Additional constraint for optimization
- Data manipulation during simulation
- Data manipulation for analysis and post-treatment

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

Outline

Implementation on HPC platforms

- MPI strategy to scale from $O(100)$ to $O(10\,000)$ core
- Hybrid strategy to migrate on many-core platform
- Additional constraint for optimization
- Data manipulation during simulation
- Data manipulation for analysis and post-treatment

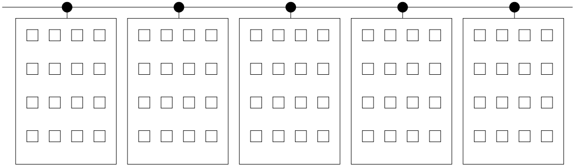
Constraints related to modern many-cores platforms

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



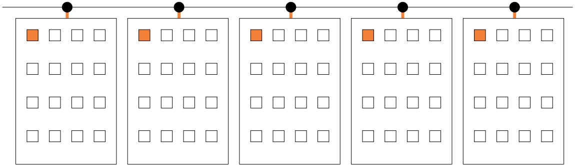
Constraints related to modern many-cores platforms

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



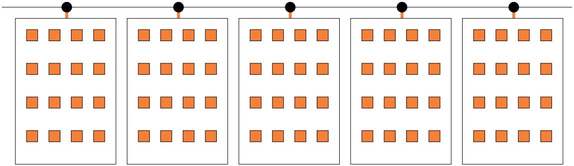
Constraints related to modern many-cores platforms

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



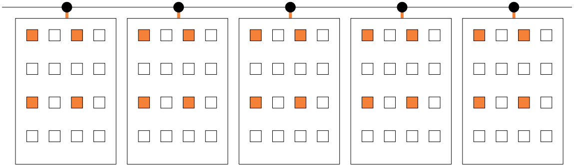
Constraints related to modern many-cores platforms

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



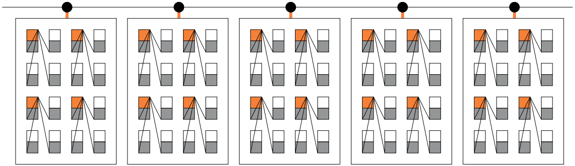
Constraints related to modern many-cores platforms

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.

Outline

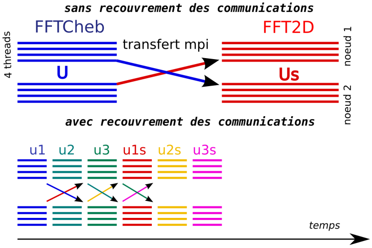
Implementation on HPC platforms

- MPI strategy to scale from $O(100)$ to $O(10\,000)$ core
- Hybrid strategy to migrate on many-core platform
- Additional constraint for optimization
- Data manipulation during simulation
- Data manipulation for analysis and post-treatment

More than domain decomposition ...

Tasks parallelization : mask communications by execution time

- reduces by 20% time per iteration
- less loss in communications - waist ∼ 10%



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

Outline

Implementation on HPC platforms

- MPI strategy to scale from $O(100)$ to $O(10\,000)$ core
- Hybrid strategy to migrate on many-core platform
- Additional constraint for optimization
- Data manipulation during simulation
- Data manipulation for analysis and post-treatment

Problems related to the very large calculations
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

HPC simulations require every layer of HPC ressources

Tier-0, PRACE

1. JUGENE and JUQUEEN, Jülich, Germany
2. CURIE, Bruyères-le-Châtel, France
3. SuperMUC, Garching, Germany

Tier-1, GENCI

1. IDRIS, Orsay
2. CINES, Montpellier
3. TGCC, Bruyères-le-Châtel

Tier-2, Fédération Lyonnaise de Modélisation et Sciences Numériques

P2CHPD, la Doua
Many thanks to Christophe Péra

Problems related to the very large calculations
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 - batch mode

Requirements and constraints

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

$\Rightarrow$ Should be done locally, i.e. implies data transfer and storage

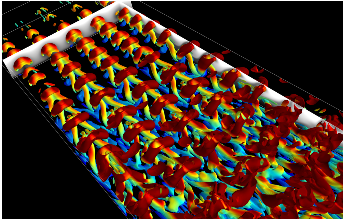
$\Rightarrow$ Must be performed from remote access

$\Rightarrow$ Must be parallel, but on a smaller scale

Example

Simulation (multi-run batch) on LRZ SUPERMUC
$\sim$ 5 billions of modes
$34560 \times 192 \times 768$
run with $\sim 1s/dt$ on 16 384 cores
2048 partitions

Analysis of the Big Data



in-situ analysis not possible (batch)
$\Longrightarrow$transfert of data
$\Longrightarrow$parallel analysis mandatory
$\Longrightarrow$script mode mandatory (reproductible)

Software review

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

Limitations

- linear interpolation
- no repartitioning of the data
- no resampling of the data
- no zonal analysis

Parallel client-server analysis tools

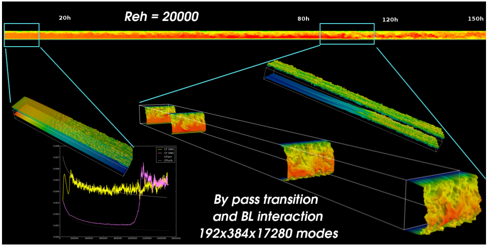
Parallel server

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

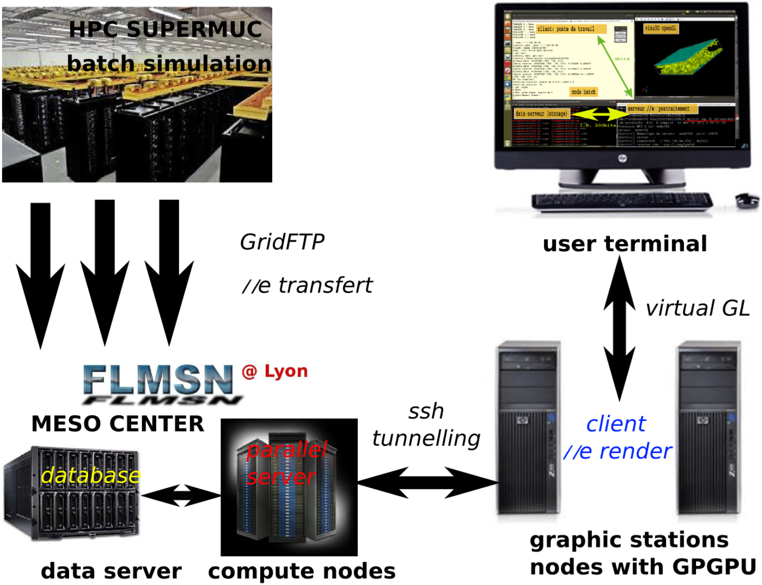
Multiple clients

1. matplotlib 1D + 2D
2. mayavi lib 3D visualization
3. VisIt 3D //e visualization (using libsim, i.e. wo file)

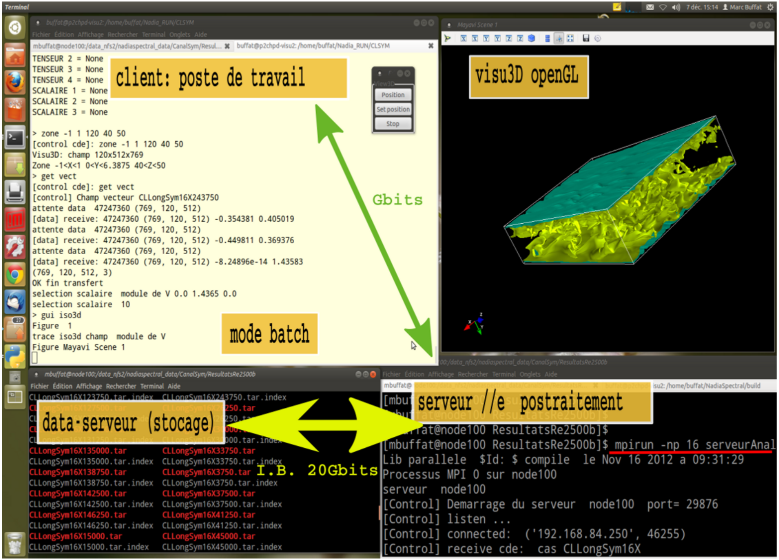
- Python + UDF + script
- TCP connection



Workflow for the analysis



Client screen



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

Resulting method

Characterictics

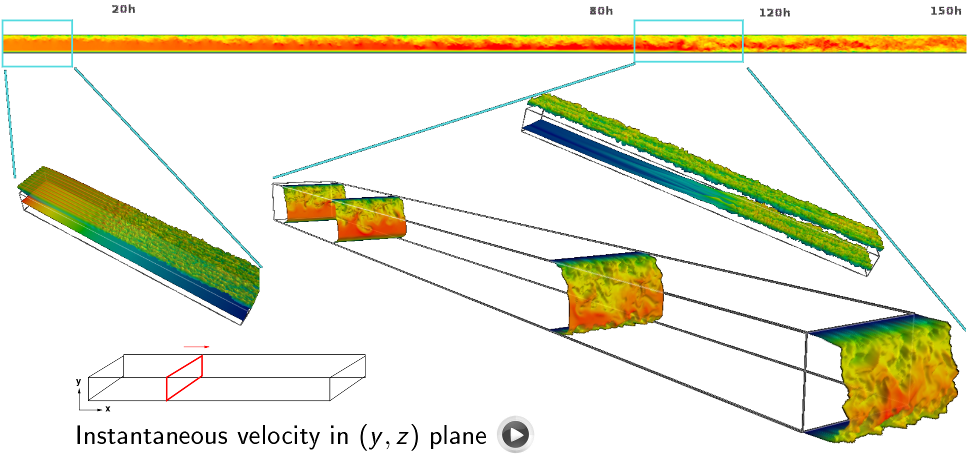
- Efficient solver for hybrid multicore massively parallel platforms
  - Original coarse grained MPI/OpenMP strategy
  - Tasks overlapping
- Pre- and post- processing tools for smaller MPI platforms
  - parallel VTK format (paraview)
  - Parallel Client/Server programs in Python calling a spectral library
  - 2D/3D parallel visualization - (matplotlib/mayavi/visit)

Properties

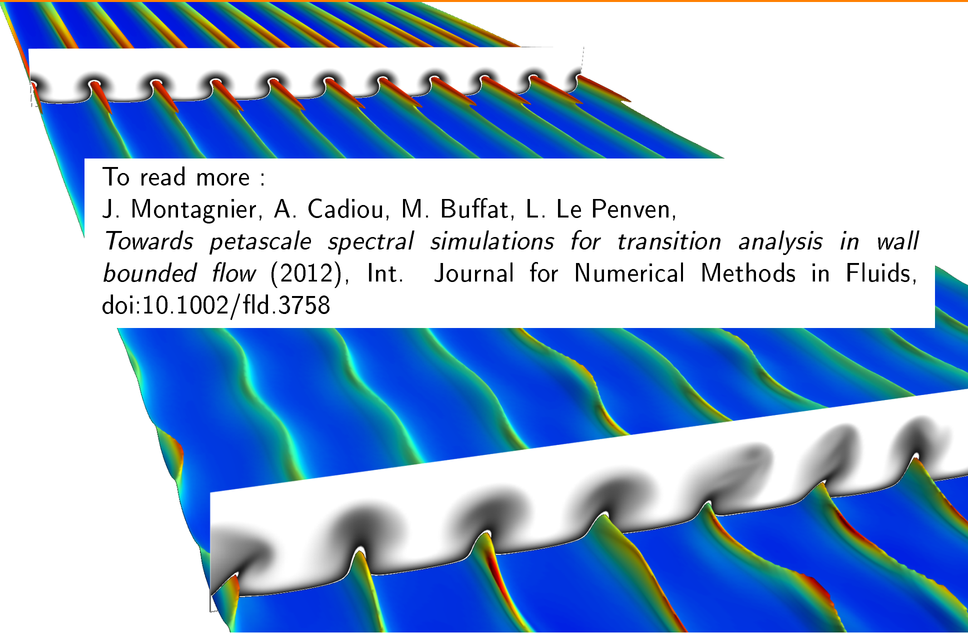
- Fairly portable
- Small time spent in communications $\sim 10\%$
- Rapid wall clock time for a global scheme
  (1 billion of modes: 1.3s/it on BlueGene/P - 0.2s/it on SuperMUC)

DNS of turbulent transition in channel entrance flow

First transition Second transition



Instantaneous velocity in (y, z) plane



To read more :

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, doi:10.1002/fld.3758

2D Parallel strategy - illustration

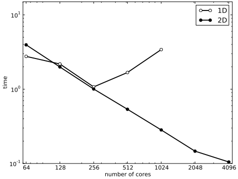


Figure: Time per iteration for a $1024 \times 256 \times 256$ case.

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

OpenMP

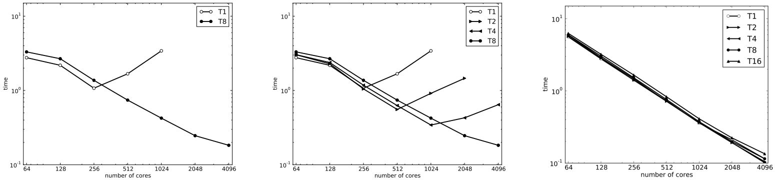
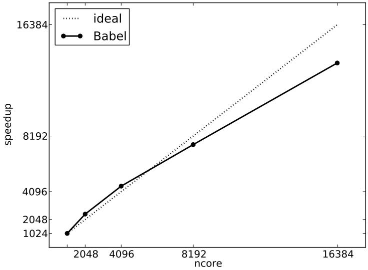


Figure: Time per iteration for a $1024 \times 256 \times 256$ case.

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

Speedup and efficiency



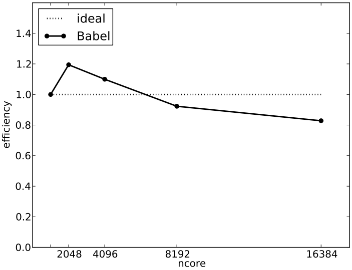


Figure: $4096 \times 512 \times 512 \sim 10^9$ modes

Decent wall clock time :
10^9 modes : 0.9 s/iteration for 16384 cores

Communications

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

Hybrid MPI/OpenMP

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.