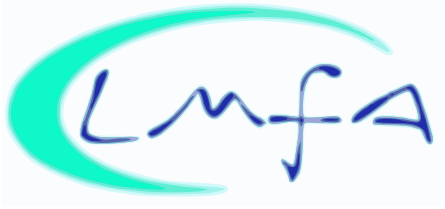
Calcul intensif pour l'étude de la transition turbulente en canal plan,

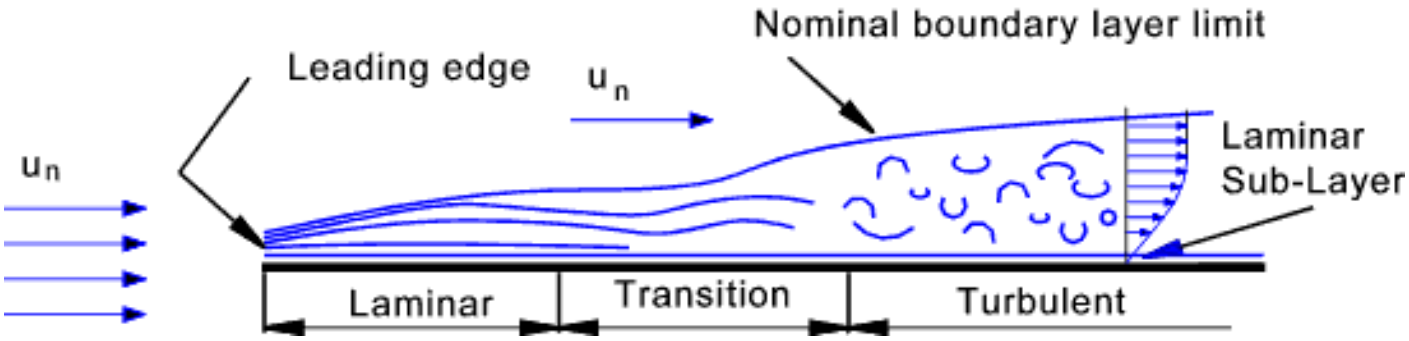
J. Montagnier[1], M. Buffat[1], L. Le Penven[1], A. Cadiou[1]

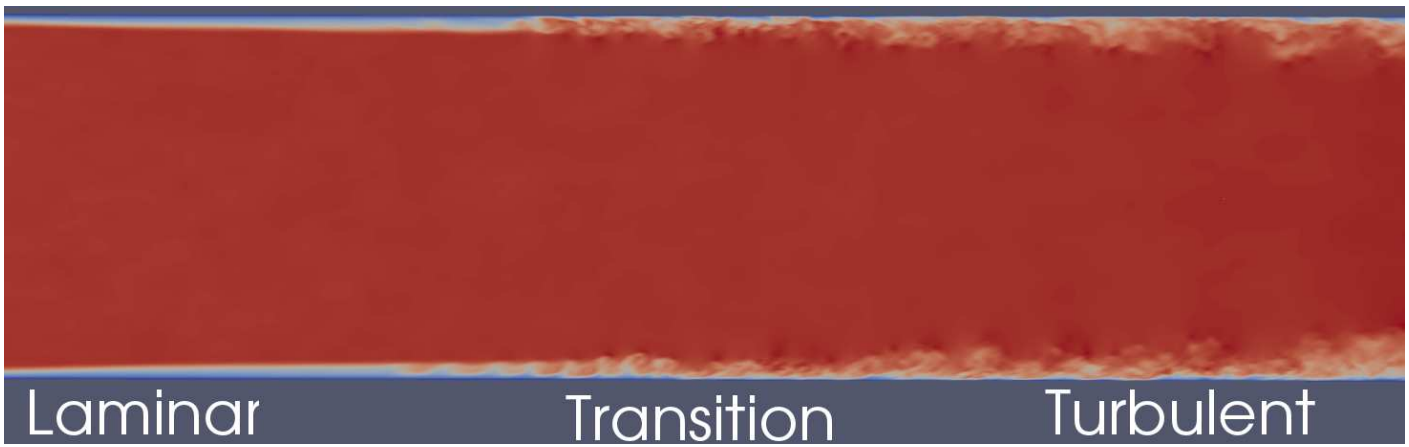
[1]Université de Lyon, Université Lyon I, CNRS, ECL, INSA Lyon

CIRA 2011



Boundary Layer transition





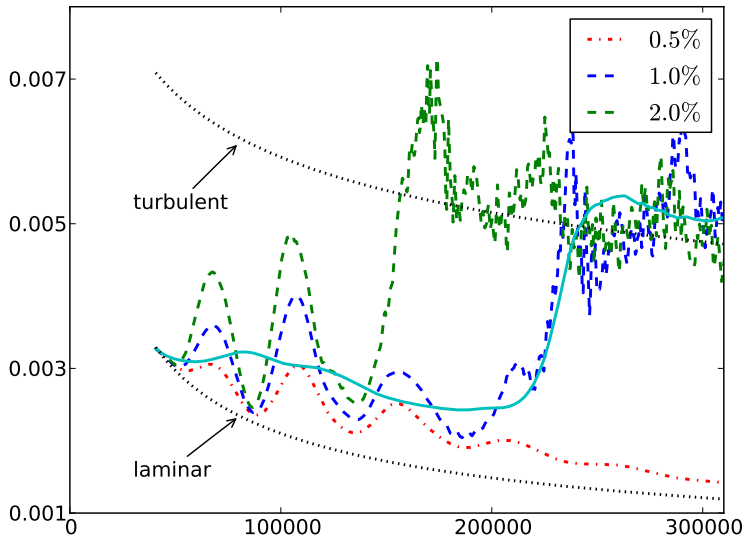


Figure: Instataneous C_f versus Re_x for different levels of perturbations

- Increase of drag (aerodynamic of airplanes / car / gas turbine blades...) and pressure fluctuations (cavitation, ...)

Complex problem / classical stability theory fails

By pass transition induced by Free Stream Turbulence

By Pass transition

Free stream perturbation

▼

Non modal growth of BL streaks

▼

Secondary Instability of streaks

▼

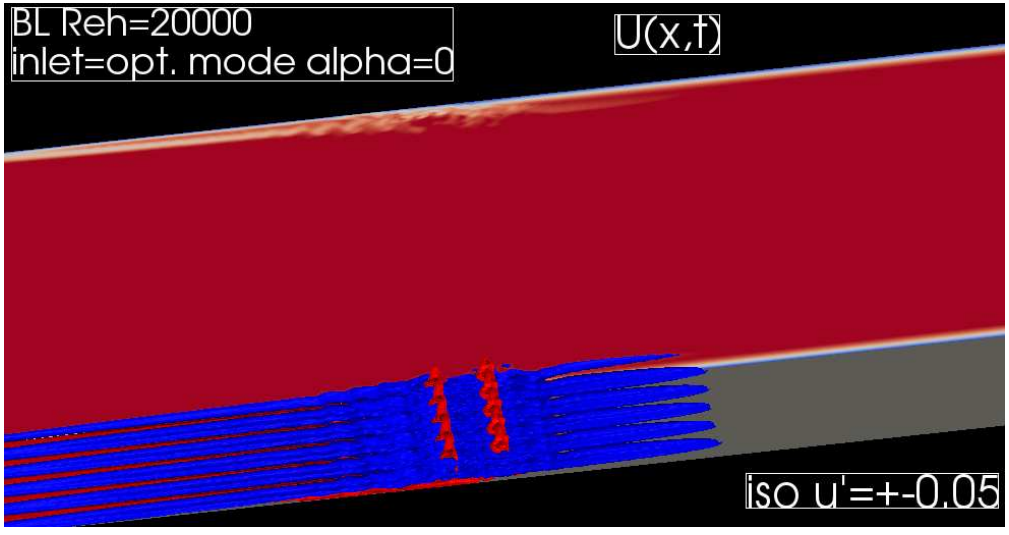
Turbulent spots

▼

Turbulence

- Zaki & Durbin JFM 2005, Durbin & Wu Annual Review 2007, Schlatter et al. POF 2008

Study Objectives



Stability analysis

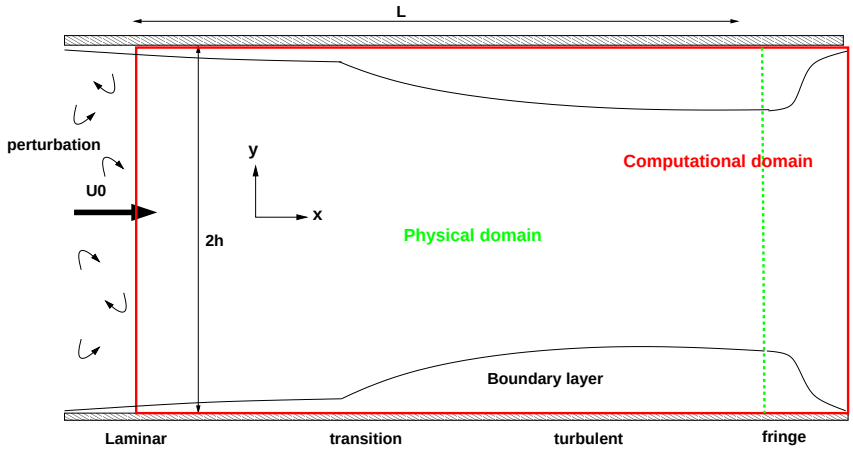
- Better understanding of by-pass transition
- Study varicose and sinuous instabilities

Numerical experiments

- Well controlled perturbations (linear stability)
- Parametric studies

Numerical setup

Channel flow with thin boundary layers $Re_h = 20000$



Requirements

- high accuracy $\rightarrowtail$ DNS with spectral method
- Fourier in (x, z) and Chebyshev in y
- $\delta_0 \ll h \ll L$
- parametric studies with moderate resolutions ($\sim 10^7$ modes)
- high resolutions simulations ($\sim 10^9$ modes)

NadiaSpectral code

Orthogonal decomposition of the velocity

$$\boldsymbol{U} = \boldsymbol{U}_{os}(v) + \boldsymbol{U}_{sq}(\omega) \quad \text{with } \boldsymbol{U}_{os} \perp_{L_2} \boldsymbol{U}_{sq}, \quad \nabla.\boldsymbol{U}_{os} = \nabla.\boldsymbol{U}_{sq} = 0$$

- Optimal representation of a solenoidal velocity field
- Elimination of the pressure
- Spectral accurate Galerkin formulation using a Fourier-Chebyshev approximation

- **M. Buffat, L. Le Penven, A. Cadiou** : "An efficient spectral method based on an orthogonal decomposition of the velocity for transition analysis in wall bounded flow", Computer & Fluids, 2011

Orthogonal decomposition

Fourier approximation in (x, z), Chebyshev in y

- Orthogonal decomposition of the velocity fields with *NxMxP* modes

$$\mathbf{U}(x,y,z,t) = \sum_{m=-M/2}^{M/2} \sum_{p=-P/2}^{P/2} \left[\sum_{j=0}^{N-1} \alpha_{OS,j}^{mp} \mathbf{u}_{OS,j}^{mp} + \sum_{j=0}^{N-1} \alpha_{SQ,j}^{mp} \mathbf{u}_{SQ,j}^{mp} \right]$$

- $2MxP$ differential equations **with sparse matrices** (7D)
- time integration with **Crank Nicholson/Adams Bashford** scheme

Problématique de la parallélisation

- A chaque pas en temps :

$$A\alpha_i^{mp} = b$$

- Matrice A calculée dans l'espace spectrale.

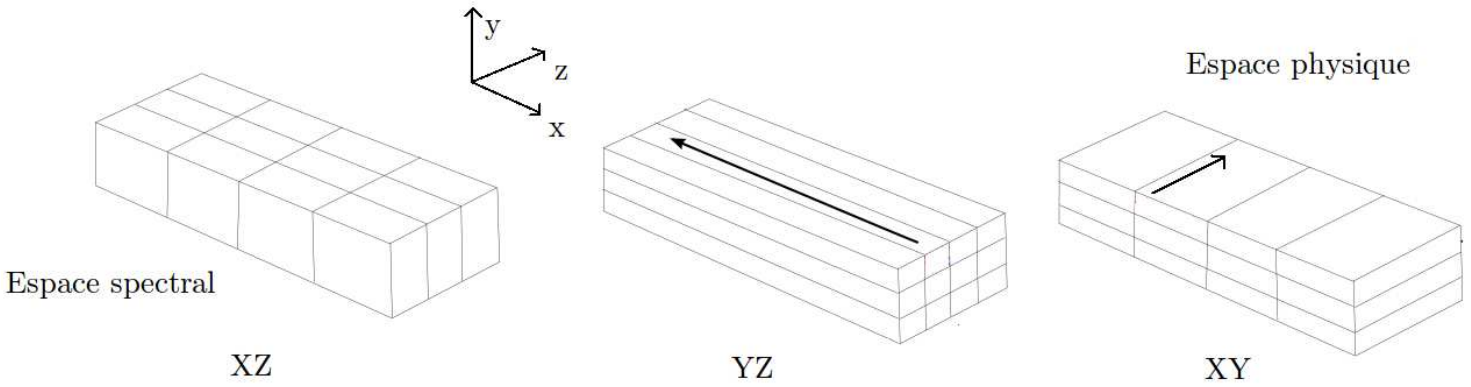
Difficulté : calcul du 2nd membre b (non linéaires)

- Produit de convolution entre coefficients spectraux : couplage entre tout les α_j^{mp}
- Nécessité d'utiliser des FFT pour passage dans l'espace physique (FFT Cheb axe y, FFT 1D axe x et z))
- Bibliothèques classiques // de calcul des FFT multidimensionnelles non adaptées ($M \gg P, N$)
- Transposition de données pour calculer efficacement les FFT multidimensionnelles

Architecture hybride des calculateurs HPC

- Objectif : $o(10\,000)$ coeurs

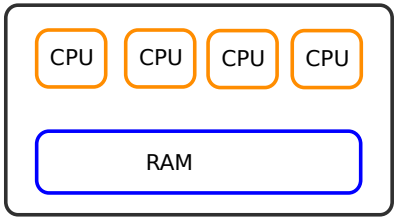
Partitionnement : localisation des données pour M>> N,P



- Partitionnnement XZ : calcul de A et résolution du système linéaire, Transformée Chebyshev
- Partitionnement YZ : FFT selon l'axe x
- Partitionnement XY : FFT selon l'axe y

Multithreading

- Multithreading : mieux utiliser les architectures actuelles
- Gain : réduction du nombre de processus MPI
  - Diminution du nombre de communications
  - Gain en mémoire (BLUEGENE)
- Approche classique : Boucle *#pragma omp for (int i = 0 ; i < n ;i++)*
- Problème **temps de création des threads important** pour des boucles rapides ($20\mu s$).
- Approche multi-threading haut niveaux



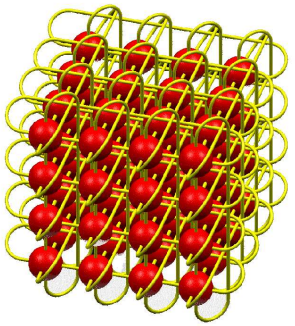
Optimisation du code

- Reécriture du calcul des fft2D à partir du partitionement introduit et des routines de calcul fft1D (provenant de la librairie fftw3).
- **Division par 2 des communications** en calculant les termes non linéaires dans l'espace physique partitionné différement de l'espace spectral.
- **Arrangement des données optimales** pour chaque partitionnement (JKI pour XZ ....) pour une meilleure utilisation du cache.
- Multithreading haut niveaux
- Structure du code qui permet une certaine souplesse dans le placement des processeurs (mapping)

Influence du mapping

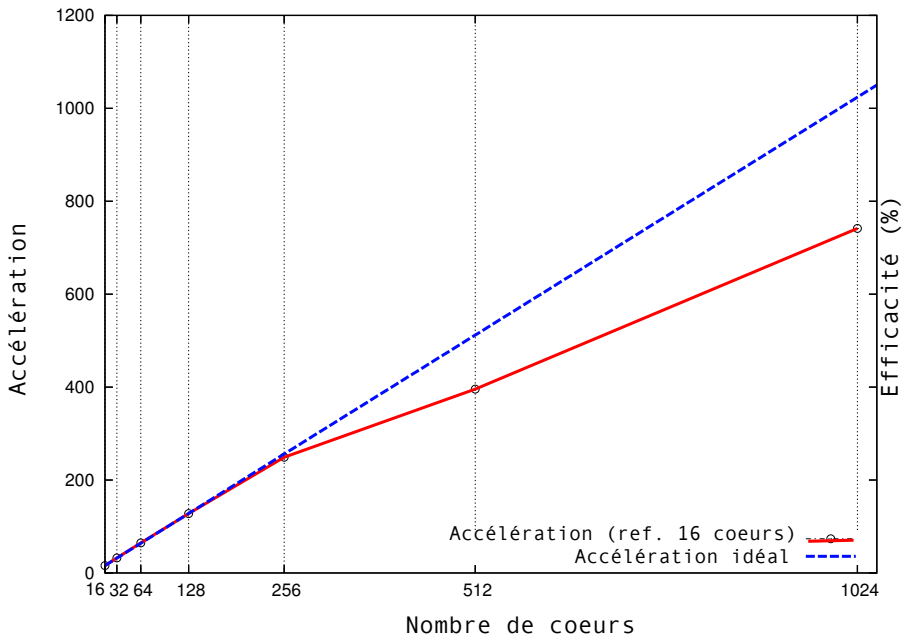
Mapping : comment placer les processeurs sur la machine de manière optimales ?

- Impact considérable sur les performances
- Réalisation de tests de placement et comparaison des performances
- Variation des performances jusqu'à 50 % sur 8096 coeurs sur BLUE GENE



Efficacité parallélisation

- Test de speed up et scale up
- Speed up :
  - Taille du problème données, variation du nombre de coeurs
  - Speed Up : $\frac{temp(nproc_{ref})}{temps(nproc)}$
  - Efficacité speed up : $\frac{speeed\ up}{speed\ up\ idéal}$
- Calcul Méso centre pour études paramétriques



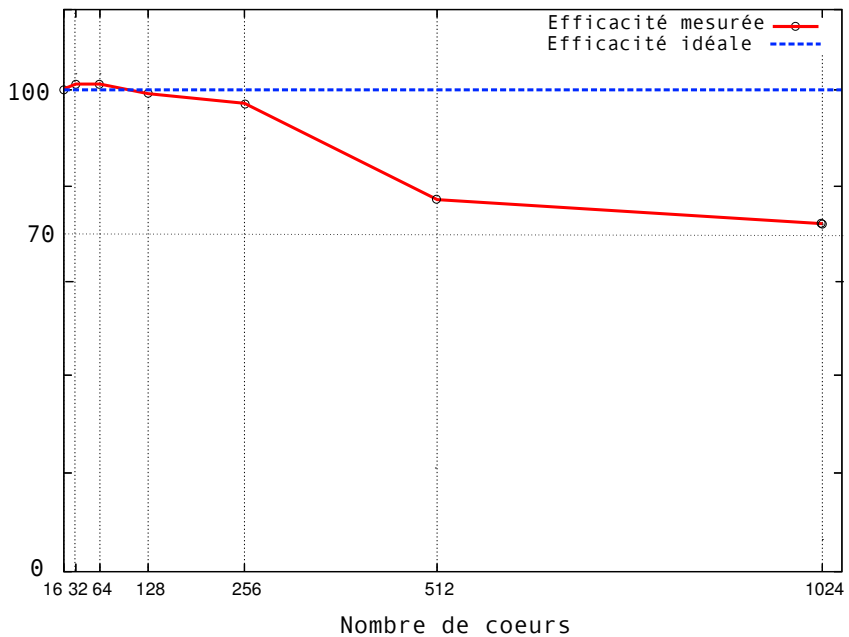


Figure: $192 \times 96 \times 768 = 14 \times 10^6$ modes

Efficacité parallélisation

- Calcul très précis sur centre nationaux (IDRIS) BLUE GENE (40 % des 40 960 coeurs)

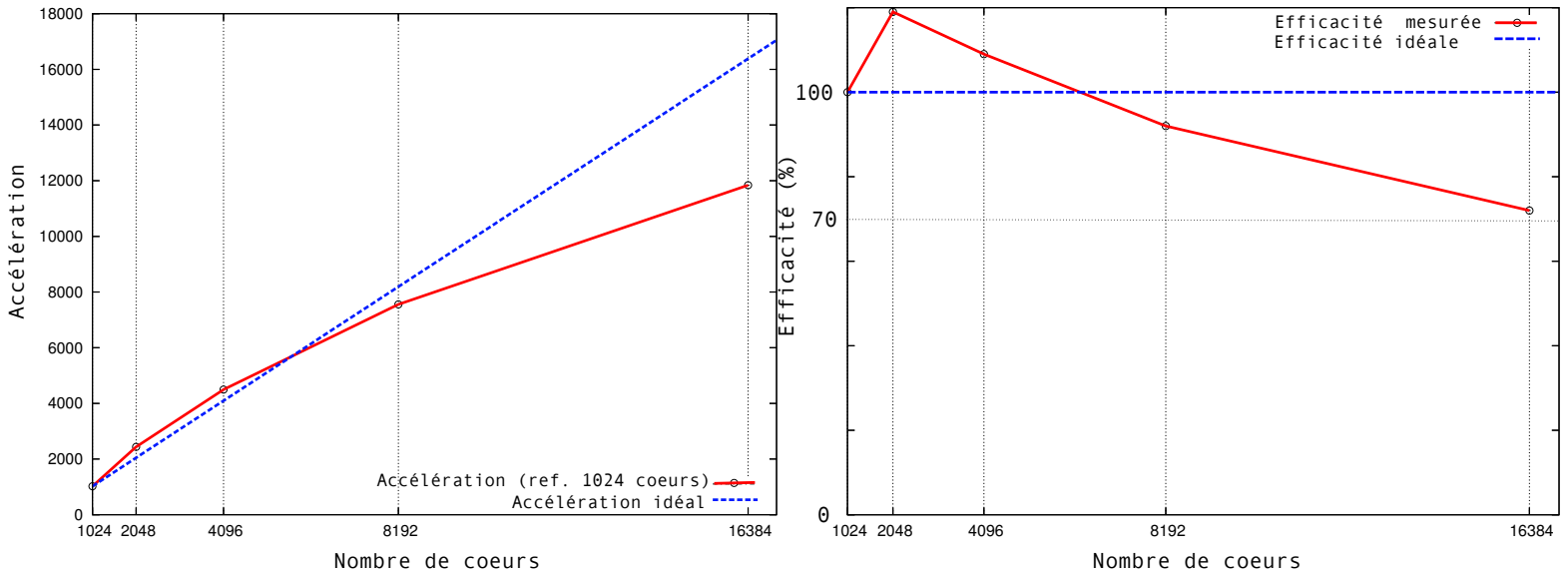
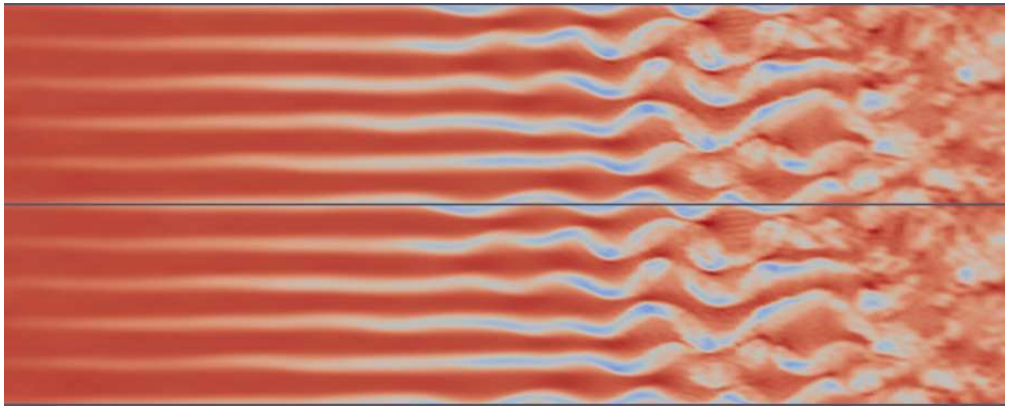


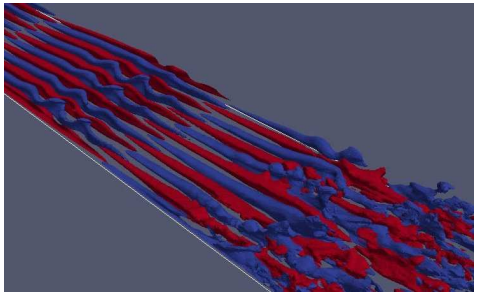
Figure: $4096 \times 512 \times 512 \approx 10^9$ modes

- 14×10^6 modes : 0.2 sec / iterations pour 1024 coeurs (13 824 modes / coeurs)
- 10^9 modes : 0.9 sec / iterations pour 16 384 coeurs (65 536 modes / coeurs)

Some numerical results on by-pass transition



Video: U 3D view



- **Video: U top view**

Conclusion

Ancienne version :

- Utilisation inefficace des multicoeurs.
- Innefficacité parallèle $>o(100)$ (limitée du nombre de processeurs par le nombre de points en x et y)

Nouvelle version :

- Augmentation du multithreading
- Diminution d'un facteur 2 du nombre de communications.
- Efficacité $o(10000)$ (calcul sur 16 384 coeurs $\approx$ 40% BLUE GENE)

Outils performant pour l'étude de la transition de la couche limite

Details Multithreading

⇒ Approche multi-threading haut niveaux :

```
#pragma omp for (int t =0 ; t < nthread ; t++)
{
...
    // Calcul termes d²/dy²
    int imin = t*n/nthread ;
    int imax = (t+1)*n/nthread ;
    for (int i = imin ;i<imax ;i++)
    {...
    }
    // Calcul fft axe y
    int imin = t*nfft_y/nthread ;
    int imax = (t+1)*nfft_y/nthread ;
    for (int i = imin ;i<imax ;i++)
    {...
    }
...
}
```

```
...
  // Calcul termes d²/dy²
  #pragma omp for (int i=0 ; i < n ; i++)
  {

  }
  // Calcul fft axe y
  #pragma omp for (int i=0 ; i < nfft_y ; i++)
  {

  }
...
```

- Problèmes rencontrés :
  - Directive de reduction (maximum et minimum) non supporté pour certaines implémentations d'openmp
  - Certaines implémentations du code interdise l'utilisation de barrier de synchronisation de thread → recodage d'une barrier (#pragma omp barrier)