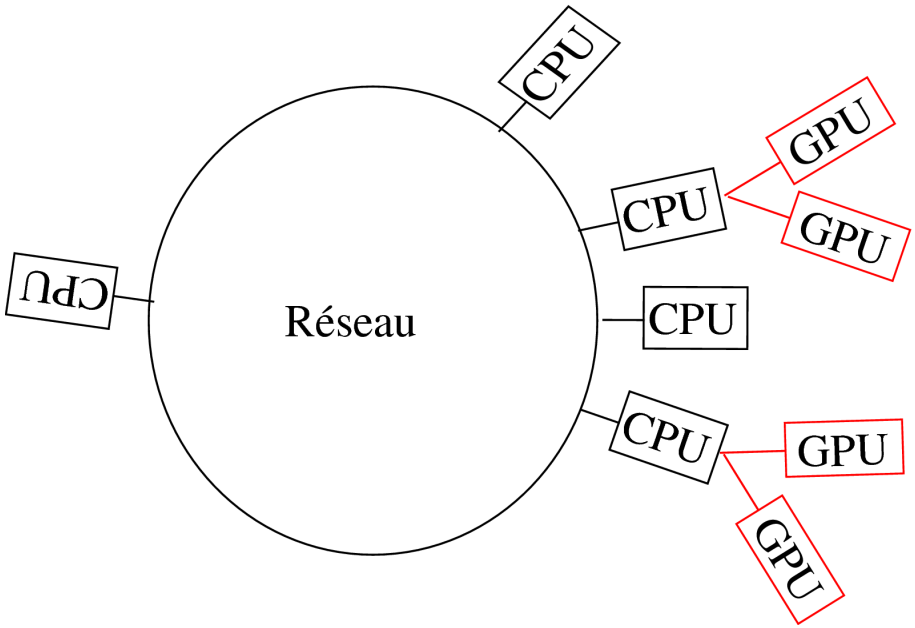
Exploitation et partage de GPU dans des grappes de calcul hybrides

Matthieu Ospici (BULL - LIG/MESCAL - CEA/L_SIM)

Luigi Genovese (ESRF)
Thierry Deutsh (CEA/INAC/L_SIM)

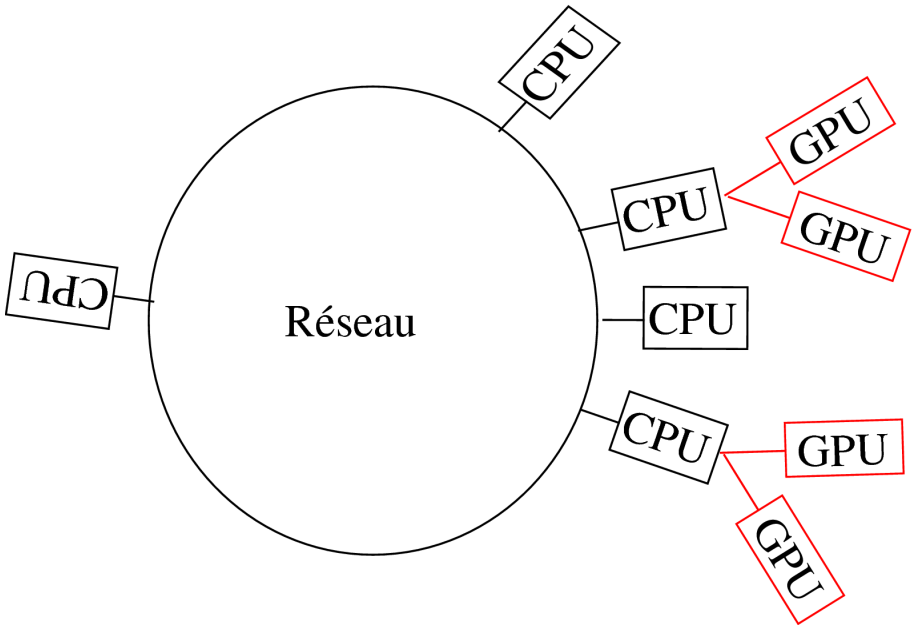
BULL Échirolles
Laboratoire d'informatique de Grenoble - INRIA – MESCAL
Laboratoire de simulation atomistique (L_SIM)
European Synchrotron Radiation Facility (ESRF)

Contexte technologique et scientifique



- Grappe de calcul hybride (GPU) et hétérogène
- Simulation scientifique et calcul haute performance

Contexte



- Nombre de GPUs et de cœurs CPU différent
- Répartition des données entre les CPUs et les GPUs

- Problématique
  - Accès concurrents de processus, placés sur différents cœurs, aux GPUs
- Contribution
  - Gestion et partage de GPU
  - Bibliothèque logicielle S_GPU

Expérimentations effectuées sur le code de simulation pour les nanosciences **BigDFT**

Plan de l'exposé

- Exploitation des grappes de calcul hybrides
- Présentation de S_GPU
- Description de BigDFT
- Évaluation des performances

Exploitation des grappes hybrides

GPU NVIDIA

- Programmation avec CUDA
- Espaces mémoire CPU / GPU disjoints
  - transfert mémoire
- Possibilité de recouvrir les transferts par des calculs (GT 200)

Applications utilisant les GPUs

- Répartition statique des données entre les GPUs et les CPUs (Linpack de NVIDIA)
- Utilisation de grappes hybrides homogènes (FEAST, NAMD)

→ Décision d'évaluer une stratégie où un groupe de cœurs CPU partage des GPUs (S_GPU).

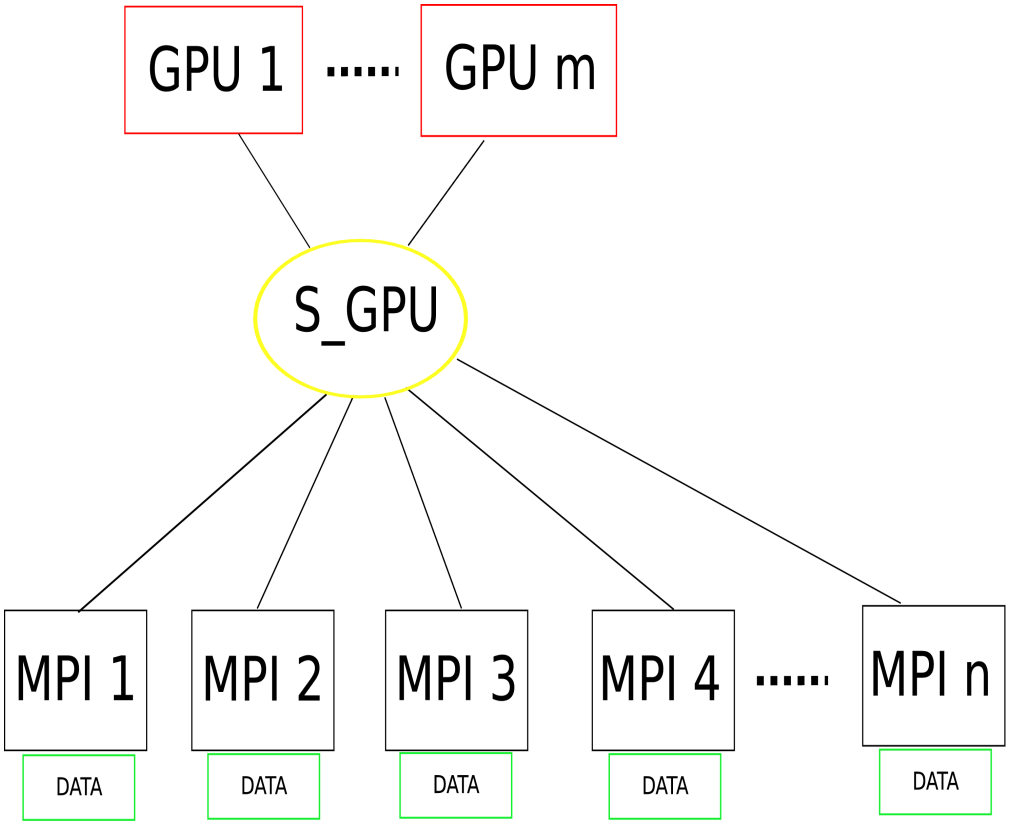
– Les cœurs CPU possèdent la même quantité de données

S_GPU

S_GPU

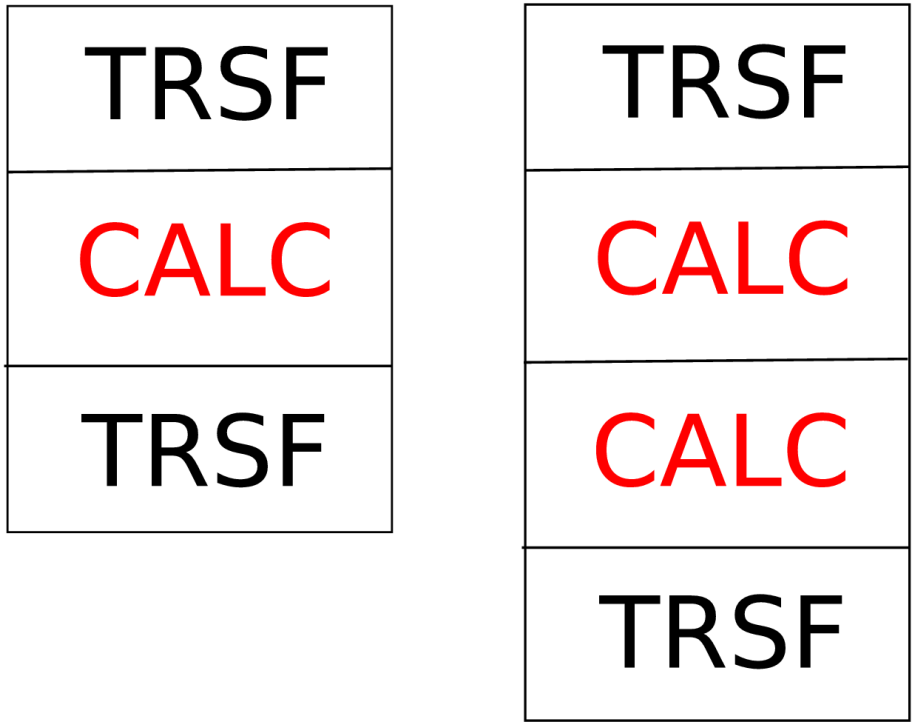
- Partage de plusieurs GPUs NVIDIA avec plusieurs processus (MPI) placés sur des cœurs CPU
- Recouvrement calcul / transfert mémoire sur GT200
- Codé en C++, CUDA. Possibilité d'appel depuis Fortran

S_GPU



- Initialisation : les GPUs sont attribués aux processus
- Utilisation :
  - Création de flots
  - Exécutions des flots

Flots



- Chaque flot contient une liste de dépendances
- La fonction `sg_launch_all_streams()` exécute les flots

Exemple

```
1       call sg_create_stream(stream_ptr)
2
3       do iorb=1,num_orbs
4          call sg_mem_copy(...,
5                           stream_ptr)
6          call sg_gpu_send(...,
7                           stream_ptr)
8       end do
9       call calcul(...,
10                  stream_ptr)
11      call sg_gpu_recv(...,
12                          stream_ptr)
13      call sg_mem_copy(...,
14                          stream_ptr)
15
16      call sg_launch_all_streams()
```

Création du flot

Copie en mémoire "pinnée"

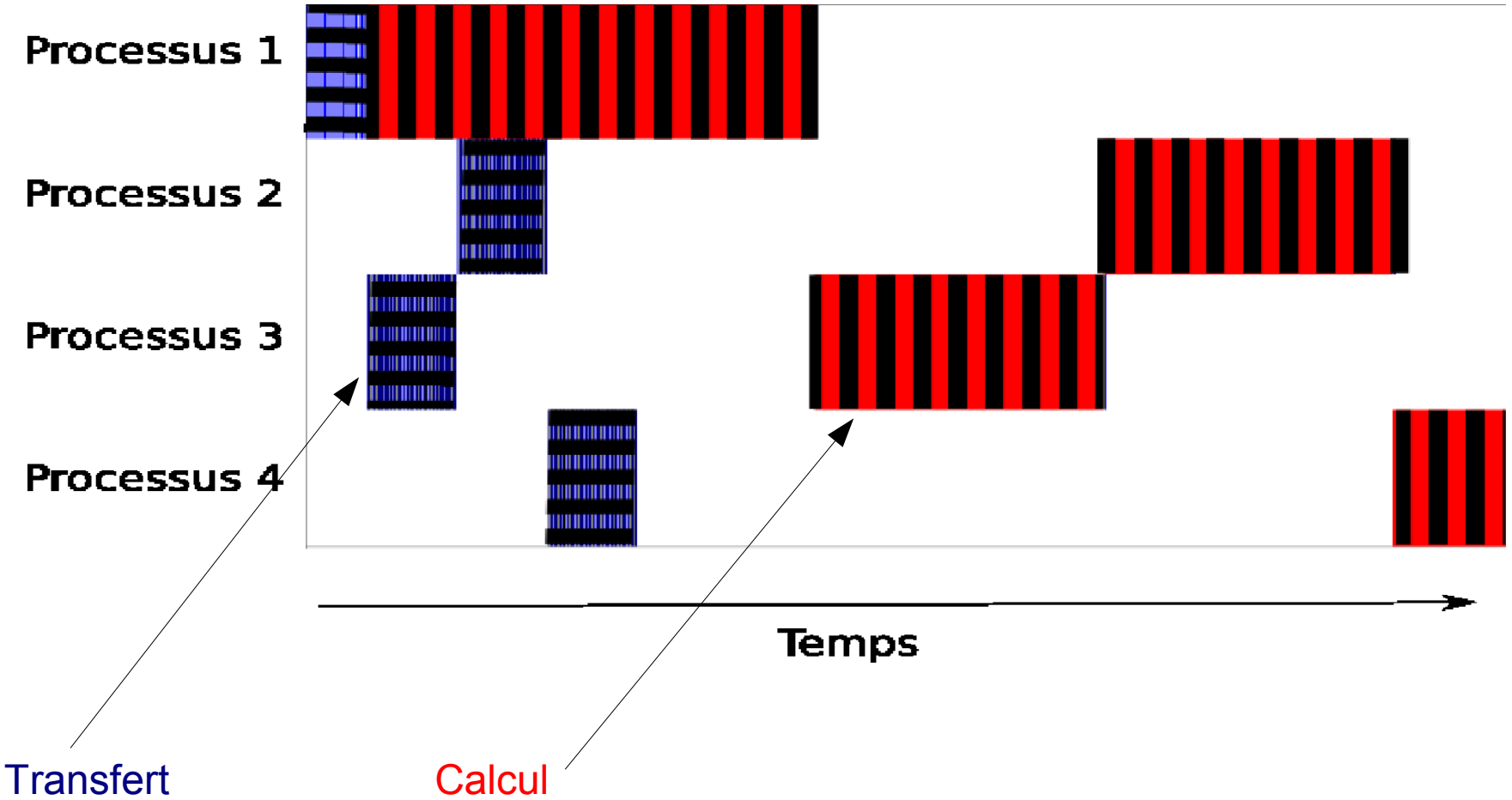
Copie vers le GPU

Exécution

Mise en œuvre de S_GPU

- Utilisation de deux sémaphores par carte
  - Le premier synchronise les transferts, le second les calculs
  - Impossible de faire deux calculs ou deux transferts simultanés
  - Calcul et transfert simultanés possibles
    - Permet de faire le recouvrement calcul / transfert

Trace d'exécution



1 GPU partagé entre 4 processus (cœurs)

Utilisation de S_GPU dans BigDFT

BigDFT

BigDFT

- Code de simulation ab-initio
  - Basé sur la mécanique quantique
  - Beaucoup de calculs et de données
- Code très régulier
- Développé en fortran, utilise MPI et certains calculs ont été déportés sur GPU avec CUDA (double précision)
  - Convolutions
  - BLAS (avec CUBLAS)

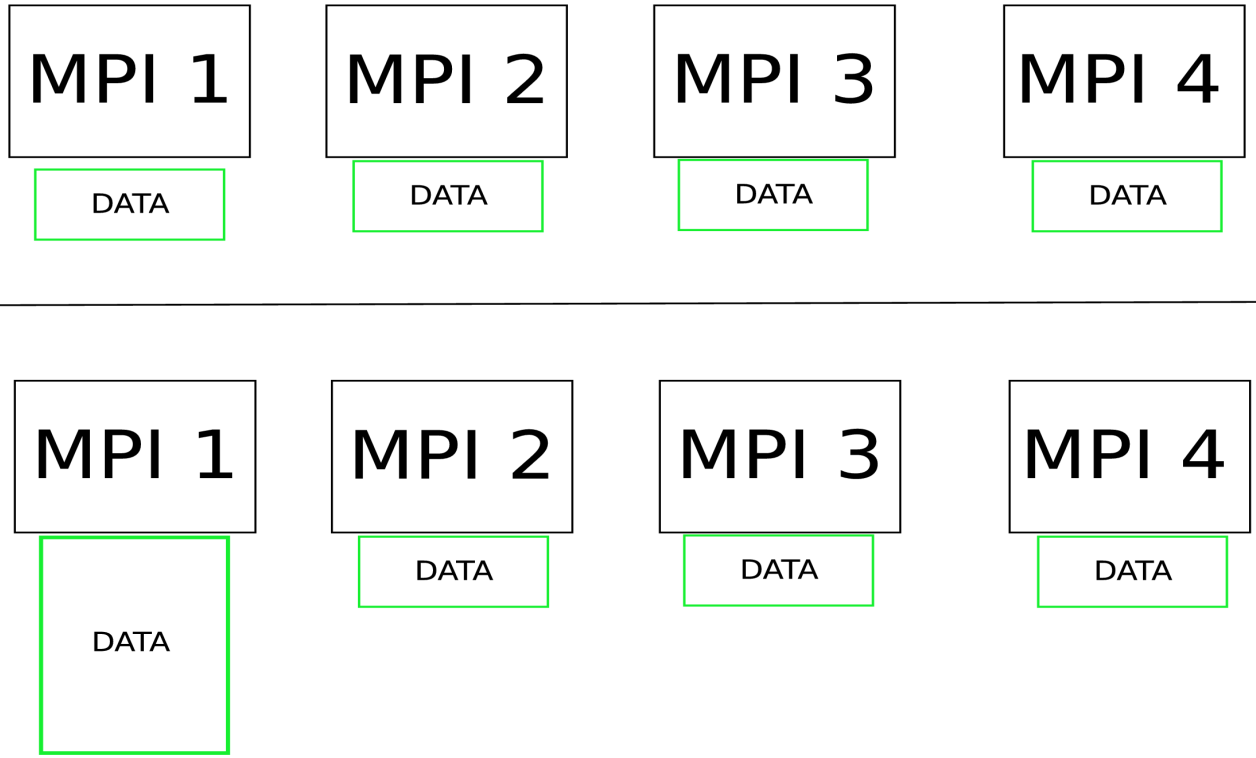
BigDFT

- Parallélisation
  - Chaque processus (tâche MPI) peut utiliser CUDA
  - Une tâche MPI par cœur CPU
- Efficacité de la parallélisation très bonne (de l'ordre de 90% pour 1000 cœurs)

Distribution des orbitales dans les tâches MPI

- Orbitales : type de données élémentaires (fonction d'onde)
- Les orbitales sont partagées entre les tâches MPI

Distribution des orbitales dans les tâches MPI



- Régulière
- Irrégulière

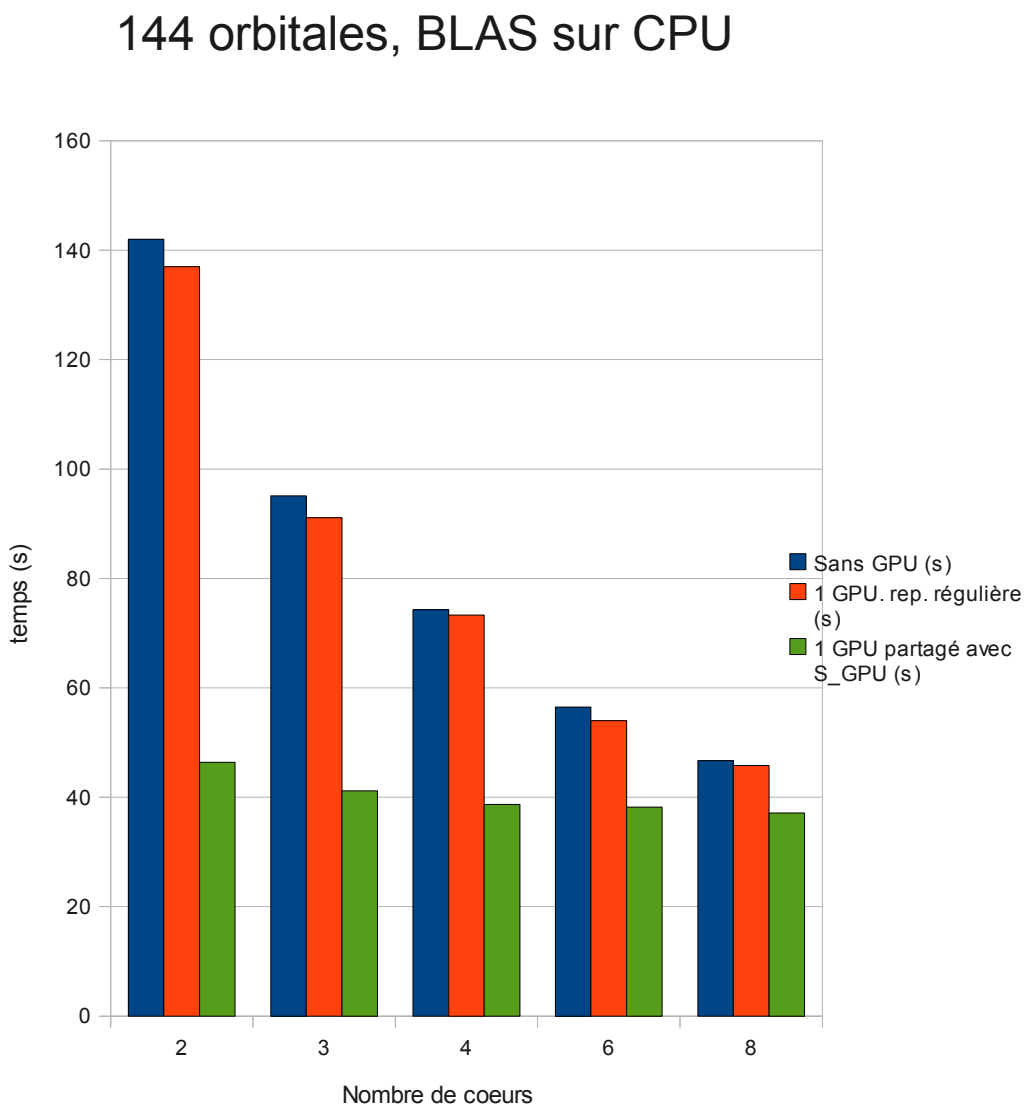
- Analyse des performances de ces différentes répartitions

Évaluation des performances

Contexte expérimental

- Machines
  - Nœud hybride
    - Deux Intel Xeon X5472 quadri-cœur, 16Go et deux NVIDIA TESLA C1060
  - Grappe hybride
    - IBLIS du CINES, configuration identique sauf pour la mémoire (8Go par nœud)
- Évaluation en répartissant les données de manière régulière et irrégulière sur le nœud hybride

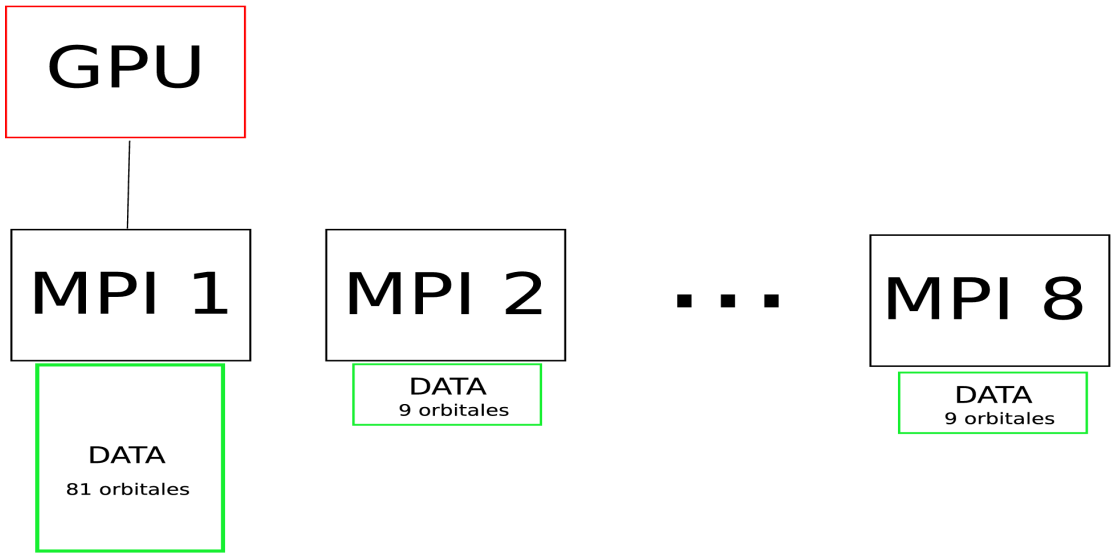
Répartition régulière (double prec.)



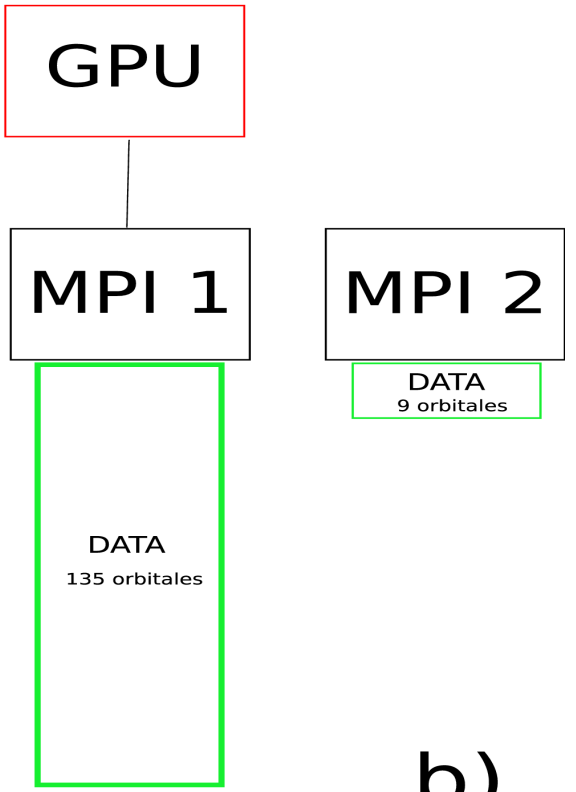
- Le temps d'exécution sur GPU est constant.
- Plus le nombre de tâches augmente, moins le GPU est intéressant

Répartition irrégulière

144 orbitales



a)



b)

Comparaison S_GPU / irrégulière (double précision)

	S_GPU	Irrégulière
2 tâches	46,4	52,4
8 tâches	37,1	34,9

- Irrégulière
  - Les parties non-accélérées deviennent beaucoup plus lentes.
  - Calcul simultané CPU – GPU

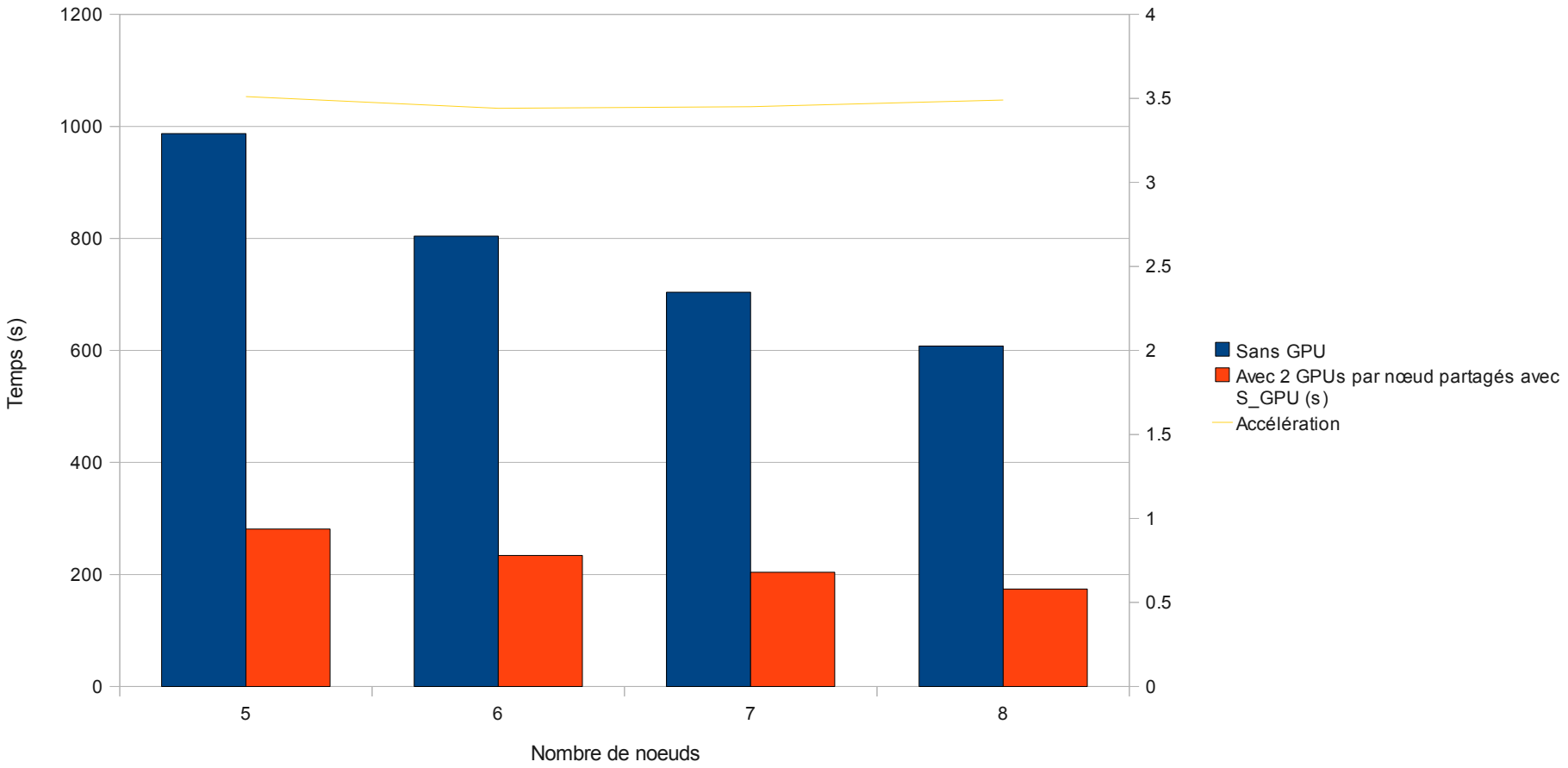
Comparaison S_GPU / irrégulière (double prec.)

- S_GPU
  - Bonne performances
  - Aucun calcul de répartitions de données

→ Utilisation de S_GPU dans la grappe hybride

Sur la grappe Iblis (double prec.)

2 GPU par nœud partagés avec S_GPU



Résultats récents

- Grand Challenge TITANE
  - Exécution de BigDFT sur plus de 50 nœuds hybrides
- Prix BULL – GENCI – FOURIER (juin 2009)
  - L. Genovese + BigDFT + GPU

- Conclusion
  - S_GPU : partage efficace des cœurs généralistes avec des GPUs
  - Intégration de S_GPU à une application « réelle » de calcul scientifique
  - Comparaison à d'autres approches de partage des GPUs
  - Utilisation avec succès de S_GPU sur une grappe hybride
- Perspectives
  - Utilisation de S_GPU sur des codes moins réguliers que BigDFT
  - Évaluation et modélisation de la consommation énergétique

QUESTIONS ?