POLYTECH SORBONNE

SORBONNE UNIVERSITÉ

PROJET HPC

P. FORTIN - L. ABBAS TURKI

Paola ALLEGRINI
Mahshid KHEZRI NEJAD

2018-2019

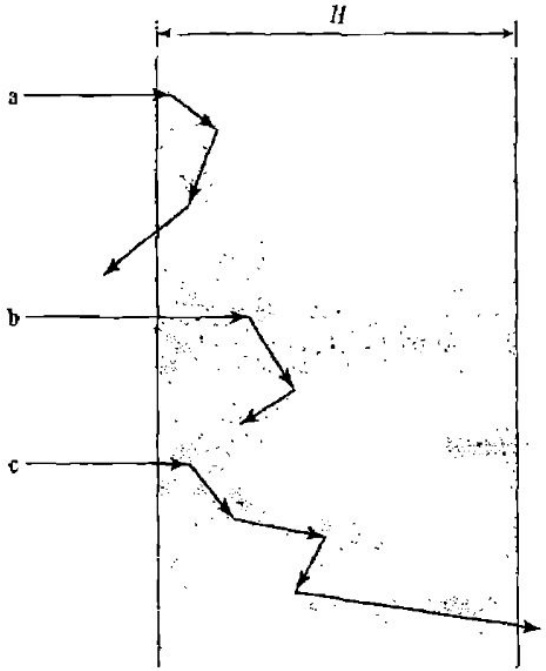
Présentation du problème

Objectif:

- **Simulation** parallèle de transport de neutrons
- A l'aide d'une méthode de type **Monte Carlo**

Paramètres :

- H : largeur de la plaque
- n : nombre de neutrons



Version GPU : Nombres aléatoires

Générateur de nombres aléatoires :

- Libraire : <curand_kernel.h>
- curand_init() pour initialiser le générateur
- curand_uniform (&state)

Chaque thread avec sa propre séquence de nombres aléatoires :

- Définition d'un vecteur d'état : chaque thread avec une séquence de nombres aléatoires différente

Version GPU : Vérification résultat correct

- Lancement du programme plusieurs fois
- Comparaison des % de neutrons absorbés, transmis et réfléchis avec la version séquentielle
- Vérification du remplissage du tableau absorbed

Version GPU : Mise à jour des compteurs

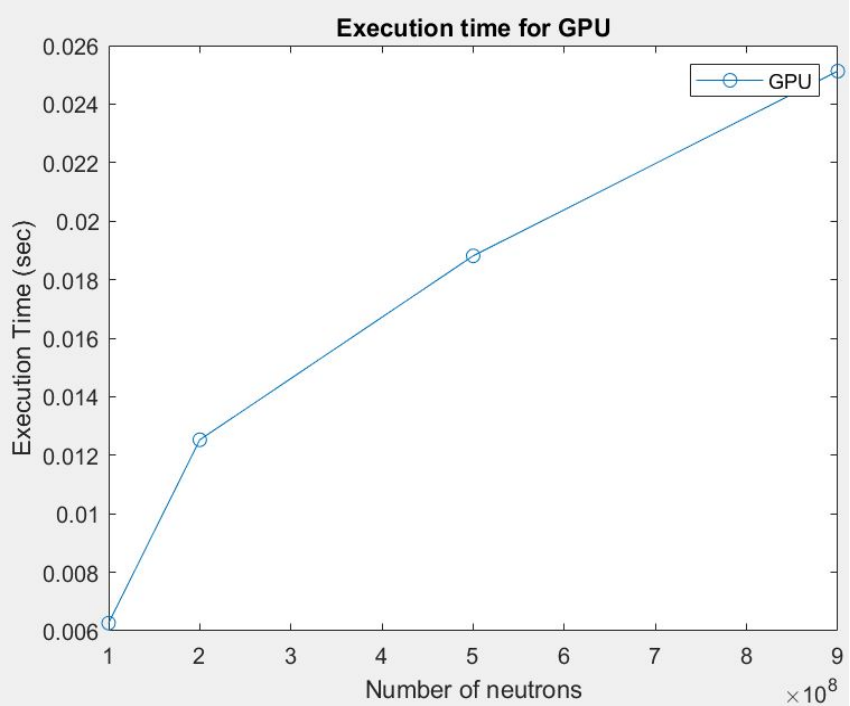
Compteurs r, b, t :

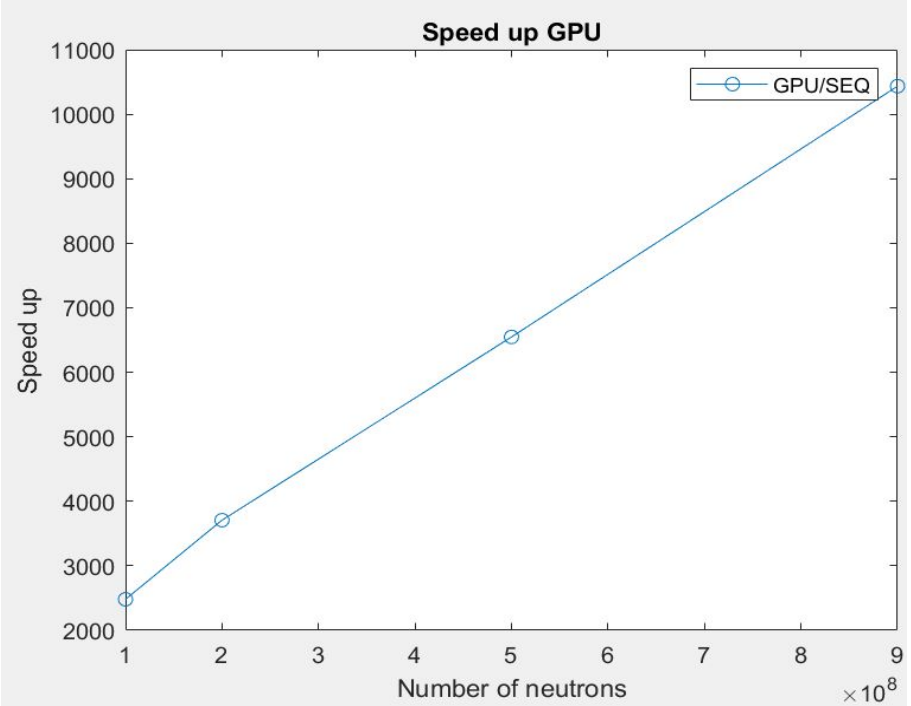
- Allocation mémoire partagée par bloc de thread :
  - 3 tableaux par bloc : R, B, T
  - taille : nbThreads par bloc
- Synchronisation puis réduction du tableau d'un bloc
- Atomicadd de R[0], B[0], T[0] d'un bloc

Stockage contigu dans absorbed :

- Utilisation d'un compteur global j
- AtomicAdd pour mettre à jour le compteur avant le stockage de la position

Version GPU : Comparaison performances





Version CPU : Parallélisation

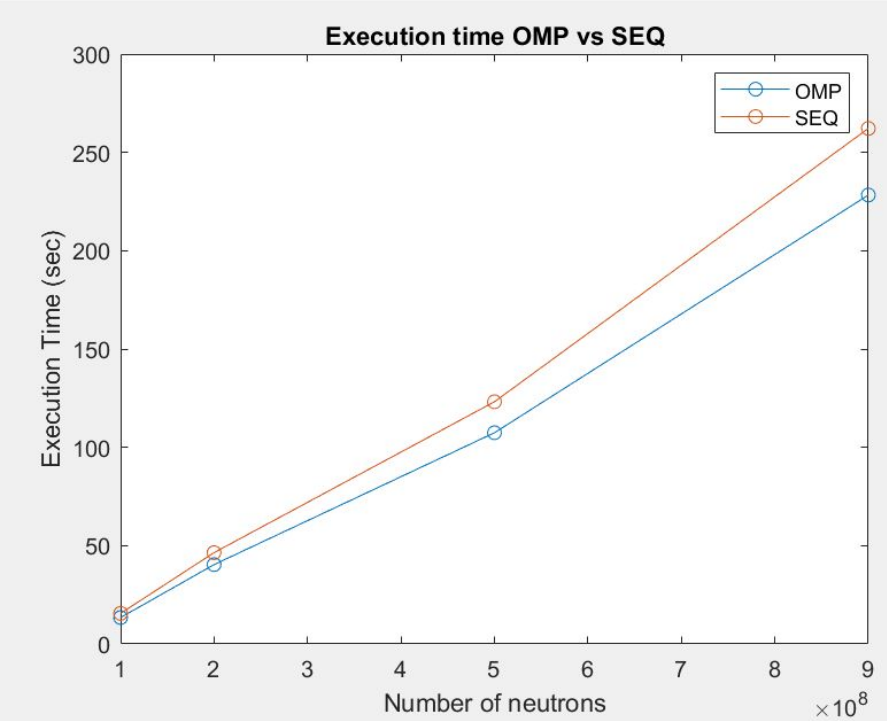
Parallélisation :

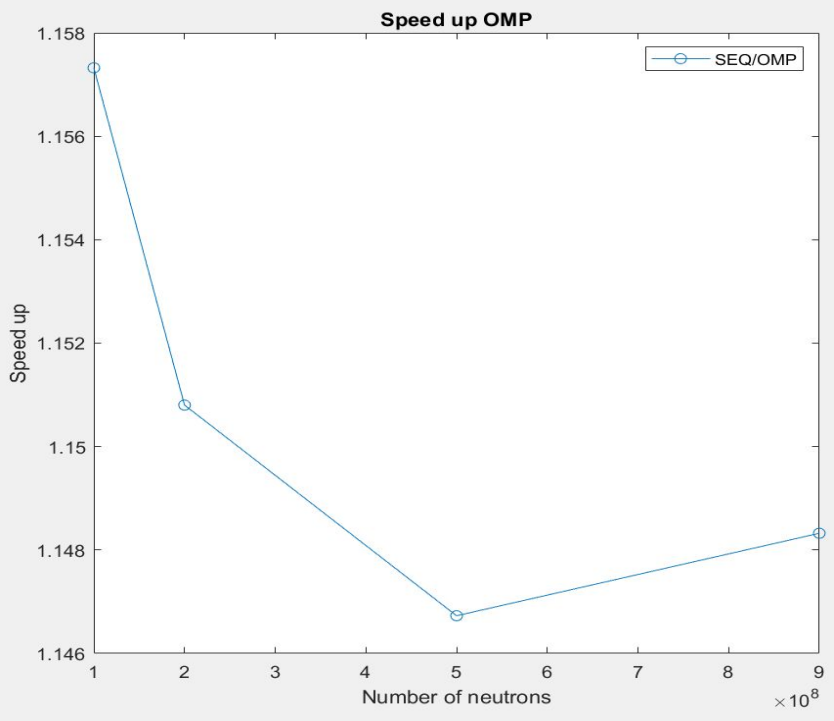
- num_threads()
- Boucle for avec réduction sur r, b et t
- Atomic update pour indice du tableau absorbed

Séquence de nombres aléatoires :

- Utilisation de la fonction private dans OpenMP
- private(x, u, d, L)

Version CPU : Comparaison performances

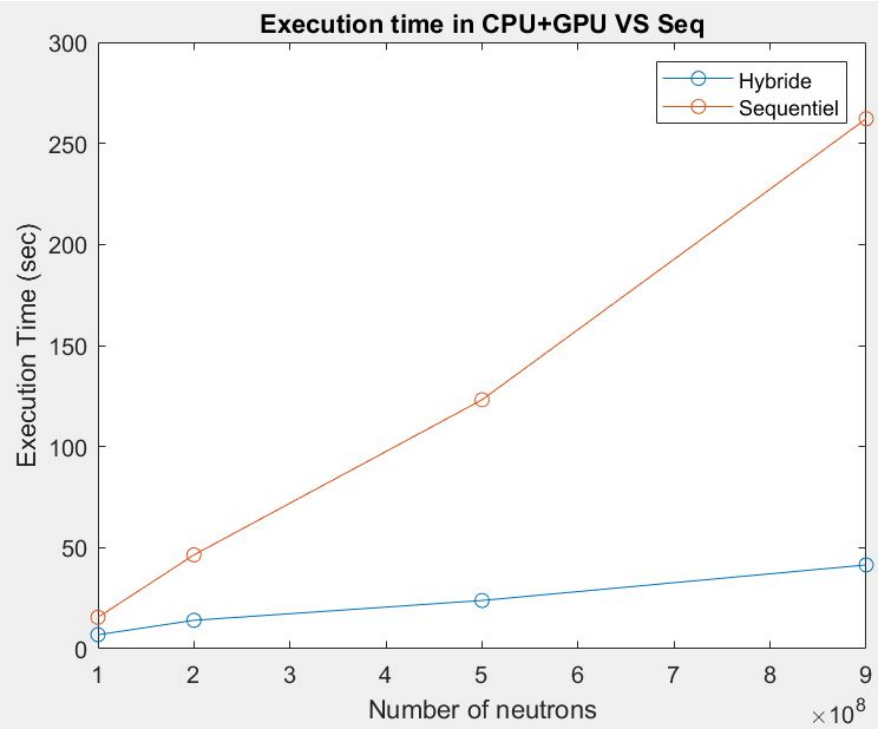


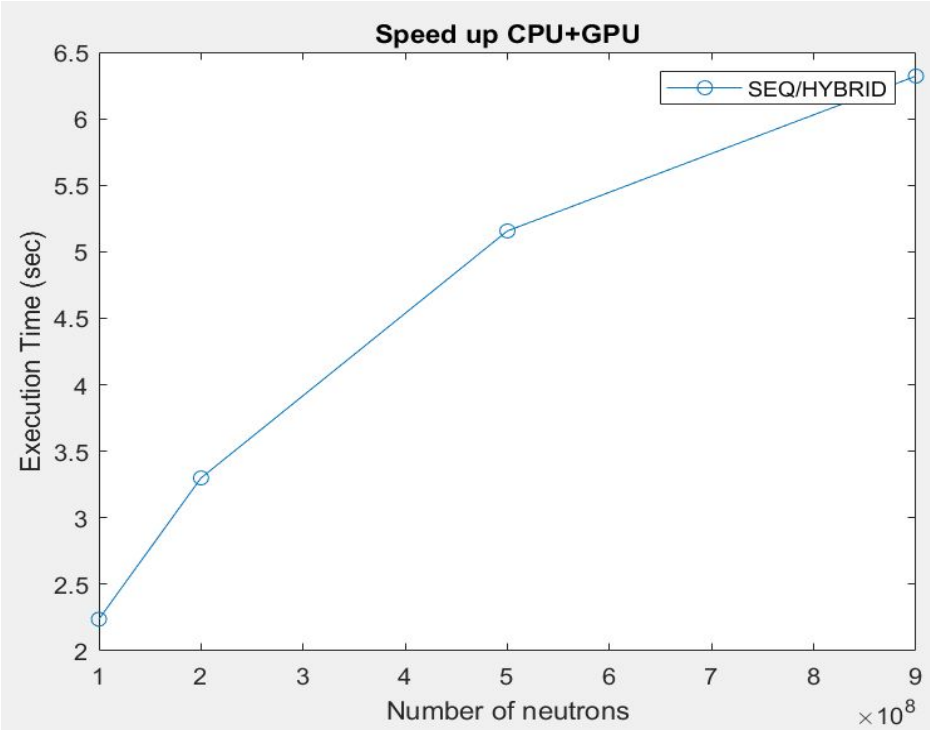


Version GPU + CPU : Parallélisation

- **Partage de tâche avec OpenMP** :
  - 1 thread pour lancer le kernel GPU
  - 1 thread CPU
- **Tests** :
  - 80% sur GPU et 20% sur CPU
- **Allocation mémoire CUDA :**
  - Effectuée dans le thread principal

Performances





Conclusion

- Connaissances en programmation GPU
- Comparaison des performances intéressante
- Meilleure version : GPU seul
- Perspectives :
    - Exploitation MPI pour version hybride