Projet HPCA : transport de neutrons

Version du 13 octobre 2018

1 Présentation

Le but de ce projet est d'effectuer une simulation parallèle de transport de neutrons à l'aide d'une méthode de type Monte Carlo. Nous étudions ici un modèle simplifié en 2D (voir figure 1). Une source émet des neutrons contre une plaque homogène d'épaisseur H et de hauteur infinie (ce qui ramène ce problème 2D à un problème 1D). Un neutron peut être réfléchi par la plaque, absorbé dans celle-ci ou transmis à travers celle-ci. Nous souhaitons calculer les pourcentages d'occurrence de ces trois cas.

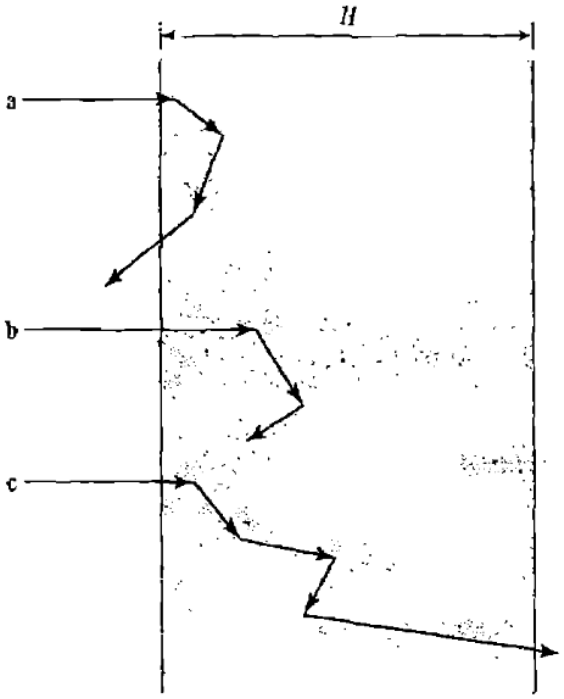

FIGURE 1 – Un neutron rencontrant un milieu homogème de longueur H peut être (a) réfléchi, (b) absorbé, ou (c) transmis.

Deux constantes décrivent l'interaction des neutrons dans la plaque : la « section efficace » de capture C_c et la « section efficace » de diffusion C_s. La section efficace totale est $C = C_c + C_s$.

La distance L qu'un neutron parcourt dans la plaque avant d'interagir avec un atome est modélisée par une distribution exponentielle de moyenne $1/C$. Si u est un nombre aléatoire issu d'une distribution uniforme sur $[0.1[$, la formule $L = -(1/C) \ln u$ fournit un nombre aléatoire de la distribution exponentielle appropriée.

Quand un neutron interagit avec un atome dans la plaque, sa probabilité de « rebondir » sur l'atome est C_s/C, alors que la probabilité qu'il soit absorbé par l'atome est C_c/C. Nous pouvons utiliser un nombre aléatoire d'une distribution uniforme sur $[0, 1[$ pour déterminer le résultat d'une interaction neutron-atome.

Si un neutron est diffusé, il a la même probabilité de partir dans chaque direction. Ainsi son nouvel angle de direction D (mesuré en radians) peut être modélisé par une variable aléatoire distribuée uniformément dans $[0, \pi[$. Comme la plaque a une hauteur infinie, nous ne faisons en effet pas de distinction entre un rebond vers le haut et un rebond vers le bas. Pour un angle de direction D, la distance correspondante sur l'axe « x » que le neutron parcourt dans la plaque entre deux collisions est $L\,cos(D)$.

La simulation d'un neutron se poursuit jusqu'à ce que

— le neutron soit absorbé par un atome ;
— ou que la position en « x » du neutron soit inférieure à 0, ce qui signifie que le neutron a été réfléchi par la plaque ;
— ou que la position en « x » du neutron soit supérieure à H, ce qui signifie que le neutron a été transmis à travers la plaque.

Le code séquentiel effectuant cette simulation est fourni (l'aide est affichée lors d'une exécution sans arguments). Ce code récupère les positions des neutrons absorbés et les écrit dans un fichier (dans le répertoire `/tmp/`) : le temps d'écriture dans un fichier n'est pas pris en compte.

Le cas de référence par défaut sera $H = 1.0$, $n = 500000000$, $c_c = 0,5$ et $c_s = 0,5$. Au besoin, d'autres valeurs de ces paramètres pourront être explorées.

2 Partie 1 : parallélisation sur [CPU+]GPU

2.1 Version GPU

Déployez en CUDA la simulation de transport de neutrons sur un seul GPU.

Quel générateur de nombres aléatoires utilisez-vous sur GPU ?

Comment faites-vous en sorte que chaque thread GPU ait sa propre séquence de nombres aléatoires ?

Comment vérifiez-vous que le résultat de votre simulation parallèle sur GPU est correct ?

Comment obtenez-vous la mise à jour correcte des compteurs sur GPU (nombre de neutrons réfléchis, absorbés et transmis), ainsi que le stockage contigu des positions des neutrons absorbés sur GPU ? Si nécessaire, proposez des versions plus efficaces.

2.2 Version CPU

Déployez la simulation de transport de neutrons sur un nœud de CPU multi-cœurs.

Comment faites-vous en sorte que chaque thread CPU ait sa propre séquence de nombres aléatoires ?

Comment vérifiez-vous que le résultat de votre simulation parallèle sur CPU est correct ?

Comparez-les performances obtenues entre un GPU et un nœud de CPU multi-cœurs.

Une variation de la probabilité d'absorption impacte-t-elle de la même façon la vitesse de calcul du GPU et celle sur CPU ?

2.3 Version CPU+GPU

Proposez une version hybride permettant d'exploiter à la fois les cœurs CPU et les cœurs GPU sur un des nœuds de calcul dont vous disposez.

Quel est l'impact d'un changement de nœud de calcul sur votre approche ?

La vitesse de calcul de la version CPU+GPU est-elle égale à la somme des vitesses de calcul des versions "CPU seul" et "GPU seul" ?

3 Partie 2 : portabilité et parallélisation multi-nœud

3.1 Version OpenCL

Déployez en OpenCL votre version "GPU seul" de simulation de transport de neutrons.

Les performances sont-elles équivalentes à la version CUDA sur GPU ?

Adaptez-votre code OpenCL pour GPU afin qu'il puisse s'exécuter sur un nœud de CPU multi-cœurs.

Comment les performances de cette version se comparent-elles à votre version "CPU seul" de la partie 1 ?

3.2 Version MPI

En vous appuyant sur MPI et sur la version CPU+GPU de la partie 1, proposez une solution pour exploiter plusieurs nœuds de CPU multi-cœurs et de GPU.

Comment vérifiez-vous que le résultat de votre simulation parallèle est correct ?

On pourra ici augmenter la valeur de n si cela permet d'obtenir une vitesse de calcul globale plus importante.

4 Travail à remettre

Pour chacune des deux parties, vous devrez remettre le code source, sous la forme d'une archive `tar` compressée et nommée suivant le modèle `projet_HPCA_nom1_nom2.tar.gz`. L'archive ne doit contenir aucun exécutable, et les différentes versions demandées devront être localisées dans des répertoires différents. Chaque répertoire devra contenir un fichier `Makefile` : la commande `make` devra permettre de lancer la compilation, et la commande `make exec` devra lancer une exécution parallèle représentative avec des paramètres appropriés. Un fichier `Makefile` situé à la racine de votre projet devra permettre (avec la commande `make`) de lancer la compilation de chaque version.

A la fin du projet, vous devrez remettre un rapport au format `pdf` (de 5 à 10 pages, nommé suivant le modèle `rapport_HPCA_nom1_nom2.pdf`) présentant vos algorithmes, vos choix d'implémentation (sans code source), vos résultats et vos conclusions pour les deux parties. L'analyse du comportement de vos programmes sera particulièrement appréciée. Les machines n'étant pas strictement identiques, on précisera dans le rapport la (ou les) machine(s) utilisée(s) pour les tests de performance. Vous ne serez bien sur pas pénaliser si vous utilisez un CPU ou un GPU moins puissant que les autres étudiants.

5 Quelques précisions importantes

- Le projet est à réaliser par binôme (aucun trinôme ne sera accepté).
- Le code de la première partie, accompagné des slides de votre soutenance (au format `pdf`, et donc sans animations, dans un fichier nommé suivant le modèle `projet_HPCA_slides_nom1_nom2.pdf`; prévoir une soutenance de 10 minutes, suivie de 5 minutes de questions), est à remettre au plus tard le mercredi 05 / 12 / 2018 à 22h00 (heure locale). Les soutenances de présentation de la première partie auront lieu lors de la séance de TDTP du jeudi 06 / 12 / 2018.

  Le code de la seconde partie et le rapport final sont à remettre au plus tard le dimanche 27 / 01 / 2019 à 22h00 (heure locale).
- Quel que soit votre groupe, le projet devra être remis par courriel à : `lokmane.abbas_turki@upmc.fr` et `pierre.fortin@lip6.fr`
- En cas d'imprévu ou de problème technique commun, n'hésitez pas à nous contacter pour que nous puissions vous proposer une solution ou une alternative.