TP5

Objectif

Optimisation de programmes avec cuda

Travail à réaliser

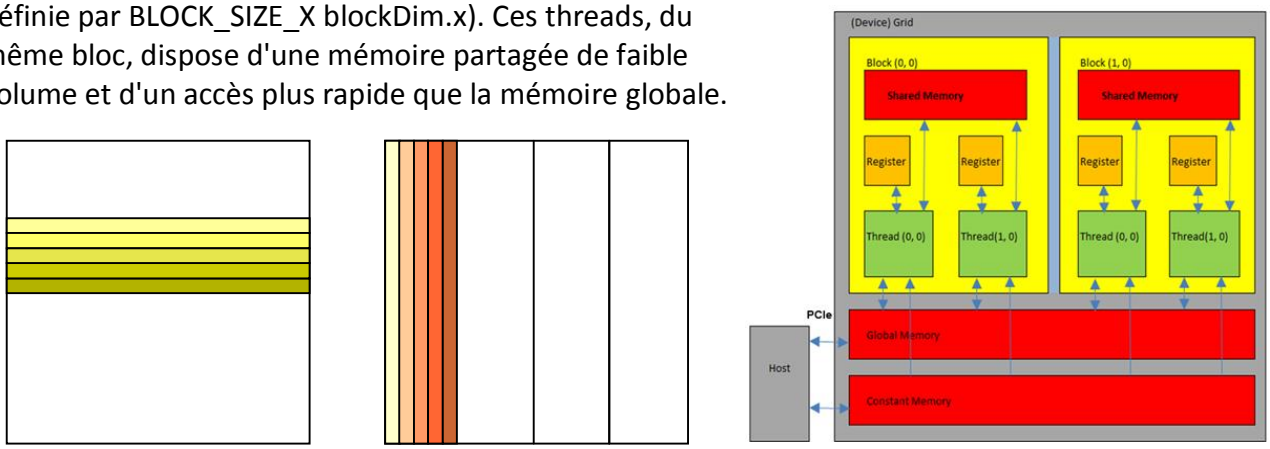
1- Prendre une copie du programme mul_matG2B2.cu (résultat de l'étape 5 du TP4). Le tester avec TM = 2048, et avec différentes valeurs de BLOCK_SIZE_X et BLOCK_SIZE-Y : 256-2, 128-4, 64-8, 32-16, 16-16, 16-32, 8-64, ... Noter les performances. Peut-on les expliquer ?

2- Les threads sont exécutés sur le GPU en fonction de leur organisation en blocs :
- un bloc dispose d'un certain nombre de registres partagés par les threads appartenant à ces blocs,
- les threads d'un même bloc se partagent un bloc de mémoire (partagée) plus rapide que la mémoire globale,
- les threads d'un même bloc sont exécutés par petits ensembles (warp) de 32 (ou 64 selon les versions des GPU). Ces warps sont exécutés de façon concurrente et se partagent les mêmes ressources (unités de calcul, accès mémoire, etc.), et peuvent bénéficier d'un certain parallélisme lorsqu'ils font appel à des ressources différentes à un instant donné.

Le déroulage boucle (duplication du code à l'intérieur d'un boucle) peut augmenter les possibilités d'entrelacement des instructions, diminuer les blocages, et améliorer le parallélisme entre les threads.

Avec BLOCK_SIZE_X = 128 et BLOCK_SIZE-Y = 2, modifier le kernel d'exécution en réalisant un déroulage de facteur 2, 4, et 8. Noter les performances. Peut-on les expliquer ?

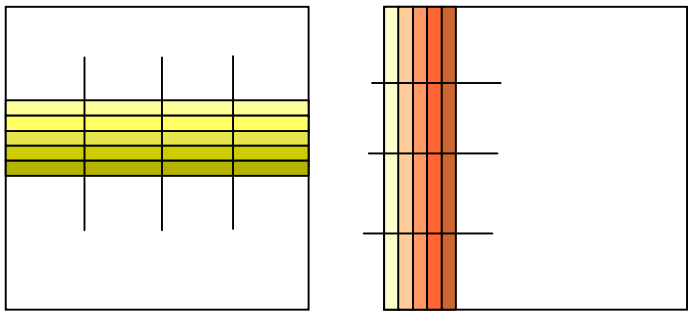
3- Le programme mul_matG2B2 découpe la matrice A en bandes horizontales (dont la largeur est définie par BLOCK_SIZE_Y ou blokDim.y), et la matrice B en bandes verticales (dont la largeur est définie par BLOCK_SIZE_X blockDim.x). Ces threads, du même bloc, dispose d'une mémoire partagée de faible volume et d'un accès plus rapide que la mémoire globale.



Mais le volume de ces bandes ne permet de les stocker dans cette mémoire partagée.
- Vérifier le volume de mémoire partagée disponible sur le GPU (deviceQuery).
- Combien de lignes de A et de colonnes de B peut-on mettre dans cette mémoire partagée ?
- Pourquoi ne peut-on exploiter cette mémoire partagée avec cet algorithme ?

- En gardant la même organisation des threads et des blocs, comment modifier l'algorithme de manière à ce que les threads d'un même bloc puissent traiter la bande horizontale de A et la bande verticale de B par morceaux gardées en mémoire partagée ?

4- On ajoute un niveau de découpage des matrices de la façon suivante :



On réalise la multiplication du premier morceau du bloc de la bande horizontale de A, qu'on appelle « tuile » avec le premier morceau de la bande verticale de B en totalité, puis on refait la même chose avec les morceaux suivants. Ainsi un morceau de A de dimension (blockDim.y *TILE_SIZE) et un morceau de B de dimension (TILE_SIZE*blockDim.x) peuvent rester en mémoire partagée, tout au long de leur traitement, et bénéficient de l'accès rapide pour chaque élément.

- La déclaration d'une matrice dans la mémoire partagée se fait de la façon suivante :

```
__shared__ float As[M][N];
```

- il faut ensuite copier le morceau correspondant de A dans cette matrice. En utilisant des morceau carrés de dimension TILE_SIZE x TILE_SIZE (BLOCK_SIZE_X = BLOCK_SIZE_Y = TILE_SIZE), il y a égalité entre le nombre de threads, le nombre d'éléments de A et le nombre d'éléments de B à copier dans la mémoire locale. Chaque thread va donc copier un éléments de A et un élémeent de B.

- ensuite, il faut synchroniser les threads pour qu'ils ne débutent le calcul qu'après le remplissage des matrices As et Bs, avec : `__syncthreads();`

- effectuer le calcul
- synchroniser de nouveau les threads pour attendre la fin des calculs avant de passer à l'étape suivante.

A- Quelles valeurs de TILE_SIZE permettent de mettre deux tuiles dans la mémoire partagée ?

B- Apporter ces modifications dans le programme mul_matG2B2. Tester les performances pour différentes tailles de tuiles.

C- Mettre en commentaire, alternativement, chacune des 2 instructions `__syncthreads();`
Vérifier les conséquences sur le résultat, et le temps qu'elles consomment.
Peut-on s'en passer ?

5- Peut-on utiliser des tuiles rectangulaires (BLOCK_SIZE_X*TILE_SIZE et TILE_SIZE*BLOCK_SIZE_Y) ?
Essayer d'implémenter ce cas, et vérifier si cela améliore les résultats.