

TP 2 : Programmation parallèle - CUDA

Dans ce TP nous allons aborder :

- La programmation sur GPU avec CUDA NVCC
- Le principe d'allocation mémoire et de transport de données avec CUDA
- Le modèle mémoire d'un GPU et les bénéfices que l'on peut en tirer

Pour compiler, utilisez la commande suivante :

```
nvcc fichier.cu
```

Exercice 1 Introduction

La base de tout langage de programmation commence avec l'écriture d'un hello world. Pour compiler le programme helloworld, entrez la ligne suivante : `nvcc -o hello helloworld.cu`

Pour exécuter, il suffit ensuite de lancer l'exécutable : `./hello`

Dans le cas présent, avec un GPU de type Kepler, vous devriez voir un "HELLO WORLD !" qui est affiché avec un caractère par ligne. Maintenant, si vous augmentez le nombre de blocs dans le programme que va t'il se passer sachant que le GPU est une plate-forme de calcul parallèle ? Comment pouvez-vous expliquer ce phénomène ?

Exercice 2 Addition de vecteurs

L'addition vectorielle est un bon exemple pour comprendre l'allocation mémoire et la distribution des données sur un GPU en gardant un kernel assez basique. Après avoir regarder les notions de `cudaMemcpy`, `cudaMalloc`, blocs et threads modifier le code `addition_vectorielle.cu` pour qu'il compile et fonctionne correctement. Utiliser la ligne de compilation suivante :

```
nvcc -o vector_addition_vectorielle.cu
```

Le résultat à obtenir est `c [4194303] = 8388606`.

Exercice 3 Stencil 1D

Dans le fichier `1DStencil.cu` vous trouverez un exemple de filtre permettant d'appréhender la notion de mémoire partagée d'un GPU. Dans un

premier temps, faites tourner le code sans utiliser la mémoire partagée et utiliser une fonction de mesure de temps pour timer le code. Essayez ensuite d'utiliser la mémoire partagée et faites tourner votre code plusieurs fois. Les résultats vous semblent-ils corrects ? Que devez-vous faire pour corriger cela ? Les performances sont-elles les mêmes avec et sans la mémoire partagée ? Utiliser la ligne de compilation suivante :

```
nvcc -o stencil 1Dstencil.cu
```

Exercice 4 Produit de vecteurs

Écrire un produit vecteur-vecteur sur GPU et vérifier que le code fonctionne. Essayer maintenant de comparer les performances sur différentes tailles de matrices avec une version CPU que vous écrirez comprenant du openmp et de la vectorisation. Que pouvez-vous en déduire de l'utilité d'un GPU dans le calcul scientifique ?