

TP Partie 1 : les bases de CUDA

Questions préliminaires

Quel est votre matériel ?

Pour découvrir quel matériel est installé sur votre machine, nous allons utiliser le programme *deviceQuery*. Il est fourni par Nvidia dans chaque distribution de CUDA. Voici les différentes étapes pour le compiler et l'utiliser.

```
$ cd 0-deviceQuery
$ mkdir build && cd build
$ cmake ..
$ make
$ ./deviceQuery
```

Servez-vous du résultat de *deviceQuery* pour répondre à ces quelques questions:

1. Combien y-a-t-il d'unités de calcul (ALU) sur votre GPU ?
2. On considère qu'une ALU peut fournir le résultat de 3 opérations flottantes par coup d'horloge. Combien d'opérations par secondes en FLOPS (FLoation-Poing Operations per Second) peut effectuer votre GPU ?
3. Quelle est la bande passante théorique du GPU ?
4. Comparez la fréquence de fonctionnement du GPU avec celle du CPU (vous pouvez utiliser *lscpu* par exemple).
Pourquoi est-il quand même intéressant d'utiliser un GPU ?

Obfuscation (Parite I)

Intro

Pour faire les choses simples quand on peut faire compliquer ? L'obfuscation est l'art de rendre quelque chose difficile à comprendre au premier abord. Dans the projet, une image a été divisée en deux parties. Prises seules, elles sont impossibles à comprendre. Vous allez devoir les recombinaison pour les déchiffrer.

Allez dans le dossier *1-Obfuscation* et visualisez les images dans *data*. Sous Linux, vous pouvez ouvrir une image depuis le terminal avec `eog file.tif`.

Vous allez devoir calculer la somme des deux images pour obtenir le résultat souhaité.

Lisez l'intégralité du sujet avant de démarrer !

Organisation du code

- CMakeLists.txt -- fichier de configuration de cmake
- data -- images d'entrée
 - big_frag_1.tif
 - big_frag_2.tif
 - small_frag_1.tif
 - small_frag_2.tif
- include
 - obfuscate.hpp -- fichier header contenant des macros utiles
- src
 - main.cpp -- structure de base du programme, vous n'avez pas besoin de faire des modifications ici
 - obfuscate.cu -- **l'endroit où vous devez faire vos modifications**
 - reference.cpp -- contient l'implémentation de référence sur CPU

Compiler the code

Tout comme *deviceQuery*, ce projet utilise cmake. Une fois dans le dossier *1-Obfuscation*, créez et allez dans le dossier *build* puis générez le Makefile avec `cmake ..` et compilez avec `make`. À chaque nouveau changement du code, vous n'avez qu'à exécuter `make`. Lancer le programme avec `./obfuscation exercise_1`.

Premiers pas en CUDA

Dans obfuscate.cu, vous allez devoir implémenter la structure de base de n'importe quel programme CUDA:

1. Allouer la mémoire sur le GPU (device).
2. Transférer les données sur cet espace mémoire.
3. Lancer le kernel.
4. Copier les données du device sur l'host (CPU).
5. Libérer la mémoire du GPU.

Il y a déjà du code de présent pour vous aider. Dans cet exercice, vous devrez faire fonctionner votre kernel sur **un seul** block. Vous devez lui donner une taille appropriée pour vos données.

Ici, l'image est représentée comme un tableau continu d'octet non-signés (niveaux de gris). Vous devez calculer la somme des deux images et stocker le résultat dans la sortie.

Vous pouvez regarder l'implémentation de référence dans *reference.cpp*. Si vous avez besoin de détails supplémentaires, la documentation de CUDA est fournie avec le projet. Vous pouvez la décompresser comme ceci : `tar xzf documentation.tar.gz`.

Questions

- Combien de temps prend votre implémentation CUDA ? Quelle est fraction de cette durée consacrée à la gestion mémoire ?
- Comparez avec l'implémentation CPU.

Obfuscation (Partie 2)

Il y quelques changements par rapport à la partie I :

- l'image est en RGB, sur 32 bits. Utilisez les macros de obfuscate.hpp pour vous aider.
- l'image est trop large pour rentrer dans un seul block. Vous aller devoir diviser votre problème dans différents blocs.

Questions

Idem qu'en partie I:

- Combien de temps prend votre implémentation CUDA ? Quelle est fraction de cette durée consacrée à la gestion mémoire ?
- Comparez avec l'implémentation CPU.

Nouvelle question:

- Quelle est l'influence des différents découpages en bloc possibles ?