Subject-Part1-fr.md

TP Partie 1 : Obfuscation

Intro

L'obfuscation est l'art de cacher les choses. Ici vous aurez deux images, qui prises séparement, ne représentent que du bruit. Pour en tirer leur secret, il va valoir en faire les sommer, pixel par pixel.

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

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'a exécuter `make`. Lancer le programe avec `./obfuscation exercise_1`.

Premiers pas en CUDA

Dans obfuscate.cu, vous allez devoir implémenter la structure de base de tout programme CUDA:

1. Allouer la mémoire sur le GPU (device).
2. Transferer 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

1. Combien de temps prend votre implémentation CUDA ? Quelle est fraction de cette durée consacrée à la gestion mémoire ?
2. 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:

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

Nouvelle question: 3. Faîtes différents essais de combinaisons taille des blocks / taille de la grille. Pour quelle découpe la temps d'exécution est-il le meilleur ? Quelle est votre justification ?