M2 - Architecture et Programmation d'accélérateurs Matériels.

(APM 2016-2017)

TP2

CUDA

hugo.taboada.ocre@cea.fr
julien.jaeger@cea.fr
patrick.carribault@cea.fr

Les objectifs de ce TP sont :
- Ecriture d'un premier code CUDA
- Utilisation efficace d'un GPU
- Calcul de l'indice global avec plusieurs dimensions

I Traduction d'un code C en code CUDA

Dans cette partie nous considérons le fichier tp2_1.c présent dans le répertoire *Emul_CPU*. Ce code simple réalise l'initialisation d'un tableau `a`, tel que `a[i]=i`, par le biais d'une indirection avec un tableau `b`. Une fonction `check_a` permet de vérifier que l'initialisation du tableau `a` s'est bien passée.

Q.1: Le squelette de la traduction en CUDA du programme contenu dans le fichier tp2_1.c vous est fourni dans le répertoire *CODE* (fichier *tp2_1.cu*. A vous de remplir ce squelette (parties notées "A COMPLETER" dans le code) pour réaliser l'initialisation du tableau `a` sur le GPU.

II Utilisation efficace d'un GPU

Q.2: Le schéma d'accès aux données du tableau `a` est-il efficace ? pourquoi ?

Q.3: Nous allons vérifier cette hypothèse. Insérer des appels à `gettimeofday` pour mesurer le temps du kernel CUDA (et uniquement du kernel). Relevez le temps mesuré.

Q.4: Nous allons maintenant changer le schéma d'accés aux données du tableau `a` en modifiant les valeurs dans le tableau d'indirection `b`. Remplacez la valeur actuelle de `STEP` par 1. Que cela change-t-il pour les accés au tableau `a` ? Relevez le temps mesuré. Est-il meilleur ? Pourquoi ?

Q.5: Jusqu'à présent, nous n'utilisions qu'un seul bloc de plusieurs threads. Nous allons changé cela. Modifiez la valeur de `nBlocks` pour la mettre à 16. Modifiez votre kernel pour avoir un calcul d'indice correct. Relevez le temps mesuré. Est-il meilleur ? Pourquoi ?

III Calculs d'indice global (si OpenCV ne marche pas)

Le calcul de l'indice global d'un thread est généralement très important dans un kernel CUDA. Nous allons augmenté le nombre de dimensions des grilles et des blocs dans le programme précédent pour s'habituer à manipuler plus de dimensions dans notre calcul d'indice global

Q.6: Passez à une taille de bloc à deux dimensions, avec 64 éléments sur la première dimension, et 16 éléments sur la deuxième dimension.

Q.7: Passez à une taille de grille à deux dimensions, avec 8 éléments sur la première dimension, et 16 éléments sur la deuxième dimension.

Q.8: Passez à des tailles de grille et de bloc à trois dimensions. Choisissez pour chaque dimension une valeur supérieure à 1.

IV Modification d'image (si OpenCV ok)

Le programme *tp2_video1.c* présent dans le répertoire *VIDEO* lit une vidéo, la traduit sous forme de tableaux de pixels, puis modifie ces pixels afin de transformer la vidéo couleur en vidéo en niveaux de gris. Vous pouvez compiler et exécuter le programme, en utilisant la petite vidéo test fournie *Wildlife.wmv*, pour voir l'effet du programme sur la vidéo générée *my_copy.wmv*.

Q.9: Dans le même répertoire se trouve le fichier tp2_video1.cu. Il s'agit du même programme sans le code permettant de transformer chaque image (frame) couleur en niveaux de gris. En vous appuyant sur les exercices précédent, à vous de réaliser le code CUDA (partie notée "A COMPLETER") permettant, pour chaque frame, de réaliser cette modification sur le GPU. Les nombres de blocs et de threads par blocs doivent être tous deux supérieurs à 32. Vous pouvez choisir le nombre de dimensions que vous voulez.

Q.10: Ce code est-il efficace ? Pourquoi ? Rendez-le plus efficace.

Q.11: Le programme *tp2_video2.c* réalise une autre modification en plus du niveaux de gris. La méthode de *SOBEL* fournie permet de détecter les contours sur une image quelconque. Ces contours sont alors affichés en blanc sur fond noir. Comme la question précédente, ecrivez le code CUDA permettant de réaliser la mèthode de SOBEL sur le GPU.