UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

M2 - Architecture et Programmation d'accélérateurs Matériels.

(APM 2016-2017)

TP1

CUDA

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

Les objectifs de ce TP sont :
- Prise en main du SDK CUDA
- Compréhension du modèle déxécution
- debugging

I Modèle d'exécution et SDK CUDA

Dans cette partie nous considérons le fichier tp1.cu. À tout moment, vous pouvez compiler le fichier l'exécuter (commande nvcc tp1.cu -o tp1).

Q.1: Quelle partie du programme doit s'exécuter sur l'hôte ? Quelle partie sur le device ?

Q.2: Quel calcule ce programme ? (si vous ne savez pas répondre à cette question, répondre à la suivante pourra vous aider).

Q.3: Combien y a-t-il de blocs au total ? Combien de threads par blocs ? Combien de threads au total ?

Q.4: Émuler sur CPU le comportement du GPU sans utiliser le SDK CUDA. Pour ce faire, réécrire le programme en C/C++ avec les contraintes suivantes :

1. utilisation d'une fonction `kernel`
2. utillisation des grilles de blocs et de threads

II Chaîne de compilation

Pour observer certaines étapes du processus de compilation, compiler le fichier *tp1_check.cu* avec la commande :
nvcc –keep –keep-dir CUDAIMG -c tp1_check.cu (il faudra avoir créé le répertoire avant de lancer la commande).
Des fichiers intermédiaires sont créés dans le répertoire CUDAIMG.

Q.5: Quel fichier correspond au PTX ? Lequel correspond au binaire CUDA ?

Q.6: Quelles parties des sources de *tp1_check.cu* ont été conservées pour être compilées en PTX ?

Q.7: À quoi correspond le suffixe ***__10*** des fichiers créés ?

Q.8: Noter le message d'avertissement sur le (non) support de la double précision. À partir de quelle *compute capability* la double précision est-elle supportée ?

Q.9: Recompiler en ajoutant l'option *-arch=sm_20*. Que remarquez-vous sur les fichiers créés et sur le message d'avertissement ?

III Debugging

Nous considérons à présent le fichier *err1.cu*. Ce programme est censé calculer la somme de deux vecteurs. À la fin de l'exécution, une erreur relative est calculée entre le vecteur issu du GPGPU et celui calculé sur CPU.

Q.10: Compiler et exécuter le programme. Le résultat est-il correct ?

Q.11: Encadrer chaque appel à CUDA par la macro `checkCudaErrors` (fichier *helper_cuda.h* dans le répertoire *$NVIDIA_SDK/common/inc*).

Q.12: Calculer le nombre total de threads. Comparer le à N. Qu'en déduire ?

Q.13: Corriger le code CUDA selon les deux possibilités suivantes :

1. Corriger le nombre de blocs pour traiter tous les indices des tableaux.
2. Sans changer les tailles de la grille et des blocs, modifier le kernel (prendre en compte le nombre total de threads).