UE : Dimensionnement et évaluation des architectures (I5AISE51)

Institut National des Sciences Appliquées de Toulouse
P.-E. Hladik, pehladik@insa-toulouse.fr

—

TD 1 : mon premier programme en CUDA

Version bêta (4 janvier 2021)

Message à caractère informatif

Afin de disposer d'un GPU de Nvidia, vous travaillerez via ssh sur le serveur du GEI `srv-gei-gpu1` ou `srv-gei-gpu2`. La connexion se fait avec votre compte habituel. Pour s'y connecter depuis l'extérieur de l'INSA il vous faut passer par le VPN.
Afin de faciliter le travail, il est proposé d'utiliser l'IDE Visual Studio Code en mode Remote (compilation à distance). Une vidéo est disponible sur moodle qui explique comme installer cet éditeur et comment ajouter les extensions nécessaires.

1 Parallèle Hello World

Objectif 1.1

— Compiler et exécuter un code en CUDA.

(1.1) Travail à faire : Hello world!

1. Récupérer le fichier `hello.cu` sur moodle et le déposer sur votre compte (un simple glisser/déposer marche très bien avec VSC Remote).
2. Compiler le fichier `hello.cu`.
3. Lancer l'exécutable, vous venez d'exécuter votre premier code sur un GPU !

(1.1) Comment faire : compiler un code CUDA

Pour compiler un fichier en CUDA (extension communément utilisée `.cu`) il vous faut un compilateur dédié, par exemple `nvcc`. Vous pouvez consulter la documentation sur :
https://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/index.html
La commande de base pour compiler est :

```
nvcc -o outputFile inputFile.cu
```

avec `outputFile` le nom du fichier de sortie et `inputFile` le fichier source à compiler.

(1.2) Comment faire : lancer l'exécutable

L'exécution d'un code CUDA se fait exactement de la même manière que pour un programme standard en faisant un simple appel à `./outputFile`.

2 Connaître sa machine

Objectif 2.1

— Récupérer et afficher les caractéristiques du GPU

(2.1) Travail à faire : Lecture des paramètres du GPU

1. Récupérer le template `properties.cu`.
2. Modifier le code pour connaître le nombre de GPU sur la machine et pour afficher le nom de la carte GPU que vous utilisez. Compiler, exécuter.
3. Ajouter dans le code les éléments pour afficher la fréquence de son horloge et le nombre de multiprocesseurs disponibles. Compiler et exécuter.

(2.1) Comment faire : La structure `struct cudaDeviceProp`

La structure `cudaDeviceProp` contient un ensemble de champs [a] décrivant un *device*. La structure est récupérée à l'aide de la fonction `cudaGetDeviceProperties` [b].

a. https://docs.nvidia.com/cuda/cuda-runtime-api/structcudaDeviceProp.html
b. https://docs.nvidia.com/cuda/cuda-runtime-api/group__CUDART__DEVICE.html

(2.2) Comment faire : Affichage des champs de la structure `cudaDeviceProp`

Pour afficher les paramètres du GPU il suffit de lire le champ de la structure `cudaDeviceProp` comme n'importe quelle structure en C, par exemple :

```
int deviceCount;
cudaGetDeviceCount(&deviceCount);
int device;
for (device = 0; device < deviceCount; ++device) {
    cudaDeviceProp deviceProp;
    cudaGetDeviceProperties(&deviceProp, device);
    printf("Device %d has compute capability %d.%d.\n",
           device, deviceProp.major, deviceProp.minor);
}
```