Faculté des Sciences & Techniques — Université de Limoges

Master 1ère année

Dév GPGPU

TP n°2

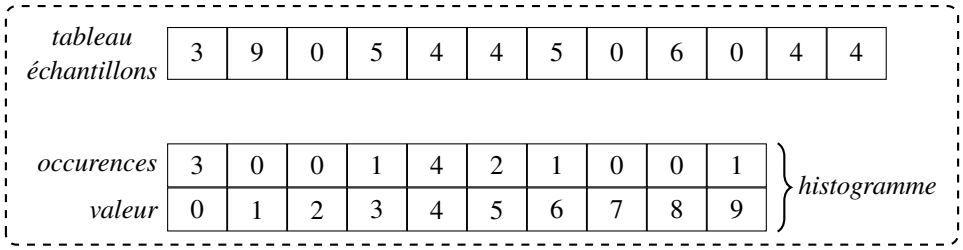
Cuda grilles, blocs & mémoire partagée

Histogramme

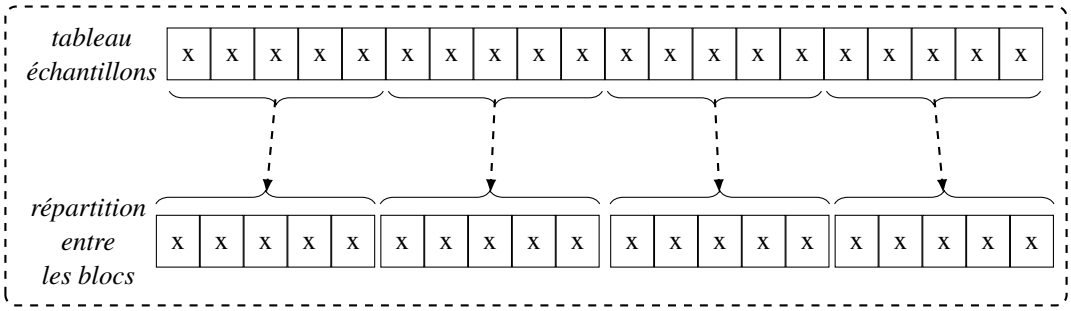
On a un tableau d'échantillons sur 10bits, c-à-d dont la valeur est comprise entre 0 et 1023 ($2^{10} = 1024$).

On veut en réaliser **l'histogramme**, c-à-d compter le nombre de fois où chaque valeur apparait :

- ▷ on parcourt les cases du tableau de valeurs ;
- ▷ pour chaque valeur rencontrée on augmente le nombre d'occurences associé.



On veut répartir le tableau d'échantillons entre différents blocs :



1 – Soit le programme Python suivant créant un fichier d'échantillons sur 2^{10} :

```
#!/usr/bin/python3

import random, struct, sys

nom_fichier = 'data.bin'
nombre_données = 536870912
taille_échantillon = 2**10

try:
    f = open(nom_fichier, 'wb')
except Exception as e:
    print(e.args)
    sys.exit(1)

# créer valeurs
for i in range(0,nombre_données):
    f.write(struct.pack('i', random.randint(0,taille_échantillon-1)))
f.close()
```

*Vous adapterez la **taille des échantillons** à la capacité de traitement de la carte CUDA installée dans votre machine.*

Écrivez le programme réalisant le **calcul de l'histogramme** et affichant les 10 premières valeurs.

Vous vous servirez de ces valeurs pour vérifier le résultat de votre implémentation CUDA.

Première méthode

On veut utiliser la méthode où les *threads* :

- ▷ sont associées à une **tranche de valeurs** du tableau d'échantillons : une thread peut traiter seule une séquence de valeurs du tableau d'échantillons ;
- ▷ partagent l'accès au tableau histogramme.

2 – a. Est-ce qu'un **problème** peut survenir lors du travail de chaque *thread* ?

b. Écrivez le *kernel* CUDA utilisant le fichier de données générées par le programme Python et calculant l'histogramme sur ces données;
Vous ferez varier la définition de la grille, des blocs et des tranches, « *SLICE* » :

```
#define N 536870912
#define SLICE 512
#define SAMPLE 1024
#define filename "data.bin"
```

Ici, une thread traite une tranche de 512 échantillons.

Pour chaque variation vous noterez le temps d'exécution obtenu avec `nvprof`.

```
xterm
$ nsys nvprof --print-gpu-trace ./compute_histo
```

c. Expliquez les différents résultats obtenus.

3 – Écrivez une nouvelle version de votre kernel CUDA utilisant la **mémoire partagée** :

- ◇ les threads d'un même bloc utilisent un histogramme **local au bloc** ;
- ◇ à la fin les différents histogrammes sont **combinés** dans **l'histogramme final**.

Vous ferez également varier les différents paramètres et étudierez les résultats obtenus.

Seconde méthode

On veut utiliser la **seconde méthode** suivante :

- ▷ chaque thread est associée à une valeur possible des échantillons, c-à-d à **une case de l'histogramme**.

4 – a. Écrivez un kernel CUDA **simple** pour cette méthode et comparez les résultats à ceux donnés par la première méthode.

b. Comment **tirer parti au maximum** de l'organisation parallèle offerte par CUDA ?

c. Est-ce que l'utilisation de la **mémoire partagée** peut être intéressante ?

d. Écrivez une version de kernel **efficace** utilisant la **mémoire partagée** et comparez les résultats obtenus.