TP2

Objectifs : Concepts avancés

Ressources :

- *http://docs.nvidia.com/cuda/index.html*
- *https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf*

Setup :

- haswell-cuda ou 192.168.80.64
- Login : nom.prenom (minuscule)
- Passwd : toto (à changer)
- Vérification bashrc, PATH et LD_LIBRARY_PATH

Depot Git :

- https://github.com/hpc-apps/Cuda.git

Exercice 1

- Modifier l'exemple d'addition de vecteurs afin de d'utiliser une version utilisant les **streams** CUDA
    - Chaque **stream** opère sur une sous-partie du vecteur (e.g deux streams).
    - Utiliser les fonctions *CudaMallocHost, CudaStreamCreate* + destructeurs et *CudaMemcpyAsync*

Exercice 2

1. Utiliser le squelette de programme proposé afin de créer deux matrices de taille N x N et de les additionner en utilisant des blocs 2D de threads.
2. Optimiser en fonction de la taille des données ainsi que des caractéristiques du GPU.

Exercice 3

Version CPU

- Récupérer le programme grayscale_ini.cpp
- Remplir tous les pixels à la valeur zero *(cv::Mat m_out(m_in.rows, m_in.cols, CV_8UC1, g.data()))*
- Compiler cet exemple
  g++ -o exe grayscale_init.cpp $(pkg-config --libs --cflags opencv)

- La conversion en niveau de gris suit la formule ci-dessous. Compléter le code fourni afin de convertir une image en niveau de gris (:qchaque pixel)

$$Grey = (307*\mathbf{Red} + 604*\mathbf{Green} + 113*\mathbf{Blue}) / 1024$$

Version GPU

- Créer le code CUDA permettant de réaliser cette conversion
- Utiliser d'abord une grille 1D puis une grille 2D de threads

Optimisation

- Instrumenter les codes CPU et GPU afin de comparer les temps d'exécution (évaluer l'impact de différentes tailles de blocs pour la version CUDA)