RAPPORT CUDA 2020

NAIL Théo

RONCIN David

Pour ce projet, nous avons décidé d’implémenter plusieurs filtres graphiques en C classique, en CUDA, en CUDA avec le shared, en CUDA avec stream et enfin en CUDA avec stream et shared. Cela permet donc de voir les effets concrets des differentes techniques. Tout les tests on été réalisé sur la machine de l’université.

Les filtres que nous avons choisi, après recherches internet, sont les suivants :

Un blur classique, qui donne ce graphe de performance(temps en milisecondes)  :

Ici, on peut voir que la partie C est légèrement plus performante que la partie CUDA classique. En revanche, on voit que les optimisations subséquentes améliorent le temps d’exécution.

La difficulté de ce filtre a été les parties qui impliquent les streams, car les manipuler efficacement est assez complexe et plante facilement si on ne fait pas attention.

Le second filtre testé est un Dilate, qui donne le graphe suivant :

Ici, on voit le gain de performance énorme dans le passage de C a CUDA(de 10.911 ms a 2.03 ms). En revanche, on remarque aussi un apport de performance présent dans les autres partie, mais minime comparé au début.

Encore une fois, la difficulté ici a été sur la partie streams.

Le filtre d’après est le filtre Emboss, qui donne ce graphe :

LE gain des performances dans ce cas est encore plus flagrant : on passe de 20.303 ms en C a 1.89 ms en CUDA. On note également une stagnation des performances après le passage en CUDA.

Le filtre suivant et un filtre erode, qui donne ce graphe :

Ici, même type de graphe. On passe de 10.9 ms a 2 ms pour le passage de C a CUDA avec une stagnation similaire après cette étape.

Le prochain filtre est un grayscale :

On remarque tout d’abords que ce graphe ne comporte que 3 entrées au lieu des 5 habituelle. En effet, nous n’avons pas réussi à le faire fonctionner en mode stream. Les données que nous avons obtenues malgré tout montre quand même une progression similaire aux graphes précèdent ; on passe de 2.6 ms en C a 0.6 ms en CUDA.

Il y a également le sobel, qui donne ce graphe :

Ce graphe montre la différence de performance la plus flagrante de nos tests. En effet, nous sommes passé de 36.8 ms en C a 2 ms en CUDA !

On remarque aussi que les performances ne changent pas vraiment après cette étape. Comme les autres graphes, il y as une baisse, mais elle ne dépasse pas les 0.4 ms.

**Problèmes et erreurs :**

Durant ce projet, nous avons rencontré beaucoup d’erreur et de bugs. Les plus présents étais des erreurs d’accès mémoires qui n’ont pas été simple à trouver et corriger vu le peu d’information fournie en message d’erreur. Des calculs mal pensés étaient aussi de la partie, qui donnait des résultat aberrant au meilleur des ; qui plantaient le programme dans les cas plus grave.

Les streams ont également été la partie la plus difficile sur ce projet. En effet, les manipuler correctement a demandé beaucoup d’effort de recherche afin d’implémenter quelque chose de fonctionnel dans un premier temps, et performant après. Ces problèmes ont été tels que le grayscale n’as pas été implémenté dans les parties streams car trop complexe et manque de temps vis-à-vis des autres matières.

Des problèmes de détections de librairies et d’installation ont également retardé notre progression sur ce projet.