

TP5 : Calcul scientifique sur GPU (2)

0 Introduction

Le but de ce TP est d'implanter un certain nombre d'algorithmes de calcul sur GPU et de comparer les temps d'exécution avec les programmes correspondant s'exécutant sur CPU.

1 SAXPY

L'URL <http://www.mathematik.uni-dortmund.de/~goeddeke/gpgpu/tutorial.html> fournit un tutoriel illustrant le calcul de SAXPY ($Y+=AX$) où X et Y sont des vecteurs de flottants simple précision et A est un scalaire (*float*) et fournit le code correspondant

- Mesurer les temps d'exécution GPU et le temps d'exécution CPU du programme fourni
- Examiner l'évolution du temps d'exécution de la version GPU et de la version CPU en fonction du nombre d'éléments du vecteur

2 Produit scalaire

En utilisant le programme précédent et le programme maximum/somme des éléments d'un vecteur du TP4, écrire le programme pour calculer le produit scalaire de deux vecteurs X et Y de N éléments. Mesurer les temps d'exécution GPU et CPU en fonction du nombre d'éléments des vecteurs

3 Résolution de l'équation de Laplace

Soit l'équation de Laplace

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \text{ sur le carré unité}$$

Conditions initiales sur les bords

$$y=0 \rightarrow u=0$$

$$y=1 \rightarrow u=1$$

$$x=0 \rightarrow u=0$$

$$x=1 \rightarrow u=0$$

Le « carré unité » est décomposé en un carré de N x N points.

Avec l'itération de Jacobi, cette équation se résout en calculant

$$u_k[i][j] = 0.25 * (u_{k-1}[i-1][j] + u_{k-1}[i+1][j] + u_{k-1}[i][j-1] + u_{k-1}[i][j+1]);$$

sur tous les points jusqu'à ce que l'erreur devienne inférieure à une valeur définie.

L'erreur est définie comme la racine carrée de la somme sur tous les points de $(u_k[i][j] - u_{k-1}[i][j])^2$

Une version (non optimale) de la version CPU est fournie sur le site Web du cours.

- a) Ecrire une version GPU pour la résolution de l'équation de Laplace
- b) Comparer les versions CPU et GPU en fonction de N (temps d'exécution et précision des calculs).