Développement GPGPU
TP n.2

Exercice 1 - Preuve de travail MD5

Prise en main du programme

a. Vous récupérerez les fichiers suivants sur `http://www.p-fb.net` :

   - cuda_md5gpu.cu
   - tester_md5.cu

b. Vous compilerez manuellement ces sources et testerez le résultat :

   - echo -n 'toto' — openssl dgst -md5

c. Vous adapterez votre programme pour pouvoir réaliser plusieurs hachés en parallèle. Attention à allouer assez de mémoire partagée dans cuda_md5gpu.cu (ligne 16) - elle doit pouvoir contenir tous les mots à hacher

Preuve de travail

La preuve de travail est un mécanisme utilisé par certaines blockchains pour récompenser les participants proportionnellement à la puissance de calcul qu'ils investissent. Elle peut être réalisée à partir d'une fonction de hachage de la manière suivante :

- Une valeur aléatoire x est fournie par la blockchain aux participants (ici on considère que sa longueur est de 32 bits par exemple)
- Les participants doivent trouver un mot commençant par x dont le haché commence par n "0" en binaire : n correspond à la difficulté du problème ; plus il est grand, plus le nombre de hachés à effectuer est grand

Vous réaliserez le travail suivant :

a. Adaptez votre programme pour qu'il réalise des hachés jusqu'à en obtenir un dont la représentation binaire commence par n "0", en utilisant un mécanisme de salage

b. A partir d'un x aléatoire, réalisez un programme permettant de fournir une preuve de travail

c. Comparez le temps nécessaire pour différentes valeurs de n

Exercice 2 - Produit de matrices

Le but de cet exercice est de réaliser le produit de matrices carrées :

$$\begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_5 \\ a_6 & a_7 & a_8 \end{bmatrix} \begin{bmatrix} b_0 & b_1 & b_2 \\ b_3 & b_4 & b_5 \\ b_6 & b_7 & b_8 \end{bmatrix} = \begin{bmatrix} a_0b_0 + a_1b_3 + a_2b_6 & a_0b_1 + a_1b_4 + a_2b_7 & a_0b_2 + a_1b_5 + a_2b_8 \\ a_3b_0 + a_4b_3 + a_5b_6 & a_3b_1 + a_4b_4 + a_5b_7 & a_3b_2 + a_4b_5 + a_5b_8 \\ a_6b_0 + a_7b_3 + a_8b_6 & a_6b_1 + a_7b_4 + a_8b_7 & a_8b_2 + a_7b_5 + a_8b_8 \end{bmatrix}$$

$$\begin{array}{c} \textit{bloc0} \\ \begin{bmatrix} a_0b_0 + a_1b_3 + a_2b_6 & a_0b_1 + a_1b_4 + a_2b_7 & a_0b_2 + a_1b_5 + a_2b_8 \end{bmatrix} \\ \textit{bloc1} \\ \begin{bmatrix} a_3b_0 + a_4b_3 + a_5b_6 & a_3b_1 + a_4b_4 + a_5b_7 & a_3b_2 + a_4b_5 + a_5b_8 \end{bmatrix} \\ \textit{bloc2} \\ \begin{bmatrix} a_6b_0 + a_7b_3 + a_8b_6 & a_6b_1 + a_7b_4 + a_8b_7 & a_8b_2 + a_7b_5 + a_8b_8 \end{bmatrix} \end{array}$$

Pour la répartition en *threads* dans chaque bloc, chaque thread va réaliser un produit :

$$\begin{array}{ccccccccccccc} \textit{t0} & & \textit{t3} & & \textit{t6} & & \textit{t1} & & \textit{t4} & & \textit{t7} & & \textit{t2} & & \textit{t5} & & \textit{t8} \\ a_0b_0 & + & a_1b_3 & + & a_2b_6 & & a_0b_1 & + & a_1b_4 & + & a_2b_7 & & a_0b_2 & + & a_1b_5 & + & a_2b_8 \end{array}$$

Les threads 0, 1 et 2 réaliseront ensuite la somme des valeurs calculées par les autres threads et contenues dans le tableau partagé par bloc.

Vous réaliserez également ce produit de manière séquentielle pour vérifier les résultats.