OpenCL - Examen sur machine

Sylvain Lefebvre, Dmitry Sokolov

April 25, 2014

1 Instructions

- Envoyez votre DM par email à `sylvain.lefebvre@inria.fr` avant le 16 mai 2014 (exclu). Indiquez `[opencl] DM` en sujet. **Tout TD non re cu avant cette date ou envoyé avec un autre sujet entrainera une note de zéro**. Envoyez vous fichiers sous forme d'une archive zip.

  Les réponses aux questions doivent être intégralement dans les fichiers code source, sous la forme de **code commenté**. La qualité des commentaires et des explications entrera pour une grande part dans la note finale.

  Ce travail est personnel. Toute détection d'échange de code ou de copie depuis une source extérieure entraînera automatiquement une note de zéro pour tous les étudiants impliqués.

  Il peut être est nécessaire de modifier le code donné pour réaliser les exercices (paramtres du kernel, etc.)

2 Introduction

Dans ce DM nous allons nous intéresser à fouiller un dictionnaire de mots à l'aide du parallélisme.

Le fichier `exam-td.cpp` montre comment charger un dictionnaire de mots. Une table `allwords` est créée. Elle contient, pour chaque mot, `longest` charactères. Sa taille est donc de `N x longest` avec N le nombre total de mots. Les mots plus courts que `longest` charactères sont complétés avec des zéros.

3 Exercice 1: Histogrammes

Ecrire un kernel qui calcule en un seul appel l'histogramme des charactères (nombre d'occurence de chaque lettre de l'alphabet) ainsi que l'histogramme des longueurs de mots. Afficher le résultat.

Note: Les charactères étant codés en ASCII leur valeur numérique est dans [0,255].

4 Exercice 2: Plus longue sous-séquence commune

Nous nous intéressons au problème de trouver dans une table de chaînes de charactères celles ayant la plus longue sous séquence commune (Longest Common Sub-Sequence, LCSS) avec une chaine donnée en entrée.

Le pdf joint, `lcss.pdf` explique comment calculer la LCSS et donne le pseudo code (dernier slide).

1. Implémenter en C++ sur CPU la fonction `int lcss(char *str0,char *str1)` qui renvoie la LCSS entre les chaînes str0 et str1 en utilisant une table de programmation dynamique.

2. Implémenter un kernel qui compare chaque chaîne du dictionnaire à une même chaîne requête `query`. Chaque thread calcule une LCSS entre `query` et une chaîne du dictionnaire, et écrit dans un tableau la valeur trouvée.

3. Implémenter un kernel qui renvoie la liste de toutes les chaînes ayant la plus petite valeur de LCSS par rapport à la requête.

5 Exercice 3: LCSS entre deux très longues chaînes

Au lieu d'avoir un grand nombre de petites chaînes, nous souhaitons calculer la LCSS entre deux très grandes chaînes. Est-il possible de paralléliser le calcul d'une seule LCSS? Si oui, comment? Proposez une implmentation.