TP : méthode de la puissance sur GPU

Version du 9 novembre 2020

(d'après un sujet de STR de C. Bouillaguet)

1 Introduction

La méthode de la puissance permet de calculer (pour certaines matrices) la plus grande valeur propre (en valeur absolue) d'une matrice ainsi qu'un vecteur propre associé à cette valeur propre. Cette méthode est décrite par exemple ici : `https://en.wikipedia.org/wiki/Power_iteration`
Cet algorithme est assez simple, mais il est au coeur d'applications réelles comme l'algorithme PageRank de Google.

Le problème est que pour des grosses matrices le temps de calcul devient (trop) long. Nous allons donc déployer ce calcul sur GPU pour l'accélérer. Etant donnés les GPU dont nous disposons, les calculs seront faits en simple précision, et le seuil d'erreur sera fixé à 10^{-5}.

2 Travail à faire

1. Vérifiez que vous avez accès aux machines de Grid'5000 en lançant depuis une salle de TP sous linux à Polytech (depuis votre compte où est stockée votre clé ssh) :

   ```
   $ ssh -t votre_login@access.grid5000.fr ssh lille
   ```

   Vous devez vous retrouver sur une machine appelée "flille".

   Pour l'édition des fichiers nous allons monter le système de fichier via ssh avec les commandes (à lancer sur votre machine locale, dans un autre terminal) :

   ```
   $ mkdir G5K_lille
   $ sshfs -o idmap=user votre_login@access.grid5000.fr:/mnt/home/lille/votre_login   G5K_lille
   ```

   Vérifiez que vous pouvez alors créer des répertoires et éditer des fichiers en local sur votre machine dans le répertoire `G5K_lille`, et que ceux-ci sont visibles (commandes `ls` et `cat`) sur la machine de Grid'5000.

   Important : à la fin de votre session de travail vous devez "démonter" le système de fichier monté via ssh avec la commande

   ```
   $ fusermount -u G5K_lille/
   ```

2. Pour réserver et accéder à une machine avec un GPU Nvidia GTX 1080 Ti, vous devez lancer la commande :

   ```
   $ oarsub -l /cpu=1/gpu=1/core=2/,walltime=2:0:0 -I -p "gpu_model = 'GeForce GTX 1080 Ti'"
   ```

   Vous devez vous retrouver sur une machine appelée "chifflet-X".

   Exécuter le programme :

   ```
   $ /usr/local/cuda/extras/demo_suite/deviceQuery
   ```

   ou la commande :

   ```
   $ nvidia-smi -a
   ```

   pour afficher les informations disponibles sur le GPU présent dans votre machine.

3. Récupérer les codes suivants :
   — `sequential.c` : une implémentation séquentielle de la méthode de la puissance. Ce code génère une matrice à peu près aléatoire de la taille $n \times n$ de votre choix, puis va en calculer un vecteur propre (de taille n), et enfin sauvegarder ce vecteur dans un fichier.
   — `checker.c` : code source d'un programme qui sert à vérifier le résultat des calculs (à compiler avec `-fopenmp`).

On pourra écrire tout le code (code pour l'hôte et code pour le GPU) dans un unique fichier avec le suffix `.cu` et le compiler avec le compilateur CUDA `nvcc` avec les options suivantes (pour GTX 1080 Ti, de *compute capability* 6.1). : `--generate-code arch=compute_61,code=sm_61 -O3`

4. Combien de kernels CUDA différents sont nécessaires pour déployer la méthode de la puissance sur GPU ?
5. A-t-on besoin d'effectuer des transferts de données CPU-GPU entre ces kernels ? Et entre plusieurs itérations de la méthode de la puissance ?
6. Mettez en place un premier déploiement (simple), et comparez les performances du GPU par rapport au CPU.
7. Comment optimisez-les performances ?
8. (facultatif) Une implémentation des routines BLAS est disponible en CUDA : CUBLAS (voir `https://developer.nvidia.com/cublas`).
   Réécrivez une version de votre code avec ces routines CUBLAS et comparez les performances de votre meilleure version sans BLAS avec cette version CUBLAS.