Modèles et techniques en programmation parallèle hybride et multi-cœurs

Travail pratique 3

Marc Tajchman

CEA - DEN/DM2S/STMF/LMES

24/1/2023

Travail pratique 3

On part de deux codes qui calculent une solution approchée du problème suivant :

Chercher u*:* $(x,t) \mapsto u(x,t)$*, où* $x \in \Omega = [0,1]^3$ *et* $t \geq 0$*, qui vérifie :*

$$\begin{aligned} \frac{\partial u}{\partial t} &= \Delta u + f(x,t) \\ u(x,0) &= g(x) && x \in \Omega \\ u(x,t) &= g(x) && x \in \partial\Omega, t > 0 \end{aligned}$$

où f *et* g *sont des fonctions données.*

On utilise des différences finies pour approcher les dérivées partielles et on découpe Ω en $n_0 \times n_1 \times n_2$ subdivisions.

Codes de départ

Récupérer et décompresser le fichier `TP3_incomplet.tar.gz`.

Cette archive contient 2 sous-répertoires :

- la version séquentielle (`PoissonSeq`),
- une version incomplète, parallélisée avec Cuda (`PoissonCuda`),

La version séquentielle sert de référence.

Le but du TP est de compléter la version Cuda.
La partie du code à modifier se trouve dans les fichiers:

`PoissonCuda/src/cuda/iteration.cu`
`PoissonCuda/src/cuda/user.cu`

Pour `iteration.cu`, on prendra les lignes 69 à 79 de `scheme.cxx` comme modèle.
Pour `user.cu`, on prendra comme modèle le fichier `user.cxx`

Un exemple de code écrit en cuda pour initialiser le tableau de valeurs :

- la fonction `init` entre les lignes 68 à 78 et
- le noyau cuda `initValue` entre les lignes 51 à 66

du fichier `PoissonCuda/src/cuda/values.cu`

Sur le cluster Cholesky (où Cuda est installé et les nœuds de la partition “gpu” contiennent des cartes graphiques)

Pour compiler et exécuter

- dans le répertoire `PoissonSeq`, tapez :

  ```
  python submit_run.py
  ```

- dans le répertoire `PoissonCuda`, tapez :

  ```
  python submit_run.py
  ```

`submit_run.py (re)compile avant d'exécuter`

Les affichages sont dans le fichier `output.txt`.

Si vous n'utilisez pas le cluster Cholesky, il faut utiliser une machine

- avec une carte graphique compatible (ç-à-d. une carte Nvidia), et
- où le toolkit Cuda est installé (https://developer.nvidia.com/cuda-downloads)

Pour compiler et exécuter

- dans le répertoire `PoissonSeq`, les 2 commandes sont :

  ```
  python build.py
  ./install/release/PoissonSeq
  ```

- dans le répertoire `PoissonCuda`, les 2 commandes sont :

  ```
  python build.py
  ./install/release/PoissonCuda
  ```

Envoyez votre code source (dans une archive compressée) par mail à marc.tajchman@cea.fr **avant le 18/2/2024**.