Programmation hybride et multi-coeurs

Romain Horcada

Février 2021

1 Présentation du problème

1.1 Système d'équations

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

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

où f et g sont des fonctions données.

1.2 Programmes utilisés

On dispose initialement de plusieurs codes servant à résoudre ce problème. Premièrement, une version séquentielle utilisant les différences finies. Ensuite, trois versions parallélisées, via OpenMP (grain fin), MPI et Cuda.

2 Modifications concernant le code

Une des options possible est de partir de la version MPI et de "l'hybrider" avec du code CUDA au fur et à mesure. Dans cette optique, il faudrait modifier principalement le fichier "scheme" pour pouvoir, à la fin, créer des échanges CPU/GPU. Une fois les modifications effectuées sur les fichiers, il faudrait modifier les scripts en eux-mêmes.

Initialisation

Premièrement, on exécute le main.cxx MPI (inchangé) et on initialise les paramètres du problème ainsi que le schéma associé qui est représenté par la structure scheme de type MPI.

Calcul sur GPU et transferts

Une fois l'initialisation réalisée, on rentre dans la boucle en temps via l'utilisation de la fonction iteration() de scheme (scheme : :iteration). On commence par calculer les indices associés aux sous-domaines affecté à un thread MPI. Ce calcul est fait de la même manière que dans la version MPI classique. On utilise ensuite la version CUDA de la fonction "iteration domaine" de manière à pouvoir calculer la solution du sous-domaine MPI sur GPU. Cette opération peut se découper en 2 étapes dont les calculs sont réalisés sur GPU : 1) Calcul de la solution locale sur GPU via iterationWrapper(), mise en place d'une barrière CUDA et envoi de la solution au CPU. 2) Utilisation de variationWrapper() pour calculer la variation locale GPU, mise en place d'une barrière CUDA et envoi de la valeur de la variation au CPU.

Réduction

Une fois ces étapes de calculs/communications GPU → CPU effectués, on place une barrière MPI pour que la solution et sa variation soient connues pour chaque sous-domaine. On peut ensuite réaliser une réduction MPI pour obtenir la valeur de la variation globale. On ne peut pas à ce stade réduire la valeur de la solution car il faut prendre en compte la valeur de la solution aux bords des sous-domaines.

Gestion des valeurs aux bords

Il reste alors à échanger les valeurs aux bords. On utilise alors, dans le main MPI, la fonction synchronize. A ce stade, il ne reste alors qu'à retourner la valeur de la solution et du temps de calcul si on le désire.

3 Modifications concernant les données

Stockage des solutions

Le calcul étant effectué sur GPU, il faut adapter la manière dont sont échangées les données et les résultats de calculs. On utilise alors les pointeurs qui sont présents dans les scripts values.cxx et values.hxx. Il y a un pointeur vers le lieu de stockage de la solution CPU (h_u), un autre pour la solution GPU (d_u). Enfin, le booléen $h_{synchronized}$ nous informe si ces deux solutions sont égales ou non. Pour chaque nouvelle itération, il faudrait affecter la valeur false à ce booléen tant que la valeur de la solution calculée sur GPU n'a pas été envoyée vers le CPU. Ainsi, chaque solution est stockée grâce à des pointeurs constants associés à chacune des solutions, et le booléen permet de décrire l'évolution de l'échange CPU/GPU.

Echange de données

Comme nous l'avons vu jusqu'à présent, une version hybride CUDA/MPI est possible à condition de réaliser un certain nombre de communications entre le CPU et le GPU. Nous allons détailler dans cette section comment s'effectue les échanges. Premièrement, les calculs et échanges effectués concernant la valeur de la solution :

1. Calcul des sous-domaines MPI $(i_1, i_2, ..., i_n)$
2. Calcul en parallèle de la solution sur chacun des sous-domaine MPI i_k
3. Envoi des valeurs obtenues au GPU
4. Calcul de la solution GPU sur chacun des i_k.
5. Barrière CUDA et communication de la valeur de la solution vers le CPU

Ensuite, les calculs et échanges concernant la variation :

1. Réception par le GPU des dernières valeurs de la solution calculées sur le CPU
2. Calcul GPU en parallèle des variations locales de la solution sur les sous-domaines i_k
3. Barrière CUDA et envoi au CPU de la valeur de la variation sur chaque sous-domaine i_k
4. Barrière MPI et réduction pour calculer la valeur de la variation globale (calcul effectué par le CPU)

Enfin, le CPU se charge d'échanger les données aux bords pertinentes (solution, variation ...) entre threads MPI.