TP2 : OpenMP 2

Version du 4 février 2022

Exercice 1 – Calcul de π (suite)

On reprend l'exercice du calcul de π de la feuille précédente.

1. Comment paralléliser de façon efficace ce calcul en utilisant la directive de partage de travail `for` ?
   Comparez la performance ainsi obtenue à la meilleure performance obtenue précédemment.

Exercice 2 – Calcul de fractales (suite)

On reprend l'exercice sur le calcul de fractales (ensemble de Mandelbrot) de la feuille précédente.

1. Réécrire les deux versions parallèles (écrites en mode SPMD) de la feuille précédente en utilisant la directive de partage de travail `for`, et vérifier les performances obtenues.
   Quel est l'intérêt de programmer avec des directives de partage de travail plutôt qu'en mode SPMD ?

Sauf mention contraire, on privilégiera désormais en OpenMP l'utilisation des directives de partage de travail à la programmation SPMD.

Exercice 3 – Produit matriciel

On considère un code de produit matriciel du type $C = A \times B$ pour des matrices carrées d'ordre N. Pour rappel, le produit matriciel consiste à calculer chaque élément de la matrice C ainsi :

$$C_{i,j} = \sum_{k=0}^{N-1} A_{i,k}.B_{k,j}, \quad \forall (i,j) \in [\![0, N-1]\!]^2.$$

En d'autres termes, l'élément $C_{i,j}$ est le résultat de produit scalaire de la ligne i de A et de la colonne j de B.

Le fichier `matmul.c`, à utiliser par défaut avec des matrices carrées d'ordre $N = 512$, contient le code source d'un programme de multiplication de matrices.

— Après avoir compris et exécuté le code séquentiel fourni sur quelques exemples, identifier la ou les boucle(s) parallèles du produit matriciel dans le code séquentiel.
— Etablir la stratégie de parallélisation multi-thread adaptée (choix de la ou des boucle(s) à paralléliser, choix de l'équilibrage de charge adapté ...).
— Implémenter votre parallélisation en OpenMP (avec des directives de compilation).
— Vérifier que le résultat de votre exécution parallèle est correct au niveau des « coins » de la matrice.
— Calculer les efficacités parallèles obtenues avec différents nombres de threads (1 thread par cœur physique), et analyser les performances obtenues.

