TD OpenMP

Pierre-Etienne Polet (pierre-etienne.polet@inria.fr)

Le but de ce TD est de vous faire résoudre des systèmes d'équations différentielles ordinaires en utilisant la méthode dite de Euler explicite (voir le pseudo code fourni).

Le programme à réaliser Le code que vous devez compléter est fourni. Le chargement des fichiers de données, la définition des structures de données (qui peuvent être changées si vous le souhaitez) ainsi que la validation (vérification de l'exactitude des résultats) sont donnés. Vous n'avez qu'à compléter le code permettant la résolution du problème.

Le peudo-code de résolution d'un système est le suivant pour le faire tourner pendant *nbStep* pas de temps pour un système avec *system_size* equations. Les tableaux *variable_value_t* et *variable_value_prev_t* contiennent respectivement les valeurs des variables du système au pas de temps t (*i.e.* celui qu'on est entrain de calculer) et celui de $t-1$ (*i.e.* ceux qui ont déjà été calculés). La matrice *value_matrix* contient les poids pour chaque équation. Par exemple, la ligne 0 contient les poids de l'équation 0. La case 0 de la ligne 0 contient le poids (a_0) pour la variable $x0$ et ainsi de suite. Pour un système de taille 3, on aura donc les équations suivantes $x0_t = a_0 * x0_{t-1} + a_1 * x1_{t-1} + a_2 * x2_{t-1}$. Afin d'éviter des problèmes de dépassements de taille de codage des doubles (et vu que les valeurs sont tirées en aléatoire, c'est possible que cela arrive), les valeurs calculées sont normalisées.

```
for t = 0; t < nb_step; t++ do
    min = variable_value_prev_t[0]
    max = variable_value_prev_t[0]
    for i = 0; i < system_size; i++ do
        variable_value_t[i] = 0
        if min > variable_value_prev_t[i] then
            min = variable_value_prev_t[i]
        end if
        if max < variable_value_prev_t[i] then
            max = variable_value_prev_t[i]
        end if
    end for
    for i = 0; i < system_size; i++ do
        for j = 0; j < system_size; j++ do
            variable_value_t[i]+ = variable_value_prev_t[j] * value_matrix[i][j]
        end for
        variable_value_t[i] = (variable_value_t[i] − min)/(max − min)
    end for
    for i = 0; i < system_size; i++ do
        variable_value_prev_t[i] = variable_value_t[i]
    end for
end for
```

Evaluation de performance Il est nécessaire, en plus d'avoir un code fonctionnel, de fournir les performances en passage à l'échelle fort et faible. Le programme affiche le temps de calcul (*e.g.* DURATION,nombre de système ou taille du système, nombre de pas de temps, temps de calcul).

Passage à l'échelle fort En passage à l'échelle fort, on garde la taille du problème (taille des vecteurs et des matrices) et on augmente le nombre de processeurs. Le minimum est de tester sur 1, 2 et 4 cœurs dans votre cas. Si possible, allez plus loin.

Passage à l'échelle faible En passage à l'échelle faible, on augmente la taille du problème avec l'augmentation du nombre de cœurs. En pratique, vous multipliez la taille du problème par 2 pour 2 coeurs et 4 pour 4 coeurs.

Jeux de tests Des jeux de tests pour un système vous sont fournis pour des tailles (*i.e.* nombre d'équations) de 64, 128, 256, 512, 1024, 2048, 4096 et 8192. Les valeurs ont été tirées aléatoirement.

Des jeux de tests pour des systèmes vous sont fournis pour des tailles (*i.e.* nombre de systèmes) de 64, 128, 256, 512 et 1024. Les valeurs et le nombre d'équations de chaque système ont été tirés aléatoirement.

Utilisation de la suite logiciel Vous devez compléter les fichiers `cpp`. L'utilisation des 2 programmes est similaire *./binary fichier_en_entree nbStep fichier_de_validation*. Le premier argument (*fichier_en_entree*) est le fichier contenant les données d'initialisation. Les fichiers sont nommées en suivant le motif suivant:

- Pour la première partie, le fichier commence par $matrice\_file\_$ suivi par la taille du système.
- Pour la deuxième partie, le fichier commence par $multiplie\_matrice\_file\_$ suivi par le nombre de systèmes.

Pour le dernier argument, les fichiers sont nommées en suivant le motif suivant:

- Pour la première partie, le fichier commence par $validation\_matrice\_file\_$ suivi par la taille du système et finalement le nombre de pas de temps (*nbStep*).
- Pour la deuxième partie, le fichier commence par $validation\_multiplie\_matrice\_file\_$ suivi par le nombre de systèmes et finalement le nombre de pas de temps (*nbStep*).

1 Exercice

1.1 Résolution d'un système

Question 1.1 Compléter le code fourni

Question 1.2 Paralléliser le code avec OpenMP (vous pouvez proposer plusieurs solutions)

Question 1.3 Faire l'évaluation de performance ainsi que tracer les graphiques pour des nombres de pas de temps de 100, 500, 1000, 5000, 10000, 50000

1.2 Résolution de plusieurs systèmes d'équations

Question 1.4 Compléter le code fourni

Question 1.5 Paralléliser le code avec OpenMP (vous pouvez proposer plusieurs solutions)

Question 1.6 Faire l'évaluation de performance ainsi que tracer les graphiques pour des nombres de pas de temps de 100, 500, 1000

Question 1.7 Est-ce que vous voyez une raison pourquoi la résolution de plusieurs sytèmes d'équation est plus complexe que celui d'un seul système (en dehors du nombre d'équations)