TP3 OMP

Ariel Eddie GUIDI

19 décembre 2019

1 EXEMPLE OPENMP "GRAIN FIN"

On utilise le fichier src/sinus_fine/sinus.cxx pour évaluer le temps de calcul de la version "grain fin" (automatic attribution of loops to threads)

On obtient les résultats suivant pour plusieurs essais des codes :

sinus_seq (real/user)	4.84/4.84	4.86/4.86	4.85/4.85
sinus_fin (real/user)	1.86/5.51	1.92/5.52	1.89/5.67

Le partage des tâche sur 3 threads accélère donc comme attendu le temps de calcul. On ne divise pas exactement le temps de calcul par 3 en mettant 3 threads parce qu'il y a une partie séquentielle obligatoire dans le code.

2 EXEMPLE OPENMP "GRAIN FIN"

L'option schedule fait référence à la manière dont le travail est fait par les threads. Il repartit les itérations en paquets pour les différents threads.

Pour schedule (static...) le travail fait par les threads est décidé dès le début.
Pour schedule (dynamic...) lorsqu'un thread a terminer sur une itération, il commence automatiquement une autre itération sur laquelle aucun autre thread n'a travaillé. Ceci permet une meilleure répartition du travail.

Résultats obtenus (valeurs moyennes) :

n/...	1	4	8	12	16	24
Static (real)	4,94	2.40	2.33	2.05	2.05	1.99
Dynamic (real)	4.85	2.30	2.06	1.86	1.87	1.86

On remarque aisément que le temps minimum est plus rapidement atteint avec le dynamic car les charge de travail se réparti mieux entre les threads.

3 EXEMPLE OPENMP "GRAIN GROSSIER"

Le grain grossier consiste a faire répartir les taches entre les threads par l'utilisateur. La répartition gros grain est effectuée dans le fichier src/sinus_coarse_1/sinus.cxx

Les résultats obtenus (moyenne sur 5 essais) enexécutant les codes sinus_fine vs sinus_coarse_1 sont :

fine	1.94
coarse_1	2.12

On remarque que le version coarse est légèrement plus lente que la version fine car la répartition des taches entrée par l'utilisateur peut ne pas être optimale.

Dans la proposition de répartition des tâches proposée dans le fichier, lorsqu'il s'agit d'une entrée telle que n (modulo) nthreads != 0 le dernier thread prends le travail restant. Donc il met logiquement plus de temps à terminer que les autres.

4 PARALLELISATION D'UN MINI CODE AVEC MODELE GRAIN GROSSIER

On utilise simplement la répartition de taches utilisée en 3 pour paralléliser le schéma numérique avec le modèle grain grossier.
fichier modifié : code/PoissonOpenMP_CoarseGrain/src/scheme.cxx

compilation non réussie - fichier manquant

5 PARALLELISATION AVEC CONCEPT DE TACHES OPENMP

Le present exemple est difficile à paralléliser car on utilise une liste de taille variable. On ne peut alors pas répartir les taches sur les threads de manière conventionnelle car on ne peut n pas définir un nombre n représentant la taille de la liste pendant les itérations.

On utilise donc le concept de taches OpenMP.
Le thread principal crée des taches pour effectuer le travail au fur et a mesure qu'on avance dans la liste.
temps moyen de calcul version séquentielle : 0.018s
temps moyen de calcul version parallèle 3 threads : 0.05s