Programmation sur architectures parallèles

Devoir 2

- étudiants : Basile FETET, Luc LIBRALESSO, Olivier SOLDANO
- login: mpi14
- mdp: dAvAuHK

rendu

Sont présents dans le répertoire

- d1s.c qui est la version séquentielle de l'algorithme
- d1p.c qui est la version parallèle avec openMP
- d2p.c qui est la version parallèle avec pthread. On note que d2p utilise aussi openMP pour le calcul du temps d'exécution.
- Makefile qui continent les règles pour compiler les trois programmes et lance les tests sur l'instance la plus grosse (inst4.txt)
- gen.py qui contient le script qui permet de générer des instances de taille quelconque pour tester les programmes.
- inst[1-4].txt des instances de différentes tailles (inst4 contient 200 000 éléments)

Analyse

Nous avons effectué quelques tests sur les différents programmes. Nous obtenons des résultats avec une instance de 200 000 éléments en entrée de l'ordre de :

- 0.015 secondes pour d1s
- 0.002 secondes pour d1p
- 0.060 secondes pour d2p

De ces résultats, nous constatons que la version openMP est en moyenne 7-8 fois plus rapide que la version séquentielle, ce qui est un gain intéressant. Cependant, la version pthread est environ 4 fois plus lente que la version séquentielle.

Ce dernier résultat est surprenant car la charge de travail pthread se limite à faire un 'spawn' sur le premier appel récursif dans l'algorithme et un join après.

Note sur la réalisation des algorithmes

Pour l'implémentation des algorithmes parallèles, nous avons opté pour appliquer un cutoff afin d'améliorer les performances. L'idée est que pour les petits problèmes (dont la taille est inférieure à une constante (CUTOFF), une version séquentielle de l'algorithme sera appliquée. Et dans les autres cas, la version parallèle.