TP2 - Comptage de Schur Avec Parallélisme

Guilherme Dias da Fonseca - Programmation répartie

Ce TP est basé sur le TP précédent. On veut paralléliser l'algorithme pour résoudre le même problème plus rapidement en utilisant les plusieurs cores de la CPU. Le problème du TP1 est un problème de comptage qui on peut facilement paralléliser, sans avoir encore vu les techniques de synchronisation. La stratégie adoptée est de décomposer la boucle extérieure parmi les plusieurs threads. On ne change pas la boucle intérieure.

Dans le TP précédent, l'algo était : pour chaque i, j tels que $i + j \leq n$, vérifier si $V[i] = V[j] = V[i + j]$ et si oui, incrémenter le compteur x. Maintenant, chaque thread va s'occuper de quelques valeurs de i. Comme le temps que la boucle intérieure prend réduit quand i augmente, la stratégie la plus simple pour faire ce partage est d'alterner les valeurs de i entre les threads. Si on a 4 threads, un thread prend $i = 1, 5, 9, 13, \ldots$, un autre thread prend $i = 2, 6, 10, 14, \ldots$, etc.

Notez qu'on peut pas modifier une même variable au même temps dans plusieurs threads. Alors, chaque thread doit avoir son propre compteur. Quand tous les threads sont finis, le thread principal somme les compteurs de chaque thread et affiche la solution.

On doit ajouter une deuxième paramètre au logiciel, pour indiquer le nombre de threads à utiliser. Par exemple :

```
$ time java tp2.MainTp2 1000000.schur 2
Thread 1 est fini
Thread 2 est fini
Total : 25006122

real    0m54.243s
user    1m46.056s
sys     0m0.052s

$ time java tp2.MainTp2 1000000.schur 8
Thread 5 est fini
Thread 3 est fini
Thread 4 est fini
Thread 8 est fini
Thread 2 est fini
Thread 7 est fini
Thread 1 est fini
Thread 6 est fini
Total : 25006122

real    0m24.004s
user    3m7.452s
sys     0m0.060s
```

Vérifiez le fonctionnement correct de l'algo et que tous les threads finissent plus ou moins au même temps. Notez les temps d'exécution (utilisez le commande "time" sur linux et notez le temps réel et le temps utilisateur) avec 1, 2, 4, 8 et 16 threads. Le temps utilisateur compte la somme du temps de cpu utilisé dans tous les threads. Comparez les temps et trouvez le nombre de vrais cores de la CPU utilisée (si on a plus de threads que de vrais cores, le temps réel ne baisse pas beaucoup mais le temps utilisateur augmente bien).