Alexandre COR

2 cœurs

4 processeurs logiques

Cache 1: 128 Ko

Cache 2 : 512 Ko

Cache 3 : 3 Ko

Exercice 1)

**Syracuse simple**

On parallélise la boucle for avec openmp, voici mes résultats:

Sans paralléliser puis en parallélisant.





Les résultats sont cohérents et on s’aperçoit que l’on a un speed up de 1.7 pour 4 threads.

1.35 pour 2 threads

1.7 pour 4 threads

2.4 pour 6 threads

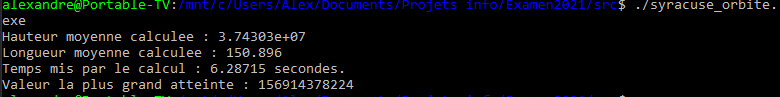
2.4 pour 8 threads

2.5 pour 12 threads

Ensuite on stagne si on augmente de nombre de threads. On s’aperçoit qu’augmenter le nombre de threads améliore la performance du calcul. C’est assez logique car l’algorithme étant limité par le nombre d’itérations à faire et par le calcul. Cependant on s’aperçoit qu’a partir d’une petite dizaine de threads, on stagne en vitesse de calcul, cela est dû à un memory bound car trop de threads veulent accéder à la mémoire et cela “bouche” l’accès à la mémoire.

**Syracuse avec orbite**

Voici le resultat sans paralléliser:



En parallélisant on obtient des speed up différents:

Pour 2 threads: 1.11

Pour 4: 1.12

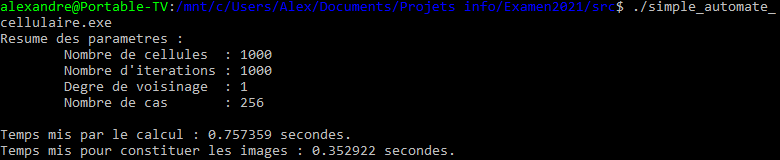
Pour 8: 1.05

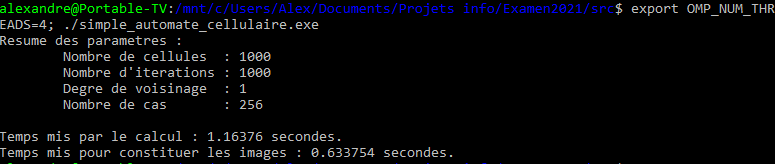
On n’obtient pas réellement de speed up conséquent. Cela est dû à un problème à l’accès mémoire car ici en plus on manipule des objets avec beaucoup de valeurs et tous les threads veulent accéder au même tableau. On a donc des problèmes d’accès mémoire. De plus l’algorithme est plus long que le premier globalement car les techniques de calculs sont moins efficaces, on s’encombre d’un tableau qu’on manipule, alors qu’il nous faut seulement des valeurs moyennes et un maximum.

Au moins, on garde les mêmes résultats.

Exercice 2)

**Parallélisation OpenMP**

Résultat sans paralléliser:



Voici le résultat avec 4 threads, on s’aperçoit que le calcul est plus long. Le speed up est de 0.62.

Plus on ajoute de thread, plus le calcul est long, avec 8 threads, le speed up est de 0.38.

On a fait le speed up avec le temps de calcul mais il est équivalent avec le temps mis pour constituer les images.

Le problème se trouve donc dans la mémoire. Les threads se bouchent en accédant à la mémoire.

**Parallélisation MPI**

On divise la dimension en parties égales en fonctiion du nombre de processus.

On doit envoyer les valeurs fantomes, donc les valeurs limites aux au bord des dimensions en utilisant des Send Recv. Et les valeurs au bord son égales à 0.