Bonneville Geoffrey - TP1

Choix

L'option -p supprime l'affichage des résultats. L'option -t permet d'éxécuter les tests.

Problèmes de mise au point

Le principal soucis à été de respecter le prototype de la fonction colimacon. J'ai écrit une première version en "trichant" : J'avais alloué un tableau représenté par un double pointeur, avant de mettre à jour le tableau passé en argument avec ce dernier.

La version finale utilise bien un tableau à une seule dimension, ce qui est au final plus clair.

Prototype préféré

On pourrait utiliser un tableau à double pointeur, avec une allocation "ligne par ligne". Ainsi, on pourrait profiter de la syntaxe du type array[ligne][colonne]. Je n'ai pas compris l'intérêt d'utiliser des entiers non-signés pour les lignes et colonnes, puisque les cases du tableau sont finalement des entiers signés.

Impact vis-à-vis des accès mémoire.

La fonction colimacon alloue un tableau de manière contigue en une seule fois. (m * n * sizeof(int)). L'algorithme parcourt ensuite exactement une seule fois chaque case du tableau. La complexité est donc de n pour n cases de tableau à remplir.

Paralellisation

Actuellement, on rempli le tableau "de l'exterieur vers l'intérieur". Pour lancer le programme sur 2 threads, il suffirait d'effectuer ce remplissage jusqu'à n/ (n = nombre de cases / 2) avec un premier thread, alors que le deuxième thread s'occuperait de remplir de la case n/2 jusqu'à la dernière case.

Après quelques essais en utilisant des "pthread", il apparait que l'intérêt est nul pour ce programme, car la vitesse d'exécution semble de toute manière limitée par les accès mémoire, du moins sur ma machine...

Autre autre solution pourait être de remplir avec 2 threads une case sur deux.

Temps

Estimation initiale: 3H30 Au final: environ 6h. (+2 heures pour les essais avec pthread)

Commentaires des étudiants

C'est un élève qui m'a indiqué qu'il ne fallait pas passer par une solution avec double pointeur. Après une demande d'explications par le professeur, j'ai relayé l'information et aidé d'autres élèves.

Questions subsidiaires

Temps de calculs

affichages (option -p) et tests (option -t) supprimés. options GCC: -march=native -O3

pour le temps d'éxécution, utilisation de la commande "time" pour la mémoire occupée, utilisation de "valgrind".

10x10: 0m0.001s - 400 bytes allocated

100x100: 0m0.002s - 40 000 bytes allocated

1000x1000: 0m0.008s - 4 000 000 bytes allocated

10 000x10 000: 0m0.861s - 400,000,000 bytes allocated