Tri parallèle d'un tableau avec OpenMP

Pierre AYOUB – Océane FLAMANT

9 novembre 2018

Table des matières

Introduction

L'objectif de ce projet est, dans un premier temps, d'implémenter en c, un algorithme de trie pour une grande base de données de manière à ce que l'efficacité soit la plus grande possible.

Dans un second temps, il faut calculer le temps d'exécution de notre programme en fonction de plusieurs paramètres tels que le nombre de threads utilisé ou la taille des données.

Comme contraintes nous avions :

— le squelette du programme : donné dans l'énoncé.

— les trois fonctions :

generator : remplit les tableaux avec des nombres réels.

tri : prend en entrée un tableau à une dimension de taille K non triés et le renvoie trié et enfin une fonction.

tri-merge : prend en entrée deux tableaux à une dimension de taille K triés et renvoie deux tableaux triés, toutes les valeurs du premier tableau seront inférieures à celles du deuxième.

— le parallélisme avec OpenMP : nous avons donc utilisé la bibliothèque de programmation parallèle OpenMP.

Développement du programme

Dans cette partie nous allons vous expliquer les choix que nous avons fait pour implémenter l'algorithme demandé ainsi que les modifications qui ont dû être effectuées.

Structure et choix

La structure globale suit l'algorithme défini dans l'énoncé mais nous allons vous détailler, dans cette partie, certaines partie du programme.

Avant d'implémenter les fonctions, nous avons de créer des macroconstante qui permettent de calculer le minimum et le maximum dans le but d'optimiser et de faciliter la compréhension du code. Nous avons aussi mis les tailles des données en static car elles ne doivent pas être modifiées au cours de l'exécution du programme.

Pour l'algorithme de tri nous avons choisi le tri à bulles car même si ce n'est pas le plus rapide que l'on connaît aujourd'hui, il est simple à comprendre et à implémenter et il n'a pas besoin de tableau annexe pour que le tri s'effectue. De plus nous avons choisi de nous concentrer sur le parallélisme du programme et la différence de son temps d'exécution suivant les différents critères, non pas sur le temps d'exécution global.

Modification de l'algorithme

Nous avons rencontré quelques déconvenues avec l'algorithme donné.

En effet une fois nos fonctions réalisées et vérifiées nous avons lancé le programme et le résultat obtenu n'était pas le bon. Les valeurs contenues dans les tableaux étaient triés mais pour les tableaux entre cela n'était pas le cas. Nous avons donc dû effectuer de léger changement.

Tout d'abord, chaque boucle for commence à 0 et s'arrête à N. Ensuite nous avons supprimé les +1 des variables k, b1 et b2 car les boucles commencent à 0.Et enfin pour b2, le K est devenu k.

En résumé, k=(j%2) ; b1=1+(k+2*i)%N -> b1=(k+2*i)%N ; b2=1+(K+2*i+1)%N -> b2=(k+2*i+1)%N.

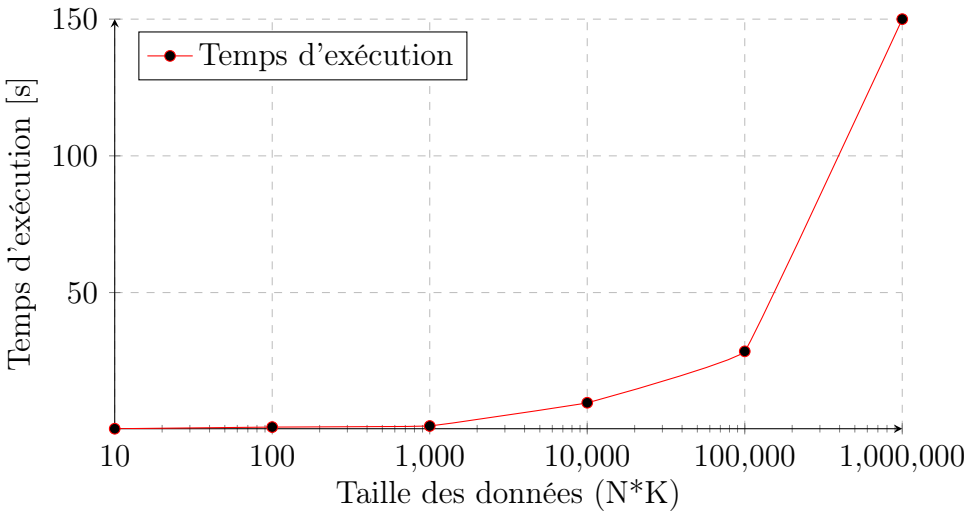

FIGURE 1 – Temps d'exécution en fonction de la taille des données (N*K)

Performances

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Mesures

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Perspectives d'amélioration

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Conclusion

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.