Dan Nakache

Le tri « radix » :

1. Gestion du développement :

Je suis reparti du TP2 comme base de code en utilisant les fonctions du préfixe et du suffixe.

Mon code est structuré en plein de fonctions que j'appelle les unes dans les autres (structure imbriquée).

Il n'y a qu'un seul fichier source qui contient tout le code.

2. Problèmes techniques spécifiques rencontrés :

Tout d'abord, dû au retard accumulé dans les TP, j'ai dû passer du temps sur la version suffixe de la descente puisque c'est la seule étape qui change par rapport au préfixe.

Le passage d'une taille de tableau de 2^n à n'importe quelle taille a été problématique. En effet, il a fallu décomposer le tableau en sous-tableaux de 2^n puis faire les opérations sur chacun de ces sous-tableaux pour ensuite réassembler tous les sous-tableaux en un seul.

À cause des fuites mémoires, quand on lançait le programme avec 1 milliard d'éléments, le processus était arrêté puisque la capacité de RAM demandée par celui-ci dépassait la capacité possible pourtant déjà conséquente.

3. Choix de parallélisation des primitives :

J'ai utilisé des « #pragma omp parallel for » dans les boucles où il n'y avait pas d'accès en lecture et en écriture concurrents. À beaucoup d'endroit dans l'algorithme, on peut se permettre de le faire ainsi car les opérations préfixe et suffixe sont parallélisables.

Le choix du niveau d'imbrication de la parallélisation s'est fait par expérience : essayer différentes valeurs pour trouver une valeur stable et optimisée.

4. Indication sur l'utilisation du programme :

Le temps d'exécution est écrit à la fin dans la sortie mais l'écriture du tableau initiale et du tableau final (trié) peut prendre beaucoup de temps selon la taille.

À la compilation, il y a besoin d'utiliser « -lm » pour compiler avec la librairie mathématique.

ATTENTION

Le fichier d'input ne doit pas avoir comme premier caractère un chiffre sinon le programme risque de planter (exemple à ne pas faire : 456_input.txt).