Travaux Pratique 5

Contexte : Les données indiquées dans ce rapport ont été obtenues avec un Processeur 6 cœurs et 12 processeurs logiques.

Exercice 1 & 2 :

Ici, on implémente le patron partition en séquentiel avec 2 boucles et on l'utilise pour le tri base.

Le temps d'exécution n'est pas terrible et est linéaire par rapport au nombre de données :

263 us pour 1024, 2653us pour 10240, 26573us pour 102400…

Exercice 3 :

J'ai utilisé mes patrons précédents pour réaliser l'exercice. Celui-ci « semble fonctionner ».

J'utilise un transformIterator pour réaliser calculer l'index. Je n'ai pas réussi à l'utiliser pour ne pas avoir à calculer le not(predicat). De plus, j'ai voulu utiliser les reverse iterator pour le scan inclusif mais je n'ai pas réussi complètement réussi, j'utilise donc encore std::reverse pour inverser le résultat.

Le code du tri base est globalement le même que pour l'exercice 2.

Au niveau des performances, mon patron parallèle est plus lent que le séquentiel. J'ai un doute sur la justesse de mes algorithmes mais je suppose que le problème vient du coût d'instanciation des threads et de l'accès à la mémoire.

Au niveau de la différence entre threads et pool de threads, le pool de threads sera plus rapide car si les patrons sont implémentés avec des threads alors ils seront instanciés au début de chaque patron parallèle. Avec le pool de threads, cette instanciation n'est faite qu'au début, on ne fait que donner de nouvelles tâches. Donc on gagne en temps sur l'instanciation des threads.

Exercice 4 & 5 :

Mon code n'est pas fonctionnel, surtout pour le tri base.

Pour ce qui est du patron partition, j'ai essayé de fusionner les deux scans, sans grand succès. Tout d'abord je fais la première étape du scan en partant du début pour le scan exclusif et de la fin pour le scan inclusif. Ensuite, le dernier thread s'occupe de l'étape séquentiel. Utiliser le dernier thread me permet d'initialiser simplement mon vecteur partialSumsFalse. J'ai préféré l'ordonner de manière croissante contrairement à l'ordre du reverse scan.

Il me semble que le souci vient des sommes partielles, je n'ai pas réussi à plus l'identifier car il n'arrive surtout que lorsque le nombre de données dans l'exercice 4 est suffisamment grand, ce qui n'est pas très pratique pour débugger.

En testant, le temps d'exécution, j'obtiens des résultats plus proches de l'exercice 1.

L'algorithme fusionnant les deux scans semblent donc plus rapides que l'algorithme utilisant les patrons les uns à la suite des autres.