TP Algorithmique Parallèle : quelques tris en MPI

Philippe SWARTVAGHER
philippe.swartvagher@enseirb-matmeca.fr

19 avril 2024

On a p processus MPI pour trier un tableau de taille n.

Pour trier un tableau localement, on peut utiliser la fonction `qsort()`.

Pour mesurer la durée des exécutions, on peut utiliser la fonction `MPI_Wtime()`, qui renvoie un `double` correspondant au nombre de secondes écoulées depuis un instant défini dans le passé.

Partir du code fourni avec le sujet du TP.

1 Tri fusion

On va paralléliser en MPI le tri fusion. Ici, le tableau de taille n est initialement stocké sur le processus 0, les autres processus vont participer au tri, puis le tableau trié sera à nouveau intégralement stocké sur le processus 0.

Idée générale :

1. Le processus 0 dispose d'un tableau à trier.
2. Le processus 0 divise le tableau en p sous-tableaux de taille $\frac{n}{p}$ et envoie chaque sous-tableau à un processus.
3. Chaque processus trie le sous-tableau qu'il a reçu.
4. Chaque processus avec un identifiant pair récupère le tableau trié de son processus voisin avec un processus impair, puis fusionne son sous-tableau trié avec le sous-tableau reçu du processus voisin.
5. On répète l'étape précédente jusqu'à ce que le processus 0 récupère tout le tableau trié.

Questions

1. Quels paramètres sera-t-il possible de faire varier pour évaluer les performances du programme ?
2. Quelles sont les contraintes sur ces paramètres ?
3. Quelle fonction MPI utiliser pour réaliser l'étape 2 ?
4. Quelle est la complexité de cet algorithme ?

5. Implémenter cet algorithme en vérifiant que le tableau est bien trié à la fin.
6. Évaluer les performances de ce programme.
7. Quels sont les inconvénients de cet algorithme ?

2 Tri pair-impair

Pour le tri pair-impair, le tableau de taille n sera réparti sur les p processus MPI (chaque processus a $\frac{n}{p}$ valeurs). À la fin de l'exécution de l'algorithme, le tableau est trié, de façon à ce que toutes les valeurs du processus i soient inférieures aux valeurs du processus $i+1$ et les valeurs de chaque processus sont aussi triées localement.

Idée générale :

1. Chaque processus dispose d'un sous-tableau de taille $\frac{n}{p}$.
2. Chaque processus trie son sous-tableau.
3. Phase impaire :
   (a) Les processus de rang pair k envoient au processus de rang $k+1$ la moitié supérieure de leur tableau.
   (b) Les processus de rang impair k envoient au processus de rang k la moitié inférieure de leur tableau.
   (c) Les processus de rang pair trient la moitié supérieure de leur tableau et ce qu'ils viennent de recevoir.
   Les processus de rang impair trient la moitié inférieure de leur tableau et ce qu'ils viennent de recevoir.
   (d) Les processus de rang pair remplacent la moitié supérieure de leur tableau par la moitié inférieure de ce qu'ils viennent de trier.
   Les processus de rang impair remplacent la moitié inférieur de leur tableau par la moitié supérieur de ce qu'ils viennent de trier.
4. Phase paire : comme la phase impaire, mais en inversant ce que font les processus pairs et impairs.
5. Chaque processus trie son sous-tableau.
6. Les étapes 3 à 5 sont répétées n fois.

Questions

1. Quels paramètres sera-t-il possible de faire varier pour évaluer les performances du programme ?
2. Quelles sont les contraintes sur ces paramètres ?
3. Quelle est la complexité de cet algorithme ?
4. Implémenter cet algorithme en vérifiant que le tableau est bien trié à la fin.
5. Évaluer les performances de ce programme.
6. Quels sont les avantages et inconvénients de cet algorithme ?

3 Tri rapide (*quicksort*)

Question ouverte : comment implémenter un tri rapide avec MPI ?