TD1 : algorithmique parallèle

Version du 25 janvier 2019

Exercice 1 – Tri pair-impair

Nous allons étudier une adaptation parallèle de l'algorithme séquentiel du tri par bulle, dont le principe est donné par l'algorithme 1. La complexité en nombre de comparaisons de l'algorithme séquentiel du tri par bulle est :

$$\sum_{i=0}^{N-2} N-1-i = \sum_{j=1}^{N-1} j = N(N-1)/2 = \Theta(N^2)$$

Algorithme 1 Tri par bulle

Entrées : T : tableau de N données à trier
1: **Pour** $i = 0$ à $N-2$ **faire**
2: **Pour** $j = 1$ à $N-1-i$ **faire**
3: **Si** $T[j] < T[j-1]$ **Alors**
4: échanger $T[j]$ et $T[j-1]$
5: **Fin si**
6: **Fin pour**
7: **Fin pour**

Pour la version parallèle, on dispose d'un réseau linéaire de P processeurs (architecture MIMD-DM), sur lequel les processeurs communiquent par échange de message.

1. On suppose tout d'abord qu'on dispose de $P = N$ processeurs (avec P pair). Chaque processeur (P_i) possède donc un élément du tableau ($T[i]$). Ecrire un algorithme SPMD de tri parallèle basé sur les procédures `compare_echange_min` et `compare_echange_max`. A la première étape, les processeurs de numéro pair comparent leur élément avec leur voisin de droite (et réciproquement). A l'étape suivante les processeurs de numéro pair comparent leur élément avec leur voisin de gauche (et réciproquement), et ainsi de suite...

Procédure 2 compare_echange_{min,max}(id_voisin)

Entrées : id_voisin : paramètre d'entrée
Entrées : id : numéro du processeur courant (de 0 a $P-1$)
1: **Si** $id\_voisin \geq 0$ et $id\_voisin \leq P-1$ **Alors**
2: envoyer l'élément courant à id_voisin et recevoir l'élément de id_voisin
3: affecter {MIN,MAX}(élément courant, élément de id_voisin) à l'élément courant
4: **Fin si**

On pourra tester l'algorithme sur le tableau suivant :

34	87	65	15	71	45	32	10

2. On suppose maintenant qu'on dispose de $P = N/k$ processeurs. Ecrire l'algorithme SPMD de tri parallèle correspondant en se basant sur les procédures `compare_echange_mins` et `compare_echange_maxs`.
3. En supposant qu'on utilise un algorithme de tri (par comparaison) optimal en $\Theta(N \ln N)$ pour le tri initial local à chaque processeur, quelle est la complexité théorique de cet algorithme en parallèle ? Quelle est l'accélération correspondante ?

Procédure 3 compare_echange_{mins,maxs}(id_voisin, k)

Entrées : id_voisin : paramètre d'entrée,
k : paramètre d'entrée
Entrées : id : numéro du processeur courant (de 0 a $P-1$)
1: **Si** $id\_voisin \geq 0$ et $id\_voisin \leq P-1$ **Alors**
2: envoyer les k éléments de id à id_voisin, et recevoir de la part de id_voisin les k éléments de id_voisin
3: conserver les k plus {**petits**, **grands**} éléments de l'ensemble des k éléments de id et des k éléments de id_voisin
4: **Fin si**

Exercice 2 – Produit matriciel

On dispose de P processeurs et on souhaite effectuer le produit matriciel : $C = A \times B$, où A, B et C sont des mactrices carrées de $N \times N$ éléments.

On suppose que N est divisible par P. Chaque processeur est identifié par un numéro $0 \leq r \leq P-1$ (son « rang »).

1. On attribue un bloc continu de $h = N/P$ lignes complètes de C à chaque processeur. Donner un algorithme simple du produit matriciel parallèle correspondant.
2. Quel est l'inconvénient de cet algorithme ?
3. En considérant que les processeurs forment un anneau, donner un autre algorithme du produit matriciel parallèle qui ne présente pas cette contrainte.

Exercice 3

On dispose de P processus et on souhaite implémenter un algorithme de réduction effectuant la sommation des valeurs entières possédées par chaque processus. La racine de l'algorithme de réduction sera le processus de numéro 0 (et c'est donc lui qui affichera le résultat).

Définissez un algorithme efficace approprié *sans routine de communication collective*.