Parallélisme et Répartition

M1 IFI/MBDS, Janvier 2011, Session 2

2 heures, Feuille A4 manuscrite recto-verso autorisée

1 Recherche du premier élément d'une séquence

Le but de cet exercice est de proposer des algorithmes PRAM permettant de chercher la première occurrence d'un élément dans une liste ou un tableau. Pour simplifier, nous allons considérer un tableau T de N éléments pouvant prendre la valeur 0 ou 1 et l'algorithme donnera l'indice du **premier** 1.

1.1 Algorithme A

Dans un premier temps, nous allons implémenter une version simple de l'algorithme. Il construit un tableau $RESULT$ de taille N dont les cases sont remplies de la façon suivante. Pour chaque case i de T contenant un 1, si il existe un élément 1 d'indice plus petit, alors $RESULT[i] = 0$, sinon $RESULT[i] = 1$. Il suffit ensuite de regarder le contenu de $RESULT$ pour connaître l'indice du premier élément 1.

1. Exécutez l'algorithme sur le tableau $[0, 0, 0, 0, 1, 0, 1]$
2. Ecrivez l'algorithme qui remplit le tableau $RESULT$
3. Ecrivez l'algorithme qui donne l'indice du premier élément 1.
4. Combien de processeurs cet algorithme nécessite-t-il pour avoir un parallélisme maximal ?

1.2 Réduction du nombre de processeurs

Nous allons essayer de réduire le nombre de processeurs nécessaires pour trouver le premier élément. On commence par implémenter un algorithme B qui produit/affiche 1 si il existe au moins un élément 1 dans le tableau en entrée, et 0 sinon.

1. Écrivez l'algorithme B
2. Combien de processeurs nécessite-t-il pour être de parallélisme maximal?

Finalement, on propose l'algorithme suivant :

- Le tableau T de taille N est divisé en sous-tableaux de taille X
- B est exécuté en parallèle sur chacun des sous tableaux

- A est exécuté en utilisant les résultats de B pour identifier le sous-tableau contenant le premier 1
- A est exécuté sur le sous-tableau et donne la position du premier élément 1

3. Exécutez cet algorithme sur le tableau $[0, 0, 1, 1, 0, 1, 0, 0, 0]$. Vous pourrez prendre $X = 3$.
4. Quelle est la taille des données produit par B ?
5. Quelle est la taille des données sur lesquelles A travaille maintenant ?
6. Ecrivez l'algorithme combinant A et B.
7. Comment choisir X en fonction de N pour utiliser $O(N)$ processeurs ?
8. Quelle est la complexité en temps de cet algorithme ?

2 Calculs sur topologie Grille 2D avec du pseudo-MPI

Le but de cet exercice est d'écrire du code pseudo-MPI utilisant une topologie de Grille 2D.

1. Rappelez les caractéristiques principales (diamètre, nombre de liens, degré) d'une Grille 2D de P processeurs
2. Soit un processeur de rang i, quel est le rang de ses voisins ? (*indication: vous devez traiter les cas particuliers des processeurs aux frontières de la grille*)

On veut implémenter l'algorithme suivant. Chaque processeur possède initialement une valeur réelle V_i. L'algorithme est itératif, à chaque tour de boucle, chaque processeur envoie sa valeur V_i à ses voisins, et lit leurs valeurs en échange. La nouvelle valeur est la moyenne des valeurs reçues et de V_i. L'algorithme s'arrête après MAX itérations.

3. On implémente le comportement suivant pour chaque processeur : envoyer V_i à son voisin nord, puis est, puis sud, puis ouest, et ensuite lire la valeur des voisins dans le même ordre. Quel problème cela va-t-il produire à l'exécution?
4. En vous basant sur le rang du processeur et des émissions et réceptions bien ordonnées, proposez une méthode *echangerAvecVoisins(int rang, float vi, float[] values)* permettant d'envoyer sa valeur aux voisins et de récupérer la leur sans provoquer de deadlock. (*on ne demande pas de code MPI, du pseudo-code avec Send, Recv, Nord, Sud... est suffisant*).

3 Tri bitonique

1. Rappelez la définition d'une séquence bitonique
2. Comment peut-on trier une séquence bitonique?
3. Combien de comparateurs sont-ils nécessaires?
4. Comment trier une séquence de nombres aléatoires en utilisant le tri de séquence bitonique?