Ecole Polytechnique

Cours : "Parallélisme"
Travaux dirigés - Révisions
E. Goubault & M. Martel

TD 9

25 mars 2003

1 PRAM

Soit L une liste contenant n objets coloriés soit en bleu, soit en rouge. Concevoir un algorithme EREW efficace qui sépare les éléments bleus des éléments rouges (c'est-à-dire qui construit une nouvelle liste ne contenant que les éléments bleus).

Corrigé:

On utilise la technique du saut de pointeur, mais cette fois-ci sur deux listes en même temps: on peut imaginer qu'en fait, chaque cellule de la liste d'origine comporte un champ `bleu` qui va correspondre au pointeur suivant de la liste des bleus, et un champ `suivant` qui correspond au pointeur suivant de la liste de départ. On utilisera également à des fins de marquage un champ `fini` à chaque cellule. On associe bien évidemment un processeur par cellule des listes en question:

```
forall i  /* cellule */
  if ((i.suivant == nil) || (i.suivant.couleur == bleu)) {
    i.fini = true;
    i.bleu = i.suivant;
  }

while exist i such that (i.fini == false)
  forall i
    if (i.fini == false) {
      i.fini = i.suivant.fini;
      if (i.fini == true)
        i.bleu = i.suivant.bleu;
      i.suivant = i.suivant.suivant;
    }
```

2 Rotations de Givens

Pour trianguliser une matrice A d'ordre n de façon numériquement stable, on peut utiliser les rotations de Givens. L'opération de base `Rot(i,j,k)` consiste à combiner les deux lignes `i` et `j`, qui doivent toutes deux commencer par `k-1` zéros pour annuler l'élément en position `(j,k)` (avec le bon θ):

$$\begin{pmatrix} 0 & \dots & 0 & \mathbf{a'_{i,k}} & a'_{i,k+1} & \dots & a'_{i,n} \\ 0 & \dots & 0 & \mathbf{0} & a'_{j,k+1} & \dots & a'_{j,n} \end{pmatrix} \leftarrow \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} 0 & \dots & 0 & \mathbf{a_{i,k}} & a_{i,k+1} & \dots & a_{i,n} \\ 0 & \dots & 0 & \mathbf{a_{j,k}} & a_{j,k+1} & \dots & a_{j,n} \end{pmatrix}$$

L'algorithme séquentiel peut s'écrire:

```
Givens(A)
1: for (k=1;k<n;k++)
2:   for (i=n;i>k;i--)
3:     Rot(i-1,i,k);
```

On considère qu'une rotation `Rot(i,j,k)` s'exécute en temps unité, indépendamment de k.

Question 1 : Mettre en oeuvre cet algorithme sur un réseau linéaire de n processeurs.

Question 2 : Mettre en oeuvre cet algorithme sur un réseau linéaire comportant seulement $\lfloor \frac{n}{2} \rfloor$ processeurs

Corrigé:

Pour la question 1: le processeur P_k va être responsable de toutes les rotations `Rot(i-1,i,k)`, et le fonctionnement va se faire "en pipeline", c'est-à-dire que la dernière ligne arrive sur P_1 d'abord, puis l'avant dernière ligne qui arrive permet à P_1 d'effectuer son premier appel à `Rot`. Quand la quatrième avant dernère ligne arrive sur P_1 puis sur P_2, P_2 peut effectuer son premier appel à `Rot`, et ainsi de suite. On donne l'illustration de l'exécution pour $n = 8$ dans le tableau ci-dessous, en indiquant l'arrivée de la $i$ème ligne sur un processeur par $\boxed{i}$:

Temps	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
1	$\boxed{8}$							
2	$Rot(7,8,1)$							
3	$Rot(6,7,1)$	$\boxed{8}$						
4	$Rot(5,6,1)$	$Rot(7,8,2)$						
5	$Rot(4,5,1)$	$Rot(6,7,2)$	$\boxed{8}$					
6	$Rot(3,4,1)$	$Rot(5,6,2)$	$Rot(7,8,3)$					
7	$Rot(2,3,1)$	$Rot(4,5,2)$	$Rot(6,7,3)$	$\boxed{8}$				
8	$Rot(1,2,1)$	$Rot(3,4,2)$	$Rot(5,6,3)$	$Rot(7,8,4)$				
9	$\boxed{1}$	$Rot(2,3,2)$	$Rot(4,5,3)$	$Rot(6,7,4)$	$\boxed{8}$			
10	$\boxed{1}$	$\boxed{2}$	$Rot(3,4,3)$	$Rot(5,6,4)$	$Rot(7,8,5)$			
11	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$Rot(4,5,4)$	$Rot(6,7,5)$	$\boxed{8}$		
12	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$Rot(5,6,5)$	$Rot(7,8,6)$		
13	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$Rot(6,7,6)$	$\boxed{8}$	
14	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	$Rot(7,8,7)$	
15	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	$\boxed{7}$	$\boxed{8}$

On remarque qu'il n'y a jamais plus de $\frac{n}{2}$ processeurs actifs en même temps. En "repliant" cette allocation, on obtient (en fait les lignes passent d'abord de P_1 vers $P_{\frac{n}{2}}$ et ensuite de $P_{\frac{n}{2}}$ vers P_1):

$Temps$	P_1	P_2	P_3	P_4
1	$\boxed{8}$			
2	$Rot(7,8,1)$			
3	$Rot(6,7,1)$	$\boxed{8}$		
4	$Rot(5,6,1)$	$Rot(7,8,2)$		
5	$Rot(4,5,1)$	$Rot(6,7,2)$	$\boxed{8}$	
6	$Rot(3,4,1)$	$Rot(5,6,2)$	$Rot(7,8,3)$	
7	$Rot(2,3,1)$	$Rot(4,5,2)$	$Rot(6,7,3)$	$\boxed{8}$
8	$Rot(1,2,1)$	$Rot(3,4,2)$	$Rot(5,6,3)$	$Rot(7,8,4)$
9	$Rot(2,3,2)$	$Rot(4,5,3)$	$Rot(6,7,4)$	$\boxed{8}$
10	$Rot(3,4,3)$	$Rot(5,6,4)$	$Rot(7,8,5)$	
11	$Rot(4,5,4)$	$Rot(6,7,5)$	$\boxed{8}$	
12	$Rot(5,6,5)$	$Rot(7,8,6)$		
13	$Rot(6,7,6)$	$\boxed{8}$		
14	$Rot(7,8,7)$			
15	$\boxed{8}$			

3 Allocation homogène/inhomogène

On considère un calcul sur une matrice M de taille $n \times n$ avec p processeurs. Ce calcul, que l'on ne précise pas, a la propriété suivante: le calcul de $M_{i,j}$ ne peut se faire que si $M_{i-1,j}$ et $M_{i,j-1}$ ont été calculés.

Question 1 : Comment faire une allocation statique optimale dans le cas homogène?

Question 2 : Comment faire une allocation statique par blocs de $B = 10$ colonnes dans le cas où $p = 3$ et les temps de cycle sont $t_1 = 3$, $t_2 = 5$ et $t_3 = 8$?

Question 3 : Utiliser l'idée de l'algorithme d'allocation incrémental du cours pour essayer de faire mieux.

Corrigé:

Voir poly pages 114 à 117.