

1 Algorithmique PRAM

1.1 Multiplication de matrices

Le but de cet exercice est d'écrire un algorithme PRAM pour la multiplication de matrices. On dispose de deux matrices A et B carrées $n \times n$ qui sont stockées en mémoire globale sous la forme de deux tableaux de dimension 2, chaque dimension étant indiquée par les entiers de 1 à n . Après exécution de l'algorithme, le tableau C de même caractéristiques que les deux tableaux A et B devra contenir la matrice produit de A par B . Les deux tableaux A et B n'auront pas dû être modifiés.

1. Écrire un algorithme PRAM pour résoudre ce problème.
2. Donner la complexité de votre algorithme.
3. Sur quel type de PRAM cet algorithme peut-il être exécuté ?
4. Modifier votre algorithme pour qu'il puisse tirer avantage d'une PRAM CRCW. Précisez quel type de CRCW vous utilisez.

Rappels : les coefficients de la matrices C sont définis par :

$$c_{i,j} = \sum_{k=1}^n a_{i,k} b_{k,j}$$

Indications : Utilisez n^3 processeurs et un tableau auxiliaire de dimension 3, chaque dimension étant indiquée par les entiers de 1 à n . Considérez que les processeurs sont désignés par un triplet (i, j, k) d'entiers compris entre 1 et n .

1.2 Statistique

On considère une suite finie de n nombres réels x_i , $1 \leq i \leq n$. Ces données sont stockées dans un tableau D de taille n en mémoire partagée.

1. Écrire un algorithme PRAM pour calculer la statistique

$$\sum_{i=1}^n (x_i - \bar{X})^2$$

où \bar{X} désigne la moyenne des x_i .

2. Donner la complexité de votre algorithme

1.3 Calcul des préfixes (*scan*)

Étant donnés n valeurs x_0, x_1, \dots, x_{n-1} , et une opération binaire *associative* \oplus , il s'agit de calculer toutes les réductions de préfixes de la liste soit :

$$(x_0), (x_0 \oplus x_1), \dots, (x_0 \oplus x_1 \oplus \dots \oplus x_{n-1}).$$

Ce calcul est à la base de nombreux algorithmes parallèles en arithmétique, en géométrie, pour les SGBD, etc.

1. Donner un algorithme PRAM pour le *scan* (Résultat dans un tableau S). Considérez une PRAM à $p = n$ processeurs.
2. Donner sa complexité.

1.4 Sans répétition

Soit un tableau de n entiers $E(1), \dots, E(n)$ dans la mémoire d'une machine PRAM CREW de $p = n$ processeurs et une fonction f des entiers dans les entiers.

1. Définissez un algorithme parallèle dont l'effet est de rendre un tableau $S(1), \dots, S(k)$ contenant les $E(i)$ pour lesquels $f(E(i))$ est pair, en ordre croissant de i et sans répétition. La valeur de k n'est pas connue à l'avance.
2. Donnez la complexité en temps de votre algorithme.

1.5 Centre de gravité

Soient deux tableaux X et Y de taille n contenant des flottants positifs et placés dans la mémoire d'une machine PRAM. La paire $(X[i], Y[i])$ représente un point dans le plan par ses coordonnées. Donnez un algorithme PRAM dont l'effet est de déplacer chacun des n points de 10% de sa distance vers le centre de gravité des n points. Le déplacement s'effectue par affectation dans les tableaux X et Y . Vous devez supposer que $n > p$ le nombre de processeurs et que n est un multiple de p . Donnez la complexité de votre algorithme.

1.6 Le principe de Brent

On considère un algorithme s'exécutant sur une PRAM CREW à $p = 2^q$ processeurs. En entrée, on dispose d'un tableau A de taille $n = 2^k$. Les processeurs sont désignés par l'entier pid . On notera désormais $Z := X+Y$ pour la suite d'instructions :

```
gr(X, x);
gr(Y, y);
z:=x+y;
gw(z, Z);
```

L'algorithme est le suivant :

```
for h from 1 to k do
  if q >= k-h then
    if pid <= n/2^h then
      A(pid) = A(2*pid-1) + A(2*pid)
    end if
  else
    for l from 1+(pid-1)*2^(k-q-h) to pid*2^(k-q-h) do
      A(l) = A(2*l-1) + A(2*l)
    end do
  end if
end do
```

1. Que calcule cet algorithme ?
2. Donner sa complexité parallèle.
3. Comparer cet algorithme à l'algorithme de réduction vu en cours.

1.7 Tri par insertion

Le but est d'écrire un algorithmes PRAM de tri d'un tableau E d'entiers de longueur n . La version séquentielle du tri par insertion est donnée par l'algorithme suivant :

```
for i from 1 to n do
  p = 1
  for j from 1 to n do
    if (E(j) < E(i)) then
      p = p + 1
    end if
  end for
  T(p) = E(i)
end for
```

1. En précisant le nombre de processeurs que vous utilisez, proposez un algorithme PRAM qui réalise ce tri par insertion.
2. Vérifiez que vous obtenez une complexité en $O(\log n)$.
3. Précisez quel type de PRAM vous utilisez ? Est-il possible d'adapter votre algorithme à une PRAM EREW ?

1.8 Tri par fusion

1. Proposez les grandes étapes d'un algorithme PRAM basé sur le tri par fusion dont la complexité séquentielle est en $O(n \log(n))$.
2. Donnez la complexité correspondante et le nombre de processeurs que vous utilisez.