Parallélisme et Répartition TD3

1 Exercice 1 (Théorème de simulation)

Comme il y a plusieurs modèles de PRAM, une question récurrente est de savoir quel modèle est plus puissant qu'un autre. Dans cet exercice, nous allons nous limiter aux PRAM CRCW (en mode consistant) et CREW.

1) Rappelez les définitions de ces PRAM.

2) Nous souhaitons simuler le fonctionnement d'une CRCW en utilisant une CREW. Quelle opération de la CRCW va être problématique?

3) Supposons qu'un processeur P_i de la CRCW écrive la donnée x_i en mémoire à l'adresse l_i. Pour éviter tout conflit avec un autre processeur, nous allons rediriger cette écriture dans un tableau intermédiaire $A[i] = (l_i, x_i)$. Pourquoi le tableau contient bien toutes les écritures effectuées à un cycle donné par tous les processeurs ?

4) Nous disposons d'un algorithme magique (en $O(log(p))$) qui va nous permettre de trier le tableau sur notre CREW. Proposez une nouvelle étape de calcul, basée sur le tableau trié, permettant de s'assurer de l'exclusivité de l'écriture.

5) En déduire qu'il est possible de simuler une CRCW sur une CREW pour un surcout de $O(log(p))$.

6) En déduire qu'un algorithme sur une PRAM CRCW ne peut pas être plus de $O(log(p))$ fois plus rapide que le meilleur algorithme PRAM CREW à p processeurs pour le même problème.

2 Exercice 2 (Prefix divide'n'conquer)

Il existe un algorithme de préfixe en parallèle basé sur la technique du divide'n'conquer dont le principe est le suivant. Le tableau $x_1, ...x_n$ $(n = 2^m)$est découpé en deux sous tableaux $(x_1, ..., x_{\frac{n}{2}})$ et $(x_{(\frac{n}{2}+1)}, ..., x_n)$ qui sont traités en parallèle pour donner les préfixes $(p'_1, ..., p'_{\frac{n}{2}})$ et $(p'_{(\frac{n}{2}+1)}, ..., p'_n)$. Il suffit ensuite de traiter ces deux tableaux pour obtenir les préfixes $(p_1, ..., p_n)$

$$p_i = \begin{cases} p'_i & si\ i \leq n/2 \\ p'_{\frac{n}{2}} \otimes p'_i & si\ i > n/2 \end{cases} \quad (1)$$

1) Exécutez cet algorithme sur l'exemple $1, 2, 3, 4, 5, 6, 7, 8$

2) Quel type de PRAM cet algorithme nécessite-t-il ?

3) Proposez une modification pour qu'il puisse s'exécuter sur une machine EREW (*hint :* utilisez un tableau intermédiaire pour la valeur problématique).

4) Comment faire pour créer ce tableau de la manière la plus efficace possible?

3 Exercice 3 (Algo mystère)

On donne l'algorithme mystérieux suivant.

```
for each processor do in parallel
  if next[i] = NIL then
    d[i] = 0
  else
    d[i] = 1
  end if
end for
while ∃ a node i such as next[i] ≠ NIL do
  for each processor do in parallel
    if next[i] ≠ NIL then
      d[i] = d[i] + d[next[i]]
      next[i] = next[next[i]]
    end if
  end for
end while
```

1) Que fait cet algorithme appliqué sur une liste chaînée d'éléments?

2) La condition de la boucle *while* ne peut être calculé en temps constant que sur une CRCW, de quelle manière ?

3) Quelle serait la complexité sur une CREW ?

4) Modifiez l'algorithme pour obtenir une exécution sur une EREW