Machines P-RAM

Résumé: Dans ce TD, nous discutons la *puissance* comparée des modèles EREW, CREW et CRCW, et nous définissons *l'efficacité* des algorithmes P-RAM.

1 Séparation des modèles

Une P-RAM CRCW est-elle plus *puissante* qu'une P-RAM CREW ? Et une CREW est-elle plus *puissante* qu'une P-RAM EREW ? Le problème de la comparaison des modèles CRCW et CREW s'énonce ainsi : peut-on trouver un algorithme tel que, pour un même nombre de processeurs, on obtienne avec une P-RAM CRCW un temps d'exécution qu'il est impossible d'atteindre pour une P-RAM CREW ?

1.1 Séparation CREW/CRCW : Calcul du maximum d'un tableau

▷ **Question 1.** Donner un algorithme PRAM CRCW permettant de calculer avec $O(n^2)$ processeurs en temps constant le maximum d'un tableau A composé de n éléments.

▷ **Question 2.** Sur un modèle CREW, quelle est la complexité qu'on peut espérer ?

Une P-RAM CRCW est donc strictement plus puissante qu'une P-RAM CREW. Nous venons de séparer les modèles CRCW et CREW, en exhibant un algorithme pour lequel un facteur $O(\log p)$ sépare les temps d'exécution avec p processeurs d'un même problème. Ce facteur $O(\log p)$ est maximal. On peut même montrer le théorème suivant :

Théorème 1. *Tout algorithme sur une machine P-RAM CRCW à p processeurs ne peut pas être plus de $O(\log p)$ fois plus rapide que le meilleur algorithme P-RAM CREW à p processeurs pour le même problème. On a le même résultat entre les machines CREW et EREW et les machines CRCW et EREW.*

1.2 Séparation EREW/CREW : Recherche des racines dans une forêt

Soit $\mathcal{F}$ une forêt d'arbres binaires. Chaque nœud i d'un arbre est associé à un processeur $P(i)$ et possède un pointeur vers son père $pere(i)$. On va chercher des algorithmes EREW et CREW pour que chaque nœud connaisse la racine de son arbre (notée $racine(i)$), et ainsi prouver l'intérêt des lectures concurrentes.

▷ **Question 3.** Donner un algorithme PRAM CREW pour que chaque nœud détermine $racine(i)$. Démontrer que l'algorithme propose n'utilise que des lectures concurrentes et déterminer sa complexité.

▷ **Question 4.** Sur un modèle EREW, quelle est la complexité qu'on peut espérer ?

On vérifie donc ainsi que le modèle CREW est strictement plus puissant que le modèle EREW. Nous allons maintenant montrer comment simuler une lecture simultanée en $O(\log n)$ sur une EREW. La simulation des écritures concurentes en $O(\log n)$ étapes peut se faire en utilisant le même type de technique.

▷ **Question 5.** Comment simuler une lecture concurrente en $O(\log n)$ sur une EREW.

▷ **Question 6.** Donner un algorithme efficace CRCW pour calculer (en le moins de temps possible) le produit de deux matrices booléennes (dans ce cadre, l'addition est remplacée par le *ou* et la multiplication par le *et*).

2 Théorème de Brent, travail et efficacité

▷ **Question 7.** Soit $\mathcal{A}$ un algorithme comportant un nombre total de m opérations et qui s'exécute en temps t sur une P-RAM (avec un nombre de processeurs indéterminé). Montrer que l'on peur simuler $\mathcal{A}$ en temps $O\left(\frac{m}{p}+t\right)$ sur une P-RAM de même type avec p processeurs (théorème de Brent).

Le théorème de Brent permet de prédire les performances quand on réduit le nombre de processeurs.

▷ **Question 8.** Proposez un algorithme pour le calcul du maximum sur une P-RAM EREW et calculez sa complexité. Est-il possible d'utiliser moins de processeur tout en conservant la meme complexité ?

Arnaud Legrand

Donnons quelques définitions usuelles : soit P un problème de taille n à résoudre, et soit $T_{seq}(n)$ le temps du meilleur algorithme séquentiel (connu) pour résoudre P. Soit maintenant un algorithme parallèle P-RAM qui résout P en temps $T_{par}(p)$ avec p processeurs.

Définition 1. Le facteur d'accélération (*speed-up* en anglais) est défini comme

$$S_p = \frac{T_{seq}(n)}{T_{par}(p)}$$

et l'efficacité comme

$$e_p = \frac{T_{seq}(n)}{p.T_{par}(p)}.$$

Définition 2. Le travail (*work*) de l'algorithme est

$$W_p = p.T_{par}(p).$$

Intuitivement, le travail est une surface rectangulaire de taille $T_{par}(p)$, le temps d'exécution, multiplié par p, le nombre de processeurs, et est minimal si à chaque étape de calcul tous les processeurs sont utilisés à faire des choses utiles, i.e. des opérations présentes dans la version séquentielle.

Le résultat suivant montre qu'on peut conserver le travail par simulation :

▷ **Question 9.** Soit $\mathcal{A}$ un algorithme qui s'exécute en temps t sur une P-RAM avec p processeurs. Montrer que l'on peut simuler $\mathcal{A}$ sur une P-RAM de même type avec $p' \leqslant p$ processeurs, en temps $O\left(\frac{t.p}{p'}\right)$.

Une conséquence de ce résultat est que le travail d'un algorithme P-RAM est au moins de l'ordre de la complexité séquentielle (sinon la simulation de cet algorithme avec un seul processeur améliorerait cette complexité). On dit qu'un algorithme P-RAM est *efficace* quand son travail est de l'ordre de la complexité séquentielle (on n'ose pas dire *optimal*, sauf si la complexité séquentielle est établie). On ne peut pas diminuer le travail d'un algorithme efficace de plus d'un facteur constant.