UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

LICENCE Informatique
Info0604 – 2019/2020

Travaux dirigés n° 1

Algorithmes multithread

Exercice 1 (Calcul des nombres de Fibonacci)
Les nombres de Fibonacci sont définis par la récurrence suivante :

$F_0 = 0$,

$F_1 = 1$,

$F_i = F_{i-1} + F_{i-2}$ pour $i \geq 2$.

1°) Rappelez l'algorithme séquentiel récursif qui permet de calculer le n-ème nombre de Fibonacci.

2°) Expliquez pourquoi cet algorithme n'est pas efficace pour réaliser ce calcul.

Malgré son inefficacité, cet algorithme est propice pour se familiariser avec les algorithmes multithread.

3°) Expliquez pourquoi cet algorithme est propice à une exécution parallèle.

Afin d'exprimer le parallélisme dans nos algorithmes, nous disposons des mots clés de concurrence suivants :

- - **spawn** (que l'on pourrait traduire par "faire naître") : placé devant un appel de procédure, exprime le parallélisme imbriqué. La différence avec un appel de procédure ordinaire est que l'instance de procédure qui exécute le **spawn** (le *parent*) peut continuer son exécution en parallèle à la sous-routine spawnée (son *enfant*), au lieu d'attendre la fin de l'exécution de l'enfant. On notera que ce mot clé ne dit pas qu'une procédure *doit* être exécutée en même temps que ses enfants spawnés, mais seulement qu'elle le *peut* : il exprime le ***parallélisme logique*** du calcul.
- - **sync** (pour "synchronisation") : indique que la procédure doit attendre que tous ses enfants spawnés aient terminé leur exécution avant de passer à l'instruction qui suit le **sync**. Outre la synchronisation explicite fournie par cette instruction, toute procédure exécute un **sync** implicitement avant de rendre la main, garantissant ainsi que tous ses enfants se termineront avant qu'elle ne le fasse.
- - **parallèle** : placé devant une boucle pour, indique que les itérations de la boucle peuvent se faire en parallèle.

4°) En utilisant ces mots clés, proposez une version multithread de l'algorithme récursif de calcul des nombres de Fibonacci.

Pour mesurer l'efficacité théorique (sur un ordinateur parallel idéal théorique) d'un algorithme multithread, on peut employer deux métriques :

- - **travail** (de l'anglais "work") : temps total passé à exécuter l'ensemble du calcul sur un processeur. En d'autres termes, c'est la somme des temps pris par l'ensemble des groupes d'instructions.

— - **durée** (de l'anglais "span") : temps d'exécution de l'algorithme, du groupe d'instruction exécuté en premier au groupe d'instructions exécuté en dernier.

5°) En considérant que chaque groupe d'instructions séquentielles prend un temps unitaire, donnez le travail et la durée de l'algorithme défini à la question précédente, en utilisant $n = 4$. Déduisez-en ensuite l'accélération maximale de votre algorithme pour cette taille d'entrée.

Exercice 2 (Multiplication matrice-vecteur)
1°) Rappelez par un exemple le fonctionnement de la multiplication d'une matrice n x n $A = (a_{ij})$ par un vecteur $x = (x_j)$ de dimension n.

2°) Définissez la procédure séquentielle `mat-vec` qui effectue la multiplication d'une matrice par un vecteur.

3°) À l'aide des mots clés de concurrence donnés à l'exercice précédent, donnez une version multithread de la procédure donnée à la question précédente.

Exercice 3 (Attention aux conditions de course)
Soit la procédure suivante :

```
course-exemple()
     x = 0
     parallèle pour i = 1 à 2
          x = x + 1
     afficher x
```

1°) En l'illustrant par un exemple, expliquez un problème pouvant être rencontré durant l'exécution de cette procédure.

2°) Expliquez comment on peut gérer les conditions de course dans un algorithme multithread.

Exercice 4 (Multiplication matricielle)
1°) Rappelez par un exemple le fonctionnement de la multiplication de deux matrices carrées.

2°) Définissez la procédure séquentielle `multiplier-matrice-carree` qui effectue la multiplication de deux matrices carrées.

3°) À l'aide des mots clés de concurrence, donnez une version multithread de la procédure donnée à la question précédente.

Exercice 5 (Tri par fusion)
1°) Rappelez par un exemple le fonctionnement du tri par fusion.

2°) Définissez la procédure séquentielle `tri-fusion` qui trie n valeurs en utilisant le tri par fusion.

3°) À l'aide des mots clés de concurrence, donnez une version multithread de la procédure donnée à la question précédente.

Références

Cormen T. H., Leiserson C. E., Rivest R. L. et Stein C., "Algorithmique" 3e édition, Dunod, 2010.