heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

Laboratoire de Programmation Concurrente

semestre printemps 2015 - 2016

Multiplication matricielle multithreadée par moniteurs

Temps à disposition : 6 périodes (travail débutant en semaine 14)

1 Objectifs pédagogiques

— Réaliser la décomposition multi-threadée d'un problème mathématique.
— Utiliser les mécanismes de synchronisation à base de moniteur.

2 Énoncé du problème

Nous désirons minimiser le calcul de la multiplication matricielle pour une architecture multi-coeur. Nous ne traiterons que de matrices carrées, de taille $n \times n$. Le produit $C = A\ B$ correspond à :

$$\forall i, j : c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} = a_{i1} b_{1j} + a_{i2} b_{2j} + \cdots + a_{in} b_{nj} \tag{1}$$

Nous désirons tirer parti des différents coeurs à disposition pour minimiser le temps de traitement. Pour ce faire nous pouvons décomposer les matrices en sous-matrices (ou blocs), les calculs sur ces blocs pouvant alors être réalisés en concurrent. Une matrice de taille $n \times n$ sera décomposée en $nbblocks = b \times b$ blocs, la taille d'un bloc étant alors $((n/b) \times (n/b))$.

Une matrice A de taille 64×64 pourra par exemple être décomposée en 16 blocs de la manière suivante :

$$A = \begin{pmatrix} A_{00} & A_{01} & A_{02} & A_{03} \\ A_{10} & A_{11} & A_{12} & A_{13} \\ A_{20} & A_{21} & A_{22} & A_{23} \\ A_{30} & A_{31} & A_{32} & A_{33} \end{pmatrix} \tag{2}$$

Chaque bloc correspond à une matrice de taille 16×16, et le calcul de C peut alors se faire de manière indépendante pour chaque bloc :

$$\forall i, j \in [0, 3] : C_{ij} = \sum_{k=0}^{3} A_{ik} B_{kj} = A_{i0} B_{0j} + A_{i1} B_{1j} + A_{i2} B_{2j} + A_{i3} B_{3j} \tag{3}$$

L'indépendance de ce calcul permet une implémentation parallèle de la multiplication, qui peut se répartir sur les différents coeurs à disposition.

3 Cahier des charges

Réalisez le programme énoncé avec les contraintes ci-dessous.

- L'utilisateur doit pouvoir saisir par clavier les informations suivantes :
  - La taille des matrices
  - Le nombre b, permettant de décomposer la matrice en $b \times b$ blocs
  - Le nombre de threads
- Le programme doit lancer le testeur avec ces trois paramètres ;
- Le seul fichier à modifier est `threadedmatrixmultiplier.h` ;
- Pour la gestion multi-threadée, l'objet de la classe `ThreadedMatrixMultiplier` doit d'abord créer et lancer les threads, puis ensuite exploiter des mécanismes de synchronisation afin d'attribuer les tâches aux threads, sur le *modèle par délégation* ;
- La synchronisation doit passer par un moniteur dérivant de la classe `HoareMonitor`.
- La fonction `multiply` doit être réentrante. Ceci sera démontré par le test effectué par `MultiplierThreadedTester` ;
- Pour obtenir une note maximale, la solution proposée doit offrir le plus de parallélisme possible, notamment avec les test de réentrance.

4 Travail à rendre

- Les modalités du rendu se trouvent dans les consignes qui vous ont été distribuées.
- La description de l'implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans les programmes rendus. Aucun rapport n'est demandé.
- Inspirez-vous du barème de correction pour savoir où il faut mettre votre effort.
- Vous pouvez travailler en équipe de deux personnes au plus.

5 Barème de correction

Conception	35%
Exécution et fonctionnement	5%
Réentrante et offrant un parallélisme maximal	5%
Tests	10%
Codage	15%
Documentation et en-têtes des fonctions	20%
Commentaires au niveau du code	10%