

T.P. Parallélisme

Rapport

Multiplication matricielle de Fox

Réalisé par :

KHABOU Amal
LAVEGNE Julien

Troisième Année
Option PRCD
Filière Informatique
23 novembre 2008

Table des matières

1	Fontionnement	1
1.1	Principe de fonctionnement	1
1.2	Définition et conventions	1
1.2.1	Types de données	1
1.2.2	Initialisation	1
2	Implémentation	3
2.1	Organisation des communicateurs MPI	3
2.1.1	Création d'une grille	3
2.1.2	Calcul	3

Résumé

L'objectif de ce TP est d'implémenter une méthode parallèle de multiplication de matrices : l'algorithme de Fox.

Dans cette algorithme, les différents processus participant au calcul se chargent de calculer un produit matriciel sur un sous-ensemble de la matrice.

En pratique, on divise les matrices carrées A et B à multiplier en n^2 blocs carrés de même taille et l'on associe à chaque processus du calcul un bloc ainsi obtenu de A et de B.

Par un procédé d'échange des résultats que nous allons voir, chaque processus est capable de retourner le résultat du bloc de $C = A * B$ lui correspondant.

Chapitre 1

Fontionnement

1.1 Principe de fonctionnement

Pour implémenter l'algorithme de Fox en MPI, il faut traiter les différentes étapes du problème.

L'algorithme de Fox consiste à calculer et accumuler localement les termes de la somme $c_{i,j} = a_{i,0} \times b_{0,j} + a_{i,1} \times b_{1,j} + \dots + a_{i,n-1} \times b_{n-1,j}$, en fournissant au processus responsable du bloc $c_{i,j}$ les données nécessaires. Pour cela, on effectue n étapes, telles qu'à la k ème étape ($0 \leq k < n$) :

- pour chaque ligne, on diffuse à tous les processus d'une ligne le bloc de A situé k positions à droite du bloc diagonal ;
- pour chaque bloc, on accumule dans $c_{i,j}$ le produit du bloc de A reçu par diffusion et du bloc de B courant ;
- pour chaque colonne, on permute circulairement vers le haut les blocs de B.

1.2 Définition et conventions

1.2.1 Types de données

Pour implémenter cet algorithme, nous avons choisi de représenter les matrices de dimension N comme des vecteurs de dimension N^2 .

Si i et j représentent respectivement l'indice de la ligne et de la colonne de l'élément auquel nous souhaitons accéder, $j + i * N$ est l'indice de cet élément dans le vecteur. Cette représentation tient du fait que le passage en argument d'une matrice dont on ne connaît pas la taille pose problème.

Les blocs sur lesquels chaque processus travaille sont eux-mêmes représentés par des vecteurs de taille N/q avec q^2 le nombre de processus.

1.2.2 Initialisation

En théorie, chaque processus ne connaît que le bloc de A et de B qui lui est associé au lancement du programme. Pour simplifier le problème, nous supposons

que chaque processus a une connaissance entière de A et B et en extrait ses blocs au démarrage.

Cela nous permet de ne pas traiter une phase d'initialisation qui consisterais à envoyer à chaque processus les blocs qui lui appartiennent.

Les matrices A et B sont initialisées par des valeurs aléatoires petites (pour vérification des calculs) entre 0 et 4.

Chapitre 2

Implémentation

2.1 Organisation des communicateurs MPI

2.1.1 Création d'une grille

MPI permet de créer une grille de communicateur. La fonction `MPI_Cart_create()` crée un communicateur dans lequel chaque processus est repérable par un indice de ligne et de colonne.

Dans notre programme, il s'agira de notre communicateur global. Il n'est pas utilisé en tant que tel car l'algorithme ne nécessite pas d'effectuer des communications globales, mais il sert de support à la création de communicateurs pour chaque ligne et chaque colonne de la matrice.

Au sein de ce communicateur, les processus sont repéré par leur ligne et leur colonne et se veront attribué le bloc des matrices A et B qui correspondent à leur position dans la grille de communication.

2.1.2 Calcul

On effectue autant d'itération qu'il y a de processus sur une ligne. Pour chaque bloc, si l'indice d'itération correspond à notre indice de colonne, on envois sa sous-matrice A aux autres processus de la ligne. Sinon on reçoit une sous-matrice A provenant d'un autre processus de la ligne. Ces deux opération sont effectuées grace à la fonction `MPI_Bcast()`.

Une fois le bloc A reçu ou envoyé, on effectue un produit matriciel entre les bloc A et B que l'on stocke dans la matrice C puis on permute les blocs B.

Pour permuer les blocs B, chaque processus envoie son bloc B au processus du dessous (du point de vue de la grille), et reçoit un nouveau bloc B du processus du dessus. Cette opération est effectuée par la fonction `MPI_Sendrecv_replace()`.

Une fois que ce procédé à été fait autant de fois que de processus sur une ligne, chacun des processus possède le résultat du bloc de C qui correspond à sa position dans la matrice.

Il suffit ensuite de rassembler les résultats pour former la matrice C résultat complète.