TP Algorithmique Parallèle
Produit de matrices en MPI

Philippe SWARTVAGHER
philippe.swartvagher@enseirb-matmeca.fr

19 avril 2024

Ce TP sera évalué : à la fin de la séance, il faudra me rendre sur Thor, dans le projet `IF247 - Algorithmique Parallèle` :

- un petit rapport qui présente surtout des graphes montrant les performances de votre implémentation ;
- le code de votre implémentation, dans une archive `tar.gz`.

On va implémenter avec MPI un produit de matrices (aussi appelé `GEMM` : *General Matrix Multiplication*). Les matrices seront distribuées sur p processus MPI (*i.e.*, aucun processus ne stockera une matrice complète).

1 Parallélisation du produit de matrices

Le produit AB de deux matrices carrées A et B de taille $N \times N$ donne une matrice carrée C de taille $N \times N$, obtenue à l'aide de l'algorithme suivant :

```
1: for i = 1 to N do
2:     for j = 1 to N do
3:         for k = 1 to N do
4:             C_ij ← C_ij + A_ik B_kj
5:         end for
6:     end for
7: end for
```

On remarque qu'il est possible de changer l'ordre des boucles et placer la boucle sur k en premier :

```
1: for k = 1 to N do
2:     for i = 1 to N do
3:         for j = 1 to N do
4:             C_ij ← C_ij + A_ik B_kj
5:         end for
6:     end for
7: end for
```

Ainsi, si chaque C_{ij} est situé sur un processus MPI différent, les différentes itérations de la boucle sur k seront séquentielles, mais les deux boucles sur i et j peuvent s'exécuter en parallèle. Dit autrement : pour un k donné, les différents C_{ij} peuvent être incrémentés simultanément.

Il est donc nécessaire de répartir les éléments des trois matrices sur les processus MPI. Pour simplifier, on ne va considérer que la multiplication de matrices carrées et travailler avec un nombre carré de processus MPI p (4, 9, 16, ...). On va utiliser une distribution 2D : les processus MPI vont être organisés en un carré contenant $\sqrt{p}$ lignes et $\sqrt{p}$ colonnes. Chaque processus MPI aura donc des coordonnées (i, j), ce qui correspondra également aux morceaux de matrices A, B et C qu'il possède.

2 Chaque processus MPI possède un élément

Pour commencer simplement et comprendre l'idée de l'algorithme, chaque processus ne va posséder qu'un élément de chaque matrice (ce qui va nous limiter à de très petites matrices).

Prenons 9 processus MPI. Chaque processus MPI possède un élément, donc on va travailler sur des matrices 9×9. Le processus MPI de coordonnées (i, j) possédera les valeurs A_{ij}, B_{ij} et C_{ij}. Il sera aussi responsable de réaliser le calcul pour obtenir la valeur finale de C_{ij}.

Les éléments des matrices seront distribués comme suit (la distribution est identique pour les trois matrices ; le numéro correspond au rang `MPI_COMM_WORLD` du processus qui possède l'élément) :

0	1	2
3	4	5
6	7	8

Le calcul de la matrice C peut se décomposer de la façon suivante :

$$C = \underbrace{\begin{array}{|c|c|c|}\hline A_{00}B_{00} & A_{00}B_{01} & A_{00}B_{02} \\ \hline A_{10}B_{00} & A_{10}B_{01} & A_{10}B_{02} \\ \hline A_{20}B_{00} & A_{20}B_{01} & A_{20}B_{02} \\ \hline \end{array}}_{k=0} + \underbrace{\begin{array}{|c|c|c|}\hline A_{01}B_{10} & A_{01}B_{11} & A_{01}B_{12} \\ \hline A_{11}B_{10} & A_{11}B_{11} & A_{11}B_{12} \\ \hline A_{21}B_{10} & A_{21}B_{11} & A_{21}B_{12} \\ \hline \end{array}}_{k=1} + \underbrace{\begin{array}{|c|c|c|}\hline A_{02}B_{20} & A_{02}B_{21} & A_{02}B_{22} \\ \hline A_{12}B_{20} & A_{12}B_{21} & A_{12}B_{22} \\ \hline A_{22}B_{20} & A_{22}B_{21} & A_{22}B_{22} \\ \hline \end{array}}_{k=2}$$

Pour chaque k, on remarque que tous les processus d'une même ligne ont besoin de la même valeur de A et tous les processus d'une même colonne ont besoin de la même valeur de B. On va donc faire des broadcasts sur les lignes et les colonnes.

1. Écrivez un code de base qui récupère la taille de `MPI_COMM_WORLD` et le rang du processus dans ce communicateur.
2. Assurez-vous que le nombre de processus soit un nombre carré.
3. Calculez les coordonnées du processus dans la grille de la distribution 2D.
4. Créez deux communicateurs : un qui contient tous les processus de la même ligne et un autre pour les processus de la même colonne.
5. Puisque chaque processus ne gère qu'un élément de chaque matrice, déclarez trois variables `a`, `b` et `c` de type `int`, judicieusement initialisées pour faciliter le débugage.
6. Chaque itération de la boucle sur k réalise les deux broadcasts sur les colonnes et sur les lignes (qui est la racine de chaque broadcast ?) et réalise l'opération `c += a * b`.
7. Réalisez une fonction qui permet d'afficher le contenu des différentes matrices. Puisque les matrices sont distribuées sur plusieurs processus MPI, chaque processus MPI affichera une ligne ressemblant à :

```
A[1][2] = 2
```

8. N'oubliez pas de libérer les communicateurs créés.

3 Chaque processus MPI possède un bloc

Au lieu d'avoir un unique élément de chaque matrice, les processus MPI vont maintenant posséder un bloc de chaque matrice.

Les modifications à apporter au code sont les suivantes :

1. Récupérez comme paramètre du programme la taille de bloc b. Les matrices seront donc de taille $b\sqrt{p} \times b\sqrt{p}$ si on travaille avec p processus MPI.
2. Allouez et initialisez la mémoire correspondant aux blocs des trois matrices.
3. On ne broadcast plus une unique valeur, mais le bloc entier.
4. L'opération `c += a * b` devient vraiment la multiplication matricielle comme décrite dans le premier algorithme ci-dessus.
5. Il faudra adapter la fonction pour afficher les matrices.
6. Puisque vous avez alloué les matrices de façon dynamique, il ne faut pas oublier de libérer la mémoire.

4 Analyse de performances

Les temps mesurés n'auront pas vraiment de sens (les machines de l'Enseirb-Matmeca ne sont pas faites pour exécuter des applications HPC), mais l'objectif est de réfléchir à ce qu'on veut mesurer et comment.

Pour mesurer la durée des exécutions, on peut utiliser la fonction `MPI_Wtime()`, qui renvoie un `double` correspondant au nombre de secondes écoulées depuis un instant défini dans le passé.

Quelle(s) partie(s) du programme veut-on chronométrer ?

Quels paramètres faire varier pour mesurer le passage à l'échelle de notre programme ? Faites quelques graphiques.

5 Pour aller plus loin

1. Enlevez la restriction sur le nombre carré de processus MPI et sur la forme carrée des matrices.
2. Testez différentes distributions, mais avec les mêmes tailles de matrices (ce sont les tailles de bloc qui changent). Qu'observez-vous (en terme de performances) ?