ENSTA Paris – AMS301 – Calcul scientifique parallèle – Année 2021-2022

TP 3 : Utilisation du cluster `Cholesky` — Étude de la performance parallèle

---

L'objectif de cette séance est d'apprendre à utiliser un cluster et effectuer des études de performance parallèle. Les exercices seront réalisés à distance sur les stations de travail de l'ENSTA (pour le développement de vos codes) le cluster `Cholesky` du mésocentre de l'IP Paris (pour les tests de performance). Les codes de départ sont disponibles sur le site Internet du cours.

Exercice 0. Utilisation du cluster `Cholesky`

Passez en revue le tutoriel *"Utilisation du cluster `Cholesky`"*. Connectez-vous à distance au cluster `Cholesky`, et testez les commandes indiquées dans le tutoriel.

Exercice 1. Étude de performance sur le problème de différences finies 1D

Reprenez le code que vous avez parallélisé dans le cadre de l'exerice 3 *("Différences finies 1D")* du dernier TP. Étudiez la scalabilité forte et la scalabilité faible de ce code sur le cluster :

(1) Pour la scalabilité forte, mesurez les temps de calcul que vous obtenez avec 5, 10, 20, 40 et 80 processus pour une taille de domaine "suffisamment grande" *(le temps de calcul total pour toutes les itération doit être de l'ordre de plusieurs secondes)*.

(2) Pour la scalabilité faible, mesurez les temps de calcul que vous obtenez avec 5, 10, 20, 40 et 80 processus pour une taille de domaine qui augmente linéairement avec le nombre de processus.

Dans tous les cas, assurez-vous de ne pas utiliser plus de 2 nœuds de calcul en même temps *(pour ne pas monopoliser les ressources)*. Pour pouvoir conclure sur les performances de votre code, tracez la courbe d'accélération *("speedup")* pour la scalabilité forte, et la courbe d'efficacité *("efficiency")* pour la scalabilité faible. Comparez avec les performances maximales théoriques.

Conseil : Il est parfois plus pertinent de considérer des temps de calcul "par itération" que des temps de calcul "pour l'ensemble des itérations".

Exercice supplémentaire. Parallélisation du produit matriciel

On fournit un code séquentiel (`matmul.c` en `C`, avec l'extension `.cpp` en `C++`) qui effectue la multiplication de matrices carrées, $\mathbf{A} = \mathbf{BC}$, où les matrices $\mathbf{B}$ et $\mathbf{C} \in \mathbb{R}^{N\times N}$ sont données. Les matrices $\mathbf{A}$ et $\mathbf{B}$ sont stockées par lignes et la matrice $\mathbf{C}$ est stockée par colonnes.

Pour démarrer ...

Proposez une version parallèle de ce code en utilisant la stratégie *"par blocs de lignes"* : chaque processus doit calculer un bloc de lignes de la matrice $\mathbf{A}$. Pour simplifier, on suppose que tous les processus ont accès à toutes les valeurs de la matrice $\mathbf{C}$ au début du calcul *(aucune phase de communication n'est nécessaire)*. Le code final doit fonctionner pour un nombre de processus P quelconque et une taille de matrice N quelconque également *(pas forcément multiple du nombre de processus)*. Il doit prendre en entrée le paramètre N et écrire la matrice résultante $\mathbf{A}$ dans un unique fichier ASCII[1].

[1] Des fonctions séquentielles pour afficher des matrices au format Matlab et imprimer des matrices dans un

Après avoir validé votre code *(i.e. pour des matrices de départ identiques, les fichiers ASCII résultants doivent être identiques, quelque soit le nombre de processsus)*, étudiez les performances parallèles de votre code sur une station de travail de l'école et sur le cluster Cholesky.

Conseil : Pour le calcul des temps, ne tenez compte que de la partie correspondant au produit matriciel.

Pour allez plus loin ...

Proposez un deuxième code où les calculs sont parallélisés sur 2 processus et les données sont réparties : chaque processus n'a accès qu'à la moité des lignes de **B** et à la moitié des colonnes de **C**. Deux phases de calcul et des communications sont nécessaires. Pour simplifier, on suppose que N est un multiple de 2. On cherche à minimiser l'utilisation de la mémoire.

Le code final doit prendre en entrée le paramètre N *(on suppose que l'utilisateur encodera un multiple de* 2*)* et écrire la matrice résultante **A** dans un unique fichier ASCII.

fichier ASCII sont fournies dans le code de départ pour vous aider. Il n'est pas nécessaire de paralléliser l'affichage au format Matlab.