UNIVERSITÉ D'ORLÉANS **M1 Info**

Informatique Année **2018 - 2019**

Contrôle continu : Programmation Parallèle - 1h30

Exercice 1. Petites Questions (4pts)

Dans toutes les questions *A* désigne un tableau d'entiers de longueur *n*. Le nombre de processeurs est noté *nprocs* et sauf indication contraire **on ne suppose pas** que *n* est divisible par *nprocs*. Enfin on notera toujours *root* le processeur référence pour une communication collective par exemple.

1. Si *n=15* et *nprocs=4* donnez le contenu des deux tableaux *send_counts* et *displs* paramètres de la fonction ci-dessous ainsi que la valeur du paramètre *n_local* pour chaque processeur

   `MPI_Scatterv(A,send_counts,displs,MPI_INT,A_local, n_local, MPI_INT,root,MPI_COMM_WORLD)`

2. *A_local* est donc la partie de *A* que chaque processeur reçoit. Après avoir effectué du calcul en local on suppose qu'on veut réunir le tableau global sur le processeur *root*. Quelle fonction utilisez vous ? Donnez les paramètres sachant que sur *root* il y a une variable *B* déjà déclarée et allouée pour la réception des données.

3. On souhaite calculer la somme cumulée des éléments de *A* sachant qu'on a distribué le tableau (Question1). Ecrivez les lignes de code nécessaires à ce calcul sachant que seul le processeur *root* doit avoir le résultat final.

Exercice 2. La suite de Syracuse (5pts)

Une suite de Syracuse est une suite telle que $U_0 = x$ avec $x > 0$ et

$$U_i = \begin{cases} \frac{U_{i-1}}{2} & si\ U_{i-1}\ est\ pair \\ 3U_{i-1}+1 & sinon \end{cases}$$

On souhaite vérifier si un tableau d'entiers U de taille n correspond à une suite de Syracuse alors que ce tableau est initialement sur le processeur `root` mais qu'on a déjà procédé à la distribution. La partie de U sur chaque processeur est $Ulocal$.

1. Décrivez le principe de la parallélisation en indiquant un ordre de grandeur de la complexité parallèle. Vous pouvez faire un schéma sur un exemple.

2. Est-il nécessaire d'avoir des communications au-delà de la distribution initiale de U.

3. Quelles sont les fonctions MPI que vous allez utiliser (en justifiant) ?

Exercice 3. L'algorithme de Cannon (6pts)

1. Décrivez succinctement l'algorithme de Cannon pour effectuer le produit de deux matrices en parallèle.

2. Quel est l'intérêt d'utiliser une topologie cartésienne ?

3. Quelles sont les fonctions MPI qui facilitent la mise en œuvre de cette multiplication ?

4. Donnez l'implémentation[1] correspondante en supposant que les deux matrices sont déjà distribuées sur les *nprocs* processeurs initiaux (MPI_COMM_WORLD). Si on note $n \times n$ la taille des matrices, on prendra une topologie cartésienne de taille $p \times p$ construites à partir des *nprocs* et telle que n est divisible par p.

[1] Vous pouvez utiliser la fonction `void Prod(int* A, int* B, int* C, int n)` réalisant $C = A.B$ sans l'implémenter

Exercice 4. L'algorithme de Darboux (5pts)

A partir de la description de l'algorithme de remplissage d'un MNT répondez aux questions suivantes :

1. Si le fichier contenant le MNT initial est disponible sur un processeur $root$ et un seul, comment peut-on distribuer le MNT initial sur l'ensemble des processeurs ? Est-il possible d'utiliser *MPI_Scatterv* en considérant qu'on souhaite travailler sur des MNT de très grande taille ?

2. En considérant uniquement la matrice Z comment va-t-on distribuer les données ? Est ce que les distribuer par bloc a un intérêt ? Et si on suppose qu'on a $nprocs$ processeurs quelle sera la taille des données distribuées sur chaque processeur ?

3. Si on suppose la partie distribution terminée, est ce que l'algorithme nécessite des phases de communication entre les itérations ? Si oui, comment seront-elles effectuées ?

4. L'algorithme s'arrête lorsque lors d'une itération aucune valeur de W^{t-1} n'a été modifiée ($W^t = W^{t-1}$). Comment gérer cette condition lorsque le calcul est réparti sur les $nprocs$ processeurs ?