1 Exercice : Analyse de performances

Dans cette exercice, nous allons lancer une évaluation du matériel et de l'implémentation MPI sur le calculateur à disposition. Nous analyserons ensuite les résultats.

2 Exercice : Mise en pratique

Ce premier exercice est une mise en pratique des commandes MPI pour la création d'une topologie cartésienne. Ensuite nous utiliserons des communication collectives, dans le contexte du communicateur global, puis celui de la topologie.

Question1 ()

Dans le fichier `TD5/Grid/grid.c`, nous souhaitons constuire un maillage décomposé en sous-domaines suivant une topologie cartésienne.

(a) Complétez la portion de code `Zone-1-A` afin de créer, en utilisant `MPI_Dims_create` et `MPI_Cart_create`, une topologie cartesienne adaptée au nombre de processus MPI.

(b) Complétez la portion de code `Zone-1-B` dans laquelle chaque processus doit calculer les dimensions de son sous-domaine, vous aurez besoin de `MPI_Cart_coords`. Calculez aussi le nombre de maille du sous-domaine, et vérifiez que la somme des nombres de mailles de tous les sous-domaines donne bien le nombre total de mailles (utilisez `MPI_Reduce`).

Question2 ()

(a) Maintenant, nous souhaitons que chaque processus ait un tableau du nombre de mailles dans l'ensemble des autres processus. Complétez la portion de code `Zone-2-A` avec un all_gather du nombre de maille sur l'ensemble des processus.

(b) Dans la `Zone-2-B`, à l'aide d'un all_gather sur la topologie, obtenir le nombre de maille dans les processus voisins.

(c) Nous souhaitons maintenant recouvrir les all_gather avec du calcul (`Zone-2-A` et `Zone-2-B`). Reprendre les deux questions si dessus avec les équivalents non bloquant des communications collectives. Ajouter une boucle de calcul entre l'appel et la complétion de chacune des collectives, et faire croître le nombre d'itérations jusqu'à saturer l'overlapping.

(d) Comparer les nombres d'itérations obtenus pour chacune des deux collectives non-bloquantes. Interprétez.

3 Exercice : Compréhension

Dans cet exercice, nous allons voir différents exemples de ce que permet la norme, ce qu'elle interdit, ou ce sur quoi elle ne statue pas.

Question1 ()

Mélange de collectives et de communications points-à-points. Commentez le code suivant :

```
1         MPI_Request reqs[2];
2         switch(rank)
3         {
4             case 0:
5                 MPI_Ibarrier(comm, &reqs[0]);
6                 MPI_Send(buf, count, dtype, 1, tag, comm);
7                 MPI_Wait(&reqs[0], MPI_STATUS_IGNORE);
8             break;
```

```
            case 1:
                MPI_Irecv(buf, count, dtype, 0, tag, comm, &reqs[0]);
                MPI_Ibarrier(comm, &reqs[1]);
                MPI_Waitall(2, reqs, MPI_STATUSES_IGNORE);
            break;
        }
```

Question2 ()

Détail d'implémentation. Commentez le code suivant :

```
        switch(rank) {
            case 0:
                MPI_Bcast(buf1, count, type, 0, comm);
                MPI_Bcast(buf2, count, type, 1, comm);
            break;
            case 1:
                MPI_Bcast(buf2, count, type, 1, comm);
                MPI_Bcast(buf1, count, type, 0, comm);
            break;
        }
```

4 Exercice : Pipeline de réductions non-bloquantes.

Nous allons faire une réduction sur la totalité d'un tableau, potentiellement très grand. Pour cela, nous comparerons 4 approches.

Question1 ()

Dans le fichier `TD5/Pipelining/pipelining.c`, nous partons d'un tableau répliqué sur N processus.

(a) Première option : faire la réduction en mode bloquant. Complétez pour cela la fonction `allreduce_blocking`. Relevez le temps d'exécution pour différentes tailles de tableau et nombres de processus.

(b) Deuxième option : faire la réduction en mode non-bloquant. Complétez pour cela la fontion `allreduce_nonblocking`. Relevez le temps d'exécution pour différentes tailles de tableau et nombres de processus.

(c) Troisième option : faire la réduction en mode non-bloquant mais à l'aide de deux réductions, et une seule opération de complétion. Complétez pour cela la fonction `allreduce_nonblocking_pipelined2`. Relevez le temps d'exécution pour différentes tailles de tableau et nombres de processus.

(d) Troisième option : faire la réduction en mode non-bloquant mais à l'aide de deux réductions, et une seule opération de complétion. Complétez pour cela la fonction `allreduce_nonblocking_pipelined4`. Relevez le temps d'exécution pour différentes tailles de tableau et nombres de processus.

(e) Commentez les résultats obtenus en terme de performances.