TPs

Exercice 1 : Tirages

1/ Chaque processus lance un dé. On calcule la somme des lancés et on ajoute le résultat à un total (stocké par le processus 0). On recommence tant qu'aucun des dés lancés n'est tombé sur un 6.

2/ On cherche à tirer le 2 de trèfle dans un jeu de 52 cartes. Pour cela, on tire une carte :
- si c'est le 2 de trèfle, on s'arrête,
- sinon, on replace la carte tirée dans le jeu, on mélange, puis on recommence.

On veut vérifier informatiquement que le nombre moyen de tirages nécessaires à sortir le 2 de trèfle est 52. Chaque processus tirera des cartes jusqu'à sortir le 2 de trèfle, puis notera le nombre de tirages. L'opération sera recommencé X (passé en argument) fois par processus pour obtenir une moyenne par processus. Le résultat sera la moyenne des moyennes obtenues par chaque processus.

Exercice 2 : Filtres

On dit qu'on *filtre* un tableau lorsque tout élément à l'indice i, à part le premier et le dernier élément du tableau qui restent constants, est remplacé par la moyenne des éléments aux indices $i-1$, i et $i+1$. C'est une opération classique du traitement d'image.

1/ Ce programme fonctionne avec deux processus. Le processus de rang 0
1. génère aléatoirement un tableau d'un million de `double` compris entre 0 et 1,
2. l'envoie au processus 1,
3. génère un second tableau d'un million de `double` compris entre 0 et 1,
4. filtre le second tableau.

Le processus de rang 1
1. reçoit le premier tableau du processus 0,
2. filtre le tableau.

L'intégralité des étapes est répété 100 fois.

Utiliser MPI_Ssend et MPI_Recv pour les transfert de tableau. Mesurer avec MPI_Wtime le temps qu'il faut au processus 0 pour effectuer les opérations $2, 3, 4$ (sommé sur les 100 fois).

Remplacer MPI_Ssend par MPI_Isend/MPI_Wait (placés au bons endroits) et mesurer à nouveau le temps des opérations $2, 3, 4$.

2/ Cette fois-ci, le processus de rang 0
1. génère aléatoirement un tableau d'un million de `double` compris entre 0 et 1,
2. l'envoie au processus 1,
3. filtre le tableau.

Le processus de rang 1
1. génère aléatoirement un tableau d'un million de `double` compris entre 0 et 1,
2. filtre le tableau,
3. reçois le tableau envoyé par le processus 0,
4. calcule sa moyenne.

L'intégralité des étapes est répété 100 fois.

Utiliser MPI_Ssend et MPI_Recv pour les transfert de tableau. Mesurer avec MPI_Wtime le temps qu'il faut au processus 1 pour effectuer les opérations $2, 3, 4$ (sommé sur les 100 fois).

Remplacer MPI_Recv par MPI_Irecv/MPI_Wait (placés au bons endroits) et mesurer à nouveau le temps des opérations $2, 3, 4$.

Exercice 3 : Opérations sur les matrices

On considère une matrice stockée *ligne d'abord* sous la forme d'un tableau d'entier : la matrice

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 10 & 11 & 12 & 13 \\ 20 & 21 & 22 & 23 \\ 30 & 31 & 32 & 33 \\ 40 & 41 & 42 & 43 \end{bmatrix}$$

sera ainsi stockée dans le tableau :

0 1 2 3 10 11 12 13 20 21 22 23 30 31 32 33 40 41 42 43

On pourra utiliser la structure suivante.

```
typedef struct Matrix_
{
    int rows;
    int cols;
    int* data;
} Matrix;
```

1/ Écrire les fonctions *createLine*, *createCol*, *createDiag* et *createUpper* qui construisent (en fonction des dimensions de la matrices) les types MPI *Line*, *Col*, *Diag* et *Upper* permettant la transmission d'une ligne, d'une colonne, de la diagonale principale, et de la matrice triangulaire supérieure.

Transmettre la ligne 3, la colonne 2, la diagonale et la la matrice triangulaire supérieure du processus 0 au processus 1. Le processus 1 affichera et réinitialisera la matrice après chaque réception.

2/ Utiliser le type *Line* pour transmettre la matrice en inversant l'ordre des lignes. La matrice ci-dessus deviendrait alors

$$\begin{bmatrix} 40 & 41 & 42 & 43 \\ 30 & 31 & 32 & 33 \\ 20 & 21 & 22 & 23 \\ 10 & 11 & 12 & 13 \\ 0 & 1 & 2 & 3 \end{bmatrix}$$

De même, utiliser le type *Col* pour transmettre la matrice en inversant l'ordre des colonnes.

3/ Utiliser les types *Line* et *Col* pour transmettre une matrice et obtenir sa transposée.

4/ On veut calculer le produit de deux matrices A et B de dimensions $N \times N$ en parallèle sur N processus.

- Le processus 0 génère les matrices A et B, puis les broadcast.
- Le processus de rang r calcule la ligne r de la matrice résultat.
- Tous les processus envoie leurs résultats au processus 0.

Adapter l'algorithme pour que des matrices de dimensions $N \times N$ puissent être multipliés en parallèle par R processus si $R * arg = N$. On passera arg en argument du programme.

Comparer le temps de l'algorithme parallèle avec l'algorithme série sur des grosses matrices.

Exercice 4 : Propriétés de jeux de données

On considère la structure suivante :

```
typedef struct Data_
{
    char id;
    double data[TSIZE];
    short pMean;
    short pMedian;
} Data;
```

L'attribut `id` ne peut être qu'une lettre majuscule de l'alphabet.

Créer une structure MPI permettant de transmettre une variable de type `Data`. Le processus 0 remplit aléatoirement les 26 jeux de données, un pour chaque identifiant, par des double compris entre -1 et 1, puis les distribue le plus équitablement possible sur tous les processus. Chaque processus se charge alors de remplir les variables `pMean` et `pMedian` :

- `pMean` est mis à 1 si la moyenne des éléments de `data` est strictement positive, 0 sinon,
- `pMedian` est mis à 1 si le nombre d'éléments strictement positifs est supérieur au nombre d'éléments négatifs ou nuls, 0 sinon.

Le but est de compter le nombre de jeux de données pour lesquels `pMean`$\neq$`pMedian`.

Exercice 5 : Tournoi de lancés de dés

Seize personnes participent à un tournoi de lancés de dés. Quatre groupes de quatre joueurs sont formés. Les joueurs d'un même groupe jouent entre eux. À chaque tour, chaque joueur lance un dé et le plus gros score gagne. La partie est gagné par le premier joueur qui gagne trois tours d'affilés. Les quatres vainqueurs des quatre groupes forment un groupe final, et rejoue pour désigner un vainqueur.

Implémentation. Chaque personne sera représenté par un processus.

1/ On notera R_W le rang dans le communicateur par défaut. À partir du communicateur par défaut, créer quatre communicateurs de quatre processus :

- processus $R_W = 0, 1, 2, 3$,
- processus $R_W = 4, 5, 6, 7$,
- processus $R_W = 8, 9, 10, 11$,
- processus $R_W = 12, 13, 14, 15$.

Les numéros de processus donné sont ceux du communicateur par défaut. Les rangs des processus dans leurs nouveaux communicateurs devront être $0, 1, 2, 3$.

2/ Faire jouer les groupes entre eux et indiquer chaque vainqueur.

3/ Créer un nouveau communicateur avec les quatre processus vainqueurs.

4/ Le groupe joue et indique le vainqueur.