TP MPI N°3 --- Types dérivés

Les types MPI prédéfinis permettent la communication des données de même type contiguës en mémoire. Les types dérivés sont à utiliser dans la communication des données hétérogènes ou non contiguës en mémoire, surtout quand ceci est à faire régulièrement. L'énoncé de ce TP décrit quatre méthodes de définition de types dérivés, seules deux d'entre eux seront dans la pratique de ce TP.

1. Méthode générale : `MPI_Type_struct`

Un type structure utilisateur ne peut être utilisé par les fonctions de communication de MPI. Tout type de données mentionné par celles-ci doit être de type de données MPI. La méthode générale de définition du type dérivé permet la création d'un type de données MPI à partir d'une structure utilisateur. Elle consiste à spécifier l'agencement mémoire des données dans le type.

Exemple :

```
typedef struct {
       float a, b ;
       int   n;
} indata_type_t;

indata_type_t indata = {-2.0, 2.0, 20} ;
MPI_Bcast( &indata, 1 indata_type_t, 0, MPI_COMM_WORLD ); /* FAUX */
```

L'appel `MPI_Bcast` est faux, car `indata_type_t` n'est pas un type MPI, les fonctions de la bibliothèque MPI ne la connaît pas. La fonction suivante définit son type dérivé correspondant.

```
/*           Definition d'un type derive                */
void Build_derived_type( indata_type_t *indata_ptr,
                         MPI_Datatype *message_type_ptr )
{
     int          blok_lengths[3];  /* longueur de chaque champs */
     MPI_Aint     displacements[3]; /* deplacement memoire de chaque
                         champs par rapport au debut de la structure */
     MPI_Aint     addresses[4];     /* add. des champs de la struct */
     MPI_Datatype typelist[3];      /* tableau de types des champs  */

     /* Types de chaque membre */
     typelist[0] = typelist[1] = MPI_FLOAT;
     typelist[2] = MPI_INT;

     /* Nombre d'elements de chaque membre */
     block_lengths[0]= block_lengths[1]= block_lengths[2]=1;

     /* Obtenir l'add. des ≠ champs d'une structure */
     MPI_Address( indata_ptr, &addresses[0] );
     MPI_Address( &(indata_ptr->a), &addresses[1] );
     MPI_Address( &(indata_ptr->b), &addresses[2] );
     MPI_Address( &(indata_ptr->n), &addresses[3] );
```

```
    /* Calcul les ecarts en memoire des champs par
       rapport au debut de la variable */
    displacements[0] = addresses[1] - addresses[0];
    displacements[1] = addresses[2] - addresses[0];
    displacements[2] = addresses[3] - addresses[0];

    /* Creation du type derive */
    MPI_Type_struct( 3, block_lengths, displacements,
                     typelist, message_type_ptr );

    MPI_Type_commit( message_type_ptr ); /* Tout type derive    */
                    /* doit etre sauvegarde avant l'utilisation */
}
```

Après la définition du type dérivé, le contenu d'une variable de type structure peut être communiqué d'un processus à un autre comme suit :

```
MPI_Datatype INDATA_TYPE;
Build_derived_type( &indata, &INDATA_TYPE );
MPI_Bcast( &indata, 1, INDATA_TYPE, 0, MPI_COMM_WORLD );
```

Exercice :

Reprendre le programme parallèle de la multiplication Matrice-Vecteur version ligne :

- Définir une structure `tailleMat` ayant deux champs de type entier (nombre de lignes et nombre de colonnes) pour la matrice ;
- Définir le type dérivé correspondant en utilisant la méthode générale ;
- Initialiser la taille de matrice dans le processus maître, puis les diffuser à tous les esclaves ;
- Effectuer la multiplication de Matrice-Vecteur avec des matrices quelconques (pas forcément carrées).

2. Données stockées à des intervalles réguliers

Cette méthode faciliter la définition de type dérivé pour les vecteurs colonnes de matrice.

- ```
  int MPI_Type_vector( int blocnumber, int bloclength, int stride,
                       MPI_Datatype oldtype, MPI_Datatype *newtype );
  ```
  Ex : `MPI_Type_vector( 4, 1, 6, MPI_DOUBLE, newtype );`
  crée un type représente les colonnes d'une matrice `4x6.`
- ```
  int MPI_Type_commit( MPI_Datatype *newtype );
  ```

Exercice :

Soient V un vecteur d'ordre n et A une matrice carrée d'ordre n, calculer X=VA :

avec $X_i = \sum_{j=0}^{n-1} V_j A_{ij}, \quad i = 0, \ldots, n-1$.

- Faire une copie de votre programme de l'exercice 1;
- Définir le type dérivé correspondant aux colonnes de matrice A;
- Effectuer la multiplication de Vecteur-Matrice en suivant le même algorithme que celui de l'exercice 1.