Programmation parallèle

UFR SAT, CFPP - MaDSI 1

Travaux pratiques 5

REPUBLIQUE DU SENEGAL
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE

UNITÉ DE FORMATION ET DE RECHERCHE DE SCIENCES APPLIQUÉES ET DE TECHNOLOGIE
Section Informatique

Objectifs:

1. Découvrir les fonctions **MPI_Pack** et **MPI_Unpack**
2. Pouvoir encapsuler une donnée sur un buffer et le communiquer dans un réseau de processus
3. Augmenter le nombre de processus et pouvoir en observer le comportement de votre PC

Dans ce TP, vous allez communiquer différents types de données à l'aide de **MPI_Pack** et **MPI_Unpack**.
Votre programme lit un entier et un réel à partir de l'entrée standard (à partir du processus 0, comme auparavant), et le communique à tous les autres processus avec un appel **MPI_Bcast**. **MPI_Pack** permet de compresser les données dans un tampon (par exemple, vous pouvez utiliser **char packbuf [100]**; mais considérez comment utiliser plutôt **MPI_Pack_size**).

Notez que MPI_Bcast, [contrairement aux opérations MPI_Send/MPI_Recv], exige que la même quantité de données soit envoyée et reçue.

Tous les processus sont fermés lorsqu'un entier négatif est lu.

Voici les fonctions MPI utilisées dans ce TP:
MPI_Pack, MPI_Unpack, MPI_Bcast

```
#include <stdio.h>
#include "mpi.h"

int main( argc, argv )
int argc;
char **argv;
{
    int           rank;
    int           packsize, position;
    int           a;
    double        b;
    char          packbuf[100];

    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
```

```
    do {
     if (rank == 0) {
         scanf( "%d %lf", &a, &b );
         packsize = 0;
         MPI_Pack( &a, 1, MPI_INT, packbuf, 100, &packsize,
              MPI_COMM_WORLD );
        MPI_Pack( &b, 1, MPI_DOUBLE, packbuf, 100, &packsize,
              MPI_COMM_WORLD );
     }
     MPI_Bcast( &packsize, 1, MPI_INT, 0, MPI_COMM_WORLD );
     MPI_Bcast( packbuf, packsize, MPI_PACKED, 0, MPI_COMM_WORLD );
     if (rank != 0) {
         position = 0;
         MPI_Unpack( packbuf, packsize, &position, &a, 1, MPI_INT,
                MPI_COMM_WORLD );
       MPI_Unpack( packbuf, packsize, &position, &b, 1, MPI_DOUBLE,
                MPI_COMM_WORLD );
     }

     printf( "Processus %d a reçu %d et %lf\n", rank, a, b );
    } while (a >= 0);

    MPI_Finalize();
    return 0;
}
```