Programmation parallèle

UFR SAT, CFPP - MaDSI 1

Travaux pratiques 4

Objectifs:

1. Découvrir la fonction **MPI_Bcast**
2. Comprendre l'utilité des fonctions utilisées
3. Augmenter le nombre de processus et pouvoir en observer le comportement

---

Dans ce TP, vous allez communiquer différents types de données avec un seul appel de diffusion MPI (**MPI_Bcast**). Votre programme lit un entier et une valeur double précision à partir de l'entrée standard (à partir du processus 0, comme auparavant), et les communique à tous les autres processus avec un appel **MPI_Bcast**.

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

Voici les fonctions MPI utilisées dans ce TP:

MPI_Address MPI_Type_struct MPI_Type_commit MPI_Type_free MPI_Bcast

Visiter ce lien pour afficher la description de chaque fonction MPI.

Code C

Nom du programme: bcast2.c

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

int main( argc, argv )
int argc;
char **argv;
{
    int          rank;
    struct { int a;
             double b;
           } value;
    MPI_Datatype mystruct;
```

```
    int          blocklens[2];
    MPI_Aint     indices[2];
    MPI_Datatype old_types[2];

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

    /* Une valeur de chaque type */
    blocklens[0] = 1;
    blocklens[1] = 1;
    /* Les types de base */
    old_types[0] = MPI_INT;
    old_types[1] = MPI_DOUBLE;
    /* Les emplacements mémoire de chaque element */
    MPI_Address( &value.a, &indices[0] );
    MPI_Address( &value.b, &indices[1] );
    /* Etablir les correspondances */
    indices[1] = indices[1] - indices[0];
    indices[0] = 0;
    MPI_Type_struct( 2, blocklens, indices, old_types, &mystruct );
    MPI_Type_commit( &mystruct );

    do {
if (rank == 0) {
    printf( "Ce programme lit 2 nombres, un entier et un réél, et
effectue un message brodcast à tous les processus.\n" );fflush(stdout);
    printf( "Donner un entier et un réél \n" );fflush(stdout);
    scanf( "%d %lf", &value.a, &value.b );
}

MPI_Bcast( &value, 1, mystruct, 0, MPI_COMM_WORLD );

printf( "Processus %d a reçu %d and %lf\n", rank, value.a,
value.b );fflush(stdout);
 } while (value.a >= 0);

    /* Libérer la mémoire et finaliser */
    MPI_Type_free( &mystruct );
    MPI_Finalize();
    return 0;
}
```

Sortie à l'écran:

```
www:parallel babacardiop$ mpirun -np 4 ./g
Ce programme lit 2 nombre un entier et un réél et effectue un message brodcast à tous les processus.
Donner un entier et un réél
1 1.8
Processus 0 a reçu 1 and 1.800000
Ce programme lit 2 nombre un entier et un réél et effectue un message brodcast à tous les processus.
Donner un entier et un réél
Processus 1 a reçu 1 and 1.800000
Processus 2 a reçu 1 and 1.800000
Processus 3 a reçu 1 and 1.800000
2 2.87
Processus 0 a reçu 2 and 2.870000
Ce programme lit 2 nombre un entier et un réél et effectue un message brodcast à tous les processus.
Donner un entier et un réél
Processus 1 a reçu 2 and 2.870000
Processus 2 a reçu 2 and 2.870000
Processus 3 a reçu 2 and 2.870000
7 -3.6
Processus 0 a reçu 7 and -3.600000
Ce programme lit 2 nombre un entier et un réél et effectue un message brodcast à tous les processus.
Donner un entier et un réél
Processus 1 a reçu 7 and -3.600000
Processus 2 a reçu 7 and -3.600000
Processus 3 a reçu 7 and -3.600000
-1 -8.9
Processus 0 a reçu -1 and -8.900000
Processus 1 a reçu -1 and -8.900000
Processus 2 a reçu -1 and -8.900000
Processus 3 a reçu -1 and -8.900000
www:parallel babacardiop$
```