ENSEIRB-MATMECA

Parcours CISD

Langages du parallélisme

TP 3 (séance 4)

Exercice 1: Somme globale

(1) Ecrire un programme qui calcule la somme de tous les identifiants des processus et qui envoie ce résultat sur tous les processus. Ecrire deux versions, une avec réduction puis diffusion, et l'autre avec réduction globale.

(2) Redéfinissez l'opérateur `MPI_SUM` avec la fonction `MPI_Op_create`.

Exercice 2: communicateurs

(1) Dupliquez le communicateur de base à l'aide de la fonction `MPI_Comm_dup`.

(2) Diviser les processus dans deux communicateurs selon s'ils sont de rang pair ou impair à l'aide de la fonction `MPI_Comm_split`.

Exercice 3: Echange de données

Ecrire un programme dans lequel un processus envoie un entier et un double à un autre processus. Pour cela, vous devrez créer un nouveau type de données.

Exercice 4: Topologies cartésiennes

En MPI, une topologie virtuelle décrit un mapping/réordonnancement des processus MPI selon une figure géométrique. Il y a deux types de topologies principaux supportés par MPI: Cartésien (grid) et Graphe. Ces topologies sont construites sur la notion de communicateurs et de groupes.

(1) Ecrivez un programme dans lequel on crée une grille virtuelle de processus (utilisez `MPI_Dims_create` pour définir les dimensions de la grille et `MPI_Cart_create` pour créer la grille). Faites afficher par chacun des processus, leur ancien rang, leur nouveau rang ainsi que leurs coordonnées dans la grille. (utilisez `MPI_Cart_coords`). Essayez différent nombres de processus.

(2) Modifiez votre programme pour créer une grille virtuelle $q \times q$ de processus. Assurez vous de créer une grille carrée à partir du nombre de processus (pas forcément carré) mis en paramètre de l'exécution.

(3) Créez des communicateurs par ligne et par colonne à l'aide de la fonction `MPI_Cart_sub`.

(4) Modifiez votre programme pour faire tourner un jeton à partir du processus 0 sur les lignes et un sur les colonnes. Que se passe- t-il si une des dimensions ne possède pas de lien retour?

(5) Modifiez votre programme pour que chaque processus envoie un entier à tous les autres processus de sa ligne.

(6) Effectuez des sommes globales sur les colonnes.

(7) Faites des shifts sur les lignes et les colonnes et affichez le voisin de chaque processus sur la grille (utilisez `MPI_Cart_shift`). Que se passe t-il si un processus n'a pas ses 4 voisins sur la grille?

(8) Répartissez des vecteurs sur la grille de processus (utilisez `MPI_Scatter` et `MPI_Gather`).

Message Passing Interface - Quick Reference in C

Environmental

- `int MPI_Init (int *argc, char ***argv)` - Initialize MPI
- `int MPI_Finalize (void)` - Cleanup MPI

Basic communicators

- `int MPI_Comm_size (MPI_Comm comm, int *size)`
- `int MPI_Comm_rank (MPI_Comm comm, int *rank)`

Point-to-Point Communications

- `int MPI_Send (void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)` - Send a message to one process.
- `int MPI_Recv (void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)` - Receive a message from one process
- `MPI_SendRecv_replace()` - Send and receive a message using a single buffer

Collective Communications

- `int MPI_Bcast (void *buf, int count, MPI_Datatype datatype, int root, MPI_Comm comm)` - Send one message to all group members
- `int MPI_Gather (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)` - Receive from all group members
- `int MPI_Scatter (void *sendbuf, int sendcount, MPI_Datatype sendtype, void *recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)` - Send separate messages to all group members

Communicators with Topology

- `int MPI_Cart_create (MPI_Comm comm_old, int ndims, int *dims, int *periods, int reorder, MPI_Comm *comm_cart)` - Create with cartesian topology
- `int MPI_Cart_rank (MPI_Comm comm, int *coords, int *rank)` - Determine rank from cartesian coordinates
- `int MPI_Cart_coords (MPI_Comm comm, int rank, int maxdims, int *coords)` - Determine cartesian coordinates from rank
- `int MPI_Cart_shift (MPI_Comm comm, int direction, int disp, int *rank_source, int *rank_dest)` - Determine ranks for cartesian shift.
- `int MPI_Cart_sub (MPI_Comm comm, int *remain_dims, MPI_Comm *newcomm)` - Split into lower dimensional sub-grids

Constants

Datatypes: `MPI_CHAR`, `MPI_SHORT`, `MPI_INT`, `MPI_LONG`, `MPI_UNSIGNED_CHAR`, `MPI_UNSIGNED_SHORT`, `MPI_UNSIGNED`, `MPI_UNSIGNED_LONG`, `MPI_FLOAT`, `MPI_DOUBLE`, `MPI_LONG_DOUBLE`, `MPI_BYTE`, `MPI_PACKED`

Reserved Communicators: `MPI_COMM_WORLD`