Message Passing Interface (MPI)

Modèle de programmation par envois de messages

MPI ?

- Message Passing Interface
- Bibliothèque d'envois de messages
- Défini par un “forum” de chercheurs et d’industriels du calcul haute performance.

  https://www.mpi-forum.org/docs/

Objectif

- Faciliter la programmation par envois de messages
- Portabilité des programmes parallèles
- Utilisable avec une architecture hétérogène
- Cible stable pour des outils/langages de plus haut niveau

Principe

- Parallèle : Utilisation de plusieurs processeurs (1 processus chacun pour simplifier)
- Exécution séquentiel du programme qui opère des envois et des réceptions de messages sur chaque processus
- Synchronisation des processus via ces messages

Avantage

- Extensibilité : fonctionne avec 1 comme avec 1000 processus

Inconvénient

- Programmation à deux niveaux : global vs local

Programmation Simple Program Multiple Data (SPMD)

1. On écrit un programme C avec des opérations MPI pour les communications
2. On le compile et le lance avec *mpirun -np n* où n est le nombre de processus voulu.
3. n copies de l'exécutable sont créées avec pour numéros de processus 0,1,...,n −1.
4. Les processus s'exécutent en mode asynchrone et se synchronisent lors des échanges de messages.
5. Les processus se terminent tous (s'il n'y a pas eu blocage).

Contenu

- Constantes, types et fonctions pour C et F77
- Opérations de communications point-à-point
- Opérations de communications collectives
- Groupes de processus
- Et plus dans les versions suivantes...

Communication point-à-point : Envoi

Fonction :

int MPI_Send (void* buf, int count, MPI_Datatype datatype, int dst, int tag, MPI_Comm grp)

Arguments :

- buf : adresse du début du buffer
- count : nombre de valeurs à envoyer
- datatype : type de chaque valeur
- dst : rang du processus de destination
- tag : étiquette pour identifier le message
- grp : groupe de processus

MPI Types

MPI_Datatype:	Equivalent C :
- MPI_CHAR	signed short
- MPI_SHORT	signed short int
- MPI_INT	signed int
- MPI_LONG	signed long int
- MPI_FLOAT	float
- MPI_DOUBLE	double

Communication point-à-point : Réception

Fonction :

int MPI_Recv (void* buf, int count, MPI_Datatype datatype, int src, int tag, MPI_Comm grp, MPI_Status *status)

Arguments :

- buf : adresse du début du buffer
- count : nombre de valeurs à recevoir
- datatype : type de chaque valeur
- src : rang du processus d'émission
- tag : étiquette pour identifier le message
- grp : groupe de processus
- status : structure contenant des informations effectives du message

Communication point-à-point : Exemple

Ici le processus 0 envoie un message au processus 1 :

```
char msg[10]; int myrank, tag = 5; MPI_Status status;

/* récupère le rang du proc dans le groupe global MPI_COMM_WORLD */
MPI_Comm_rank(MPI_COMM_WORLD, &myrank);

/* envoi ou réception en fonction du rang */
if (myrank == 0) {
        strcpy(msg,"Hello");
        MPI_Send(msg, strlen(msg)+1, MPI_CHAR, 1, tag, MPI_COMM_WORLD);
} else if (myrank == 1) {
        MPI_Recv(msg, 10, MPI_CHAR, 0, tag, MPI_COMM_WORLD, &status);
}
```

Communication collective

Cette fonction permet au processus de rang *src* d'envoyer un message à tous les autres processus du groupe *grp* :

int MPI_Bcast (void* buf, int count, MPI_Datatype datatype, int src, MPI_Comm grp)