Message Passing Interface (MPI) – IV
Grouper les processus

Amal KHABOU
amal.khabou@lri.fr

Terminologie (I)

- Un `MPI_Group` est un objet décrivant une liste de processus qui forment une entité logique
  - sa taille est `MPI_Group_size`
  - chaque processus du groupe a un unique identifiant, son rang, compris entre 0 et (taille du groupe -1) `MPI_Group_rank`
  - un groupe ne peut pas être utiliser pour effectuer une communication

Terminologie (II)

- Un `MPI_Comm`(unicator) est un objet contenant
  - un ou deux groupes de processus (*intra* ou *inter*-communicateurs)
  - Information sur la topologie
  - Attributs
- Un communicateur a un gestionnaire d'erreurs qui y est attaché
- Un nom
- Dans la suite on se focalise sur les *intra*-communicators i.e. la liste des processus participants sont décrit par un seul groupe

Les communicateurs prédéfinis

- `MPI_COMM_WORLD`
  - contient tous les processus lancés par `mpirun`
  - ne peut pas être modifié

- `MPI_COMM_SELF`
  - contient juste le processus local , il est de taille 1
  - ne peut pas être modifié

Création de nouveaux communicateurs

- Tous les communicateurs dans MPI sont dérivés de `MPI_COMM_WORLD` ou `MPI_COMM_SELF`
- Créer ou libérer un communicateur est une opération **collective** => tous les processus du communicateur doivent appeler la fonction MPI avec les mêmes arguments
- Les méthodes de création de nouveaux communicateurs
  - diviser le communicateur de départ en n-parties
  - créer des sous groupes du communicateur de départ
  - réorganiser les processus suivant une certaine topologie
  - engendrer de nouveaux processus
  - connecter deux applications et fusionner leurs communicateurs

Diviser un communicateur

```
MPI_Comm_split ( MPI_Comm comm, int color,
int key, MPI_comm *newcomm);
```

- Divise `comm` en sous-communicateurs
  - tous les processus ayant la même couleu `color` appartiendront au même sous-communicateur
  - trier les processus ayant le même `color` suivant la valeur de la clé `key`
  - si la valeur de la clé est identique pour tous les processus avec la même `color`, alors l'ordre est le même que dans `comm`

Exemple de MPI_Comm_split (I)

```
MPI_Comm newcomm;
int color, rank;

MPI_Comm_rank (MPI_COMM_WORLD, &rank);
color = rank%2;

MPI_Comm_split (MPI_COMM_WORLD, color, rank,
&newcomm);
MPI_Comm_size (newcomm, &size);
MPI_Comm_rank (newcomm, &rank);
```

- Diviser les processus de rang pair/impair
- Un processus
  - peut faire partie d'un **seul** communicateur généré
  - ne peut pas "voir" les autres communicateurs
  - ne peut pas "voir" combier de communicateurs ont été créés

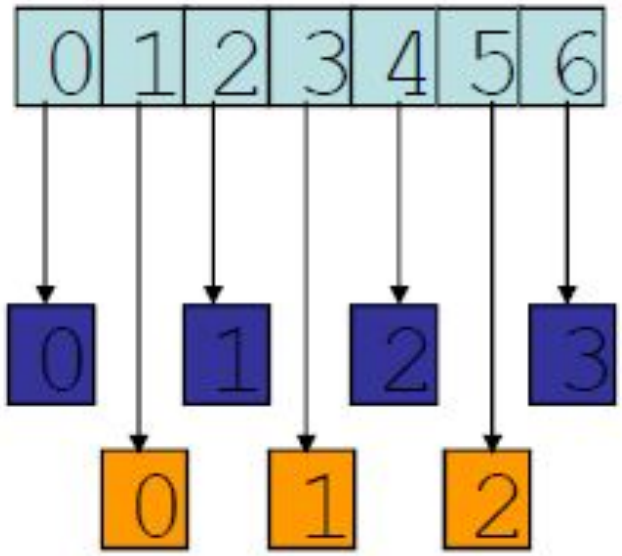
Exemple for MPI_Comm_split (II)

- rang et taille du nouveau communicateur

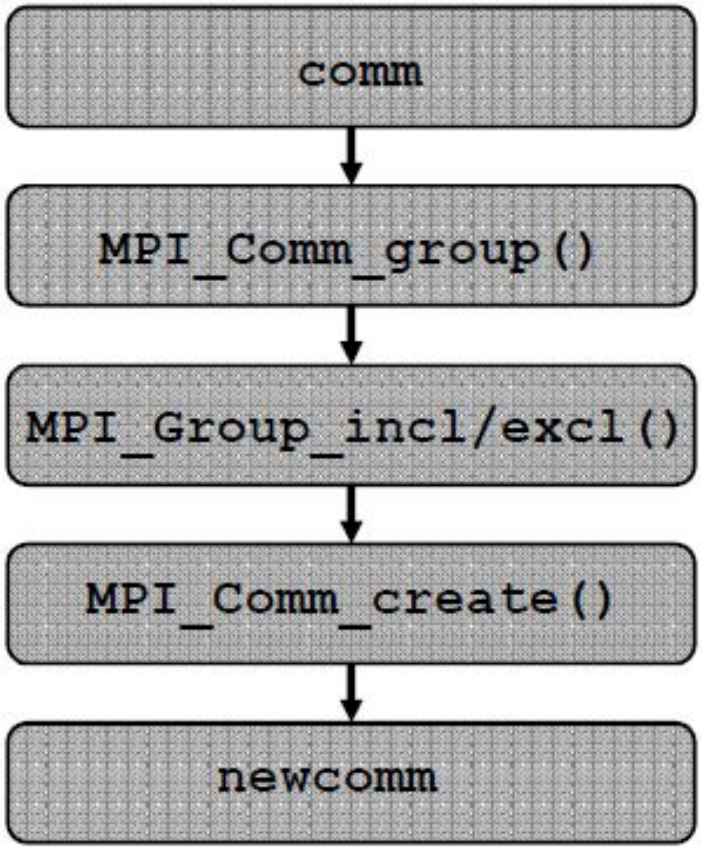
```
MPI_COMM_WORLD


newcomm, color=0, size = 4

newcomm, color=1, size = 3
```



Modifier un groupe de processus



Communicateur de départ

Extraire le groupe de processus du communicateur de départ

Modifier le groupe

Créer un nouveau communicateur basé sur le groupe modifié

Nouveau communicateur

Extraire le groupe de processus

```
MPI_Comm_group (MPI_Comm comm, MPI_Group *group);
```

- avec
  - comm: le communicateur de départ
  - group: le groupe décrivant la liste de processus faisant partie du communicateur comm

Modifier un groupe de processus (I)

```
MPI_Group_incl (MPI_Group group, int cnt, int
ranks[], MPI_Group *newgroup);
MPI_Group_excl (MPI_Group group, int cnt, int
ranks[], MPI_Group *newgroup);
```

- avec
  - `group`: le groupe décrivant la liste de processus faisant partie du communicateur comm
  - `ranks[]` : un tableau d'entiers contenant les rangs des processus faisant partie du groupe, qui doivent être
    - inclus dans le nouveau groupe `MPI_Group_incl`
    - Exclus du groupe de départ `MPI_Group_excl`
  - `newgroup`: le nouveau groupe

Modifier un groupe de processus (II)

- Il existe d'autres fonctions pour construire de nouveaux groupes
  - `MPI_Group_range_incl`
  - `MPI_Group_range_excl`
  - `MPI_Group_difference`
  - `MPI_Group_intersection`
  - `MPI_Group_union`

Créer un nouveau communicateur basé sur un groupe

```
MPI_Comm_create ( MPI_Comm comm, MPI_Group newgroup,
MPI_Comm *newcomm);
```

- Avec
  - `comm`: communicateur de départ
  - `group`: le groupe décrivant la liste de processus faisant partie du nouveau communicateur
  - `newcomm`: le nouveau communicateur
- Noter que
  - `newcomm` est toujours une sous partie de `comm`
  - on peut générer un **seul** communicateur à la fois (à l'opposé de `MPI_Comm_split`)
    - La liste des arguments doit être identique pour tous les processus de `comm`
  - `Newcomm` sera `MPI_COMM_NULL` pour les processus qui ont été exclus/pas inclus dans `newgroup`

Libérer les groupes et les communicateurs

```
MPI_Comm_free ( MPI_Comm *comm);
MPI_Group_free ( MPI_Group *group);
```

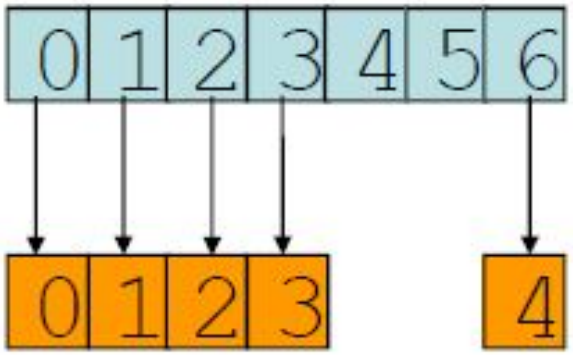
- retourne `MPI_COMM_NULL` et `MPI_GROUP_NULL` respectivement
- `MPI_Comm_free` et `MPI_Group_free` sont des opérations collectives

Exemple de MPI_Comm_create

- Générer un communicateur qui contient seulement les 4 premiers et le dernier processus du communicateur de départ

```
MPI_COMM_WORLD


newcomm, size = 5
```



1ère option: utiliser MPI_Group_incl

```
MPI_Comm newcomm;
MPI_Group group, newgroup;
int color, size, ranks[5], cnt;

MPI_Comm_size (MPI_COMM_WORLD, &size);

cnt = 5;
ranks[0] = 0; ranks[1] = 1; ranks[2] = 2;
ranks[3] = 3; ranks[4] = size-1

MPI_Comm_group (MPI_COMM_WORLD, &group);
MPI_Group_incl (group, cnt, ranks, &newgroup)
MPI_Comm_create (comm, newgroup, &newcomm);
if ( newcomm != MPI_COMM_NULL ) {
MPI_Comm_rank (newcomm, &nrank);
MPI_Comm_free (&newcomm);
MPI_Group_free (&newgroup);
}
MPI_Group_free (&group);
```

2ème option: utiliser MPI_Group_excl

```
MPI_Comm newcomm;
MPI_Group group, newgroup;
int color, size, ranks[…], cnt;
/* NOTE: Assuming that size >5, ranks is large enough
etc. */
MPI_Comm_size (MPI_COMM_WORLD, &size);
cnt = 0;
for ( i=4; i<(size-1); i++) {
ranks[cnt++] = i;
}
MPI_Comm_group (MPI_COMM_WORLD, &group);
MPI_Group_excl (group, cnt-1, ranks, &newgroup)
MPI_Comm_create (comm, newgroup, &newcomm);
if ( newcomm != MPI_COMM_NULL ) {
MPI_Comm_rank (newcomm, &nrank);
MPI_Comm_free (&newcomm);
MPI_Group_free (&newgroup);
}
MPI_Group_free (&group);
```

Les informations relatives à la topologie dans les communicateurs

- Certaines applications ont besoin d'informations, non seulement sur le communicateur, mais aussi sur l'organisation des processus dedans:
  - informations sur la topologie
  - 1D, 2D, 3D…topologie cartésienne
  - les limites de chaque dimension
  - les voisins de chaque processus
- Grille 2D
  - position suivant la direction des abscisses: rank%dimx
  - position suivant la direction des ordonnées: floor(rank%dimx)

MPI_Cart_create

```
MPI_Cart_create ( MPI_Comm comm, int ndims,
int *dims, int *periods, int reorder, MPI_Comm
*newcomm);
```

- Crée un nouveau communicateur ayant une topologie cartésienne avec
  - ndims dimensions
  - chaque dimension ayant dims[i] processus, i=0, …,ndims-1
  - periods[i] indique si la grille est périodique ou pas pour la ième dimension
  - reorder: indique si la bibliothèque MPI peut réordonner les processus