Communicators and Topologies

A communicator is a collection of processors that can send messages to each other. A topology is a structure imposed on the processors in a communicator that allows the processors to be addressed in different ways. The most basic approach consists of building a group, and then having the system associate a context with a group using MPI_Comm_group, MPI_Group_incl, and MPI_Comm_create.

1. MPI_Comm_group

```c
int MPI_Comm_group(
    MPI_Comm old_comm /* in */,
    MPI_Group* group /* out */)
```

It simply returns the group underlying the communicator `comm`.

2. MPI_Group_incl

```c
int MPI_Group_incl(
    MPI_Group old_group /* in */,
    int new_group_size /* in */,
    int ranks_in_old_group[] /* in */,
    MPI_Group* new_group /* out */)
```

It creates a new group from a list of processors in the existing group, `old_group`. The number of processors in the new group is `new_group_size`, and the processors to be included are listed in `rank_in_old_group`. Processor 0 in `new_group` has rank `ranks_in_old_group[0]` in `old_group`, processor 1 in `new_group` has rank `ranks_in_old_group[1]` in `old_group`, etc.
3. MPI_Comm_create

```c
int MPI_Comm_create(
    MPI_Comm old_comm /* in */,
    MPI_Group new_group /* in */,
    MPI_Comm* new_comm /* out */)
```

It associates a context with the group `new_group` and creates the communicator `new_comm`. All of the processors in `new_group` belong to the group underlying `old_comm`.

4. MPI_Cart_create

```c
int MPI_Cart_create(
    MPI_Comm old_comm /* in */,
    int number_of_dims /* in */,
    int dim_sizes[] /* in */,
    int wrap_around[] /* in */,
    int reorder /* in */,
    MPI_Comm* cart_comm /* out */)
```

This creates a new communicator, `cart_comm`, by caching a cartesian topology with `old_comm`. Information used on the construction of the cartesian topology are:

- `number_of_dims`(the number of dimensions in the cartesian coordinate system)
- the array `dim_sizes`(the order of each dimension)
- the array `wrap_around`
  (each dimension is circular: `wrap_around[i]=1`,
  or linear: `wrap_around[i]=0`)
- `reorder`(own position in cartesian coordinates).
5. MPI_Comm_split

```c
int MPI_Comm_split(
    MPI_Comm old_comm /* in */,
    int split_key /* in */,
    int rank_key /* in */,
    MPI_Comm* new_comm /* out */)
```

It partitions the group associated with `old_comm` into subgroups, one for each value of `split_key`. The rank in the new group is determined by the value of `rank_key`.

6. MPI_Cart_coords

```c
int MPI_Cart_coords(
    MPI_Comm cart_comm /* in */,
    int rank /* in */,
    int number_of_dims /* in */,
    int coordinates[] /* out */)
```

It takes the rank of a processor in `cart_comm` and returns its coordinates `coordinates` in the grid.

7. MPI_Cart_rank

```c
int MPI_Cart_rank(
    MPI_Comm cart_comm /* in */,
    int coordinates[] /* in */,
    int* rank /* out */)
```

It returns a processor’s rank given its coordinates.