Virtual Topologies

ICHEC
Irish Centre for High-End Computing

ICHEC
Irish Centre for High-End Computing

Topologies

- MPI process topologies allow for simple referencing scheme of processes.
  - Cartesian and graph topologies supported
  - Process topology defines a new communicator

- MPI topologies are virtual – hence 'virtual' topologies
  - Not necessarily related to the physical structure of the computer
  - Process mapping more natural only to the programmer

- Usually no performance benefits
  - BUT code more compact and readable as a result
  - Can allow MPI to optimize communications.

ICHEC
Irish Centre for High-End Computing

How to use a Virtual Topology

- Creating a topology produces a new communicator.
- MPI provides mapping functions:
  - from process ranks (old communicator), to those based on the topology naming scheme,
  - and vice versa.

ICHEC
Irish Centre for High-End Computing

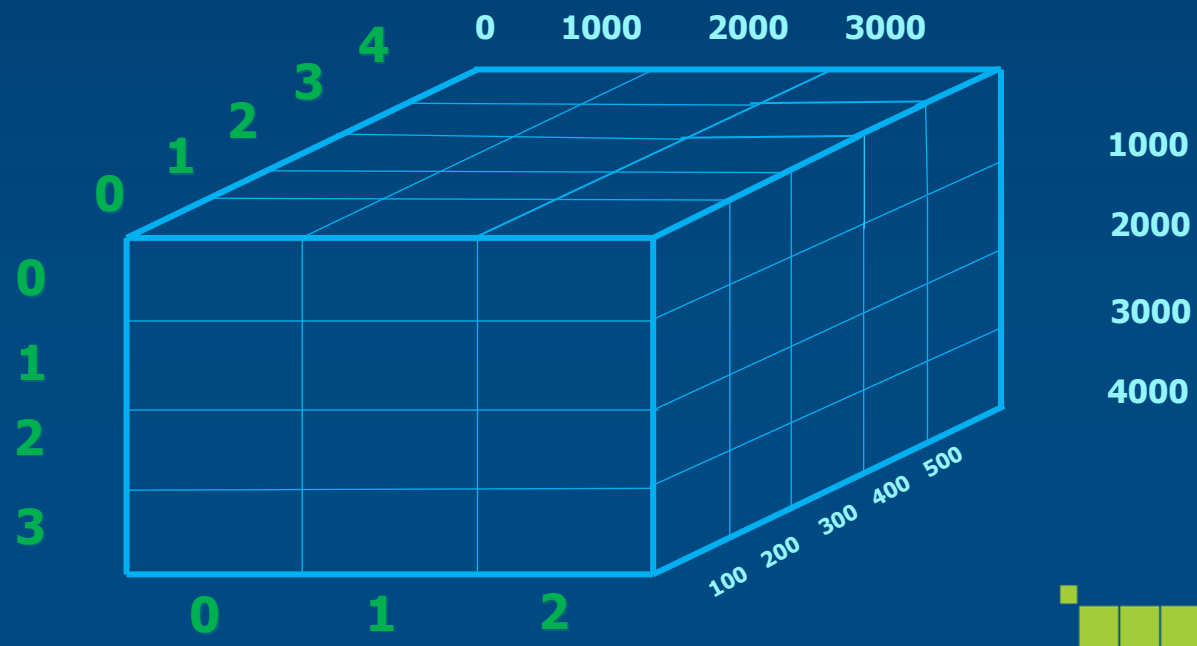
Virtual Topology

- For more complex mapping, MPI routines are available

Global array: A(1:3000, 1:4000, 1:500) = $6 \bullet 10^9$ units

On **3 x 4 x 5 = 60 processors**

Process coordinates = **(0:2), (0:3), (0:4)**

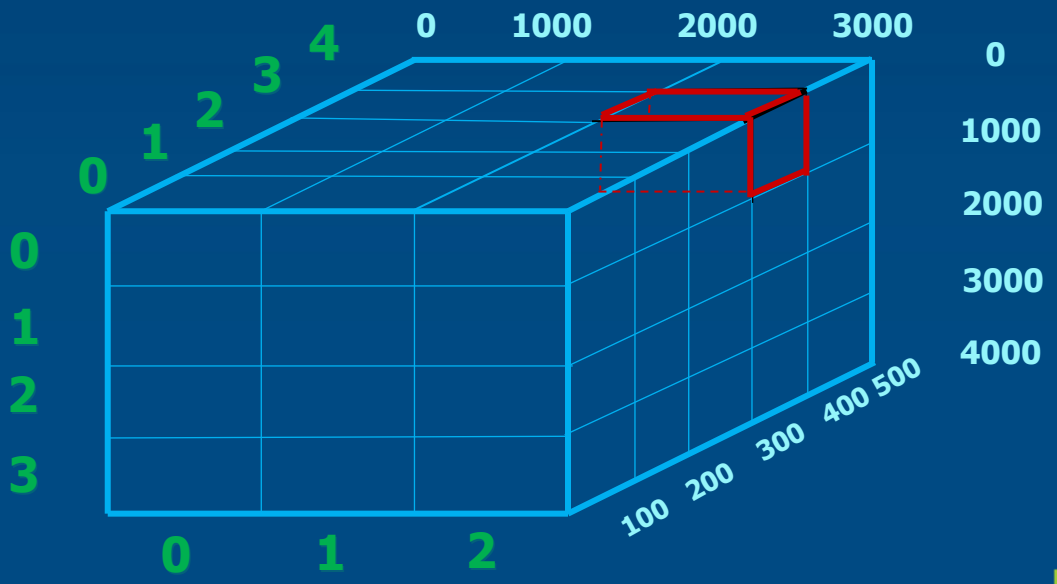


ICHEC
Irish Centre for High-End Computing

Graphical representation eg. Rank = 43

- **Process coordinates:** handled with **virtual Cartesian topologies**
  - ic_0=2, ic_1=0, ic_2= 3 → (rank=43)
  - Coordinate (2, 0, 3) represents process number 43

- Array decomposition: handled by the application program directly
  - It is being assigned the cube A(2001:3000, 1:1000, 301:400)
  - A(2001:3000, 1:1000, 301:400) = $1 \bullet 10^8$ units



ICHEC
Irish Centre for High-End Computing

Topology Types

- **Cartesian Topologies** – `MPI_Cart_create(…)`
  - each process is connected to its neighbor in a virtual grid
  - boundaries can be cyclic, or not
  - processes are identified by Cartesian coordinates
  - communication between any two processes is still allowed.

- **Graph Topologies** – `MPI_Graph_create(…)`
  - general graphs,
  - not covered here.

ICHEC
Irish Centre for High-End Computing

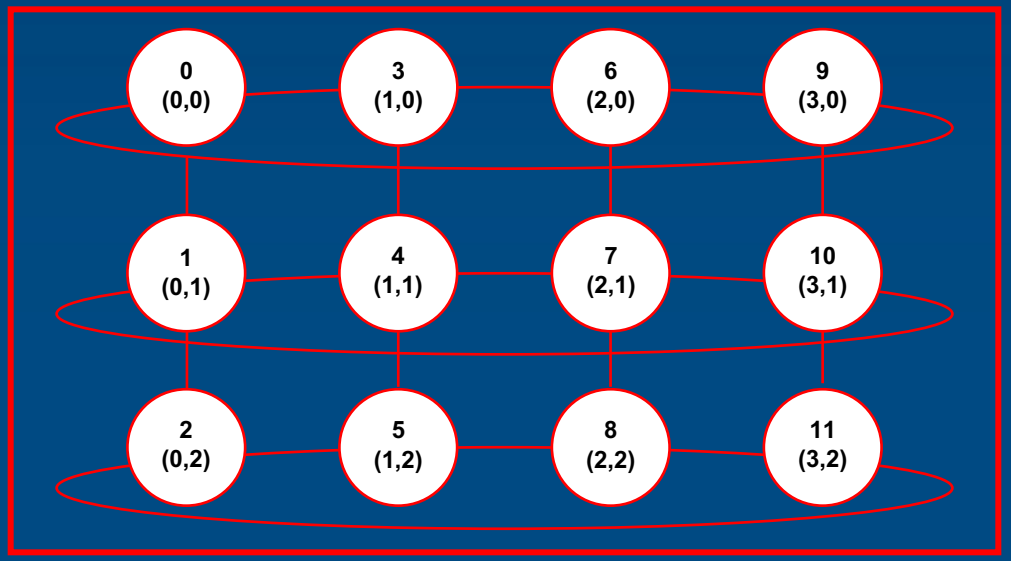
Creating a Cartesian Virtual Topology

- C:

```
int MPI_Cart_create(MPI_Comm comm_old, int ndims,
                    int *dims, int *periods, int reorder,
                    MPI_Comm  *comm_cart)
```

- Fortran:

```
MPI_Cart_create(comm_old, ndims, dims, periods,
                reorder, comm_cart, ierror)
integer :: comm_old, ndims, dims(:)
logical :: periods(:), reorder
integer :: comm_cart, ierror
```

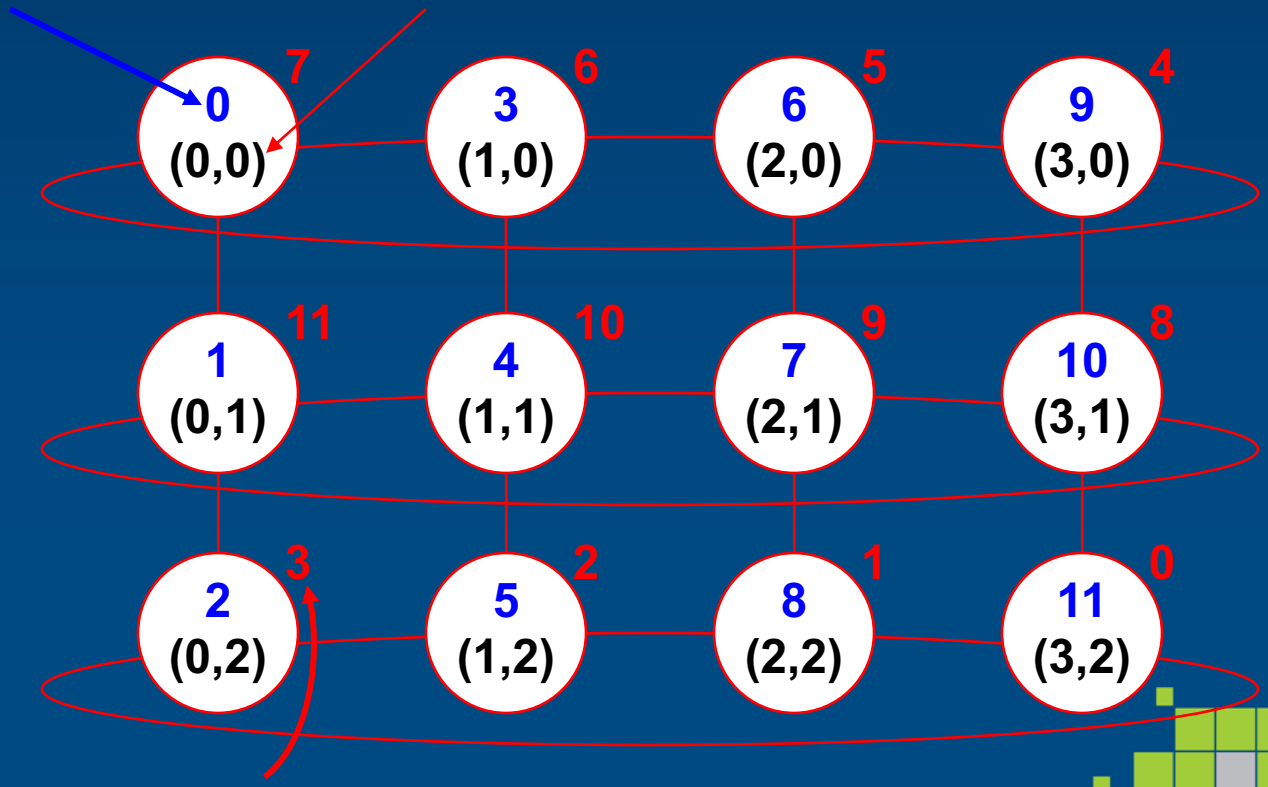


comm_old = MPI_COMM_WORLD
ndims = 2
dims = (4, 3)
periods = (1/.true., 0/.false.)
reorder = *see next slide*

ICHEC
Irish Centre for High-End Computing

Example – A 2-dimensional Cylinder

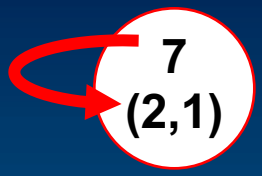
- Ranks and Cartesian process coordinates in comm_cart
- Ranks in comm and comm_cart may differ, if reorder = 1 or .TRUE.
- This reordering can allow MPI to optimize communications



ICHEC
Irish Centre for High-End Computing

Cartesian Mapping Functions

- Mapping ranks to process grid coordinates



- C:

```
int MPI_Cart_coords(MPI_Comm comm_cart,
    int rank, int maxdims, int *coords)
```

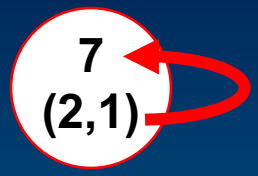
- Fortran:

```
MPI_Cart_coords(comm_cart, rank,
                maxdims, coords, ierror)
integer ::  comm_cart, rank
integer ::  maxdims, coords(*), ierror
```

ICHEC
Irish Centre for High-End Computing

Cartesian Mapping Functions

- Mapping process grid coordinates to ranks



- C:

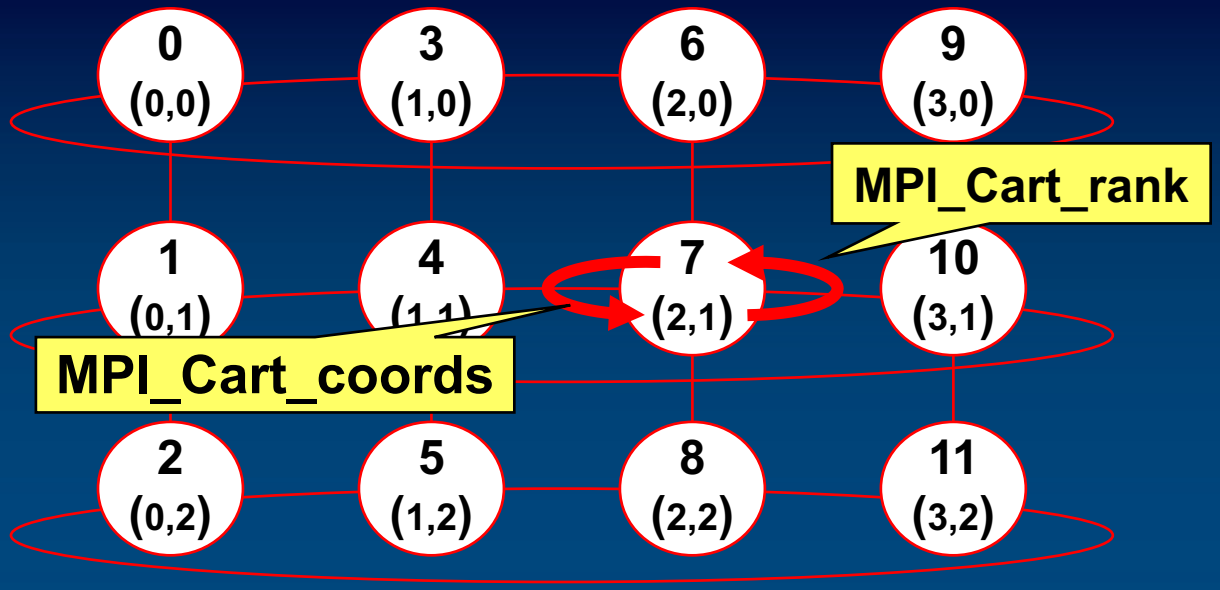
```
int MPI_Cart_rank(MPI_Comm comm_cart,
                  int *coords, int *rank)
```

- Fortran:

```
MPI_Cart_rank(comm_cart, coords, rank,
ierror)
        integer :: comm_cart, coords(:)
        integer :: rank, ierror
```

ICHEC
Irish Centre for High-End Computing

Own coordinates



- C

```
MPI_Comm_rank(comm_cart,&my_rank)
MPI_Cart_coords(comm_cart,my_rank,maxdims,my_coords)
```

- Fortran

```
MPI_Comm_rank(comm_cart, my_rank, ierror)
MPI_Cart_coords(comm_cart, my_rank, maxdims, my_coords,
ierror)
```

ICHEC
Irish Centre for High-End Computing

Cartesian Mapping Functions

- Computing ranks of neighboring processes
- C:

```
int MPI_Cart_shift(MPI_Comm comm_cart, int
    direction, int disp, int *rank_prev, int
    *rank_next)
```

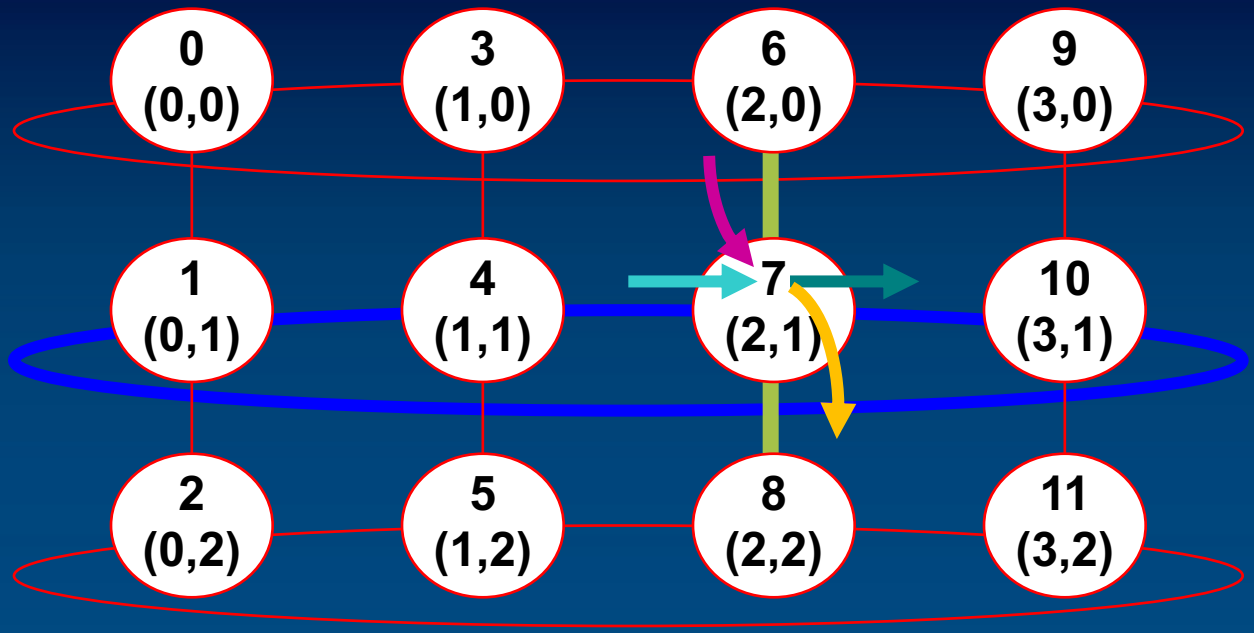
- Fortran:

```
MPI_Cart_shift(comm_cart, direction, disp,
                rank_prev, rank_next, ierror)
                integer :: comm_cart, direction
                integer :: disp, rank_source
                integer :: rank_dest, ierror
```

- Returns MPI_PROC_NULL if there is no neighbor.
- MPI_PROC_NULL can be used as source or destination rank in each communication then, this communication will be a noop!

ICHEC
Irish Centre for High-End Computing

MPI_Cart_shift – Example



invisible input argument: **my_rank** in cart

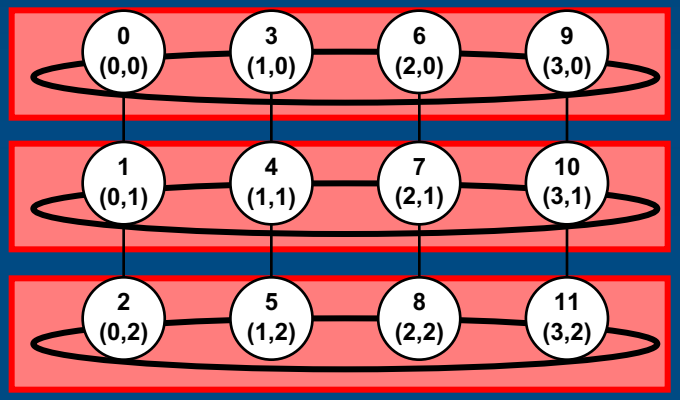
```
MPI_Cart_shift(cart, direction, displace, &rank_prev, &rank_next)
MPI_Cart_shift(cart, direction, displace, rank_prev, rank_next, ierror)
```

example on	0	+1	4	10
process rank=7	1	+1	6	8

ICHEC
Irish Centre for High-End Computing

Cartesian Partitioning

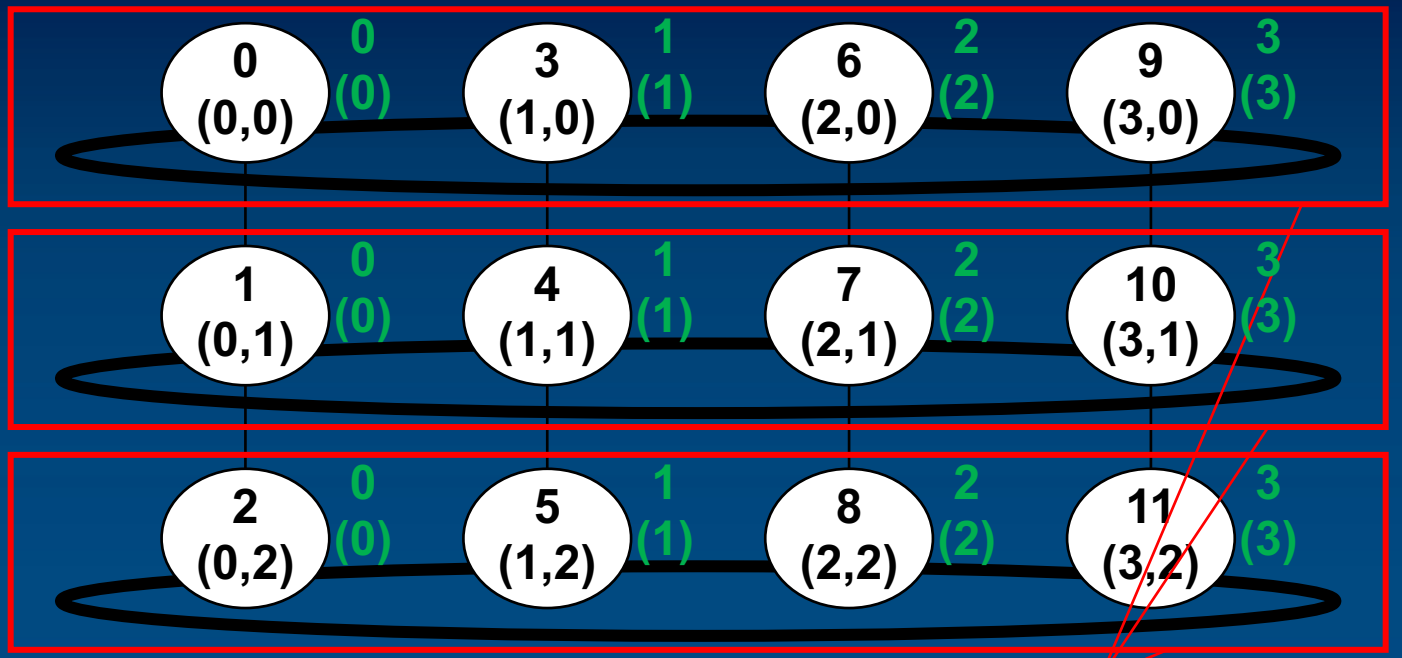
- Cut a grid up into *slices*.
- A new communicator is produced for each slice.
- Each slice can then perform its own collective communications.
- C:
```
int MPI_Cart_sub(MPI_Comm comm_cart, int *remain_dims,
                 MPI_Comm *comm_slice)
```
- Fortran:
```
MPI_Cart_sub(comm_cart, remain_dims, comm_slice, ierror)
```
```
integer :: comm_cart
logical :: remain_dims(*)
integer :: comm_slice, ierror
```



ICHEC
Irish Centre for High-End Computing

MPI_Cart_sub – Example

- Ranks and Cartesian process coordinates in comm_sub



- `MPI_Cart_sub( comm_cart, remain_dims, comm_sub)`
- `MPI_Cart_sub( comm_cart, remain_dims, comm_sub, ierror)`

(1/true, 0/false)