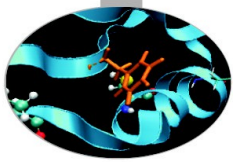
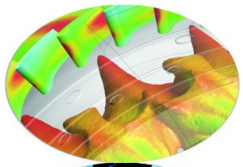
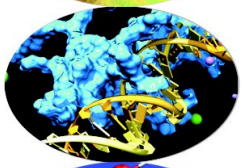
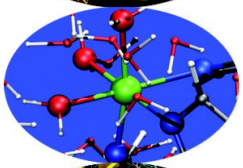
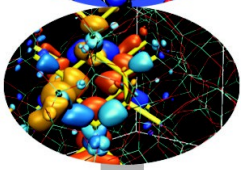
CINECA SCAI
SuperComputing Applications and Innovation



INTRODUCTION TO MPI – VIRTUAL TOPOLOGIES

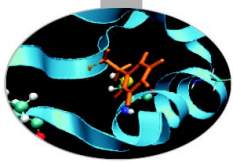
Introduction to Parallel Computing with MPI and OpenMP

18-19-20 november 2013

a.marani@cineca.it
g.muscianisi@cineca.it
l.ferraro@cineca.it

CINECA

VIRTUAL TOPOLOGY



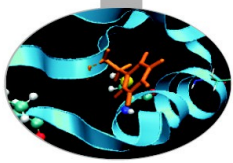
Topology:

- extra, optional attribute that can be given to an intra-communicator; topologies cannot be added to inter-communicators.
- can provide a convenient naming mechanism for the processes of a group (within a communicator), and additionally, may assist the runtime system in mapping the processes onto hardware.

A process group in MPI is a collection of n processes:

- each process in the group is assigned a rank between 0 and n-1.
- in many parallel applications a linear ranking of processes does not adequately reflect the logical communication pattern of the processes (which is usually determined by the underlying problem geometry and the numerical algorithm used).

VIRTUAL TOPOLOGY



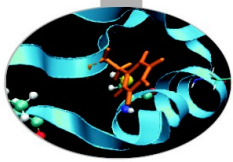
Virtual topology:

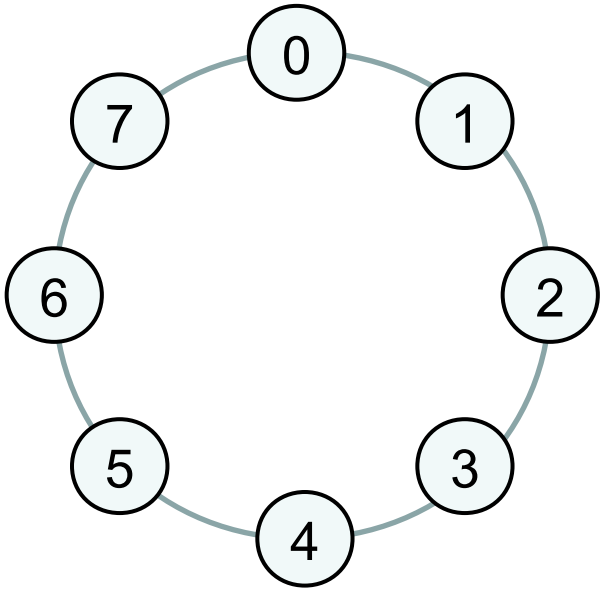
- logical process arrangement in topological patterns such as 2D or 3D grid; more generally, the logical process arrangement is described by a graph.

Virtual process topology .vs. topology of the underlying, physical hardware:

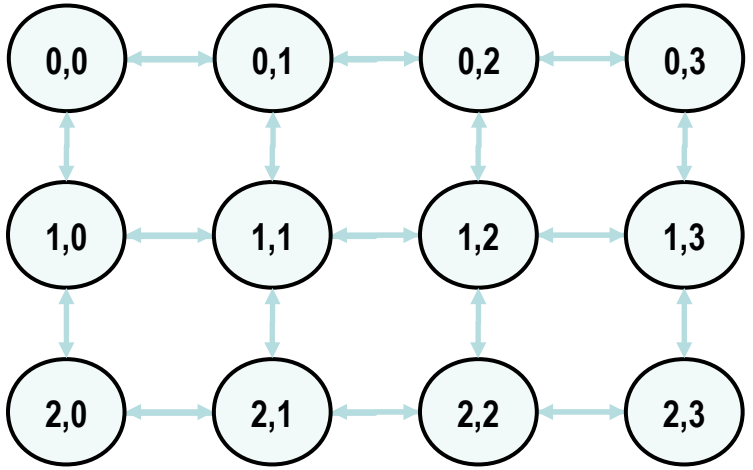
- virtual topology can be exploited by the system in the assignment of processes to physical processors, if this helps to improve the communication performance on a given machine.
- the description of the virtual topology depends only on the application, and is machine-independent.

EXAMPLES



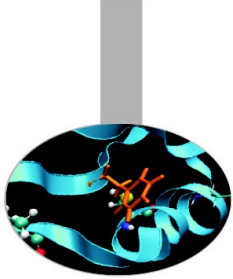


RING



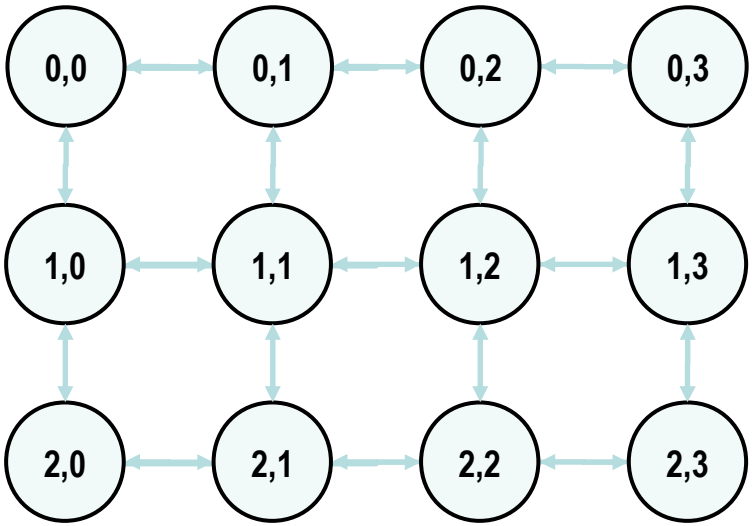
2D-GRID

CARTESIAN TOPOLOGY

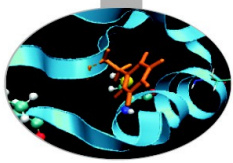


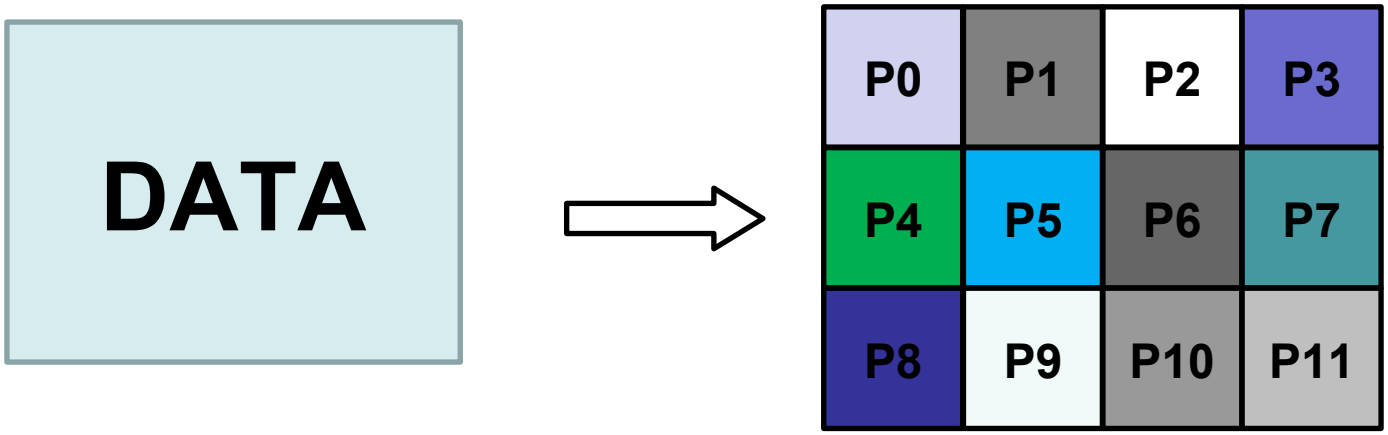
A grid of processes is easily described with a cartesian topology:

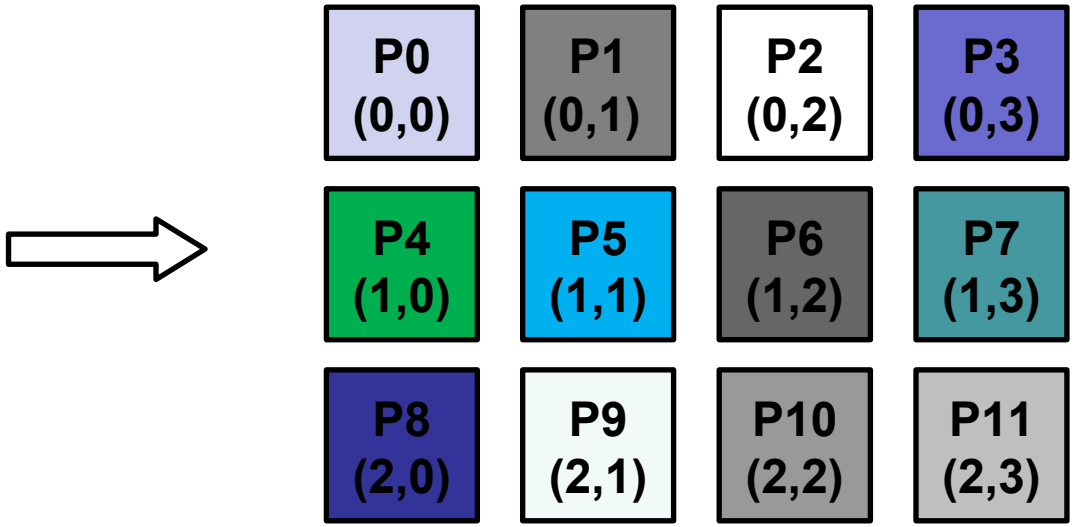
- each process can be identified by cartesian coordinates
- periodicity can be selected for each direction
- communications are performed along grid dimensions only



EXAMPLE: 2D DOMAIN DECOMPOSITION







CARTESIAN CONSTRUCTOR

MPI_CART_CREATE(comm_old, ndims, dims, periods, reorder, comm_cart)

IN comm_old: input communicator (handle)

IN ndims: number of dimensions of Cartesian grid (integer)

IN dims: integer array of size ndims specifying the number of processes in each dimension

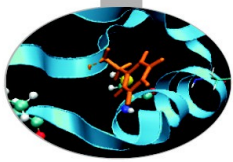
IN periods: logical array of size ndims specifying whether the grid is periodic (true) or not (false) in each dimension

IN reorder: ranking may be reordered (true) or not (false)

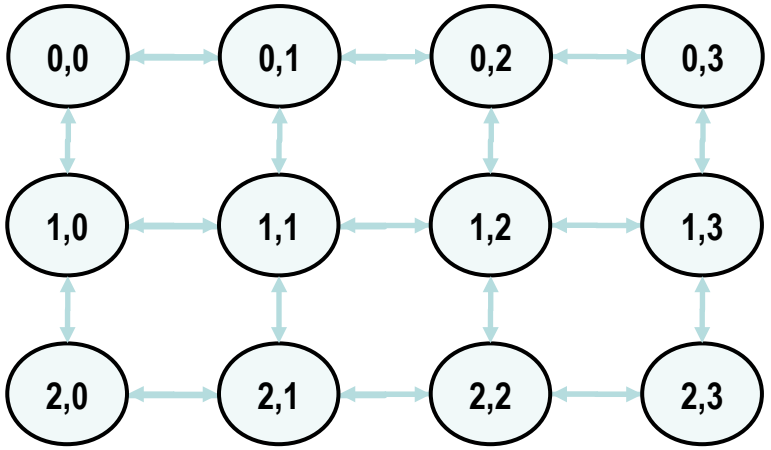
OUT comm_cart: communicator with new Cartesian topology (handle)

- Returns a handle to a new communicator to which the Cartesian topology information is attached.
- Reorder:
  - false: the rank of each process in the new group is identical to its rank in the old group.
  - True: the processes may be reordered, possibly so as to choose a good embedding of the virtual topology onto physical machine.
- If cart has less processes than starting communicator, left over processes have MPI_COMM_NULL as return

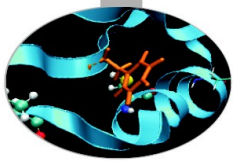
EXAMPLE (C)



```
#include <mpi.h>

int main(int argc, char *argv[])
{

  MPI_Comm cart_comm;
  int dim[] = {4, 3};
  int period[] = {1, 0};
  int reorder = 0;

  MPI_Init(&argc, &argv);

  MPI_Cart_create(MPI_COMM_WORLD, 2, dim, period, reorder, &cart_comm);
  ...
}
```



CARTESIAN TOPOLOGY UTILITIES



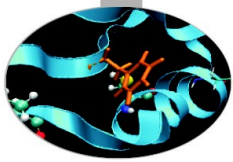
- **MPI_Dims_Create:**
  - compute optimal balanced distribution of processes per coordinate direction with respect to:
    - a given dimensionality
    - the number of processes in a group
    - optional constraints

- **MPI_Cart_coords:**
  - given a rank, returns process's coordinates

- **MPI_Cart_rank:**
  - given process's coordinates, returns the rank

- **MPI_Cart_shift:**
  - get source and destination rank ids in SendRecv operations

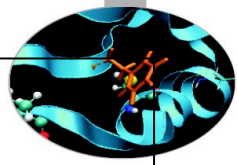
MPI DIMS CREATE



```
MPI_DIMS_CREATE(nnodes, ndims, dims)
IN nnodes: number of nodes in a grid (integer)
IN ndims: number of Cartesian dimensions (integer)
IN/OUT dims: integer array of size ndims specifying the number of
nodes in each dimension
```

- Help user to select a balanced distribution of processes per coordinate direction, depending on the number of processes in the group to be balanced and optional constraints that can be specified by the user
- if `dims[i]` is set to a positive number, the routine will not modify the number of nodes in that i dimension
- negative value of `dims[i]` are erroneous

IN/OUT OF "DIMS"



MPI_DIMS_CREATE(nnodes, ndims, dims)

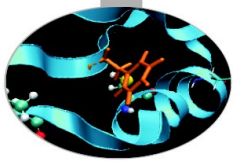
IN nnodes: number of nodes in a grid (integer)

IN ndims: number of Cartesian dimensions (integer)

IN/OUT dims: integer array of size ndims specifying the number of nodes in each dimension

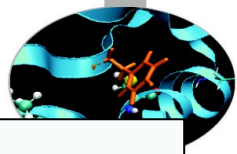
dims before call	Function call	dims on return
(0, 0)	MPI_DIMS_CREATE(6, 2, dims)	(3, 2)
(0, 0)	MPI_DIMS_CREATE(7, 2, dims)	(7, 1)
(0, 3, 0)	MPI_DIMS_CREATE(6, 3, dims)	(2, 3, 1)
(0, 3, 0)	MPI_DIMS_CREATE(7, 2, dims)	erroneous call

USING MPI_DIMS_CREATE (FORTRAN)



```
integer :: dim(3),period(3),reorder, cube_comm, ierr

CALL MPI_COMM_SIZE(MPI_COMM_WORLD, nprocs,ierr)

dim(1) = 0 ! let MPI arrange
dim(2) = 0 ! let MPI arrange
dim(3) = 3 ! I want exactly 3 planes

CALL MPI_DIMS_CREATE(nprocs, 3, dim, ierr)

if (dim(1)*dim(2)*dim(3) .LE. nprocs) then
  print *,"WARNING: some processes are not in use!"
endif

period = (1, 1, 0)
reorder = 0

CALL MPI_CART_CREATE(MPI_COMM_WORLD, 3, dim, period, reorder, &
cube_comm,ierr)
```

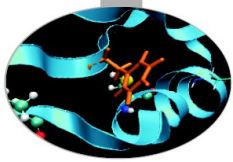
FROM COORDINATE TO RANK



```
MPI_CART_RANK(comm, coords, rank)
IN comm: communicator with Cartesian structure
IN coords: integer array (of size ndims) specifying the Cartesian
coordinates of a process
OUT rank: rank of specified process
```

- translation of the logical process coordinates to process ranks as they are used by the point-to-point routines
- if `dimension i` is periodic, when i-th coordinate is out of range, it is shifted back to the interval `0<coords(i)<dims(i)` automatically
- out-of-range coordinates are erroneous for non-periodic dimensions

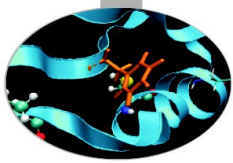
FROM RANK TO COORDINATE



```
MPI_CART_COORDS(comm, rank, maxdim, coords)
IN comm: communicator with Cartesian structure
IN rank: rank of a process within group of comm
IN maxdims: length of vector coords in the calling program
OUT coords: integer array (of size ndims) containing the Cartesain
coordinates of specified process
```

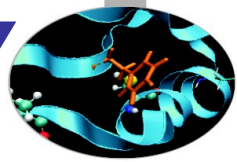

- For each MPI process in Cartesian communicator, the coordinate whitin the cartesian topology are returned

MAPPING OF COORDINATES (C)



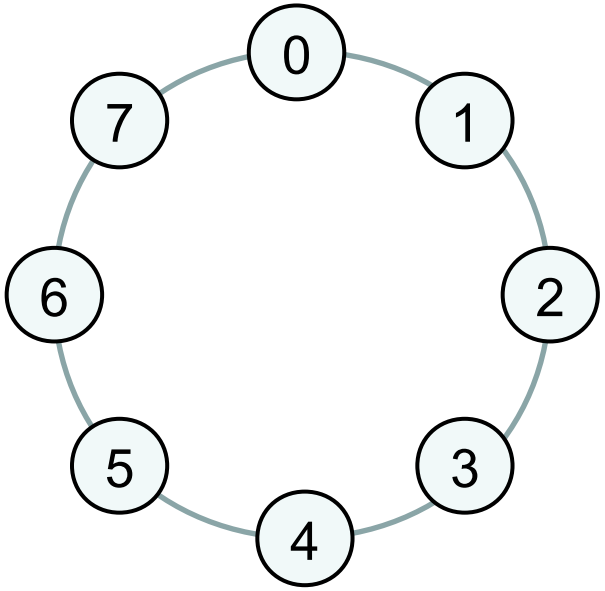
```
int cart_rank;
MPI_Comm_rank(cart_comm, &cart_rank);

int coords[2];
MPI_Cart_coords(cart_comm, cart_rank, 2, coords);

// set linear boundary values on bottom/left-hand domain
if (coords[0] == 0 || coords[1] == 0) {
  SetBoundary( linear(min, max), domain);
}

// set sinusoidal boundary values along upper domain
if (coords[0] == dim[0]) {
  SetBoundary( sinusoid(), domain);
}

// set polynomial boundary values along right-hand of domain
if (coords[1] == dim[1]) {
  SetBoundary( polynomial(order, params), domain);
}
```

CARTESIAN SHIFT: A 1D CARTESIAN TOPOLOGY

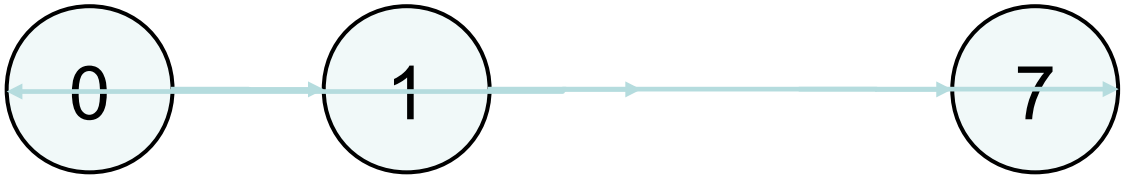


Circular shift is another tipical MPI communication pattern:

- each process communicate only with its neighbors along one direction
- periodic boundary conditions can be set for letting first and last processes partecipate in the communication



such a pattern is nothing more than a 1D cartesian grid topology with optional periodicity

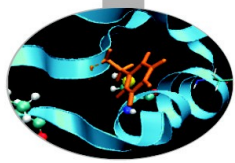


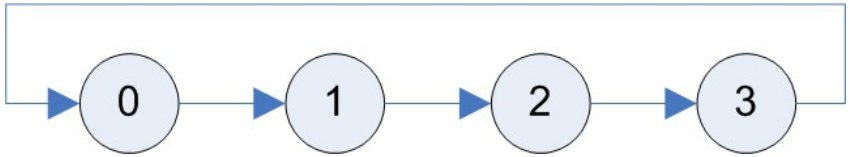
MPI CART SHIFT

```
MPI_CART_SHIFT(comm, direction, disp, rank_source, rank_dest)
IN comm: communicator with Cartesian structure
IN direction: coordinate dimension of shift
IN disp: displacement (>0: upwards shift; <0: downwards shift
OUT rank_source: rank of source process
OUT rank_dest: rank of destination process
```

- Depending on the periodicity of the Cartesian group in the specied coordinate direction, MPI_CART_SHIFT provides the identifiers for a circular or an end-o shift.
- In the case of an end-o shift, the value **MPI_PROC_NULL** may be returned in rank_source or rank_dest, indicating that the source or the destination for the shift is out of range.
- provides the calling process the ranks of source and destination processes for an MPI_SENDRECV with respect to a specified coordinate direction and step size of the shift

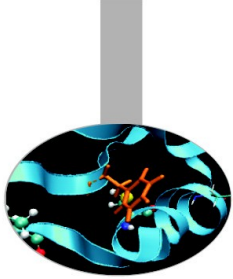
EXAMPLE (FORTRAN)






```
...

integer ::  dim = nprocs
integer ::  period = 1
integer ::  source, dest, ring_comm, status(MPI_STATUS_SIZE),ierr

CALL MPI_CART_CREATE(MPI_COMM_WORLD, 1, dim, period, 0,ring_comm,ierr)

CALL MPI_CART_SHIFT(ring_comm, 0, 1, source, dest, ierr)

CALL MPI_SENDRECV(right_bounday, n, MPI_INT, dest, rtag, left_boundary,
n, MPI_INT, source, ltag, ring_comm, status, ierr)

...
```

PARTITIONING OF CARTESIAN STRUCTURES



- It is often useful to partition a cartesian communicator into subgroups that form lower dimensional cartesian subgrids
  - new communicators are derived
  - lower dimensional communicators cannot communicate among them (unless inter-communicators are used)

MPI CART SUB

MPI_CART_SUB(comm, remain_dims, newcomm)

IN comm: communicator with Cartesian structure

IN remain_dims: the i-th entry of remain_dims specifies whether the i-th dimension is kept in the subgrid (true) or is dropped (false) (logical vector)

OUT newcomm: communicator containing the subgrid that includes the calling process

```
int dim[] = {2, 3, 4};

int remain_dims[] = {1, 0, 1}; // 3 comm with 2x4 processes 2D
grid
...
int remain_dims[] = {0, 0, 1}; // 6 comm with 4 processes 1D
topology
```