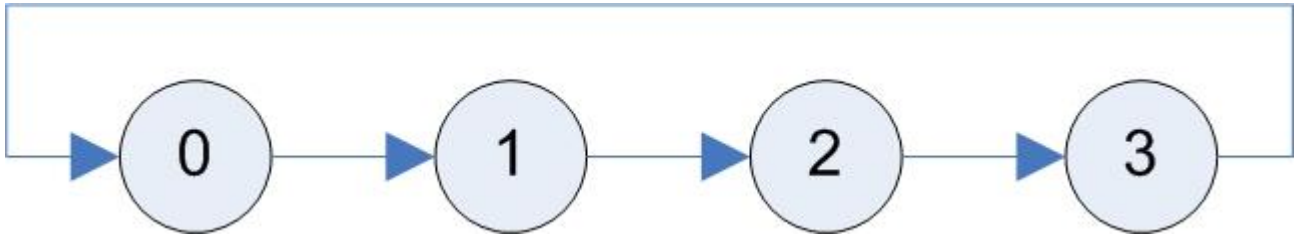
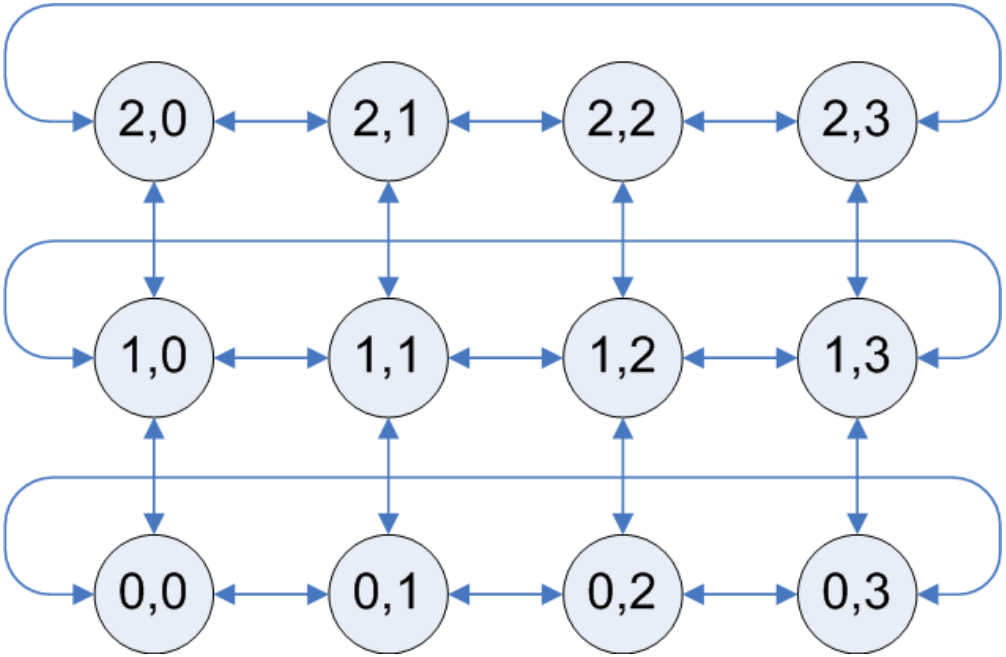
Circular shift: una topologia cartesiana 1D



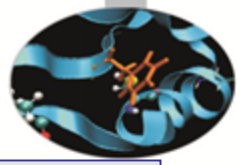
- Ogni processo comunica alla sua destra un dato
- L'ultimo processo del gruppo comunica il dato al primo

Topologia cartesiana 2D



- Ad ogni processo è associata una coppia di indici che rappresentano le sue coordinate in uno spazio cartesiano 2D
- Ad esempio, le comunicazioni possono avvenire
  - tra primi vicini *con periodicità* lungo la direzione X
  - tra primi vicini *senza periodicità* lungo la direzione Y

Creare un comunicatore con topologia cartesiana



In C

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

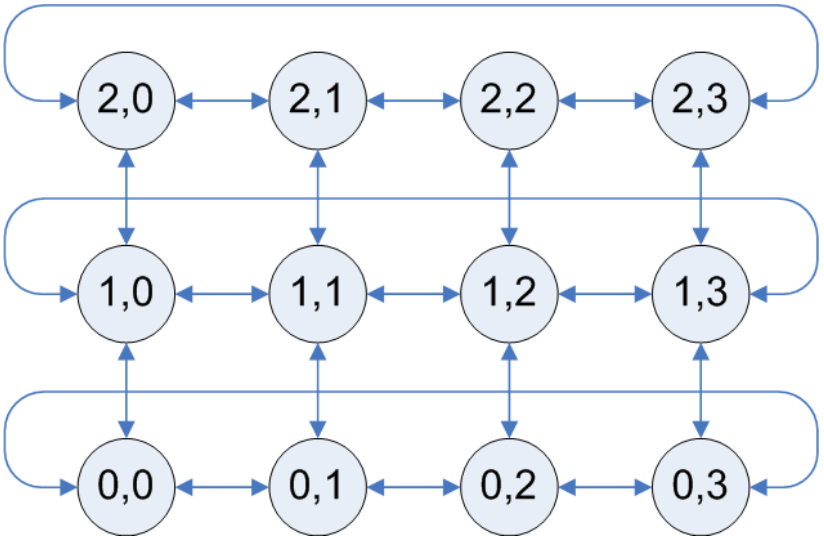
In Fortran

```
MPI_CART_CREATE(COMM_OLD, NDIMS, DIMS, PERIODS,
                REORDER, COMM_CART, IERROR)
```

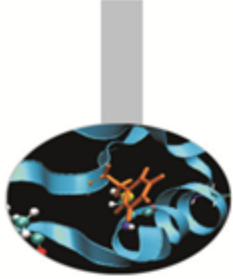
- [IN] **comm_old**: comunicatore dal quale selezionare il gruppo di processi (INTEGER)
- [IN] **ndims**: numero di dimensioni dello spazio cartesiano (INTEGER)
- [IN] **dims**: numero di processi lungo ogni direzione dello spazio cartesiano (INTEGER(*))
- [IN] **periods**: periodicità lungo le direzioni dello spazio cartesiano (LOGICAL(*))
- [IN] **reorder**: il ranking dei processi può essere riordinato per utilizzare al meglio la rete di comunicazione (LOGICAL)
- [OUT] **comm_cart**: nuovo comunicatore con l'attributo topologia cartesiana (INTEGER)

Come usare
MPI Cart create

```
int main(int argc, char **argv)
{
    ...

    int dim[2], period[2], reorder;

    ...

    dim[0]=4;
    dim[1]=3;

    period[0]=1;
    period[1]=0;
    reorder=1;

    MPI_Cart_create(MPI_COMM_WORLD,2,dim,period,reorder,&cart);

    ...

    return 0;
}
```



Alcune utili funzionalità

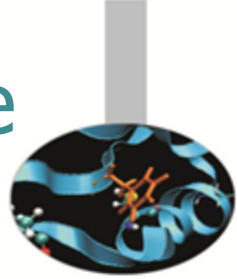


- **`MPI_Dims_Create`**:
  - Calcola le dimensioni della griglia bilanciata ottimale rispetto al numero di processi e la dimensionalità della griglia dati in input
  - Utile per calcolare un vettore `dims` di input per la funzione `MPI_Cart_Create`

- Mapping tra coordinate cartesiane e *rank*
  - **`MPI_Cart_coords`**: sulla base della topologia definita all'interno del comunicatore, ritorna le coordinate corrispondenti al processo con un fissato *rank*

  - **`MPI_Cart_rank`**: sulla base della topologia definita all'interno del comunicatore, ritorna il *rank* del processo con un fissato set di coordinate cartesiane

Binding di MPI_Dims_Create



In C

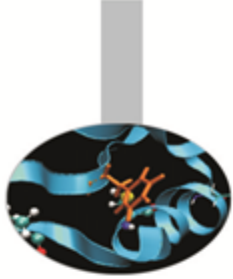
```
int MPI_Dims_create(int nnodes, int ndims, int *dims)
```

In Fortran

```
MPI_DIMS_CREATE(NNODES, NDIMS, DIMS, IERROR)
```

- [IN] **nnodes**: numero totale di processi (INTEGER)
- [IN] **ndims**: dimesionalità dello spazio cartesiano (INTEGER)
- [IN]/[OUT] **dims**: numero di processi lungo le direzioni dello spazio cartesiano (INTEGER(*))
  - Se una entry = 0 MPI_Dims_create calcola il numero di processi lungo quella direzione
  - Se una entry ≠ 0 MPI_Dims_create calcola il numero di processi lungo le direzioni "libere" compatibilmente col numero totale di processi e i valori definiti dall'utente

Rank -> Coordinate: MPI_Cart_coords



In C

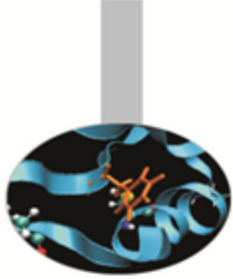
```
int MPI_Cart_coords(MPI_Comm comm, int rank,
                    int maxdims, int *coords)
```

In Fortran

```
MPI_CART_COORDS(COMM, RANK, MAXDIMS, COORDS,
                IERROR)
```

- Dato il *rank* del processo, ritorna le coordinate cartesiane associate
- Argomenti:
  - [IN] **comm**: comunicatore con topologia cartesiana (INTEGER)
  - [IN] **rank**: rank del processo del quale si vogliano conoscere le coordinate (INTEGER)
  - [IN] **maxdims**: dimensionalità dello spazio cartesiano (INTEGER)
  - [OUT] **coords**: coordinate del processo rank (INTEGER(*))

Coordinate -> Rank: MPI_Cart_rank



In C

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

In Fortran

```
MPI_CART_RANK(COMM, COORDS, RANK, IERROR)
```

- Date le coordinate cartesiane del processo, ritorna il *rank* associato
- Argomenti:
  - [IN] **comm**: comunicatore con topologia cartesiana (INTEGER)
  - [IN] **coords**: coordinate del processo (INTEGER(*))
  - [OUT] **rank**: *rank* del processo di coordinate coords (INTEGER)

Sendrecv in topologia cartesiana

In C

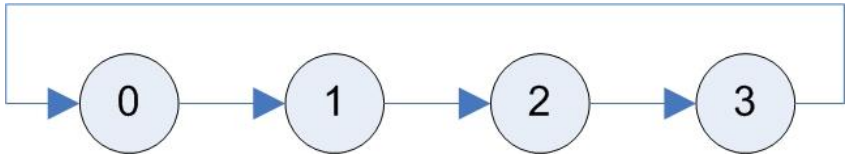
```
int MPI_Cart_shift(MPI_Comm comm, int direction,
      int disp, int *rank_source, int *rank_dest)
```

In Fortran

```
MPI_CART_SHIFT(COMM, DIRECTION, DISP, SOURCE,
      DEST, IERROR)
```

- Determina il rank del processo al/dal quale inviare/ricevere dati
- Argomenti:
  - [IN] **comm**: comunicatore con topologia cartesiana (INTEGER)
  - [IN] **direction**: indice della coordinata lungo la quale fare lo shift (INTEGER) [la numerazione degli indici parte da 0]
  - [IN] **disp**: entità dello shift (> 0: upward , < 0: downward) (INTEGER)
  - [OUT] **source**: rank del processo dal quale ricevere i dati (INTEGER)
  - [OUT] **dest**: rank del processo al quale inviare i dati (INTEGER)

Shift Circolare con topologia cartesiana 1D in C



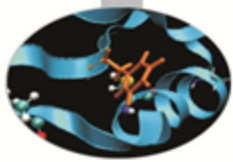
- *Circular shift* senza topologia

```
dest = (rank + 1) % size;
source = (rank + size - 1) % size;
MPI_Sendrecv(A, MSIZE, MPI_INT, dest, tag, B, MSIZE, MPI_INT, source,
                tag, MPI_COMM_WORLD, &status);
```

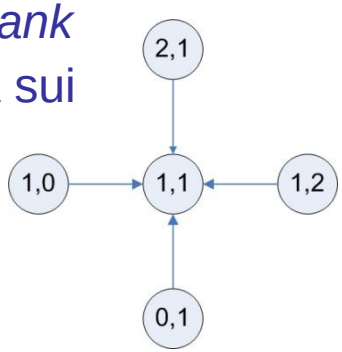
- *Circular shift* con topologia cartesiana

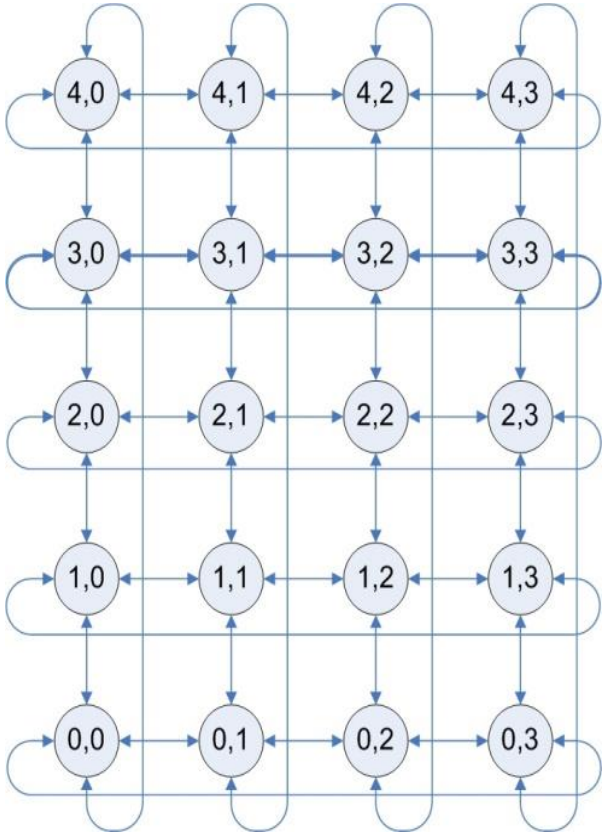
```
MPI_Cart_create(MPI_COMM_WORLD, 1, &size, periods, 0, &comm_cart);
MPI_Cart_shift(comm_cart, 0, 1, &source, &dest);
MPI_Sendrecv(A, MSIZE, MPI_INT, dest, tag, B, MSIZE, MPI_INT, source,
                tag, comm_cart, &status);
```

Media aritmetica sui primi vicini in topologia 2D



- I processi sono distribuiti secondo una griglia rettangolare
- Ogni processo:
  - Inizializza una variabile intera A con il valore del proprio *rank*
  - calcola la media di A sui primi vicini





- Il processo di rank 0:
  - Raccoglie i risultati dagli altri processi
  - Riporta in output i risultati raccolti utilizzando un formato tabella organizzato secondo le coordinate dei vari processi

Media aritm. sui primi vicini: flowchart delle comunicazioni