Message Passing Interface (MPI)

Διαχείριση Εικονικών Τοπολογιών

Εικονικές τοπολογίες (virtual topologies)

Εικονική (ή ιδεατή) τοπολογία (virtual topology): **ένας μηχανισμός αρίθμησης των διεργασιών ενός communicator έτσι ώστε η αρίθμηση αυτή να αντιστοιχεί σε μία διάταξη των διεργασιών η οποία να είναι περισσότερο συμβατή με την διάταξη επικοινωνίας (communication pattern) των διεργασιών.**

Κύριος σκοπός δημιουργίας εικονικών τοπολογιών: **η απλούστευση του παράλληλου κώδικα, όταν ειδικότερα ο αλγόριθμος ο οποίος υλοποιείται έχει αναπτυχθεί για αντίστοιχες τοπολογίες.**

Εικονικές τοπολογίες (virtual topologies)

Η δημιουργία εικονικής τοπολογίας:

- **δίνει στον προγραμματιστή τη δυνατότητα χρήσης κατάλληλων συναρτήσεων, όπως π.χ. μία συνάρτηση υπολογισμού των τάξεων των κοντινότερων διεργασιών μιας διεργασίας σε μία τοπολογία πλέγματος.**

  Οι τάξεις αυτές μπορούν στη συνέχεια, να χρησιμοποιηθούν ως παράμετροι, σε κλήσεις συναρτήσεων αποστολής ή/και παραλαβής δεδομένων.

- **δεν εμποδίζει όλες τις διεργασίες του communicator να επικοινωνούν μεταξύ τους.**

Εικονικές τοπολογίες (virtual topologies)

Το MPI παρέχει συναρτήσεις για τη δημιουργία/διαχείριση δύο ειδών εικονικών τοπολογιών:

- **Τις καρτεσιανές εικονικές τοπολογίες (cartesian virtual topologies): κάθε διεργασία είναι «συνδεδεμένη» με τους γείτονές της σε μία διάταξη εικονικού πλέγματος (με ή χωρίς εξωτερικές συνδέσεις).**

  Η τάξη κάθε διεργασίας εξαρτάται από τη θέση της στο πλέγμα και υπάρχει η έννοια των συντεταγμένων μιας διεργασίας.

- **τις εικονικές τοπολογίες γραφημάτων (graph virtual topologies).**

Συναρτήσεις δημιουργίας/διαχείρισης εικονικών τοπολογιών

- MPI_CART_CREATE
- MPI_CART_COORDS
- MPI_CART_RANK
- MPI_CART_SUB
- MPI_CARTDIM_GET
- MPI_CART_GET
- MPI_CART_SHIFT

Η συνάρτηση MPI_Cart_create

int MPI_Cart_create (MPI_Comm old_comm, int ndims, int *dims, int *periods, int reorder, MPI_Comm *cart_comm)

- **Η συνάρτηση παίρνει ως παράμετρο έναν υπάρχοντα communicator old_comm και επιστρέφει ένα νέο communicator cart_comm.**
- **Η βασική διάταξη επικοινωνίας μεταξύ των διεργασιών του cart_comm αντιστοιχεί στη δομή μιας καρτεσιανής τοπολογίας.**
- **ndims: περιέχει τον αριθμό των διαστάσεων της τοπολογίας.**
- **dims : πίνακας που περιέχει τον αριθμό των διεργασιών /διάσταση**
- **periods : πίνακας που περιέχει επίσης, ένα στοιχείο για κάθε διάσταση που δηλώνει αν είναι περιοδική ή όχι. 'Ενα στοιχείο του periods είναι TRUE αν η αντίστοιχη διάσταση έχει εξωτερική ακμή (π.χ. αν η αντίστοιχη στήλη ή γραμμή σχηματίζει δακτύλιο).**

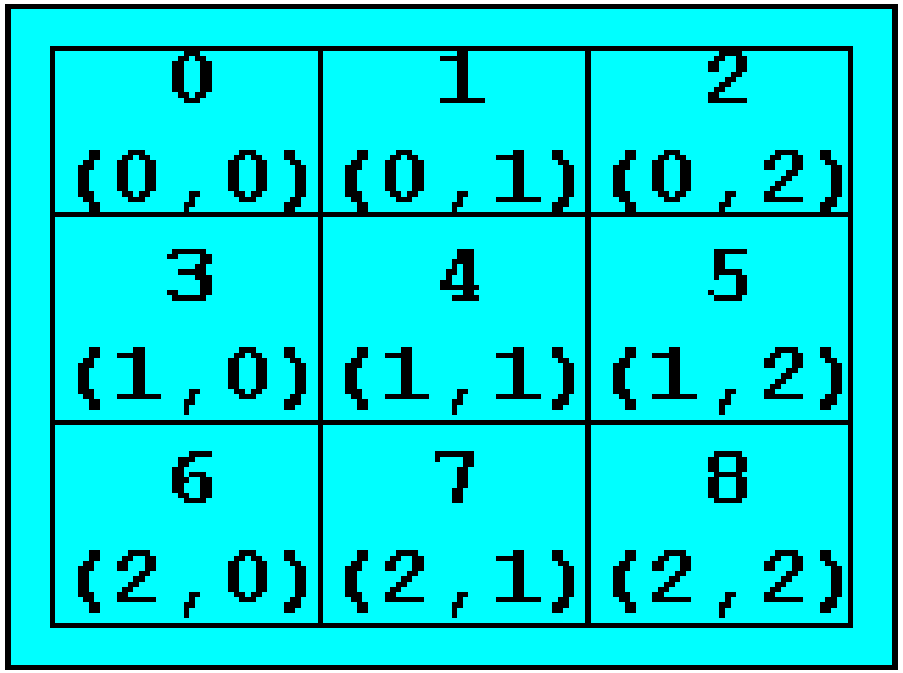
Η συνάρτηση MPI_Cart_create

- **Reorder:** **παίρνει την τιμή:**

  0 αν τα δεδομένα έχουν ήδη κατανεμηθεί στις διεργασίες. Τότε, η τάξη κάθε διεργασίας στον νέο communicator παραμένει η ίδια με αυτήν του παλαιού.

  1 αν τα δεδομένα δεν έχουν ακόμη κατανεμηθεί στις διεργασίες. Τότε, η τάξη κάθε διεργασίας στον νέο communicator μπορεί να είναι διαφορετική από αυτήν του παλαιού. Ο νέος communicator αναλαμβάνει να διαμοιράσει τα δεδομένα στις διεργασίες.

Η συνάρτηση MPI_Cart_create



Η συνάρτηση MPI_Cart_create

```
#include "stdio.h"
#include "mpi.h"
MPI_Comm old_comm, new_comm;
int ndims, reorder, periods[2], dim_size[2];

old_comm = MPI_COMM_WORLD;
ndims = 2;       /*  2D matrix/grid */
dim_size[0] = 3;          /* rows */
dim_size[1] = 2;          /* columns */
periods[0] = 1;/* row periodic (each column forms a ring) */
periods[1] = 0;/* columns non-periodic */
reorder = 1;     /* allows processes reordered for efficiency */

MPI_Cart_create(old_comm, ndims, dim_size, periods, reorder,
    &new_comm);
```

Η συνάρτηση MPI_Cart_create

- **Στο παράδειγμα η MPI_CART_CREATE αντιστοιχεί (και μετονομάζει) 6 διεργασίες (0,1,2,3,4,5) σε ένα δισδιάστατο πίνακα 3x2 διεργασιών, όπως φαίνεται στο Σχήμα.**

- **Οι αριθμοί στις παρενθέσεις είναι τα ranks των διεργασιών στην αρχική γραμμική αρίθμησή τους.**

0,0 (0)	0,1 (1)
1,0 (2)	1,1 (3)
2,0 (4)	2,1 (5)

Η συνάρτηση MPI_Cart_create

Αν ορίσουμε περιοδικότητα στις γραμμές (periods[0]=1), τότε οποιαδήποτε αναφορά πριν το πρώτο ή μετά το τελευταίο στοιχείο μιας στήλης θα ερμηνεύεται ως εξής:

- ***i = -1* αντιστοιχεί στο *i = 2*.**
- ***i = -2* αντιστοιχεί στο *i = 1*.**
- ***i = 3* αντιστοιχεί στο *i = 0*.**

	-1,0 (4)	*-1,1 (5)*	
0,-1(-1)	*0,0 (0)*	*0,1 (1)*	*0,2(-1)*
1,-1(-1)	*1,0 (2)*	*1,1 (3)*	*1,2(-1)*
2,-1(-1)	*2,0 (4)*	*2,1 (5)*	*2,2(-1)*
	3,0 (0)	*3,1 (1)*	

periods[0]=1;periods[1]=0

Αν ορίσουμε περιοδικότητα στις στήλες (periods[1]=1), τότε οποιαδήποτε αναφορά πριν το πρώτο ή μετά το τελευταίο στοιχείο μιας στήλης θα ερμηνεύεται ως εξής:

- ***j= -1* αντιστοιχεί στο *j= 1*.**
- ***j= 2* αντιστοιχεί στο *j= 0*.**

	-1,0 (-1)	*-1,1 (-1)*	
0,-1(1)	*0,0 (0)*	*0,1 (1)*	*0,2(0)*
1,-1(3)	*1,0 (2)*	*1,1 (3)*	*1,2(2)*
2,-1(5)	*2,0 (4)*	*2,1 (5)*	*2,2(4)*
	3,0 (-1)	*3,1 (-1)*	

periods[0]=0;periods[1]=1

Η συνάρτηση MPI_Cart_create

- **Η MPI_Cart_create είναι μία συνάρτηση συλλογικής επικοινωνίας. Πρέπει να καλείται από όλες τις διεργασίες της ομάδας που αντιστοιχεί στον communicator.**
- **Όπως όλες οι συναρτήσεις συλλογικής επικοινωνίας, χρησιμοποιεί αναστέλλουσα επικοινωνία. Ωστόσο, μπορεί να υπάρξει ή μπορεί να μην υπάρξει συγχρονισμός μεταξύ των διεργασιών. Η συνάρτηση MPI_Topo_test μπορεί να χρησιμοποιηθεί για να ελεγχθεί εάν μία εικονική τοπολογία έχει συσχετισθεί με έναν communicator.**
- **Αν το μέγεθος του πλέγματος είναι μικρότερο από τον αριθμό των διεργασιών, οι διεργασίες που δεν περιλαμβάνονται στο νέο communicator, επιστρέφουν MPI_COMM_NULL.**
- **Αν το μέγεθος του πλέγματος είναι μεγαλύτερο από τον αριθμό των διεργασιών, τότε η κλήση επιστρέφει λάθος.**

Η συνάρτηση MPI_Cart_coords

- **Η συνάρτηση MPI_Cart_coords επιστρέφει τις καρτεσιανές συντεταγμένες μιας διεργασίας της οποίας η τάξη είναι γνωστή.**
- **Η συνάρτηση αυτή μπορεί π.χ., να χρησιμοποιηθεί για τον προσδιορισμό των καρτεσιανών συντεταγμένων μιας συγκεκριμένης διεργασίας από την οποία μόλις παρελήφθη ένα μήνυμα.**

```
int MPI_Cart_coords(MPI_Comm cart_comm, int rank,
                    int ndims, int *coords)
```

- **Η ndims κρατάει τον αριθμό των διαστάσεων της καρτεσιανής εικονικής τοπολογίας.**

Η συνάρτηση MPI_Cart_coords

```
MPI_Cart_create(old_comm, ndims, dim_size, periods,
   reorder, &new_comm);  /* creates communicator */

if(myrank == root) {   /* only root process */
   for (rank=0; rank<p; rank++) {
        MPI_Cart_coords(new_comm, rank, ndims,&coords);
        printf("%d, %d, %d\n ",rank, coords[0], coord[1]);
   }
}
```

Η συνάρτηση MPI_Cart_rank

Η συνάρτηση MPI_Cart_rank επιστρέφει την τάξη μιας διεργασίας της οποίας οι συντεταγμένες στην ιδεατή καρτεσιανή τοπολογία είναι γνωστές, προκειμένου για παράδειγμα, να χρησιμοποιηθεί ως παράμετρος σε μία συνάρτηση αποστολής ή παραλαβής μηνύματος.

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

Η συνάρτηση MPI_Cart_rank

```
MPI_Cart_create(old_comm, ndims, dim_size, periods,
    reorder, &new_comm);

if(myrank == root) {      /* root process */
    for (i=0; i<nrow; i++) {
        for (j=0; j<mcol; j++)  {
                coords[0] = i;
                coords[1] = j;
                MPI_Cart_rank(new_comm, coords, &rank);
               printf("%d, %d, %d\n", coords[0],coords[1],rank);
        }
```

Παράδειγμα: torus_create.c

```
include <mpi.h>
#include <stdio.h>
 /* A two-dimensional torus of 12 processes in a 4x3 grid */
int main(int argc, char *argv[])
{
   int rank, size;
   MPI_Comm comm;
   int dim[2], period[2], reorder;
   int coord[2], id;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   if (size != 12)
   {
      printf("Please run with 12 processes.\n");
      MPI_Abort(MPI_COMM_WORLD, 1);
   }
```

```
dim[0]=4; dim[1]=3;
   period[0]=1; period[1]=0;
   reorder=1;
   MPI_Cart_create(MPI_COMM_WORLD, 2, dim, period, reorder,
    &comm);
   if (rank == 5)
   {
      MPI_Cart_coords(comm, rank, 2, coord);
      printf("Rank %d coordinates are %d %d\n", rank, coord[0],
    coord[1]);
   }
   if(rank==0)   {
      coord[0]= -2; coord[1]=0;
      MPI_Cart_rank(comm, coord, &id);
      printf("The processor at position (%d, %d) has rank %d\n",
    coord[0], coord[1], id);      }
   MPI_Finalize(); }
```

Παράδειγμα: cart_create.c

```
#include "mpi.h"
#include <stdio.h>

void main(int argc, char *argv[])
{
int nrow, mcol, root, me, ndim, p, rank;
int dims[2], coords[2], cyclic[2], reorder;
MPI_Comm comm2D, comm2Dp;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &me);
MPI_Comm_size(MPI_COMM_WORLD, &p);
nrow = 3; mcol = 2; ndim = 2;
root = 0; cyclic[0] = 0; cyclic[1] = 0; reorder = 1;
```

```
if(me == root) {
   printf("\n");
   printf("There are six (6) processes\n");
   printf("use all 6 to create 3x2 cartesian topology\n");
   printf(" Cart. Coords.    Cart\n");
   printf("    i       j         rank      me\n");
}
MPI_Barrier(MPI_COMM_WORLD);
```

```
/* create 3x2 cartesian topology for processes */
dims[0] = nrow;
dims[1] = mcol;
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims, cyclic, reorder,
    &comm2D);
if (comm2D != MPI_COMM_NULL) {
        MPI_Cart_coords(comm2D, me, ndim, coords);
        MPI_Cart_rank(comm2D, coords, &rank);
        printf("%8d %8d %8d %8d\n",coords[0], coords[1], rank, me);
}
MPI_Barrier(MPI_COMM_WORLD);
/* create 2x2 cartesian topology for processes */
```

```
if(me == root) {
        printf("\n");
        printf("There are six (6) processes\n");
        printf("use 4 to create 2x2 cartesian topology\n");
        printf(" Cart. Coords.    Cart\n");
        printf("    i       j        rank     me\n");
}
MPI_Barrier(MPI_COMM_WORLD);
dims[0] = 2; /* rows */
 dims[1] = 2; /* columns */
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims, cyclic, reorder, &comm2Dp);

if (comm2Dp != MPI_COMM_NULL) {
        MPI_Cart_coords(comm2Dp, me, ndim, coords);
        MPI_Cart_rank(comm2Dp, coords, &rank);
        printf("%8d %8d %8d %8d\n",coords[0], coords[1], rank, me);
}
MPI_Finalize();  }
```

Η συνάρτηση MPI_Cart_sub

Δημιουργεί communicators με δομή υποπλέγματος N-1 διαστάσεων το πολύ, από ένα πλέγμα N διαστάσεων.

int MPI_Cart_sub (MPI_Comm cart_comm, int *belongs, MPI_Comm new_cart_comm)

belongs : πίνακας ndims διαστάσεων που προσδιορίζει αν κάθε μία διάσταση του cart_comm ανήκει στον νέο communicator new_cart_comm.

Παράδειγμα: αν ο cart_comm ορίζει ένα 4×2×5 πλέγμα, και η παράμετρος belongs ορίζεται ίση με (TRUE, FALSE, TRUE) τότε, η MPI_Cart_sub θα δημιουργήσει δύο communicators καθένας με 20 διεργασίες συνδεδεμένες σύμφωνα με τη δομή ενός 4×5 πλέγματος.

MPI_CART_SUB

```
/* Create 2D Cartesian topology for processes */
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims, period,
   reorder, &comm2D);
MPI_Comm_rank(comm2D, &id2D);
MPI_Cart_coords(comm2D, id2D, ndim, coords2D);

/* Create 1D row subgrids */
belongs[0] = 0;
belongs[1] = 1;
MPI_Cart_sub(comm2D, belongs, &commrow);

/* Create 1D column subgrids */
belongs[0] = 1;
belongs[1] = 0;
MPI_Cart_sub(comm2D, belongs, &commcol);
```

MPI_CART_SUB

0,0(0)	0,1(1)
1,0(2)	1,1(3)
2,0(4)	2,1(5)

2D Cartesian Grid

0,0(0) 0(0)	0,1(1) 1(1)
1,0(2) 0(0)	1,1(3) 1(1)
2,0(4) 0(0)	2,1(5) 1(1)

belongs[0] = 0;
belongs[1] = 1;

3 υποπλέγματα γραμμής (Row Subgrids)

0,0(0) 0(0)	0,1(1) 0(0)
1,0(2) 1(1)	1,1(3) 1(1)
2,0(4) 2(2)	2,1(5) 2(2)

belongs[0] = 1;
belongs[1] = 0;

2 υποπλέγματα στήλης (column Subgrids)

MPI_CART_SUB

- MPI_CART_SUB is a collective routine. It must be called by all processes in old_comm.
- MPI_CART_SUB generated subgrid communicators are derived from Cartesian grid created with MPI_CART_CREATE.
- Full length of each dimension of the original Cartesian grid is used in the subgrids.
- Each subgrid has a corresponding communicator. It inherits properties of the parent Cartesian grid; it remains a Cartesian grid.
- It returns the communicator to which the calling process belongs.
- There is a comparable MPI_COMM_SPLIT to perform similar function.
- MPI_CARTDIM_GET and MPI_CART_GET can be used to acquire structural information of a grid (such as dimension, size, periodicity)

Παράδειγμα: cart_sub.c

- **Δημιουργούμε μία 2D (3x2) ιδεατή καρτεσιανή τοπολογία (πλέγμα). Κάθε στοιχείο αυτής της τοπολογίας αντιστοιχεί σε μία τιμή (στοιχείο) A(i,j) ενός πίνακα A. Το στοιχείο A(i,j) ορίζεται ως**

  A(i,j) = (i+1)*10 + j + 1; i=0,1,2; j=0,1

  A(0,0) = 11 , A(0,1) = 12,

  A(1,0) = 21, A(1,1) = 22,

  A(2,0) = 31, A(2,1) = 32.

- **Δημιουργούμε 2 υποπλέγματα στήλης με την MPI_CART_SUB. Κάθε υποπλέγμα είναι ένα 3x1 διάνυσμα. Στη συνέχεια, το τελευταίο στοιχείο κάθε υποπλέγματος (στήλης) συγκεντρώνει τις τιμές (δηλ. τα A(i,j)) των υπολοίπων στοιχείων του υποπλέγματος .**

Παράδειγμα: MPI_CART_SUB

```
#include "stdio.h"
#include "mpi.h"
void main(int argc, char *argv[])
{
    int nrow, mcol, lastrow, p, root;
    int i, me, id2D, colID, ndim;
    int coords1D[2], coords2D[2], dims[2], aij[1], alocal[3];
    int belongs[2], periods[2], reorder;
    MPI_Comm comm2D, commcol;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &me);
    MPI_Comm_size(MPI_COMM_WORLD, &p);
```

Παράδειγμα: MPI_CART_SUB

```
nrow = 3; mcol = 2; ndim = 2;
root = 0; periods[0] = 1; periods[1] = 0; reorder = 1;

/* create cartesian topology for processes */
dims[0] = nrow;                 /* number of rows */
dims[1] = mcol;                 /* number of columns */
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims, periods, reorder,
&comm2D);
MPI_Comm_rank(comm2D, &id2D);
MPI_Cart_coords(comm2D, id2D, ndim, coords2D);

/* Create 1D column subgrids */
belongs[0] = 1;                 /* this dimension belongs to subgrid */
belongs[1] = 0;
MPI_Cart_sub(comm2D, belongs, &commcol);
MPI_Comm_rank(commcol, &colID);
MPI_Cart_coords(commcol, colID, 1, coords1D);
```

Παράδειγμα: MPI_CART_SUB

```
MPI_Barrier(MPI_COMM_WORLD);

/* aij = (i+1)*10 + j + 1; one matrix element to each process */
aij[0] = (coords2D[0]+1)*10 + coords2D[1]+1;

if(me == root) {

        printf("\n 3x2 cartesian grid ==> 2 (3x1) column subgrids\n");
        printf("\n   Iam    2D      2D         1D      1D     aij");
        printf("\n  Rank   Rank    coords.    Rank  coords.\n");
}

/*Last element of each column gathers elements of its own column*/
    for ( i=0; i<=nrow-1; i++) {
        alocal[i] = -1;
    }
```

Παράδειγμα: MPI_CART_SUB

```
    lastrow = nrow - 1;
   MPI_Gather(aij, 1, MPI_INT, alocal, 1, MPI_INT, lastrow, commcol);

    MPI_Barrier(MPI_COMM_WORLD);

    printf("%6d|%6d|%6d %6d|%6d|%8d|",
    me,id2D,coords2D[0],coords2D[1],colID,coords1D[0]);
    for (i=0; i<=lastrow; i++) {
          printf("%6d ",alocal[i]);
    }
    printf("\n");

    MPI_Finalize();
}
```

Παράδειγμα: MPI_CART_SUB

ΕΞΟΔΟΣ

> mpirun -np 6

3x2 cartesian grid ==> 2 (3x1) column subgrids

Iam Rank	2D Rank	2D coords.		1D Rank	1D coords.			aij
0|	0|	0	0|	0|	0|	-1	-1	-1
1|	1|	0	1|	0|	0|	-1	-1	-1
2|	2|	1	0|	1|	1|	-1	-1	-1
3|	3|	1	1|	1|	1|	-1	-1	-1
4|	4|	2	0|	2|	2|	11	21	31
5|	5|	2	1|	2|	2|	12	22	32