# Виртуальные топологии

НОЦ МФТИ 2013

# Введение

- Большое количество инженерных и научных проблем в конечном счете сводится к работе с матрицами или сетками
- Обычно для таких операций используется доменная декомпозиция расчетной области, в таком случае подматрица или часть сетки обсчитывается одним процессом
- При разделении матрицы на подматрицы удобно использовать двумерную нумерацию процессов. При работе с сетками нумерация определяется размерностью задачи
- Для адресации в таких случаях необходима виртуальная топология. MPI предоставляет два вида топологий: **декартова** и **топология на графак**

# Топологии в MPI

- Удобство
  - Удобны при использовании определенных шаблонов взаимодействия процессов
  - Например декартова топология удобна когда у процесса есть 4 соседа
- Эффективность
  - Некоторое железо дает увеличение производительности если коммуникации построены правильно
  - Можно распределить процессы физически на реальную топологию железа
  - Маппинг процессов зависит от реализации MPI и может отсутствовать

# Функции в MPI

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

# MPI_Cart_create

- Используется для задания декартовых координат процессам произвольной размерности пространства
- Позволяет построить нумерацию процессов по координатам в сетке, а не по их номеру
- Позволяет делать циклические условия на границах
- В ряде случаев позволяет стоить топологию на основе реального железа

# Синтаксис

```
int MPI_Cart_create(MPI_Comm old_comm, int ndims, int
  *dim_size, int *periods, int reorder, MPI_Comm
  *new_comm)
```

- old_comm — старый коммуникатор
- ndims — размерность нового пространства
- dim_size — массив размерности ndims, указывающий размеры по каждому направлению
- periods — массив, указывающий периодичность границ по каждому направлению
- reorder — разрешать или нет MPI изменять порядок процессов
- new_comm — новый коммуникатор

# Пример

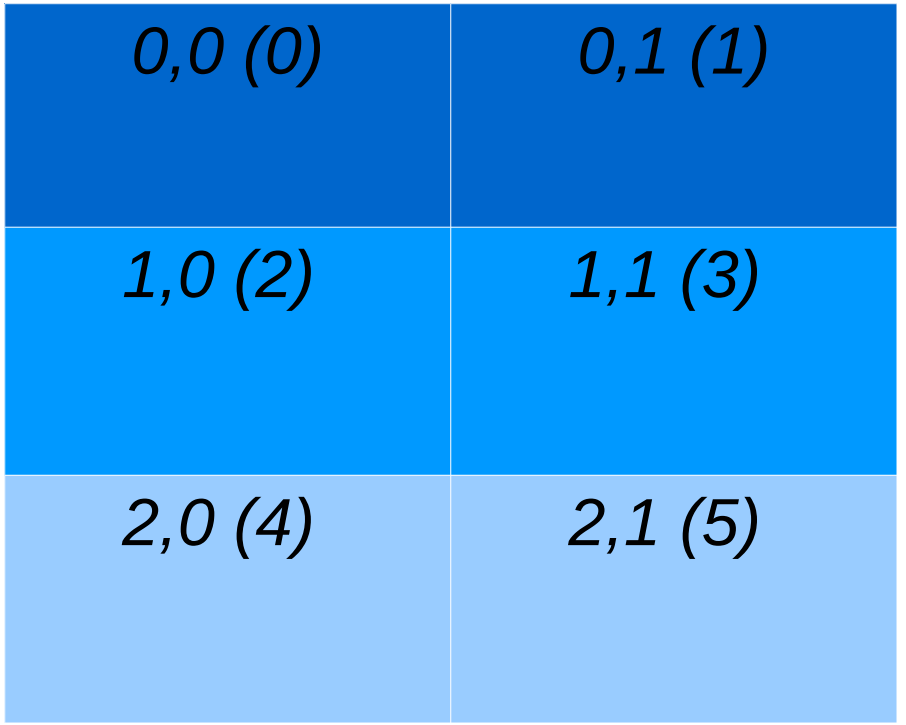
```
#include "stdio.h"
#include "mpi.h"
MPI_Comm old_comm, new_comm;
int ndims, reorder, periods[2], dim_size[2];

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);
```

# Пример

- В предыдущем примере 6 процессов перенумерованы из линейной нумерации в двумерную матрицу размером 3x2
- На рисунке показана новая нумерация. Каждый процесс имеет теперь два номера i,j, i — номер строки, j — номер столбца




Декартова сетка

# Периодичность

- Можно задавать цилиндрические граничные условия по каждому направлению
- В данном примере показано как делать периодичность по каждому направлению

| | | | |
|---|---|---|---|
| | *-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

| | | | |
|---|---|---|---|
| | *-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

# Особенности

- Коллективная операция, должна быть вызвана у всех в коммуникаторе
- Если reorder «1», то MPI может изменять порядок процессов с целью маппинга на реальное железо
- Если reorder «0», то ранк в новом коммуникаторе остается таким же
- В новом коммуникаторе существует помимо ранков еще и топология
- Если число процессов в старом коммуникаторе больше чем необходимо для создания топологии, на части процессов будет возвращено MPI_COMM_NULL
- Если число процессов меньше, то будет возвращена ошибка

# MPI_Cart_coords

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

- Служит для получения координат процесса по его ранку
- comm - коммуникатор
- rank — ранк процессора, координаты которого узнаем
- maxdirs — размерность декартовой топологии
- coords — массив с координатами

# Пример

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


if(Iam == root) {   /* only want to do this on one process */
    for (rank=0; rank<p; rank++) {
        MPI_Cart_coords(new_comm, rank, ndims,
    &coords);
        printf("%d, %d\n ",rank, coords);
    }
}
```

# MPI_Cart_rank

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

- Используется для получения ранка процесса по его координатам
- Обратная операция MPI_Cart_coords
- comm - коммуникатор
- coords — массив с координатами
- rank — номер процесса

# Пример

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

if(Iam == root) {       /* only want to do this on one process */
    for (i=0; i<nv; i++) {
        for (j=0; j<mv; j++)  {
            coords[0] = i;
            coords[1] = j;
            MPI_Cart_rank(new_comm, coords, &rank);
            printf("%d, %d, %d\n",coords[0],coords[1],rank);
        }
    }
}
```

# MPI_Cart_sub

- Используется для разделения процессов в коммуникаторе на подгруппы, когда коммуникатор был создан используя MPI_Cart_create
- Создает новые коммуникаторы декартовой топологии размерностью вплоть до N-1, где N — размерность старого коммуникатора
- Часто используется для деления пространства на меньшие, например в случае матрицы создать коммуникаторы для всех строк и столбцов

# Синтаксис

```
int MPI_Cart_sub( MPI_Comm old_comm, int *belongs,
    MPI_Comm *new_comm )
```

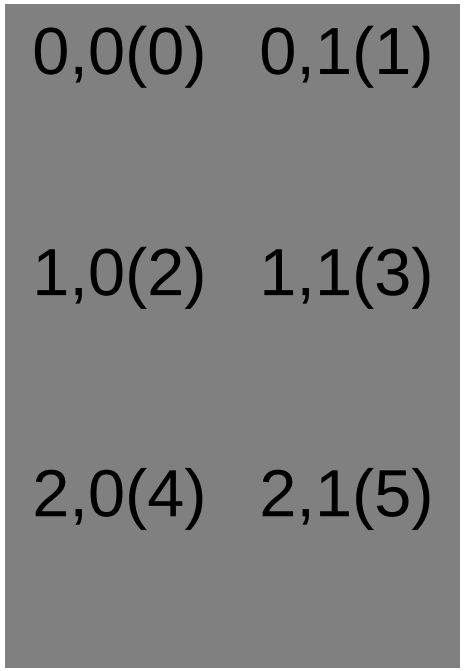
- old_comm -  старый коммуникатор
- belongs — массив размера ndims, в котором содержатся номера какие подпространства принадлежат новому комуникатору
- new_comm — новый коммуникатор

# Пример

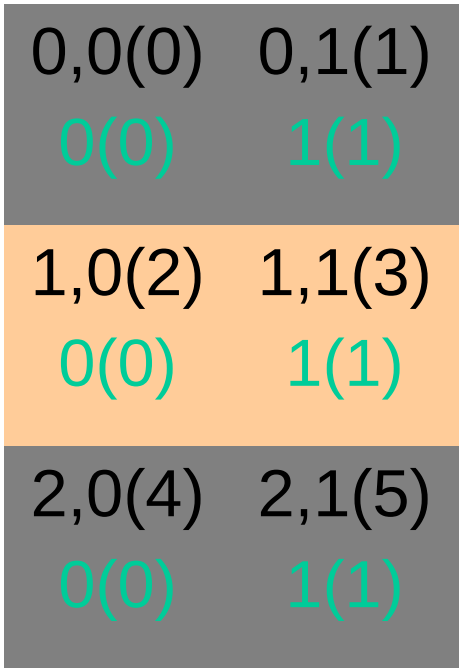
- Создание коммуникаторов для колонок и строк

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

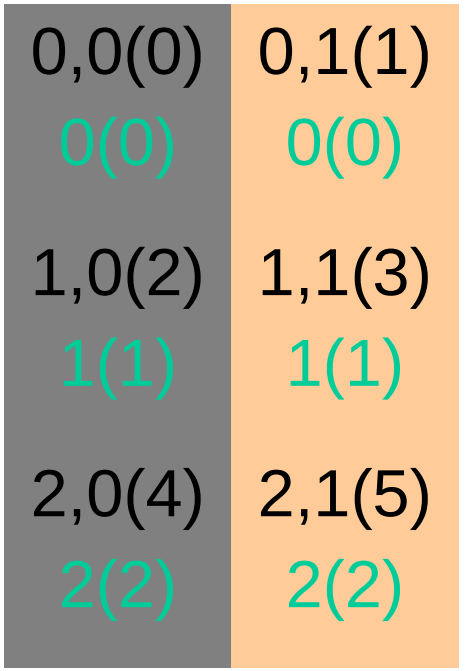
# Результат работы




Двумерная декартова сетка




Строки




Колонки

# Особенности

- Коллективная операция, должны вызывать все процессы старого коммуникатора
- Для создания нового коммуникатора используются все процессы из указанного подпространства
- Коммуникатор каждого подпространства имеет топологию декартовой сетки
- У каждого процесса возвращает коммуникатор, которому он принадлежит
- Для получения информации о новом коммуникаторе могут быть использованы функции MPI_Cartdim_get MPI_Cart_get

# MPI_Cartdim_get

int MPI_Cartdim_get( MPI_Comm comm, int* ndims )

- Используется для получения размерности пространства коммуникатора
- comm - коммуникатор
- ndims — сколько пространств имеет коммуникатор

```
/* create column subgrids */
belongs[0] = 1;
belongs[1] = 0;
MPI_Cart_sub(grid_comm, belongs, &col_comm);
/* queries number of dimensions of cartesan grid */
MPI_Cartdim_get(col_comm, &ndims);
```

# MPI_Cart_get

- Используется для получения размеров декартовой топологии по всем измерениям
- Обычно вызывается после MPI_Cartdim_get
- Получает информацию о периодичности и размере нового подпространства

```
int MPI_Cart_get( MPI_Comm subgrid_comm, int ndims, int *dims, int
    *periods, int *coords )
```

# Параметры

- subgrid_comm – коммуникатор, параметры которого узнаем
- ndims – размерность коммуникатора, полученная MPI_Cartdim_get
- dims – массив размера ndims, куда будут записаны размеры по всем измерениям
- periods – массив размера ndims, куда будут записаны периодичности всех измерений
- coords – массив размера ndims, куда будут записаны координаты текущего процесса

# Пример

```
/* create Cartesian topology for processes */
dims[0] = nrow;
dims[1] = mcol;
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims, period, reorder,
    &grid_comm);
MPI_Comm_rank(grid_comm, &me);
MPI_Cart_coords(grid_comm, me, ndim, coords);
/* create row subgrids */
belongs[0] = 1;
belongs[1] = 0;
MPI_Cart_sub(grid_comm, belongs, &row_comm);
/* Retrieve subgrid dimensions and other info */
MPI_Cartdim_get(row_comm, &mdims);
MPI_Cart_get(row_comm, mdims, dims, period, row_coords);
```

# MPI_Cart_shift

- Возвращает номера процессов в коммуникаторе, сдвинутых относительно данного в одном направлении декартовой сетки на один и тот же отступ
- Используется для получения ранков соседних процессов к данному в декартовой сетке
- Если displ > 0, то source процесс с меньшим ранком, если displ < 0, то dest процесс с меньшим ранком

```
int MPI_Cart_shift( MPI_Comm comm, int direction,
int displ, int *source, int *dest )
```

# Параметры

- comm - коммуникатор
- direction — измерение, вдоль которого ищем соседей
- displ — направление и величина сдвига, может быть >0, <0 или 0
- source — номер процесса для приема
- dest — номер процесса для отправки

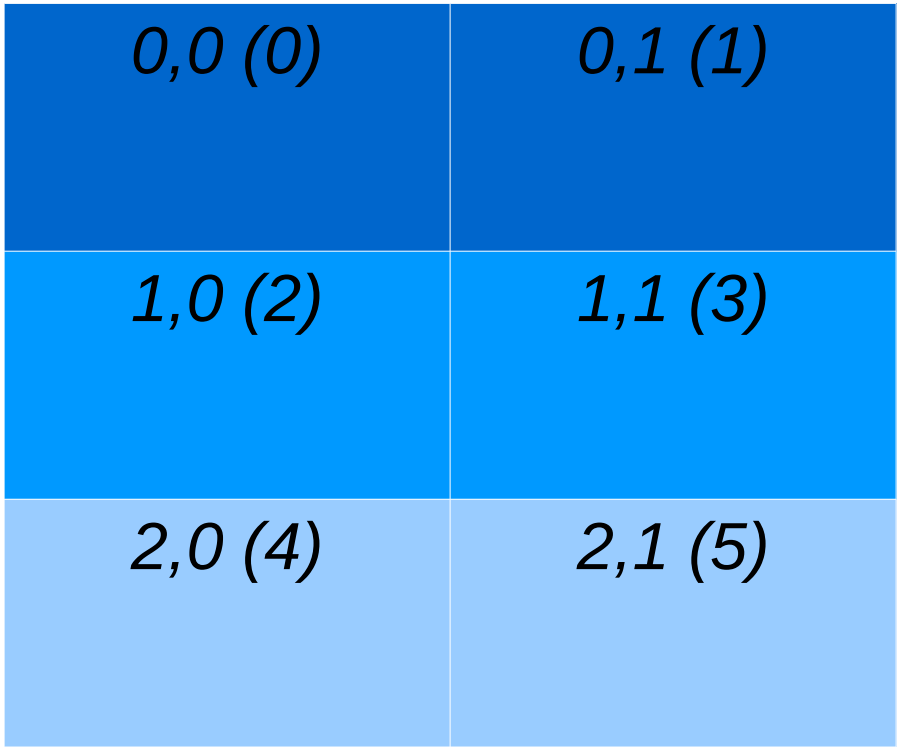
# Пример

```
/* create Cartesian topology for processes */
dims[0] = nrow;     /* number of rows     */
dims[1] = mcol;     /* number of columns  */
period[0] = 1;      /* cyclic in this direction */
period[1] = 0;      /* no cyclic in this direction */
MPI_Cart_create(MPI_COMM_WORLD, ndim, dims,
period, reorder, &comm2D);
MPI_Comm_rank(comm2D, &me);
MPI_Cart_coords(comm2D, me, ndim, coords);

index =  0;    /* shift along the 1st index (out of 2) */
displ =  1;    /* shift by  1 */
MPI_Cart_shift(comm2D, index, displ, &source, &dest1);
```

# Пример

- В данном примере если вызывает 2 процесс, то source и dest будут 0 и 4 соответственно
- Если вызывает 5, то source 1 и dest 3 из-за периодичности границ



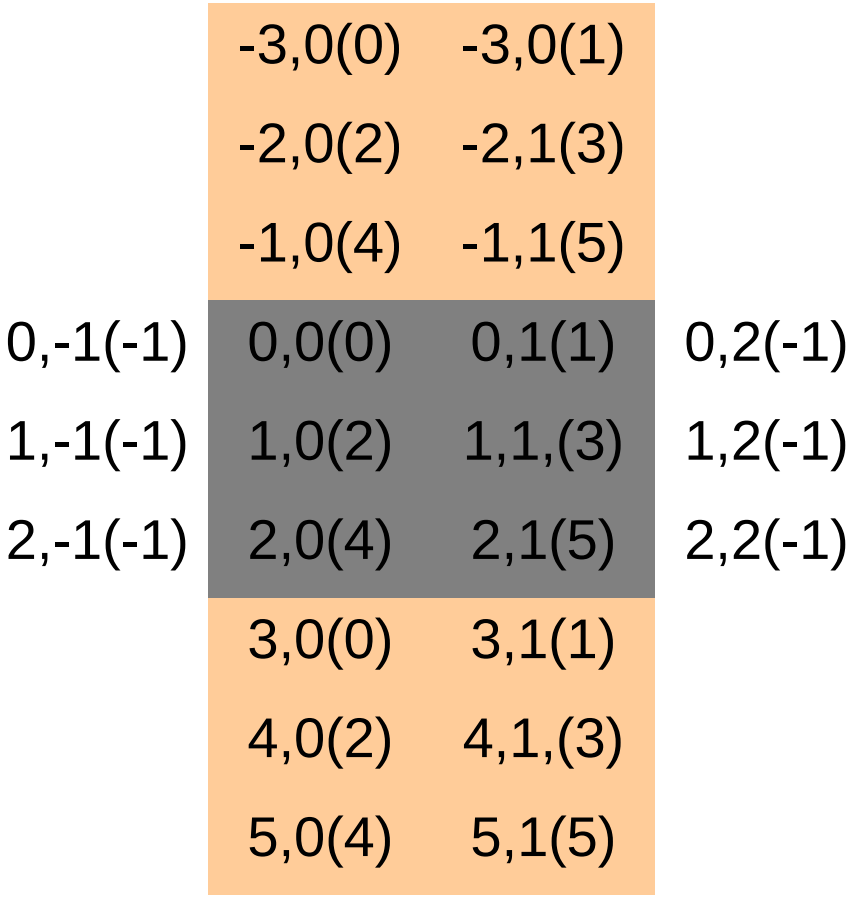

Декартова сетка

# Особенности

- Утильная функция, может вызываться где угодно и не вызывает коммуникаций
- Направления имеют номера с 0 до N-1, где N — размерность декартовой сетки в коммуникаторе
- Если source или dest отрицательные (MPI_UNDEFINED), значит вылезли за границы области
- Если граница периодичная, то возьмет процесс с другой стороны

# Вопросы

# mpi_cart_shift.c




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