Виртуални топологии

Увод в паралелното програмиране #6

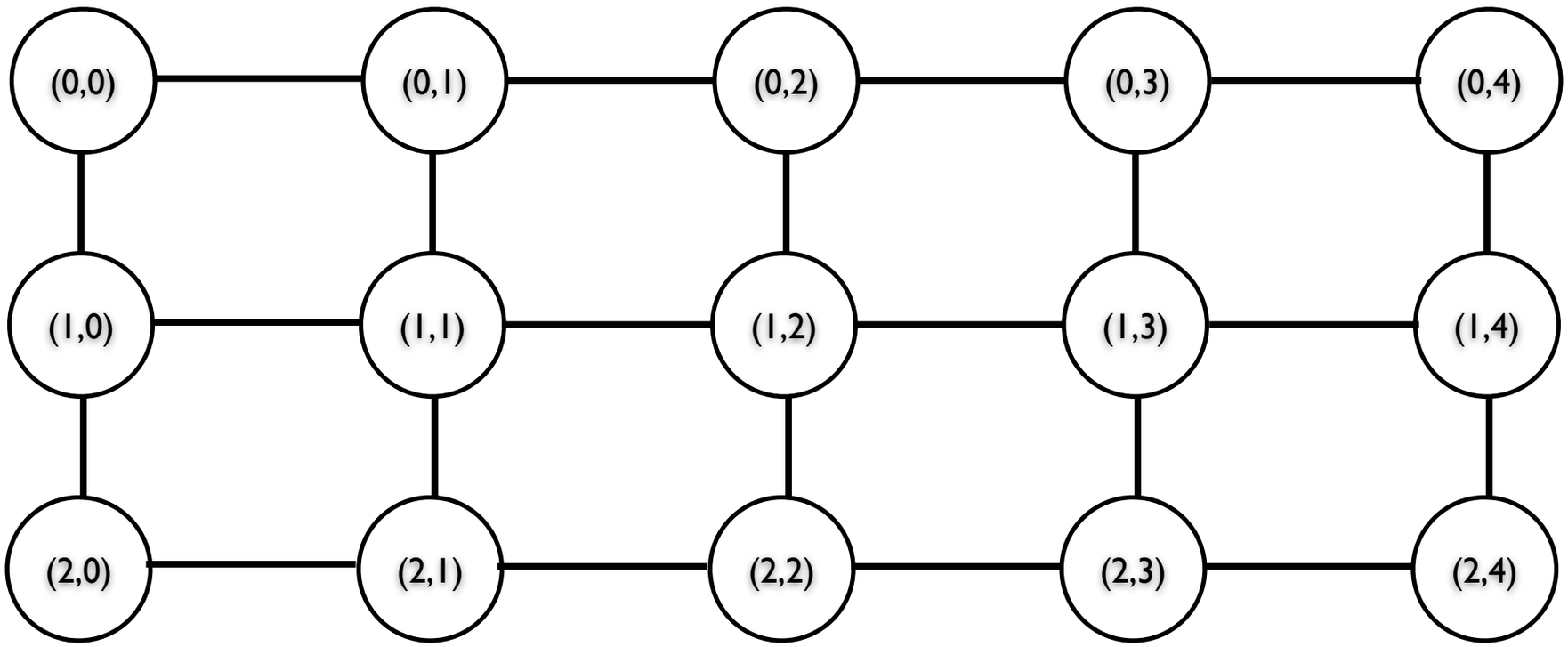
Материалите са разработени в рамките на договор ДО-02-136/2008 с Фонд Научни Изследвания
Проект: IRC-CoSiM Ръководител на проекта: проф. дфзн Ана Пройкова, ФзФ на СУ
http://www.irc-cosim.uni-sofia.bg/

Общи положения

- Всеки интракомуникатор може да има прикачена *топология*
  - декартова (правоъгълна)
  - граф (обобщена)
- По-удобно номериране на процесите в групата на комуникатора
  - съответствие на геометрията и/или симетрията на решаваната задача
  - отчитане на топологията на свързване на отделните модули на хардуера
- Функции за преобразуване на адресите в рангове и обратно

Виртуални топологии

- Схема на адресация и връзки на съседство



- Липсата на връзка не означава, че комуникация между възлите е невъзможна, а че липсва съседство
- Връзките нямат дължина и/или тегло

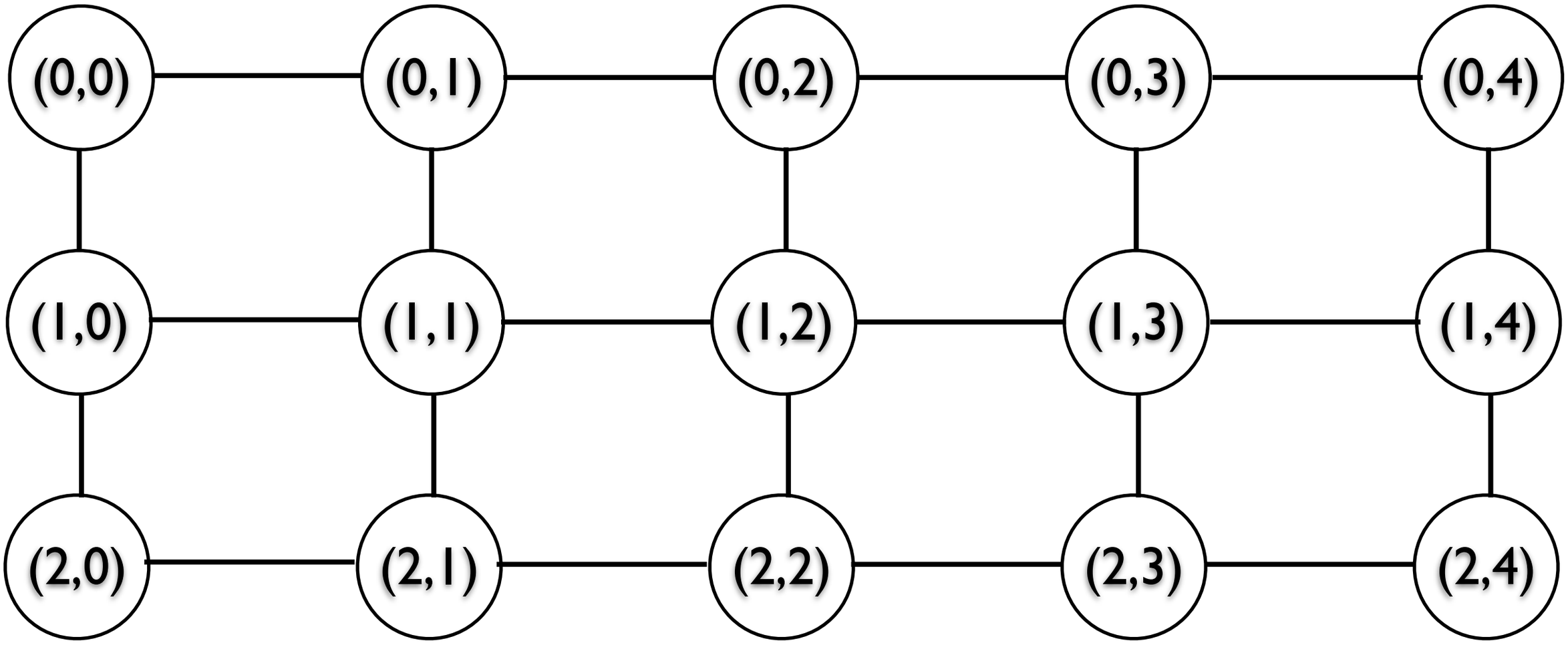
Адресация

- Всеки възел (процес) в топологията има виртуален адрес
- Виртуална адресна схема
- Операциите по изпращане и получаване на данни работят с рангове
  - преобразуване на виртуални адреси в рангове
  - преобразуване на рангове във виртуални адреси
  - специфични за топологията адресни операции

Декартови топологии

- Процесите са разположени във възлите на многомерна целочислена правоъгълна решетка
- Виртуални адреси от вида (d_1,d_2,...)
- Възможно е съседство между крайните възли в противоположните краища по дадено направление – циклични гранични условия
- 1D: нишка; пръстен
- 2D: правоъгълник; цилиндър; тор
- 3D: куб; дебел цилиндър; дебел тор; ???

Декартови топологии



Конструиране на декартова топология

MPI_Cart_create (oldcomm, ndims, dims, periods, reorder, cartcomm)

- Конструира нов комуникатор *cartcomm* от групата на процесите в *oldcomm*
- *ndims* указва размерността на мрежата
- Масивите *dims* и *periods* задават броя на процесите във всяко направление и цикличността му
- Флагът *reorder* контролира евентуалното пренареждане на процесите по процесорите в MPI вселената

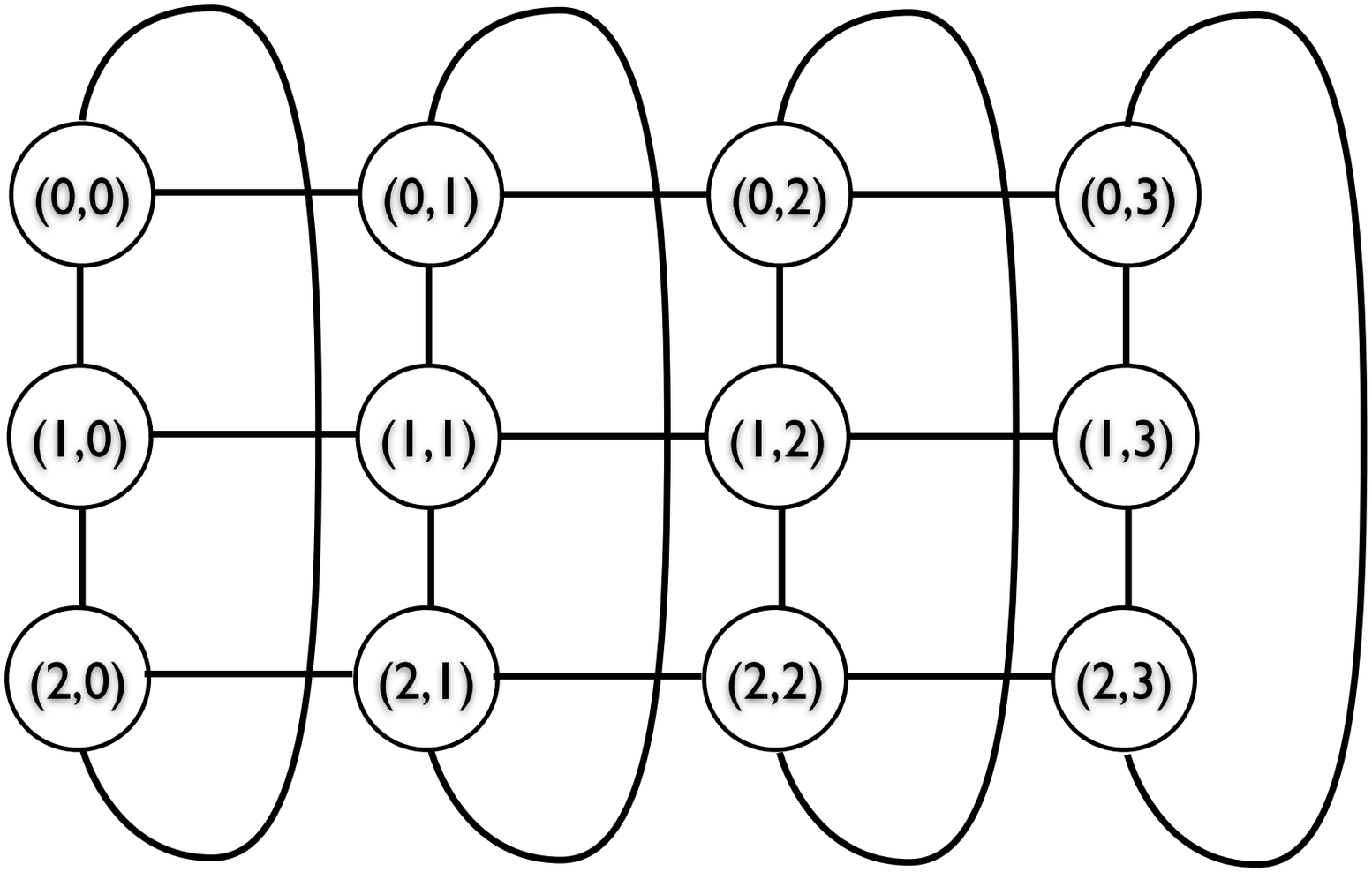
MPI_Cart_create

- Операцията е глобална – всеки един процес от *oldcomm* трябва да я изпълни за да стане член на *newcomm*
- Ако броят на възлите в решетката е по-малък от броя на процесите, в някои процеси *newcomm* == *MPI_COMM_NULL*
- Ако броят на възлите в решетката е по-голям от броя на процесите, то възниква грешка
- *reorder* == FALSE гарантира еднакви рангове на процесите в *oldcomm* и в *newcomm*

Пренареждане на процесите

- *reorder* == TRUE
- Сигнализира желание за оптимално пренареждане на процесите, но може да бъде игнорирано от страна на MPI
- MPI може да се възползва от топологията на свързване на изчислителните елементи
- Не бива да се прилага в случаите, когато данните вече са разпределени със scatter или по друг начин
  - в този случай изрично забраняваме пренареждането с *reorder* == FALSE
  - scatter след пренареждането е напълно допустим

Пример: 3 x 4



ndims = 2

dims = [3, 4]

periods = [T, F]

Забележка за потребителите на FORTRAN: координатата по всяко направление се изменя от 0 до $dims_i$-1

Картиращи функции

MPI_Cart_rank (cartcomm, coords, rank)

- Преобразува виртуален адрес в ранг на процес
- *cartcomm* трябва да бъде комуникатор с декартова топология
- *coords* е масив с поне *ndims* целочислени елемента и съдържа компонентите на адреса
- По цикличните направления координатите се взимат по модул. По нецикличните направления е грешка да се подават координати по-малки от 0 и по-големи от $dims_i$-1
- В *rank* се получава рангът на процеса в комуникатора *cartcomm*

Картиращи функции

MPI_Cart_coords (cartcomm, rank, maxdims, coords)

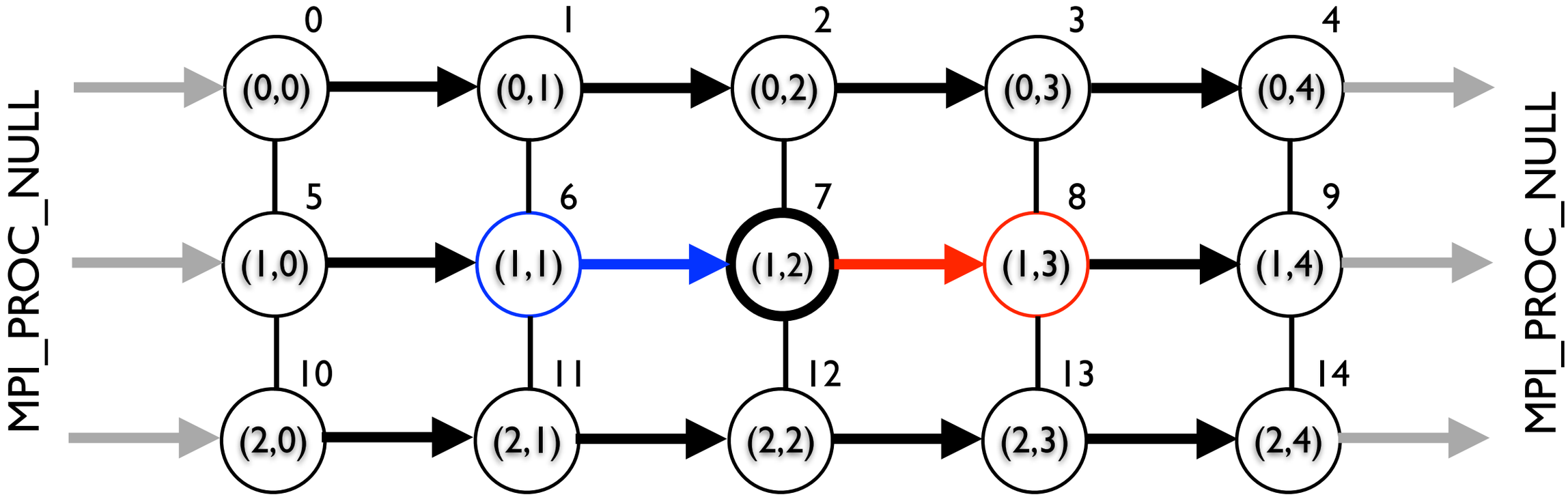
- Преобразува ранг на процес *rank* във виртуални декартови координати *coords*
- *cartcomm* трябва да бъде комуникатор с декартова топология
- *coords* е целочислен масив от поне *maxdims* елемента
- *maxdims* трябва да е поне *ndims*
- Координатите по цикличните направления получават стойности в каноничните им граници

Преместване на данни

MPI_Cart_shift (cartcomm, dir, disp, srank, drank)

- Изчислява ранговете на процеси, на които трябва да се изпратят и от които да се получат данни за осъществяване на преместване на данните на *disp* клетки по направление *dir*
- Направленията се броят от 0 до *ndims*-1
- В *srank* се получава ранга, от който следва да се получат данни
- В *drank* се получава ранга, на който следва да се изпратят данни
- *disp* може да бъде и отрицателно
- При прехвърляне на нециклична граница *srank* или *drank* ще съдържа *MPI_PROC_NULL*

Преместване на данни



rank	srank	drank
7	6	8
5	*MPI_PROC_NULL*	6
9	8	*MPI_PROC_NULL*

Нулеви обекти

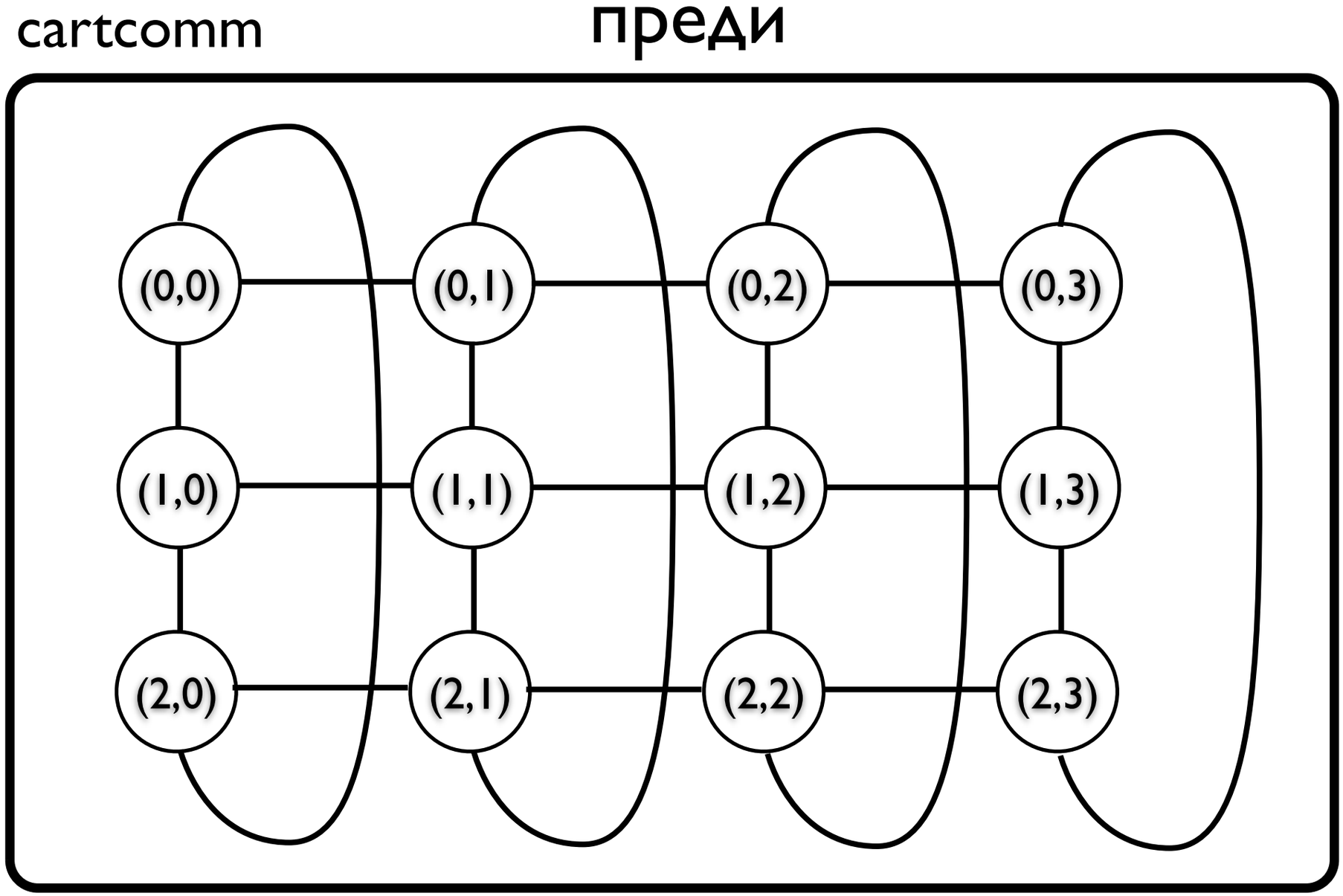
- *MPI_PROC_NULL*
  - празен ранг
  - винаги може да му се изпрати или да се получи от него съобщение с произволен маркер
  - по-симетричен програмен код
- *MPI_COMM_NULL*
  - празен комуникатор
  - операциите с него водят до грешка
  - единствено може да се сравнява за неравенство с други комуникатори с *MPI_Comm_compare()*

Декартово разбиване

MPI_Cart_sub (cartcomm, keepdims, newcomm)

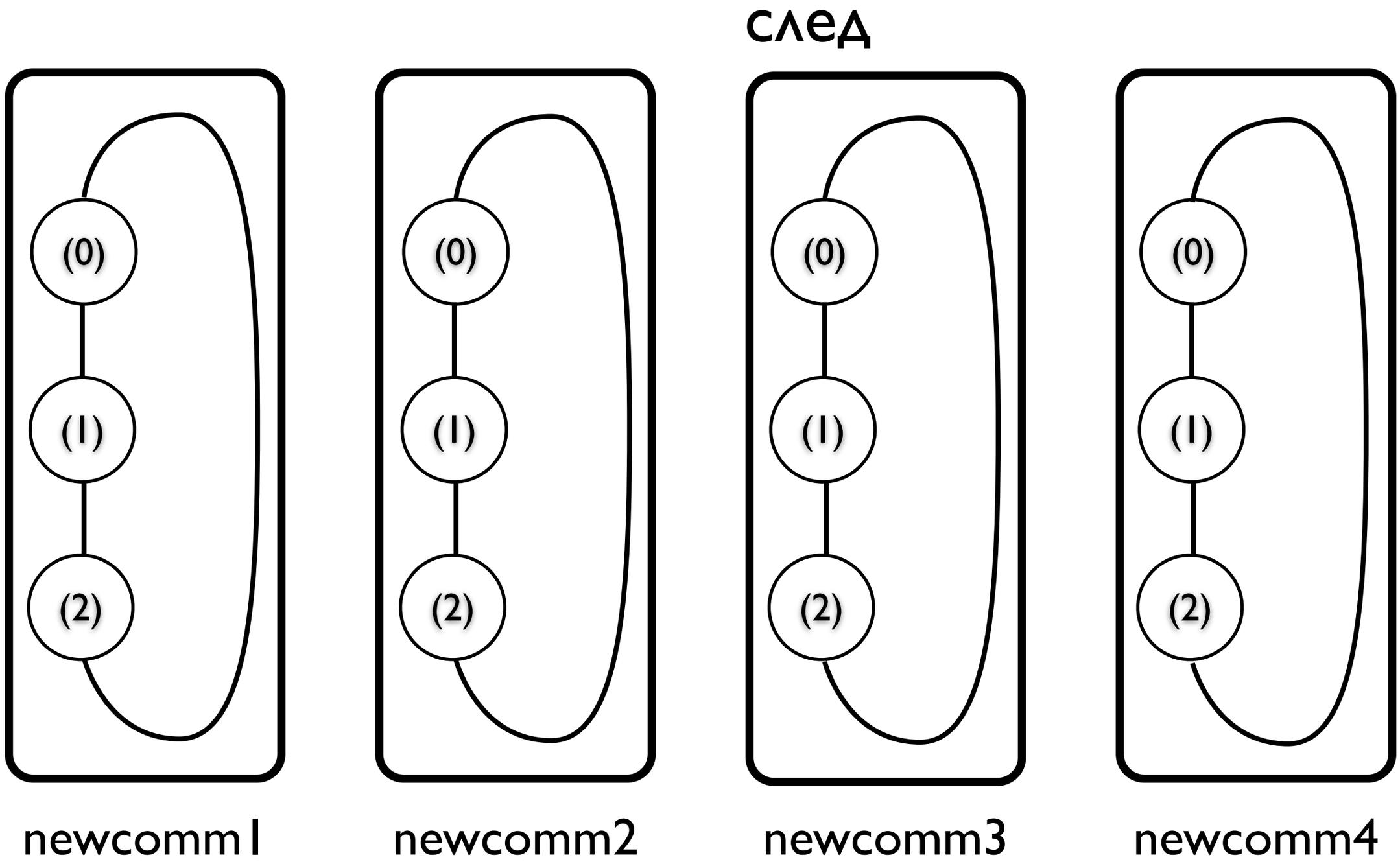
- Създава група по-нискоразмерни топологии като насича декартовата топология на *cartcomm* по направленията, за които *keepdims*$_i$ == FALSE
- От незапазващите се направления се създават нови групи в собствени комуникатори
- Викащият процес задължително присъства в *newcomm*

MPI_Cart_sub



keepdims = [T, F]

MPI_Cart_sub

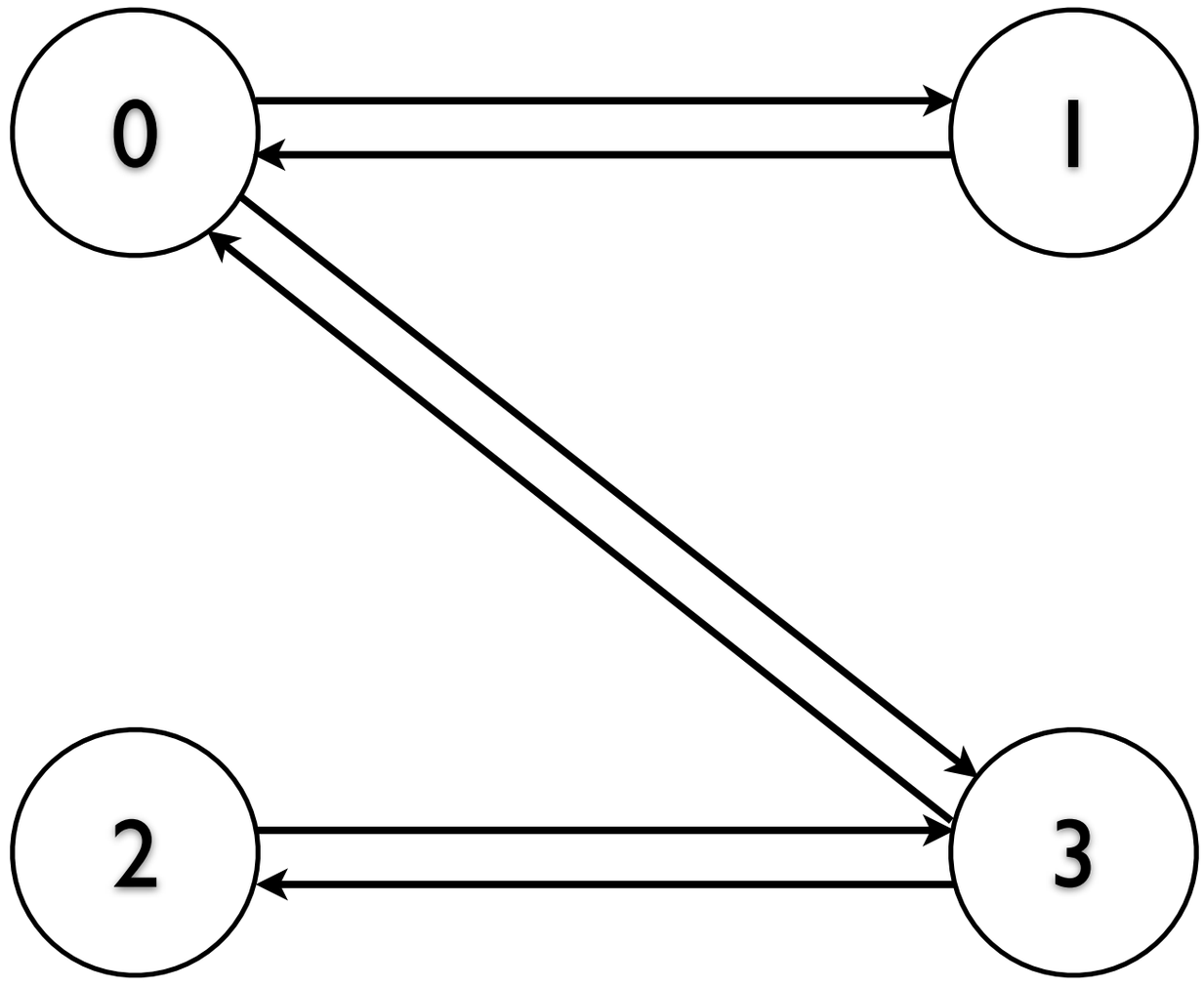


Оптимална декартова топология

MPI_Dims_create (nprocs, ndims, dims)

- Предлага оптимални размерности по различните направления на *ndims*-мерна декартова топология за разполагане на *nprocs* процеса
- Ако $dims_i$ <> 0, то размерността по направление *i* се запазва
- Произведението от ненулевите елементи на *dims* трябва да дели *nprocs*
- На излизане от извикването нулевите елементи са заместени с предложените размерности

Топологии тип граф



Топологии тип граф

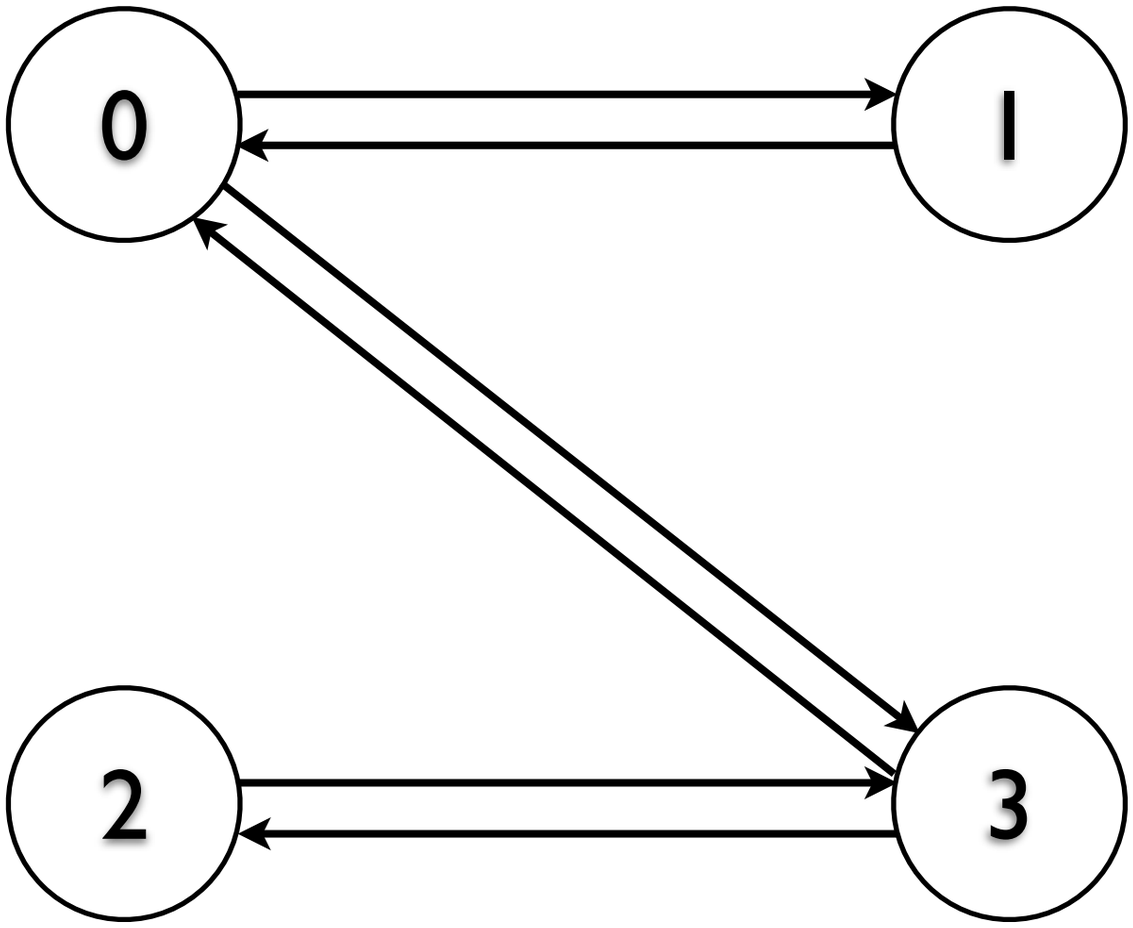
- Топологиите от тип граф не притежават регулярната структура на декартовите топологии
- Процесите нямат виртуални адреси
- Най-общо: механизъм за указване на съседство на процесите
- Възможност за преномериране на процесите за по-добро съответствие на хардуерната топология
- Полезни при реализиране на операции подобни на DFT

Конструиране на графи

MPI_Graph_create (oldcomm, nnodes, index, edges, reorder, cartcomm)

- Конструира нов комуникатор *cartcomm* от групата на процесите в *oldcomm*
- *nnodes* указва броя на върховете на графа
- Масивът *edges* задава последователно съседите на всеки един възел, а *index* указва къде в *edges* свършват съседите на даден възел
- Флагът *reorder* контролира евентуалното пренареждане на процесите по процесорите в MPI вселената

Примерен граф



Възел	Съседи
0	1, 3
1	0
2	3
3	0, 2

nnodes = 4

index = [2, 3, 4, 6]

edges = [1, 3, 0, 3, 0, 2]

Съседство

MPI_Graph_neighbors (grphcomm, rank, maxnbs, nbs)
MPI_Graph_neighbors_count (grphcomm, rank, nnbs)

- Намира ранговете на процесите, които са съседи на процес с ранг *rank* в комуникатора *grphcomm*
- *grphcomm* трябва да е комуникатор с прикачена топология от тип граф
- *nbs* е целочислен масив от поне *maxnbs* елемента
- Броят на съседите *nnbs* може да се получи с *MPI_Graph_neighbors_count()*

Проучване на топологията

MPI_Topo_test (comm, topotype)

- Връща типа на топологията, прикачена към комуникатора *comm*
- *topotype* съдържа типа на топологията:
  - *MPI_UNDEFINED* – не е прикачена топология
  - *MPI_CART* – декартова топология
  - *MPI_GRAPH* – топология от тип граф

Информация за декартова топологията

- *MPI_Cartdim_get (cartcomm, ndims)*
  - връща броя на измеренията в декартовата топология
- *MPI_Cart_get (cartcomm, maxdims, dims, periods, coords)*
  - изходните масиви *dims*, *periods* и *coords* трябва да побират поне *maxdims* елемента
  - връща размерността по всяко направление в *dims*
  - връща цикличността по всяко направление в *periods*
  - връща координатите на извикващия процес в *coords*

Информация за топология тип граф

- *MPI_Graphdims_get (grphcomm, nnodes, nedges)*
  - връща броя на възлите в графа на топологията и дължината на масива с връзки
- *MPI_Graph_get (grphcomm, maxindex, maxedges, index, edges)*
  - изходният масив *index* трябва да побира поне *maxindex* елемента
  - изходният масив *edges* трябва да побира поне *maxedges* елемента
  - в *index* се получават отместванията в *edges*, където свършват съседите на съответния процес
  - в *edges* се получават последователно съседите на всеки процес