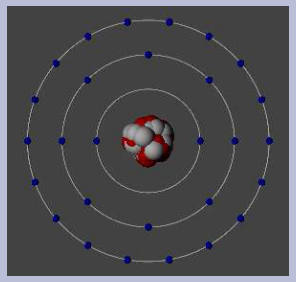
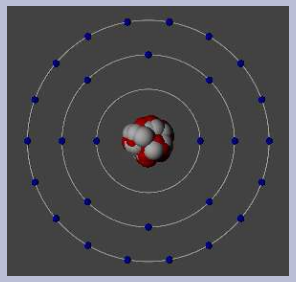


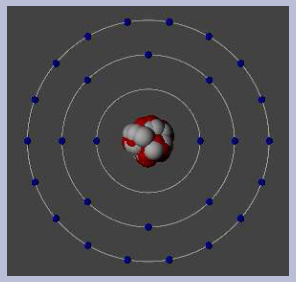
Паралелно програмиране с MPI

Производни типове данни.
Виртуални топологии.



Производни типове данни

- Когато вградените типове в MPI не са достатъчни – потребителски типове
- Структури на C и COMMON блокове на Fortran – не могат да се предават директно в хетерогенни среди
- MPI дава възможност да дефиниране на подматрици, „разредени“ вектори и структури от вече съществуващи типове, както и рекурсивното им влагане – може да се построят произволно сложни типове данни

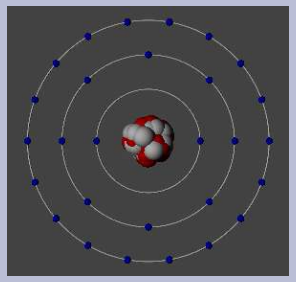


Създаване на производни типове

- Конструиране на типа

MPI_Type_contiguous	Непрекъснат масив
MPI_Type_(h)vector	Вектор, подматрица
MPI_Type_(h)indexed	Индексиран тип
MPI_Type_struct	Структура

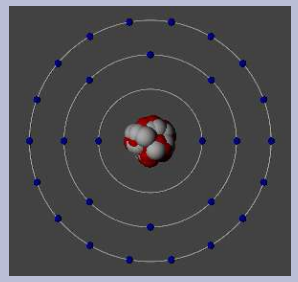
- След конструиране типът се запомня от MPI чрез ***MPI_Type_commit*** и се освобождава с ***MPI_Type_free***.



Непрекъснати типове

MPI_Type_contiguous (count, oldtype, newtype)

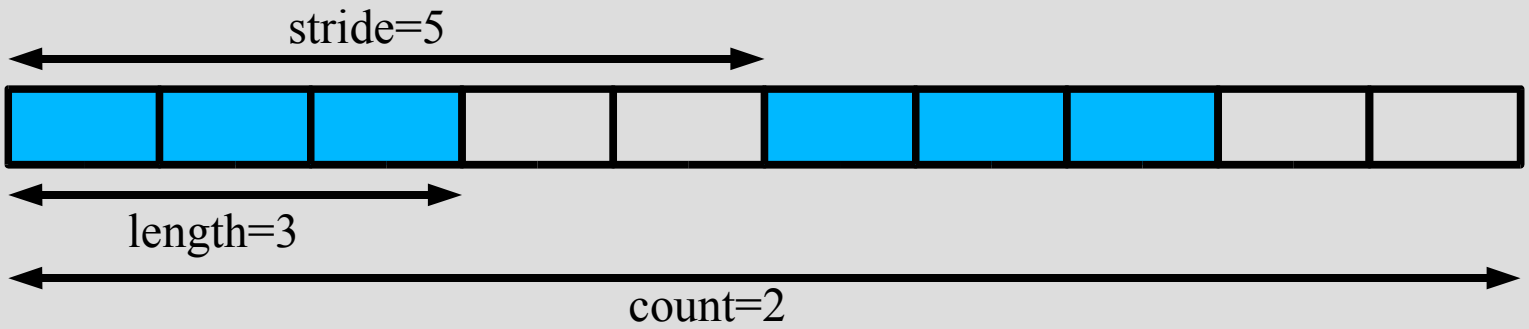
- Създава тип, представляващ непрекъсната последователност от *count* на брой елемента от тип *oldtype*, разположени последователно в паметта.
- *newtype* получава манипулатор на новия тип, който следва да се предаде на MPI чрез ***MPI_Type_commit***.
- Удобен за предаване на цели редове (C) или колони (Fortran) от някаква матрица.



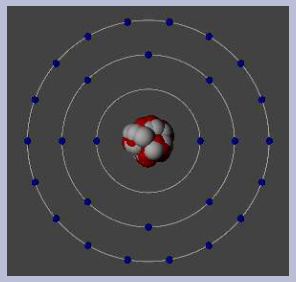
Вектори и подматрици

MPI_Type_vector (count, length, stride, oldtype, newtype)

- Създава вектор от *count* на брой групи от по *length* елемента от тип *oldtype*, началата на които отстоят на *stride* елемента един от друг.



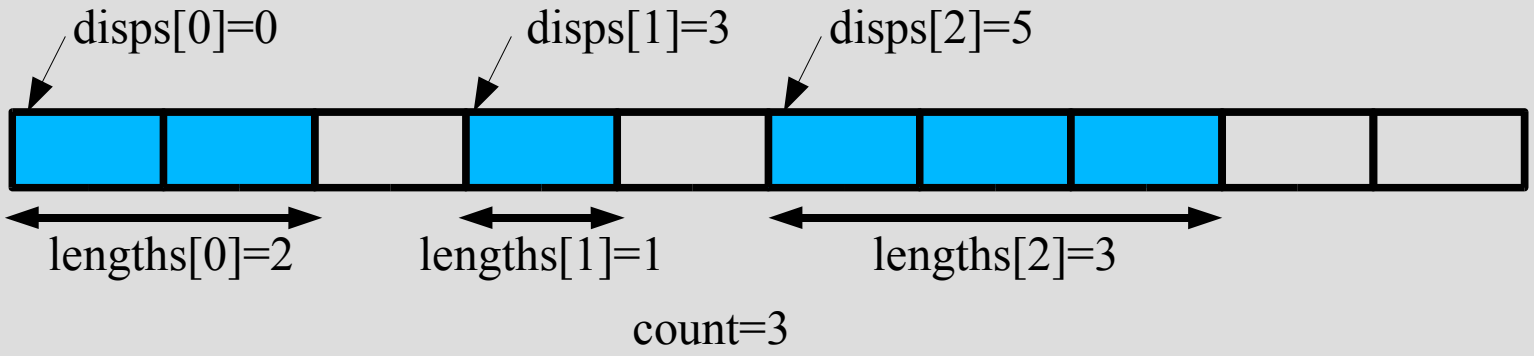
- Удобен за предаване на подматрици.

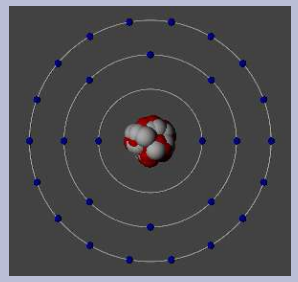


Индексирани типове

MPI_Type_indexed (count, lengths, disps, oldtype, newtype)

- Създава вектор от *count* на брой групи от по *lengths* елемента от тип *oldtype*, началата на които отстоят на *disps* елемента един от началото на типа.

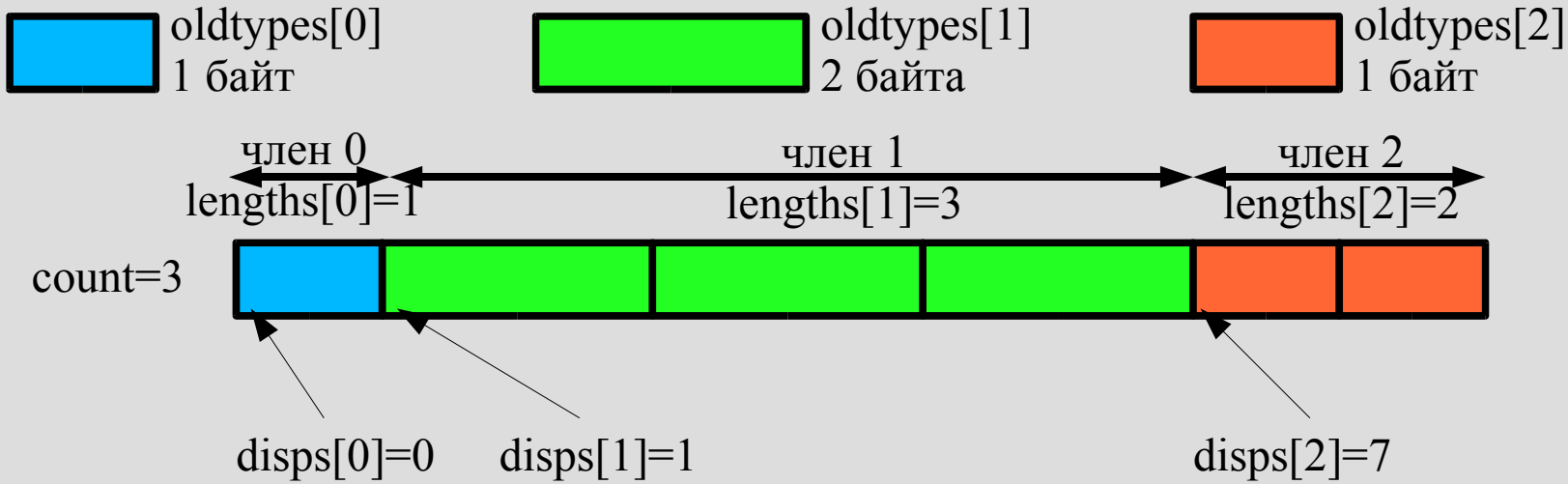


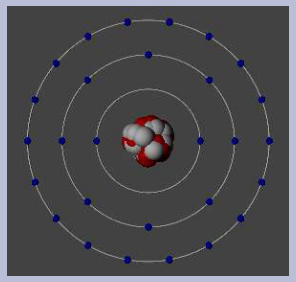


Структури

MPI_Type_struct (count, lengths, disps, oldtypes, newtype)

- Създава структура от *count* на брой члена с дължини *lengths* от типове *oldtypes*, началата на които отстоят на *disps* байта от началото на структурата.



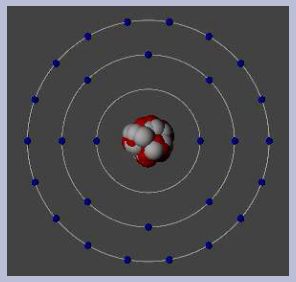


Помощни функции за типове

- За създаване на структури с MPI_Type_struct ни е необходимо да знаем дължината в байтове на участващите типове. За целта използваме

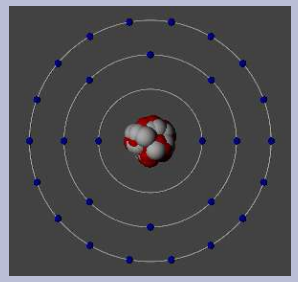
MPI_Type_extent (type, extent)

type - MPI тип
extent – големина в байтове на типа



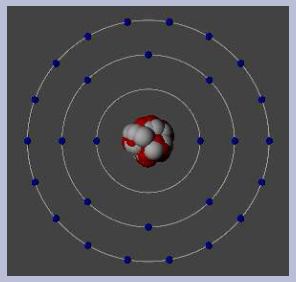
Получаване на съобщения от производни типове

- При подаване на заявка за получаване на съобщение от потребителски тип отместванията се пренебрегват и се прилагат стандартните сравнявания на базовите типове.
- Възможно е получаването на нецял брой елементи от даден тип! Тогава ***MPI_Get_count*** връща MPI_UNDEFINED.
- Броят елементи от базов тип може да се получи с ***MPI_Get_elements***.



Виртуални топологии

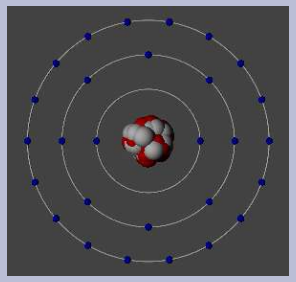
- Виртуалните топологии са механизъм за по-удобно номериране на процесите в даден комуникатор.
- Предимства:
  - Работната среда може да се възползва от исканата топология и да оптимизира физическото разположение на процесите.
  - Получаваме и удобни функции за определяне на ранковете на „съседни“ процеси, при които се взимат под внимание граничните условия.
- Първоначалните ранкове се запазват.



Декартова топология

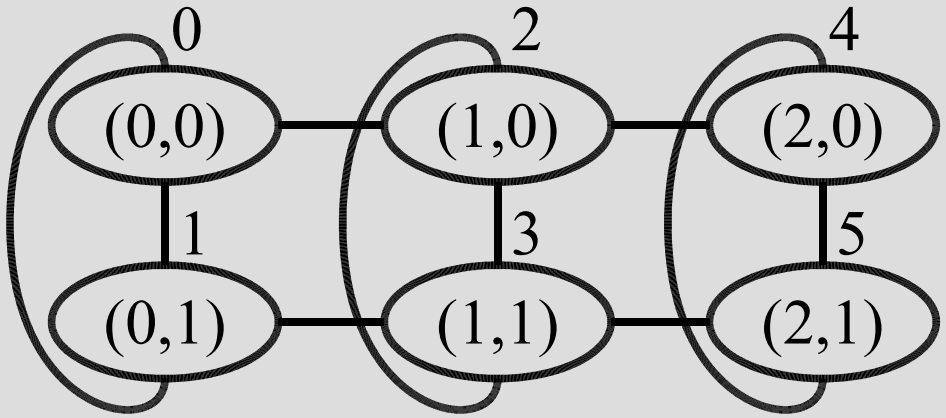
MPI_Cart_create (oldcomm, ndims, dims, periods, reorder, newcomm)

- Произволна многомерна топология тип $m_1 \times m_2 \times m_3 \times \ldots$
- *ndims* на брой размерности, всяка с размер $dims_i$ елемента.
- *periods* контролира периодичността по всяка размерност.
- *reoder* позволява на MPI да преномерира ранковете на процесите в новия комуникатор (използвайте само ако данните вече не са разпределени)



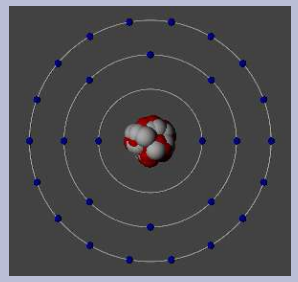
Декартова топология (2)

- Примерна 2x3 топология



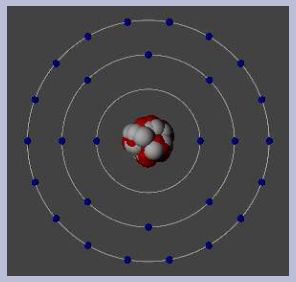
ndims=2
dims[0]=3 dims[1]=2
periods[0]=false periods[1]=true

- Координатите варират от 0 до $dims_i$-1
  (забележка за програмистите на Fortran)



Картиращи функции

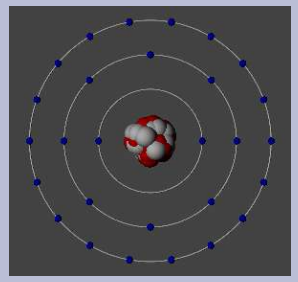
- За преобразуване на декартови координати в ранк се използва ***MPI_Cart_rank (comm, coords, rank)***
  coords – вектор от координати
  rank – ранк на съответстващия процес
- За намиране на координатите на процес по зададен ранк се използва ***MPI_Cart_coords (comm, rank, maxdims, coords)***
  coords – масив от *maxdims* на брой елемента



Преместване на данни

MPI_Cart_shift (comm, dir, disp, ranksrc, rankdst)

- Използва се за намиране на ранка на процес, чиито координати са отместени на *disp* в посока (размерност) *dir* спрямо викащия процес.
- *rankdst* и *ranksrc* са ранковете на процеси, съответно на +*disp* и -*disp*.
- Удобна функция за организация на преместване на данни (от там и shift).

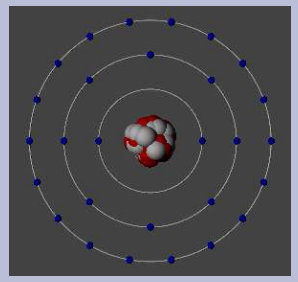


Създаване на подкомуникатор

MPI_Cart_sub (comm, keepdims, newcomm)

- Използва се за „отсичане“ на подмножество на *comm*, в което са запазени само размерностите, за които $keepdims_i$ е true.

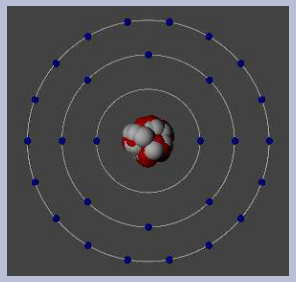
- Викащият процес задължително присъства във върнатия комуникатор *newcomm*.



Балансирано разбиване

MPI_Dims_create (nprocs, ndims, dims)

- По зададен брой процесори в *nprocs* се опитва да създаде топология с възможно най-близки един до друг процеси.
- *dims* съдържа желания брой процеси в някои направления и 0 в останалите. Функцията замества 0-те с най-подходящите стойности.
- Работи само, ако *nprocs* е кратен на произведението от ненулевите елементи.



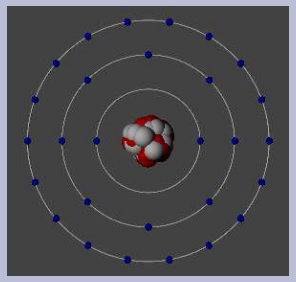
Топология граф

MPI_Graph_create (comm, nnodes, index, edges, reorder, newcomm)

- Създава произволно сложна топология.
- *nnodes* – общ брой процеси
- *index* – елемент *i* съдържа броя съседи на всички процеси от 0 до *i* вкл.
- *edges* – списък на съседство

Процес	Съседи
0	1, 3
1	0
2	3
3	0, 2

nnodes = 3
index = 2, 3, 4, 6
edges = 1, 3, 0, 3, 0, 2



Топология граф (2)

- Откриване на съседите на даден процес
  MPI_Graph_neighbors (comm, rank, maxnbrs, nbrs)

- Откриване на топологията на комуникатор
  MPI_Graphdims_get (comm, nnodes, nedges)
  MPI_Graph_get (comm, maxnodes, maxedges, nodes, edges)
  MPI_Cartdim_get (comm, ndims)