Introduction to Parallel FEM in C
Parallel Data Structure

Kengo Nakajima
Information Technology Center

Programming for Parallel Computing (616-2057)
Seminar on Advanced Computing (616-4009)

Parallel Computing

- Faster, Larger & More Complicated

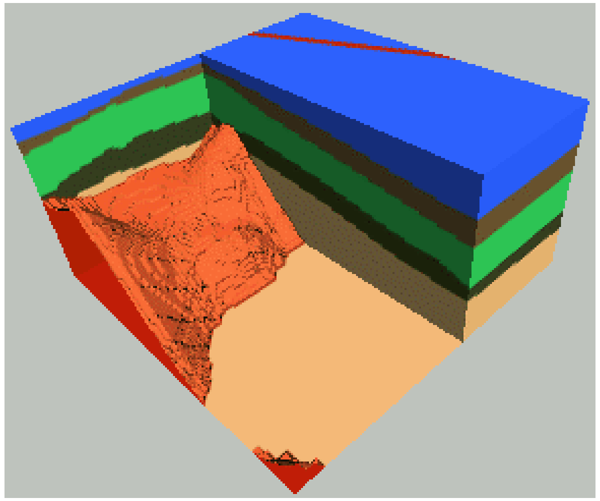
- Scalability
  - Solving N^x scale problem using N^x computational resources during same computation time
    - for large-scale problems: **Weak Scaling**
    - e.g. CG solver: more iterations needed for larger problems
  - Solving a problem using N^x computational resources during 1/N computation time
    - for faster computation: **Strong Scaling**

What is Parallel Computing ? (1/2)

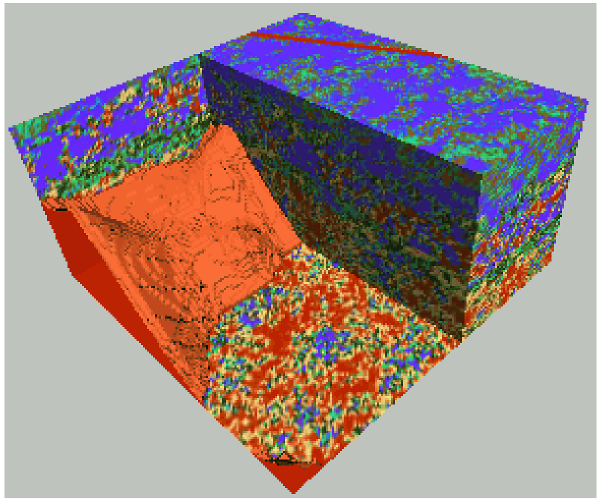
- to solve larger problems faster

Homogeneous/Heterogeneous Porous Media

Lawrence Livermore National Laboratory



Homogeneous

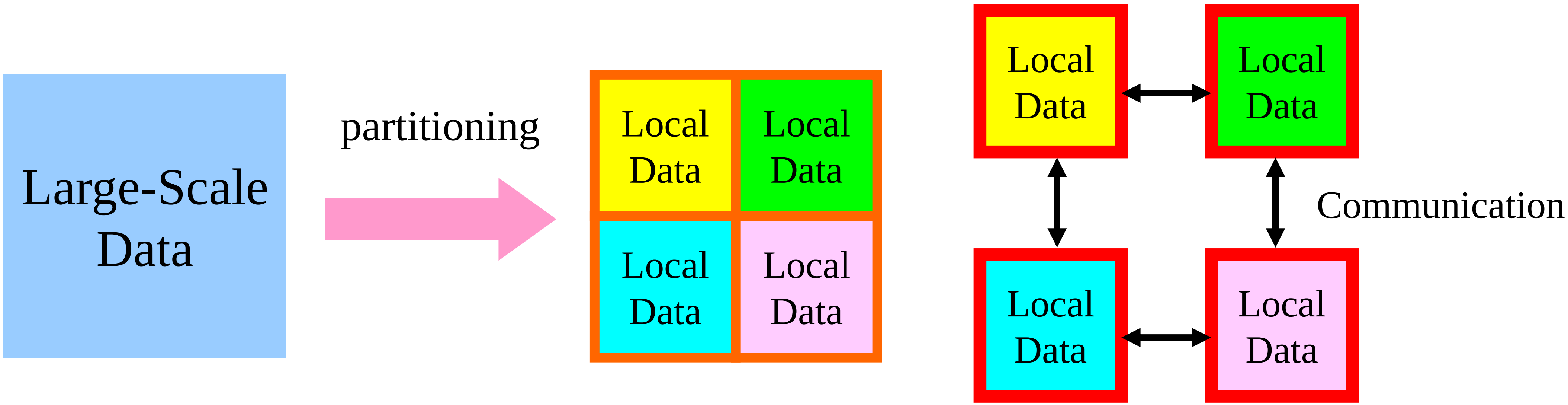


Heterogeneous

very fine meshes are required for simulations of heterogeneous field.

What is Parallel Computing ? (2/2)

- PC with 1GB memory : 1M meshes are the limit for FEM
  - Southwest Japan with 1,000km x 1,000km x 100km in 1km mesh -> 10^8 meshes
- Large Data -> Domain Decomposition -> Local Operation
- Inter-Domain Communication for Global Operation



What is Communication ?

- Parallel Computing -> Local Operations

- Communications are required in Global Operations for Consistency.

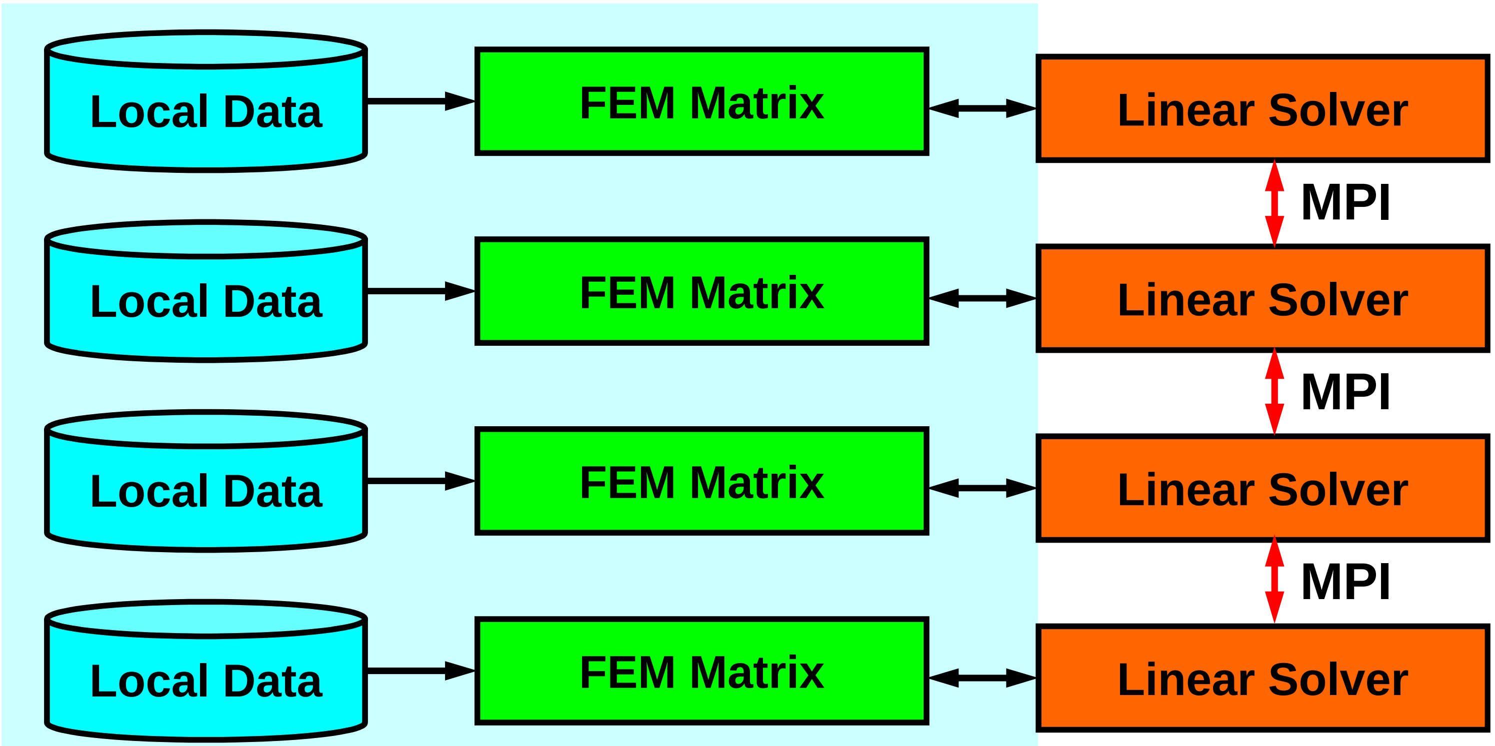
Operations in Parallel FEM

SPMD: Single-Program Multiple-Data

Large Scale Data -> partitioned into Distributed Local Data Sets.

FEM code can assembles coefficient matrix for each local data set : this part could be completely local, same as serial operations

Global Operations & Communications happen only in Linear Solvers dot products, matrix-vector multiply, preconditioning



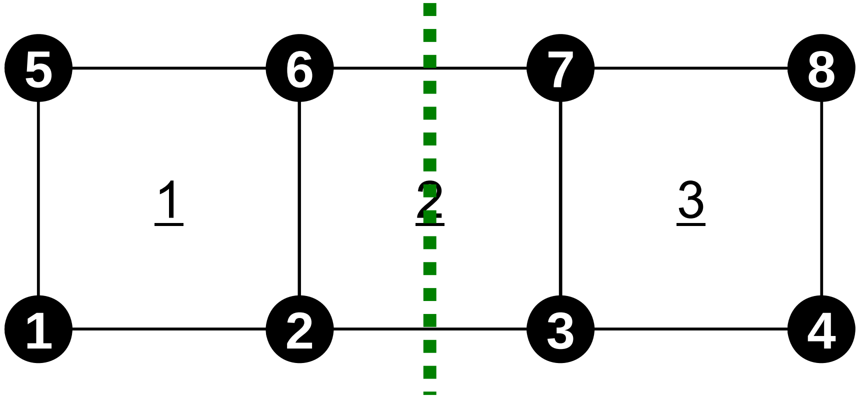
Parallel FEM Procedures

- Design on “Local Data Structure” is important
  - for SPMD-type operations in the previous page

- Matrix Generation
- Preconditioned Iterative Solvers for Linear Equations

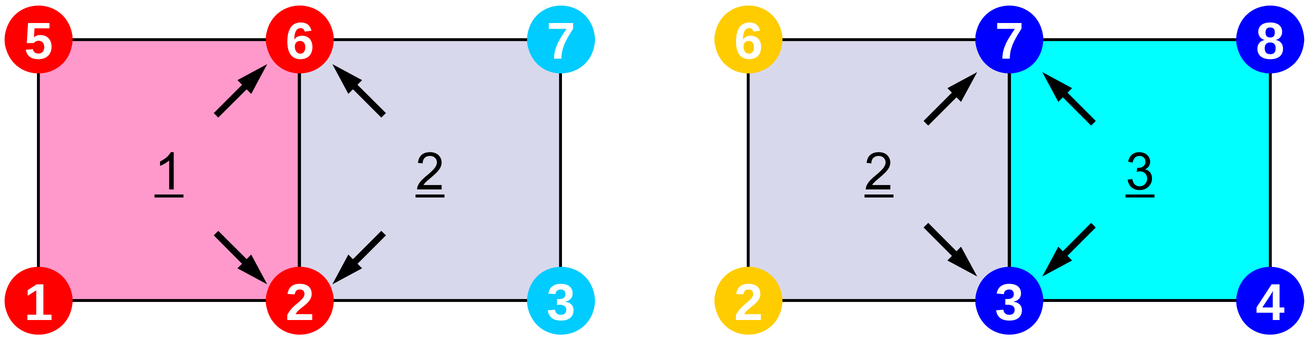
Bi-Linear Square Elements

Values are defined on each node



divide into two domains by “node-based” manner, where number of “nodes (vertices)” are balanced.

Local information is not enough for matrix assembling.



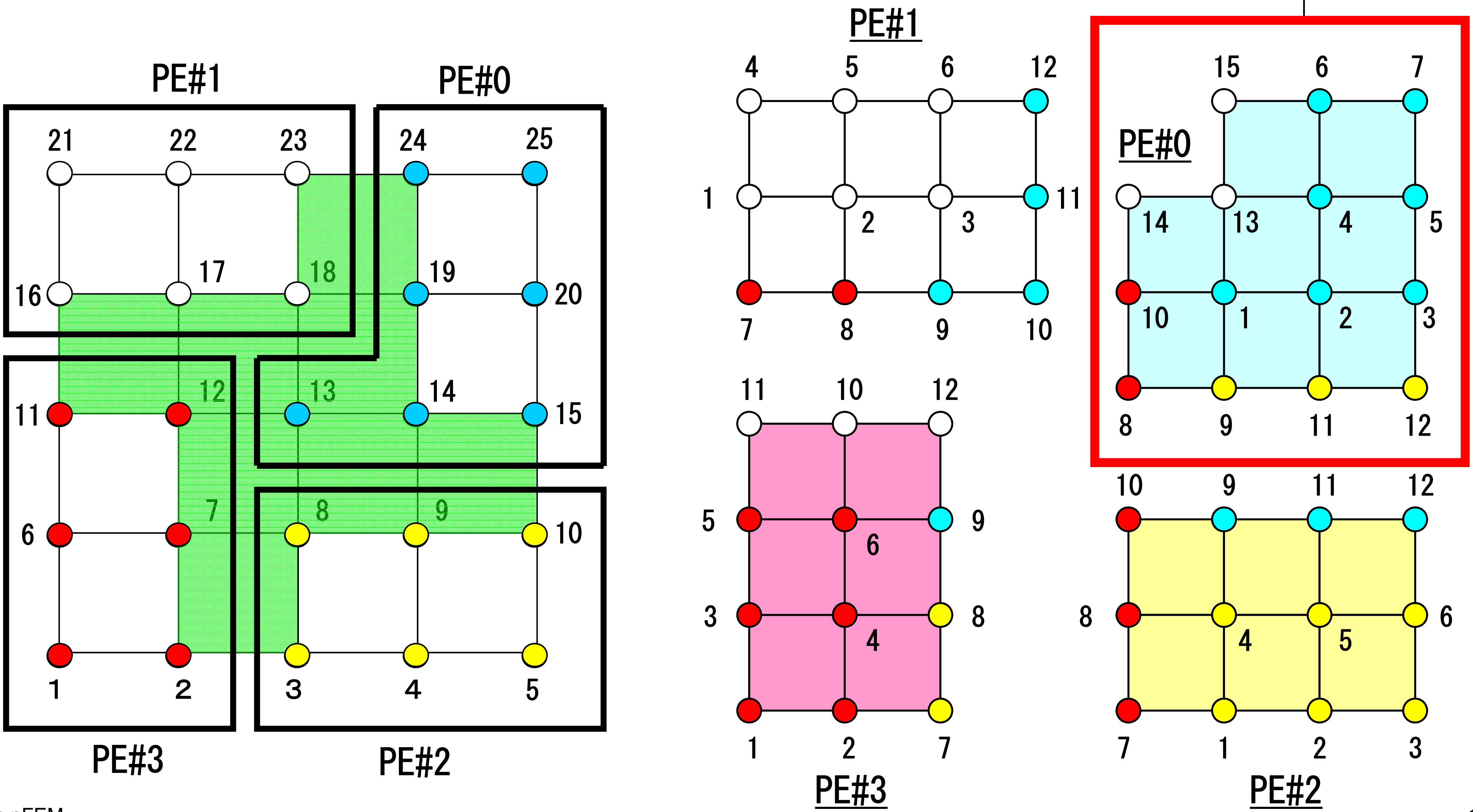
Information of overlapped elements and connected nodes are required for matrix assembling on boundary nodes.

Local Data of Parallel FEM

- Node-based partitioning for IC/ILU type preconditioning methods
- Local data includes information for :
  - Nodes originally assigned to the partition/PE
  - Elements which include the nodes : Element-based operations (Matrix Assemble) are allowed for fluid/structure subsystems.
  - All nodes which form the elements but out of the partition
- Nodes are classified into the following 3 categories from the viewpoint of the message passing
  - Internal nodes originally assigned nodes
  - External nodes in the overlapped elements but out of the partition
  - Boundary nodes *external nodes* of other partition
- Communication table between partitions
- NO global information required except partition-to-partition connectivity

Node-based Partitioning

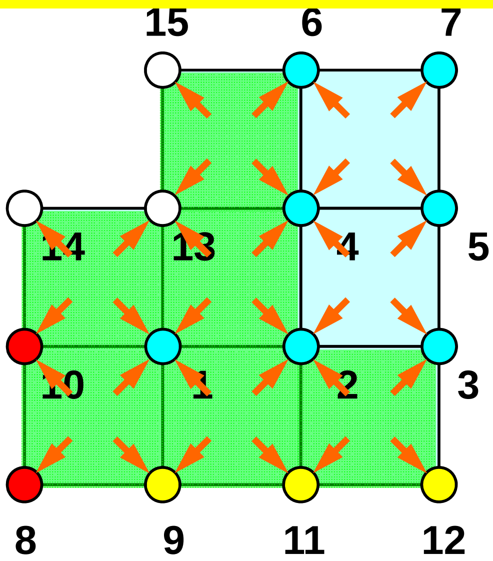
internal nodes - elements - external nodes

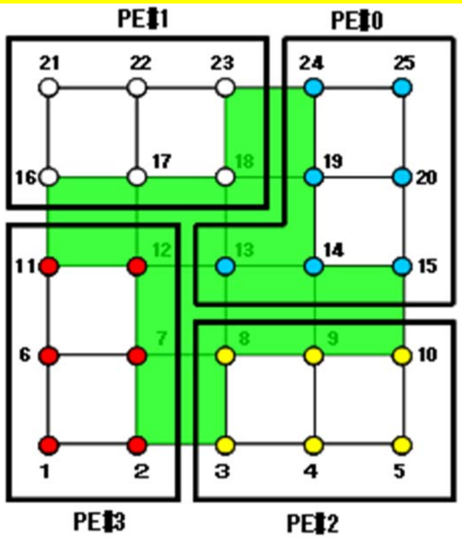


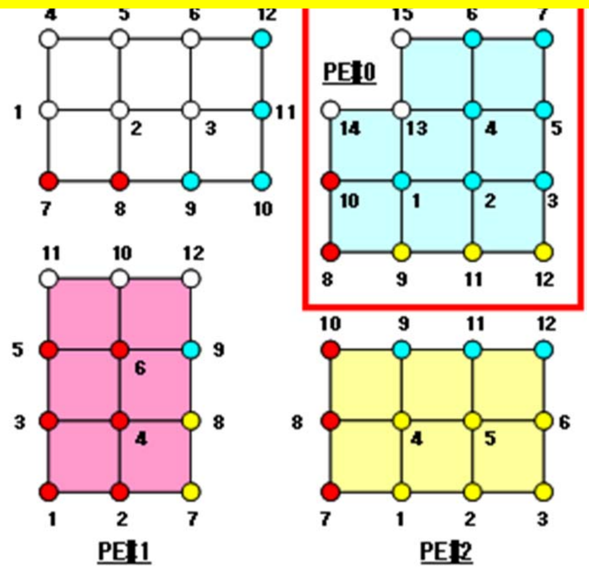
Node-based Partitioning

internal nodes - elements - external nodes

- Partitioned nodes themselves (<u>Internal Nodes</u>) 内点
- Elements which include Internal Nodes 内点を含む要素
- <u>External Nodes</u> included in the Elements 外点
  in overlapped region among partitions.
- Info of External Nodes are required for completely local element–based operations on each processor.

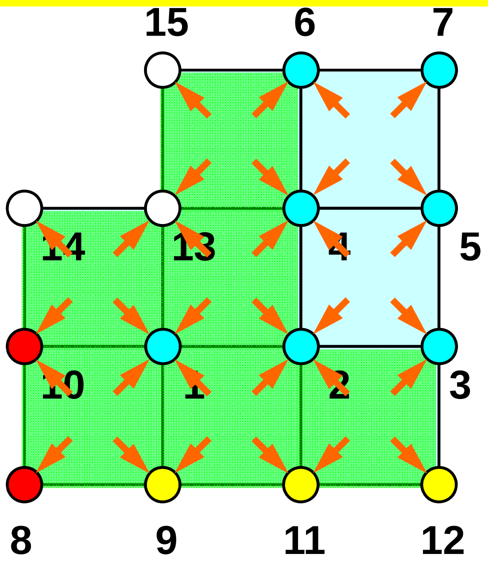


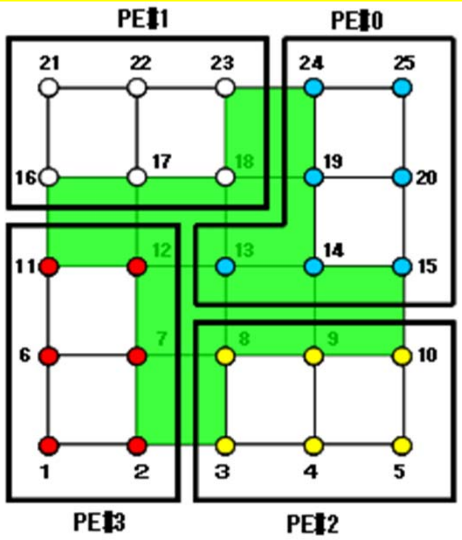


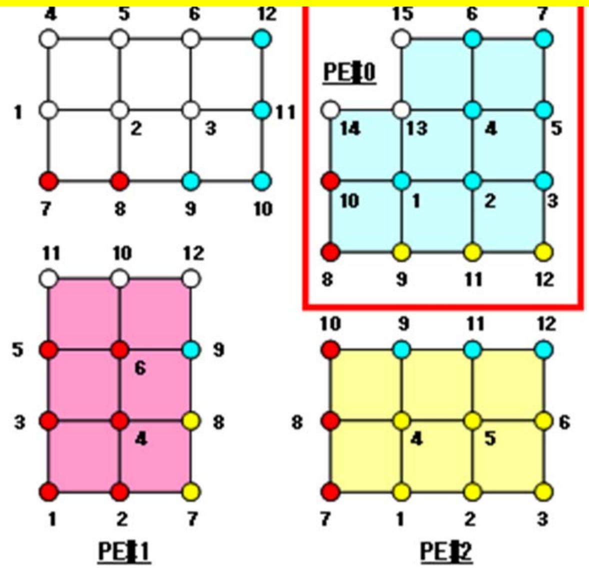


We do not need communication during matrix assemble !!

- ●Partitioned nodes themselves (<u>Internal Nodes</u>)
- ●Elements which include Internal Nodes
- ●<u>External Nodes</u> included in the Elements
  in overlapped region among partitions.
- ●Info of External Nodes are required for completely local element–based operations on each processor.

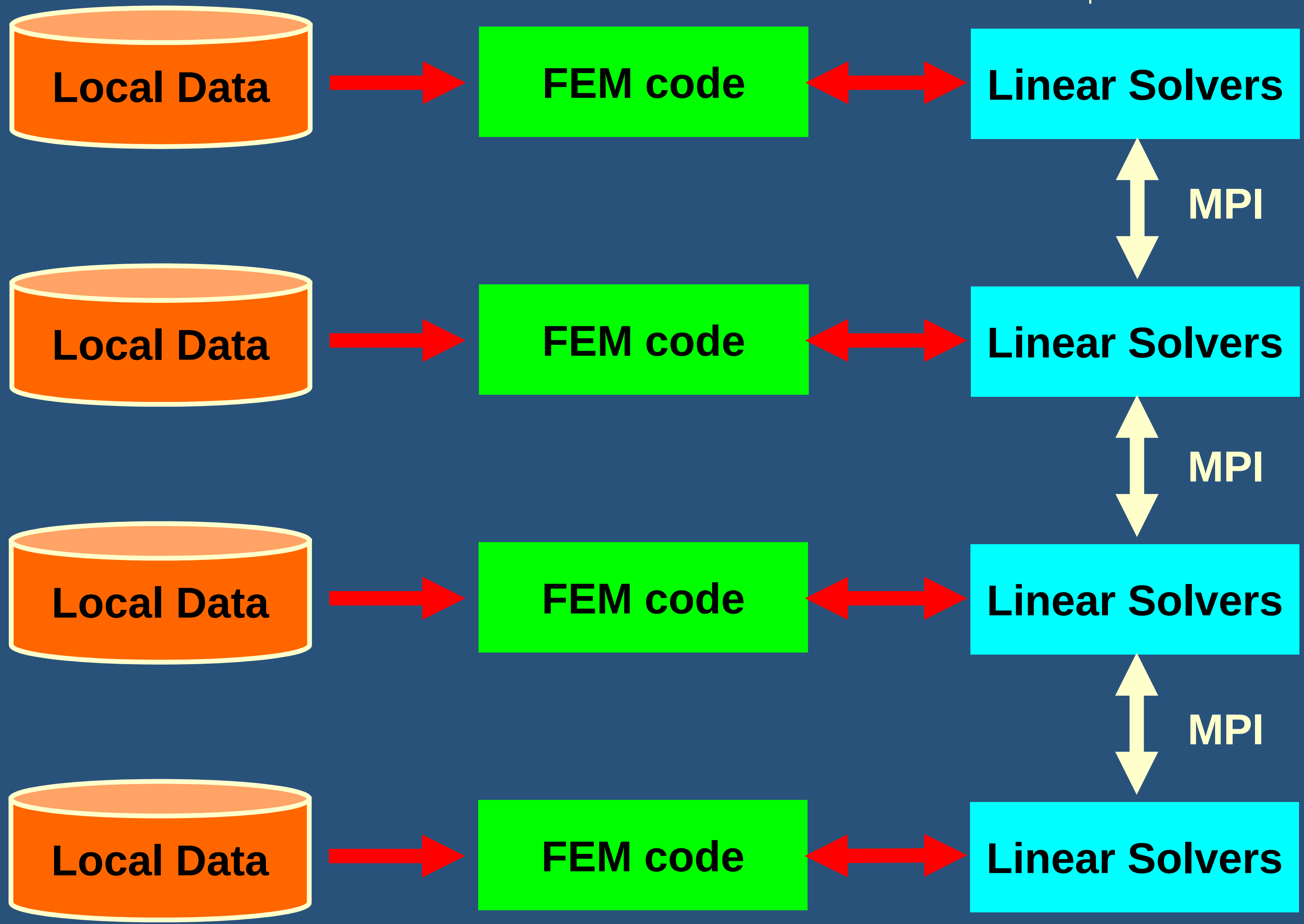




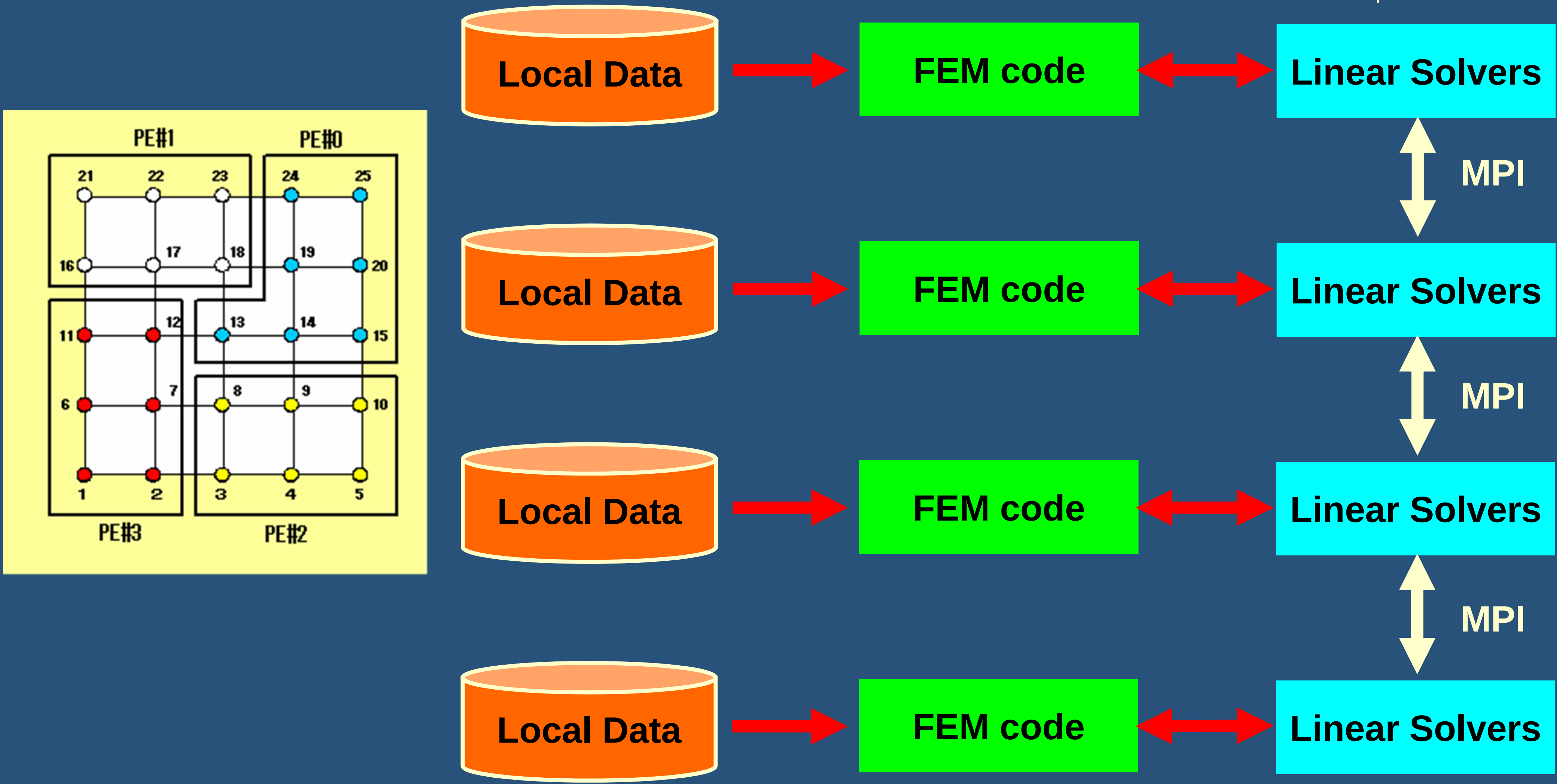


Parallel Computing in FEM

SPMD: Single-Program Multiple-Data

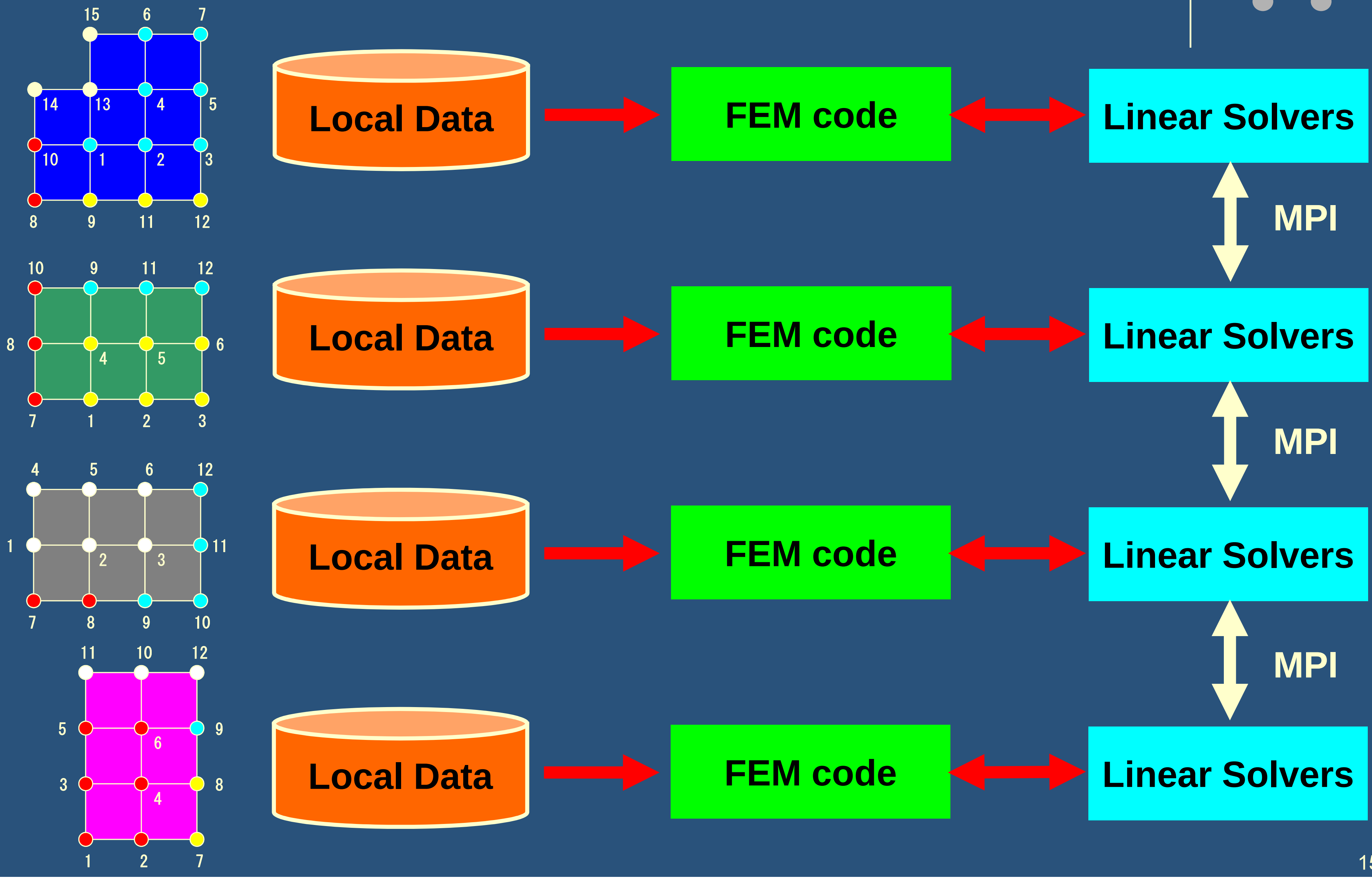


Parallel Computing in FEM
SPMD: Single-Program Multiple-Data

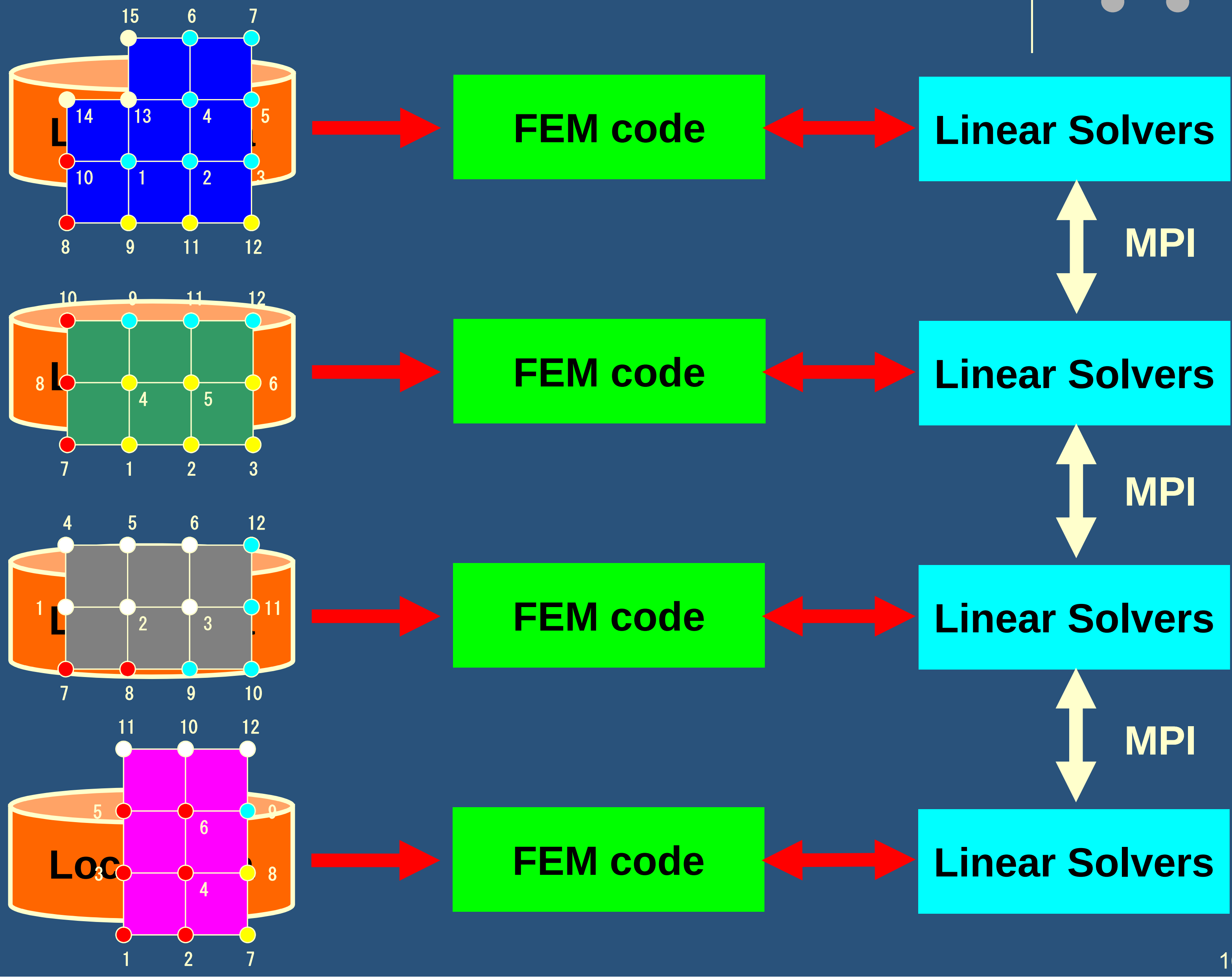


Parallel Computing in FEM

SPMD: Single-Program Multiple-Data

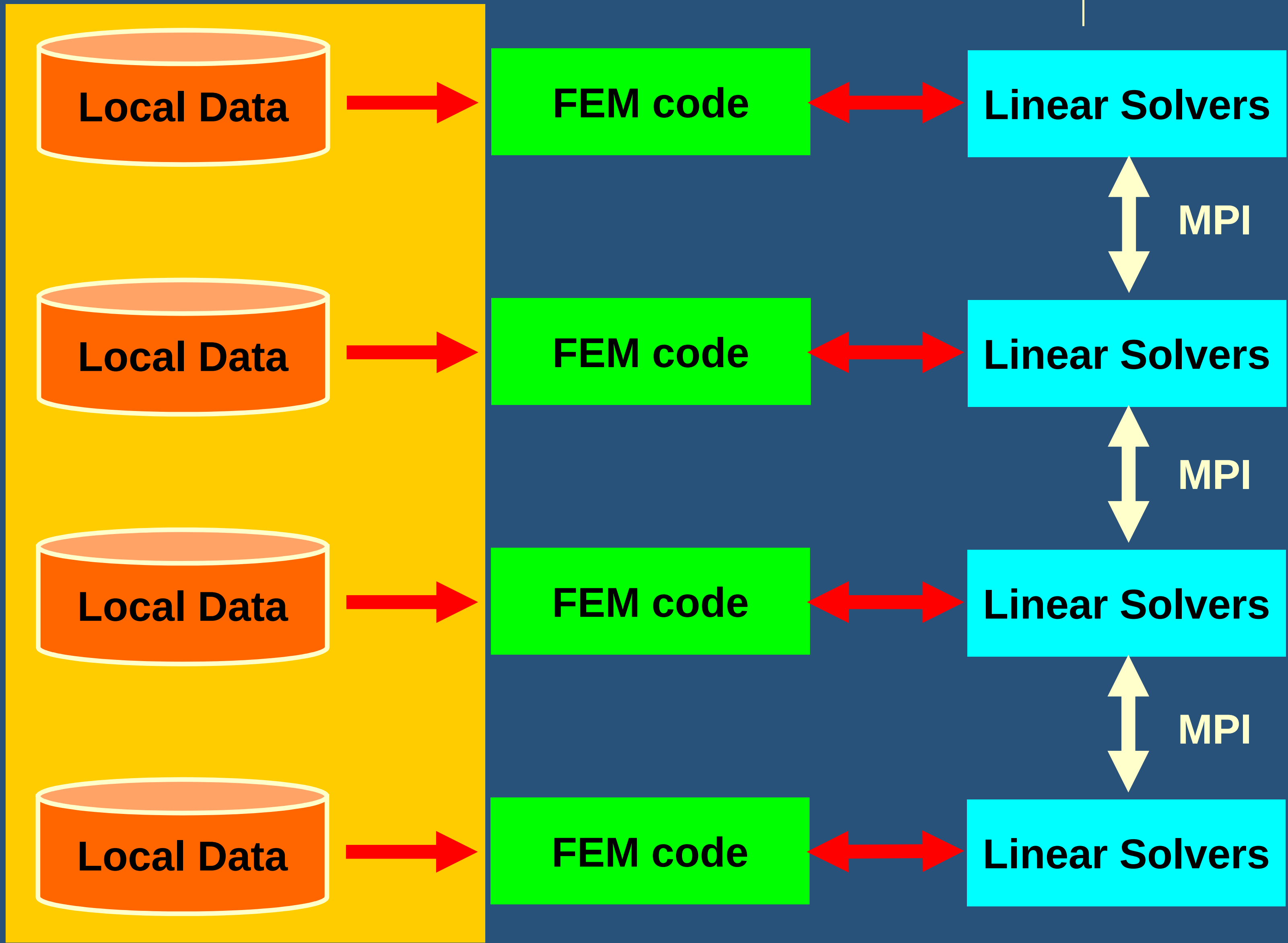


Parallel Computing in FEM
SPMD: Single-Program Multiple-Data



Parallel Computing in FEM

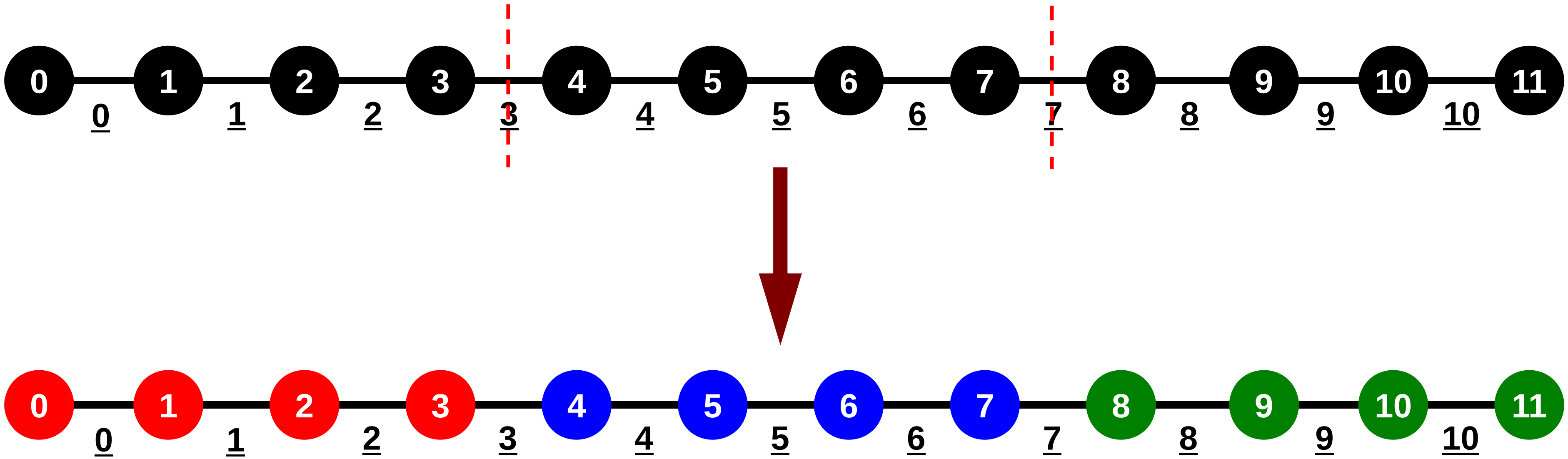
SPMD: Single-Program Multiple-Data



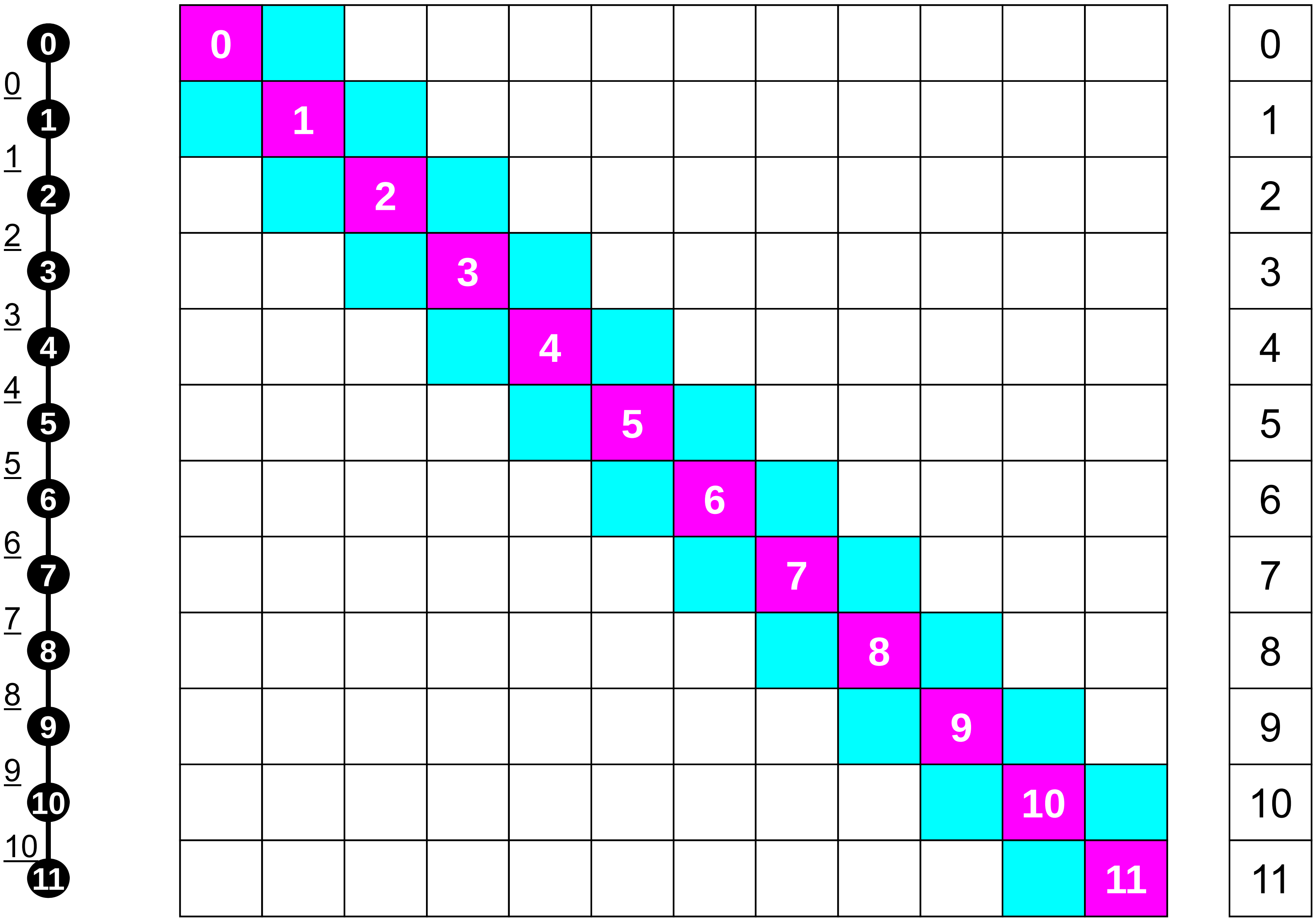
What is Communications ?

- to get information of “external nodes” from external partitions (local data)
- “Communication tables” contain the information

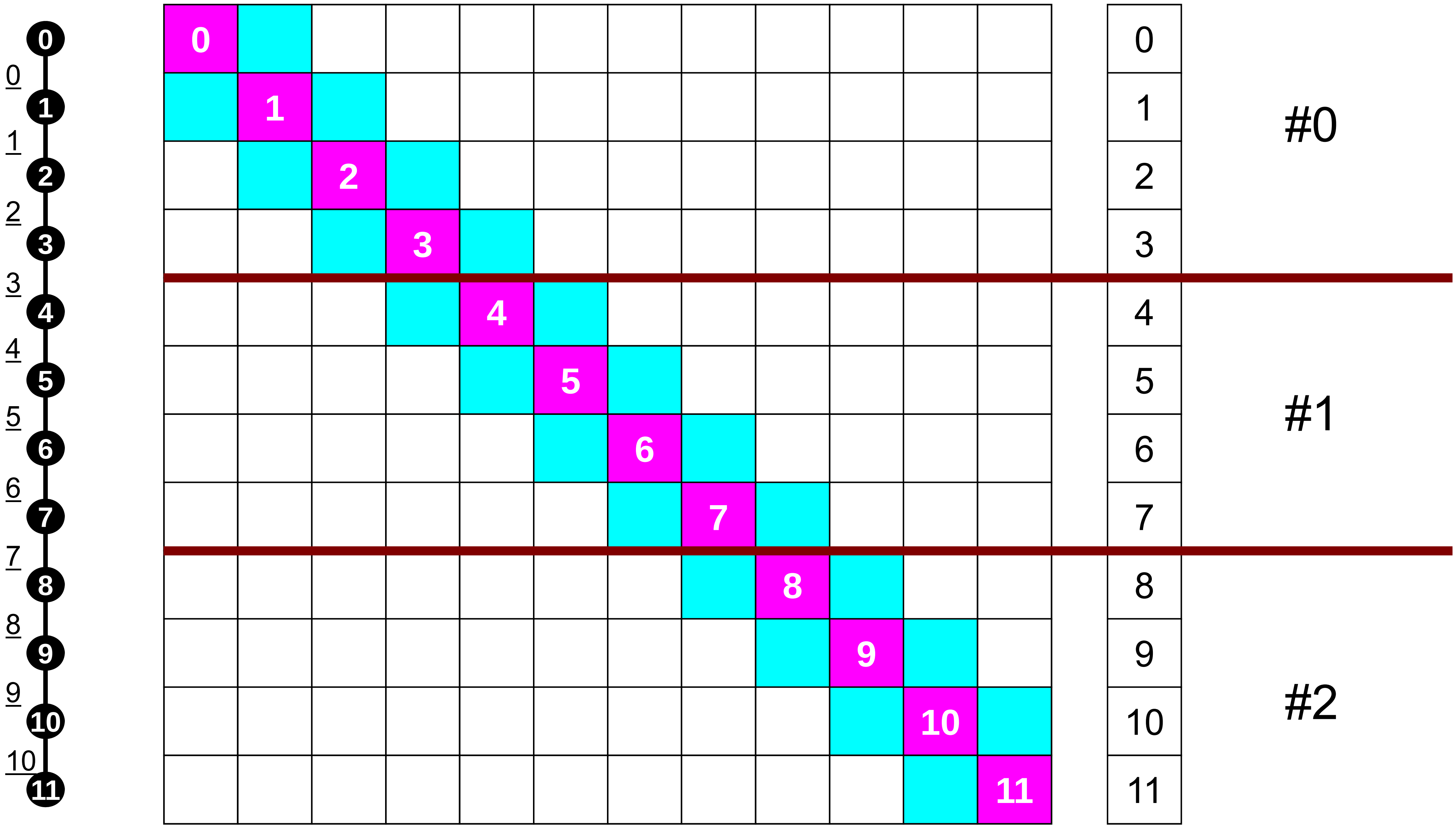
1D FEM: 12 nodes/11 elem's/3 domains



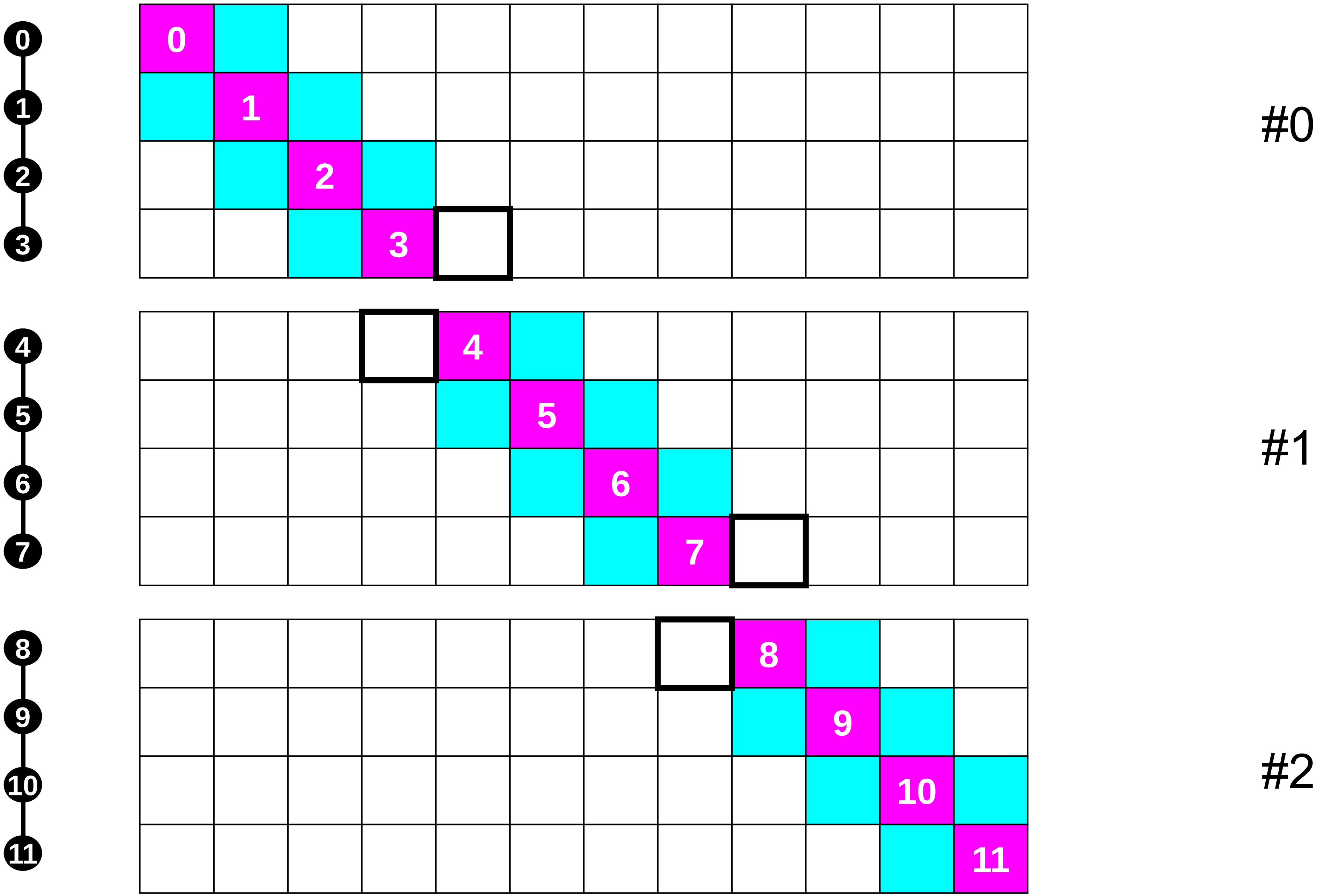
1D FEM: 12 nodes/11 elem's/3 domains



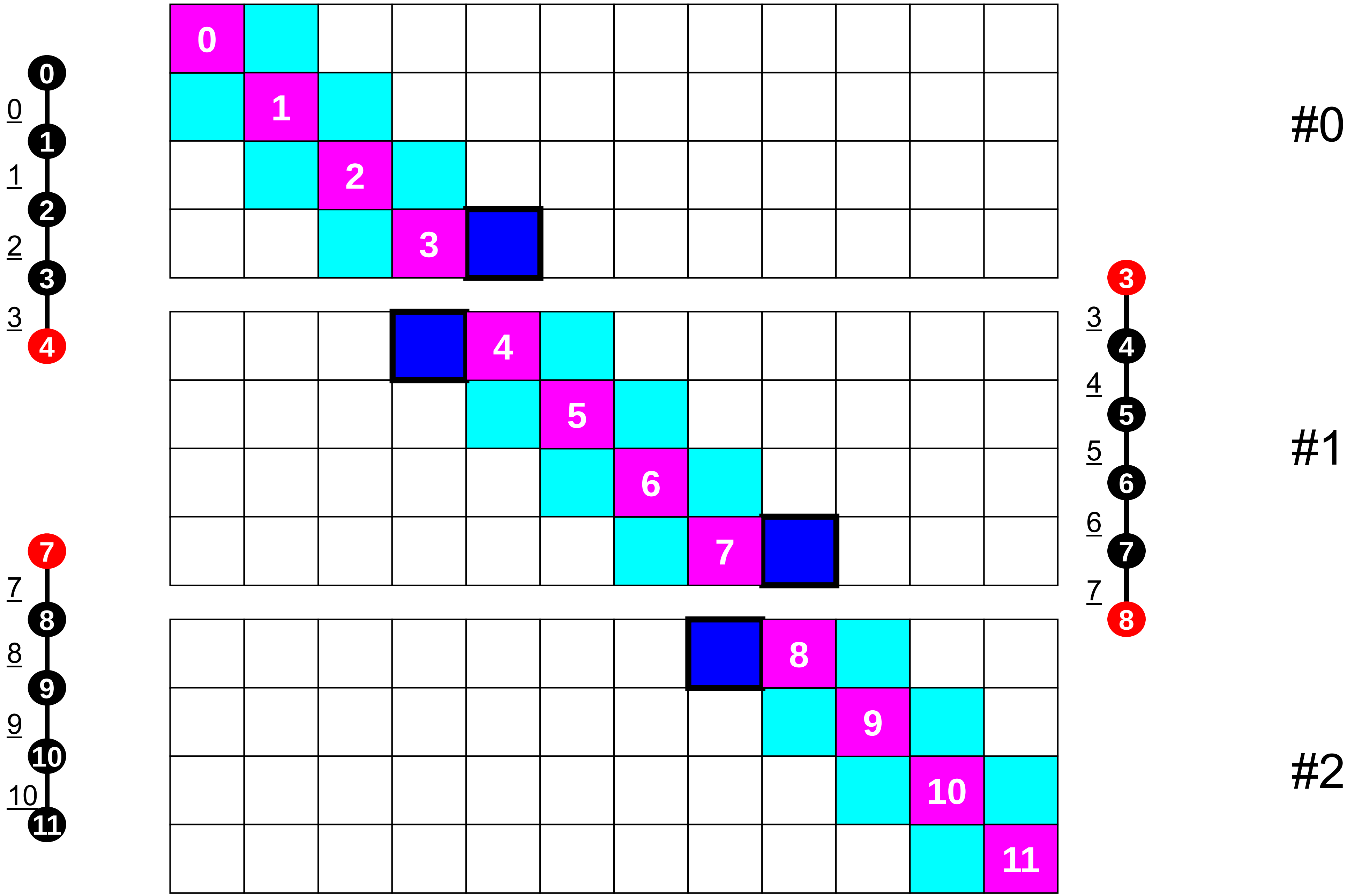
“Internal Nodes” should be balanced



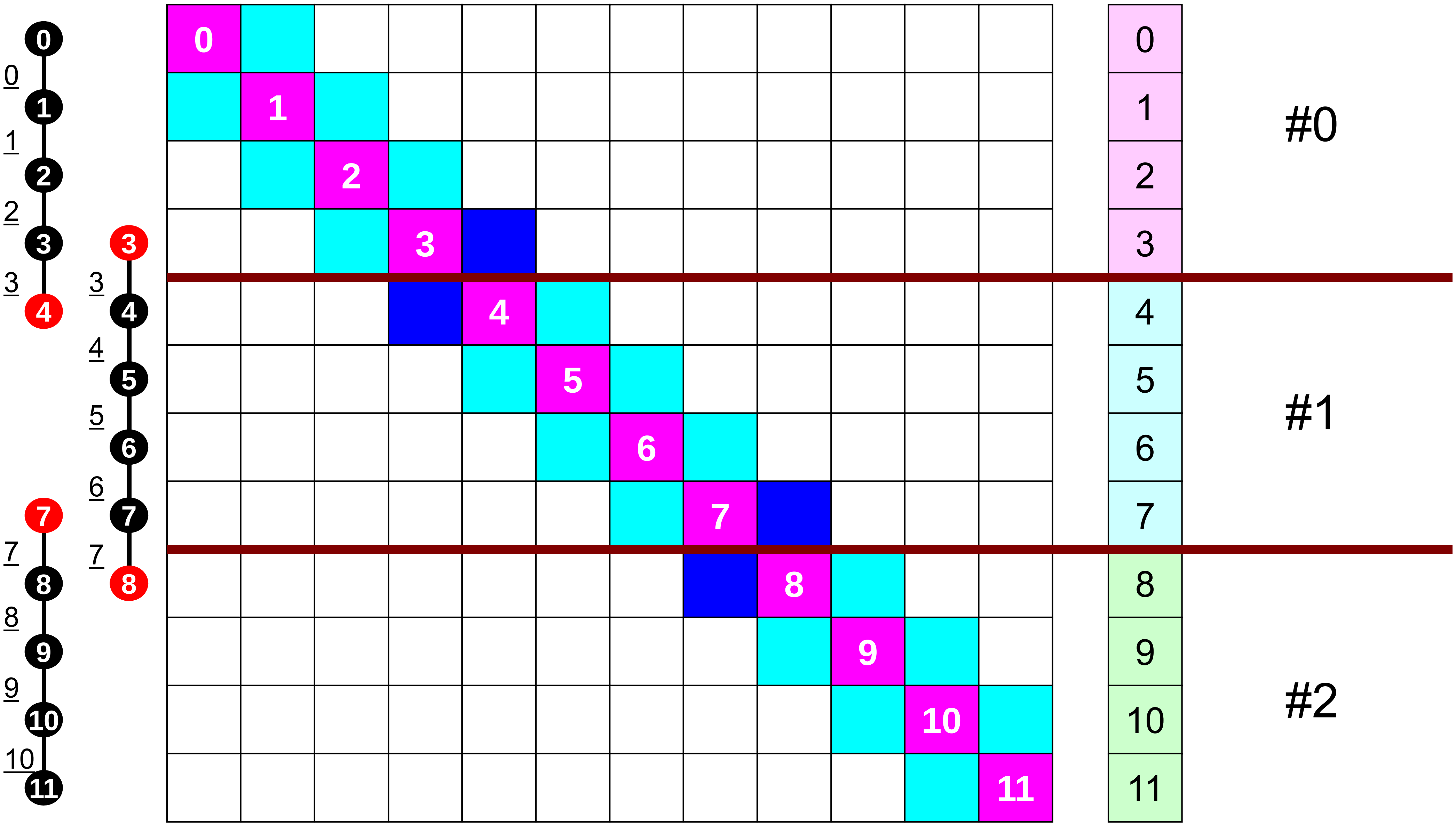
Matrices are incomplete !



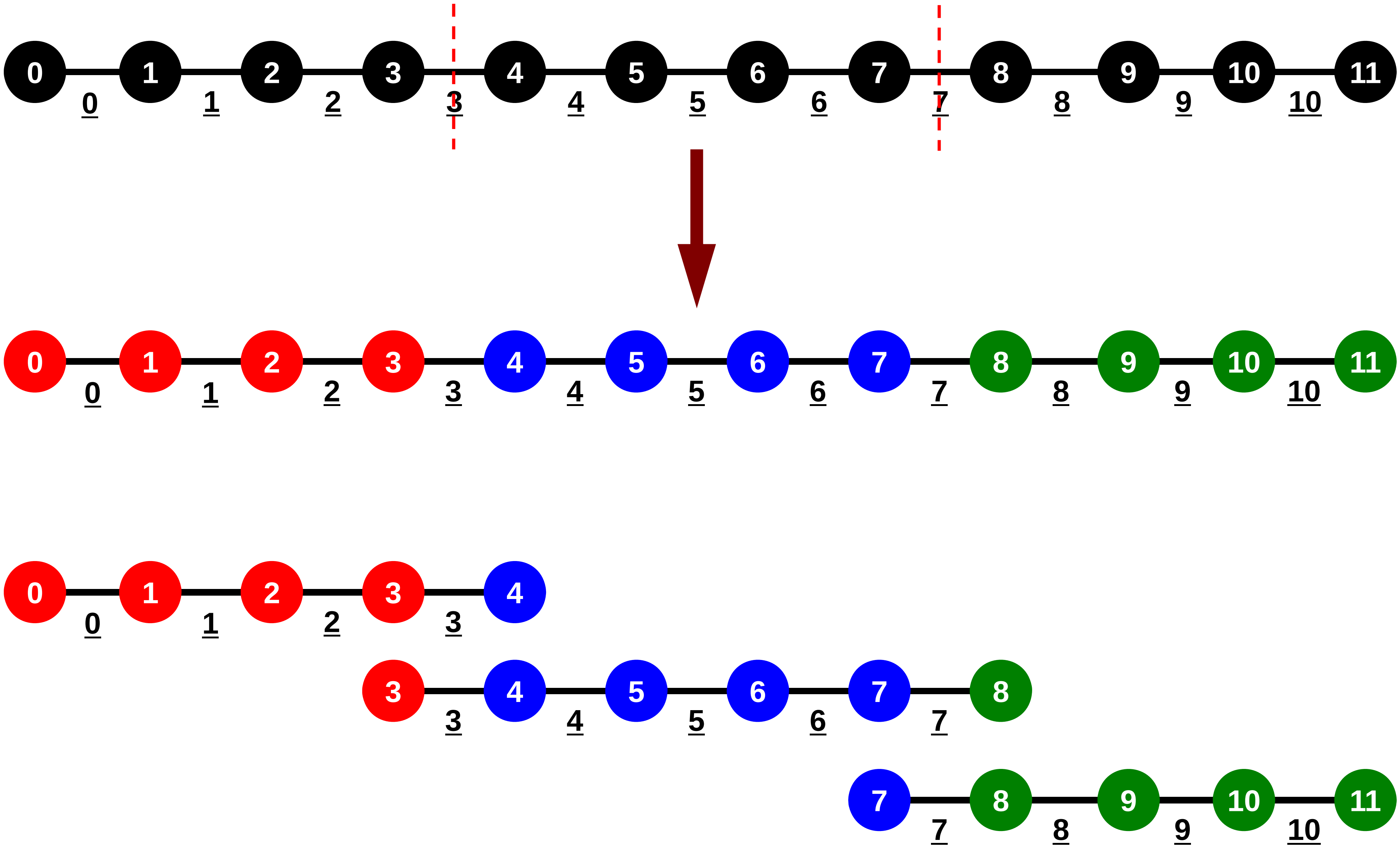
Connected Elements + External Nodes



1D FEM: 12 nodes/11 elem's/3 domains

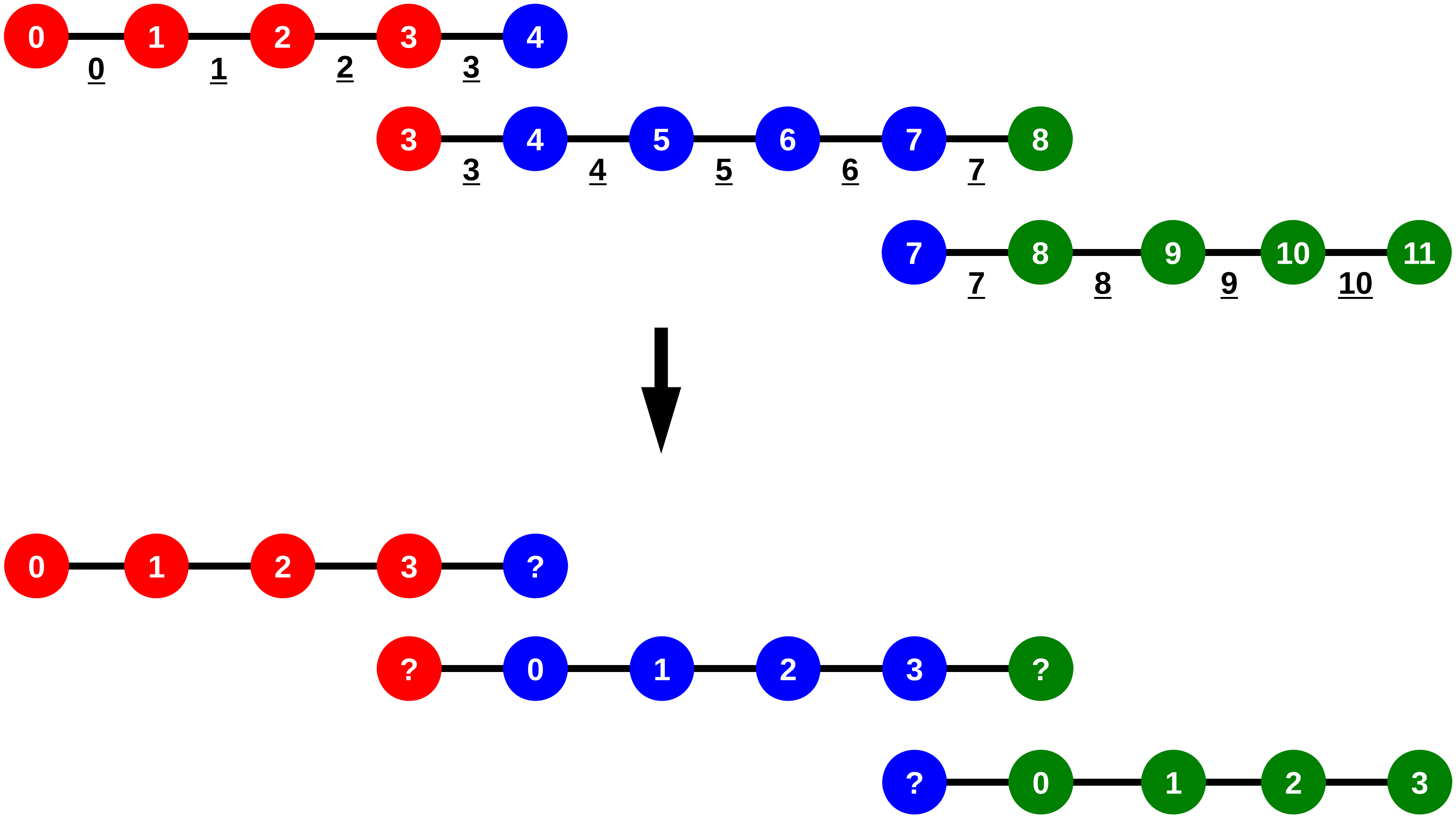


1D FEM: 12 nodes/11 elem's/3 domains



Local Numbering for SPMD

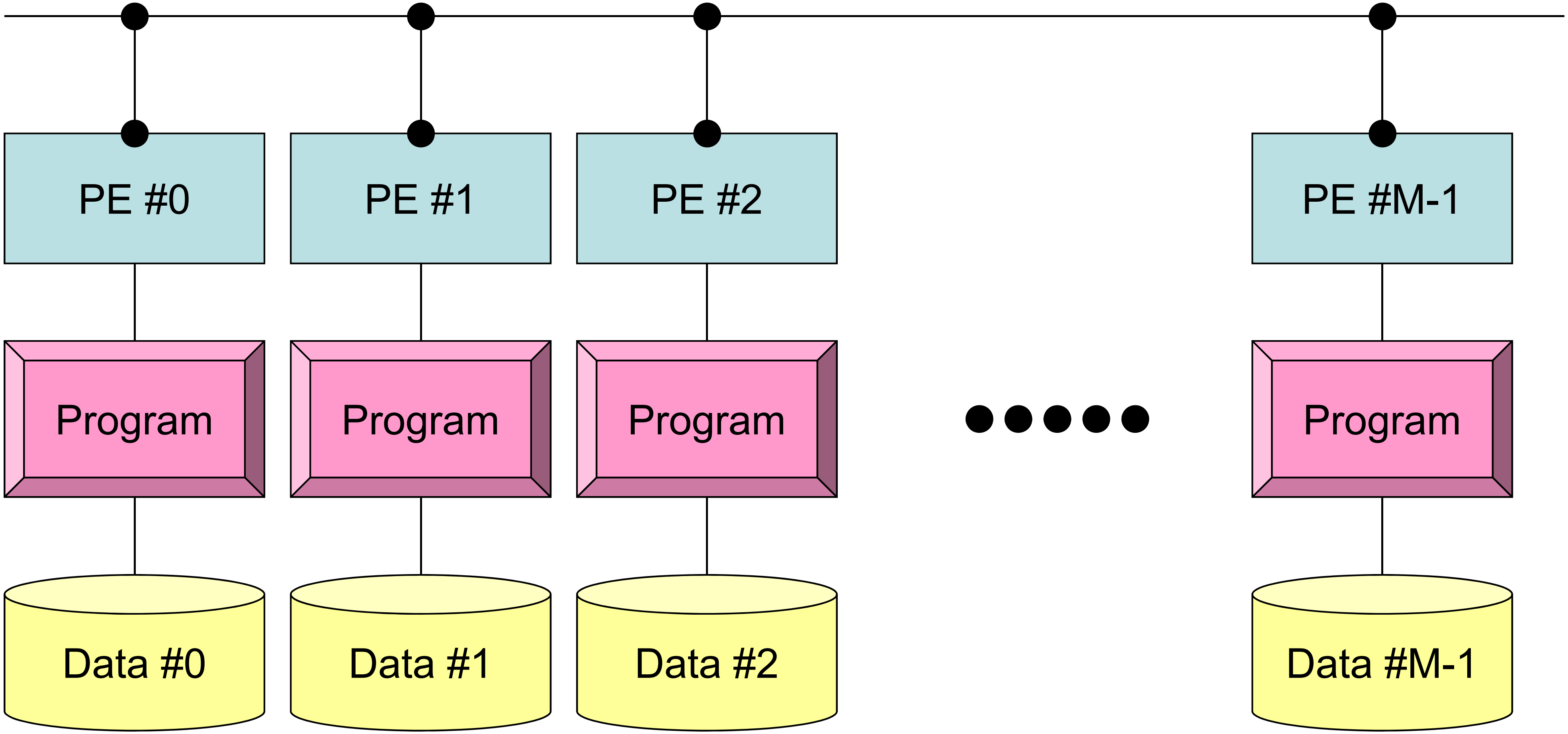
Numbering of internal nodes is 1-N (0-N-1), same operations in serial program can be applied. How about numbering of external nodes ?



PE: Processing Element
Processor, Domain, Process

SPMD

`mpirun -np M <Program>`

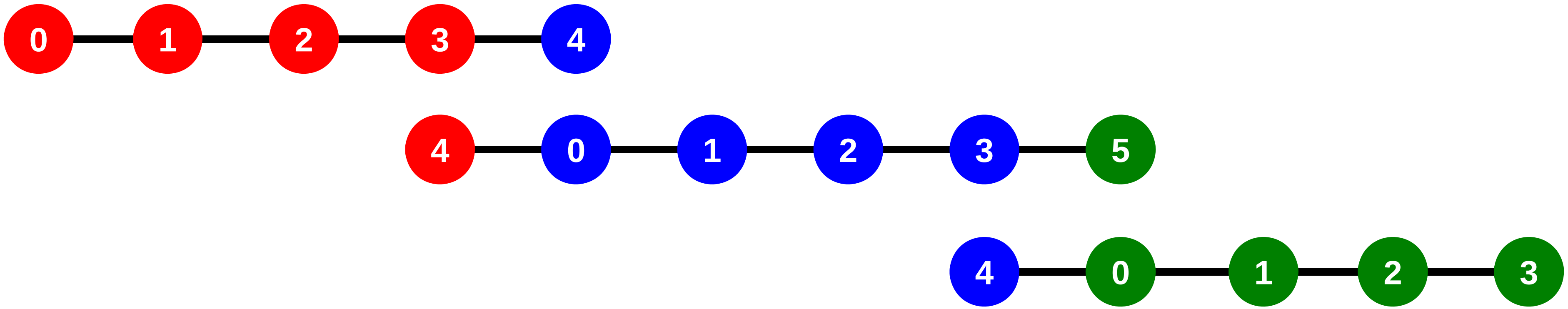


Each process does same operation for different data

Large-scale data is decomposed, and each part is computed by each process
It is ideal that parallel program is not different from serial one except communication.

Local Numbering for SPMD

Numbering of external nodes: N+1, N+2 (N,N+1)



Finite Element Procedures

- Initialization
  - Control Data
  - Node, Connectivity of Elements (N: Node#, NE: Elem#)
  - Initialization of Arrays (Global/Element Matrices)
  - Element-Global Matrix Mapping (Index, Item)
- Generation of Matrix
  - Element-by-Element Operations (do icel= 1, NE)
    - Element matrices
    - Accumulation to global matrix
  - Boundary Conditions
- Linear Solver
  - Conjugate Gradient Method

Preconditioned CG Solver

```
Compute r(0)= b-[A]x(0)
for i= 1, 2, …
    solve [M]z(i-1)= r(i-1)
    ρi-1= r(i-1) z(i-1)
    if i=1
      p(1)= z(0)
     else
      βi-1= ρi-1/ρi-2
      p(i)= z(i-1) + βi-1 p(i-1)
    endif
    q(i)= [A]p(i)
    αi = ρi-1/p(i)q(i)
    x(i)= x(i-1) + αip(i)
    r(i)= r(i-1) - αiq(i)
    check convergence |r|
end
```

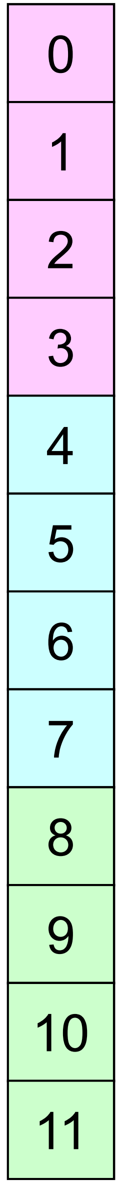
$$[M]=\begin{bmatrix} D_1 & 0 & \cdots & 0 & 0 \\ 0 & D_2 & & 0 & 0 \\ \cdots & & \cdots & & \cdots \\ 0 & 0 & & D_{N-1} & 0 \\ 0 & 0 & \cdots & 0 & D_N \end{bmatrix}$$

Preconditioning, DAXPY

Local Operations by Only Internal Points: Parallel Processing is possible

```
/*
//-- {z}= [Minv]{r}
*/
    for(i=0;i<N;i++){
        W[Z][i] = W[DD][i] * W[R][i];
    }
```

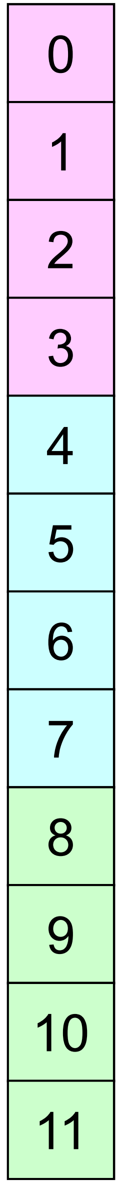
```
/*
//-- {x}= {x} + ALPHA*{p}       DAXPY: double a{x} plus {y}
//     {r}= {r} - ALPHA*{q}
*/
  for(i=0;i<N;i++){
      U[i]    += Alpha * W[P][i];
      W[R][i] -= Alpha * W[Q][i];
  }
```



Dot Products

Global Summation needed: Communication ?

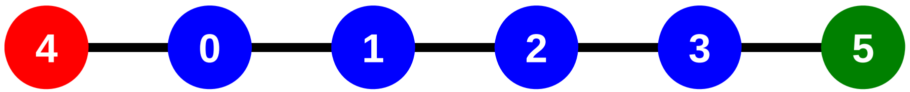
```
/*
//-- ALPHA= RHO / {p}{q}
*/
  C1 = 0.0;
  for(i=0;i<N;i++){
      C1 += W[P][i] * W[Q][i];
  }

  Alpha = Rho / C1;
```



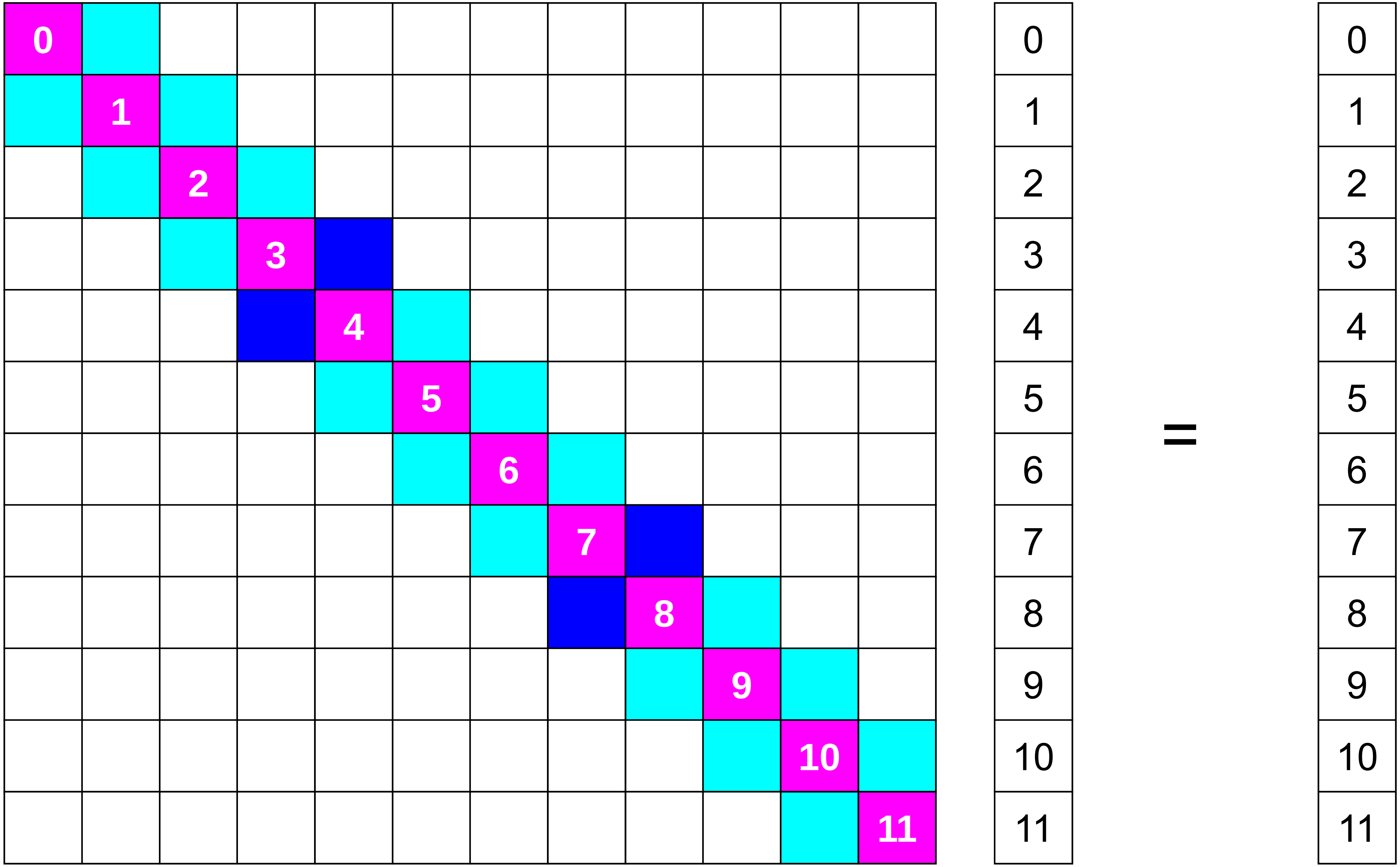
Matrix-Vector Products

Values at External Points: P-to-P Communication

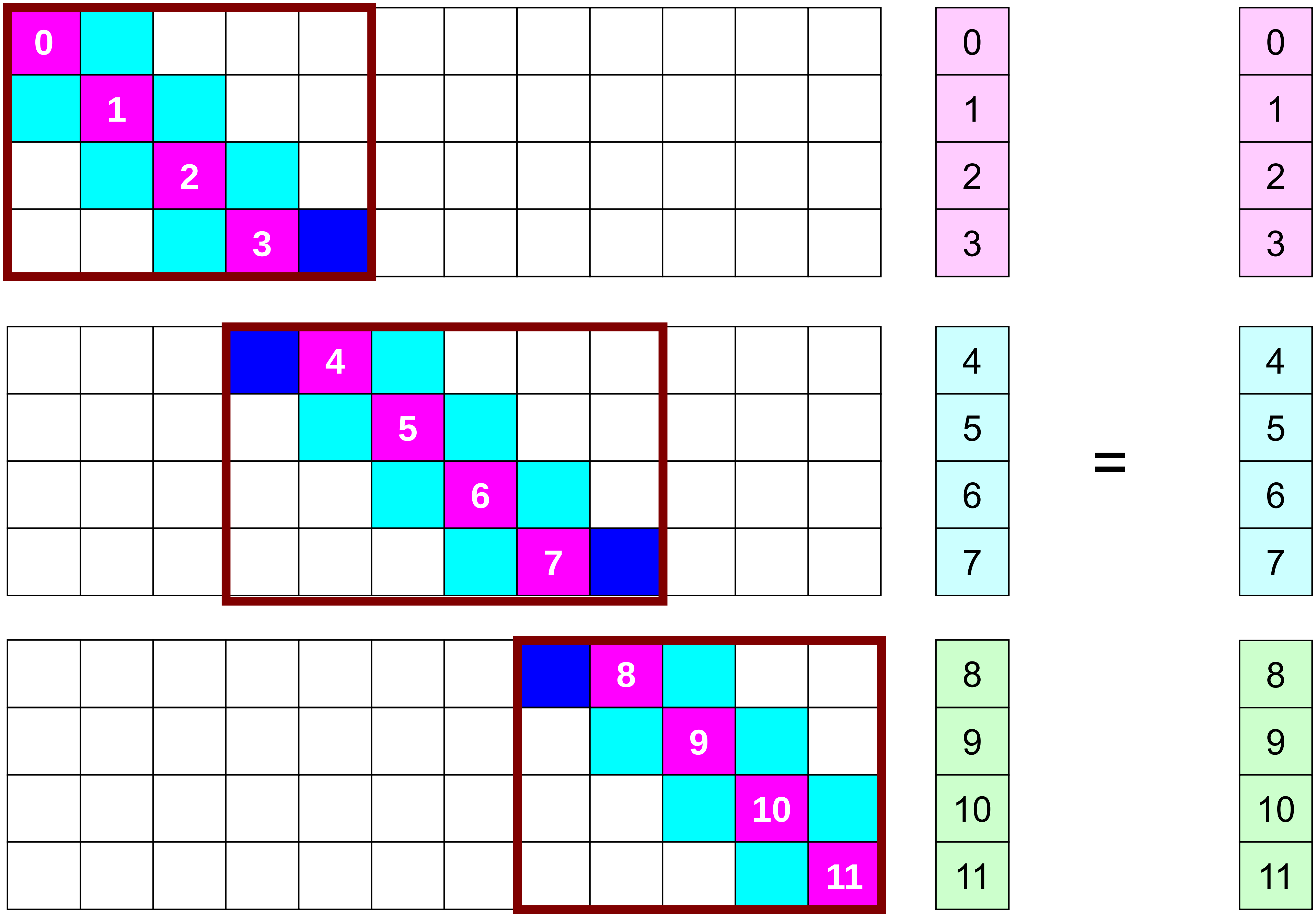
```
/*
//-- {q}= [A]{p}
*/
  for(i=0;i<N;i++){
      W[Q][i] = Diag[i] * W[P][i];
      for(j=Index[i];j<Index[i+1];j++){
          W[Q][i] += AMat[j]*W[P][Item[j]];
          }
      }
```



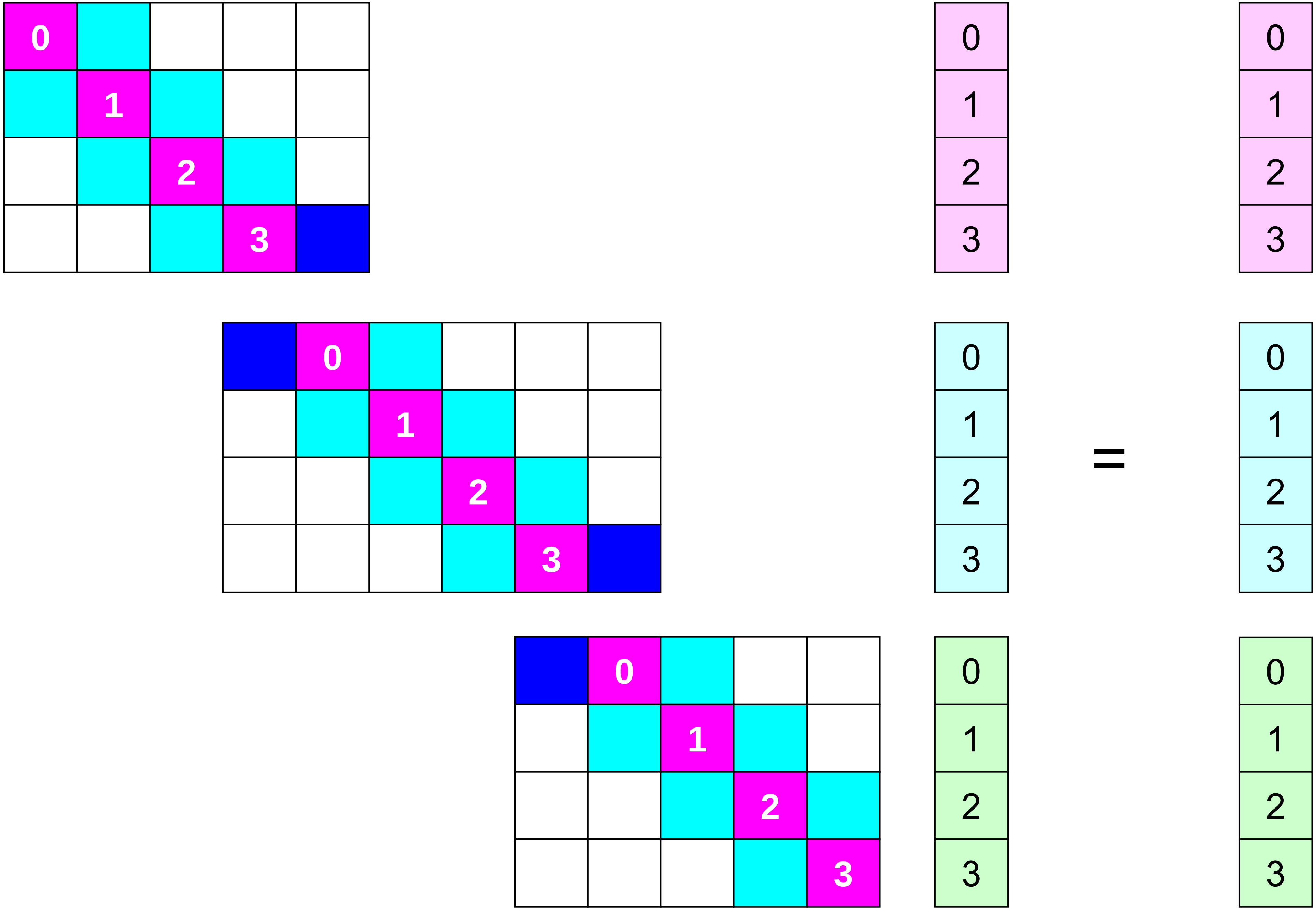
Mat-Vec Products: Local Op. Possible



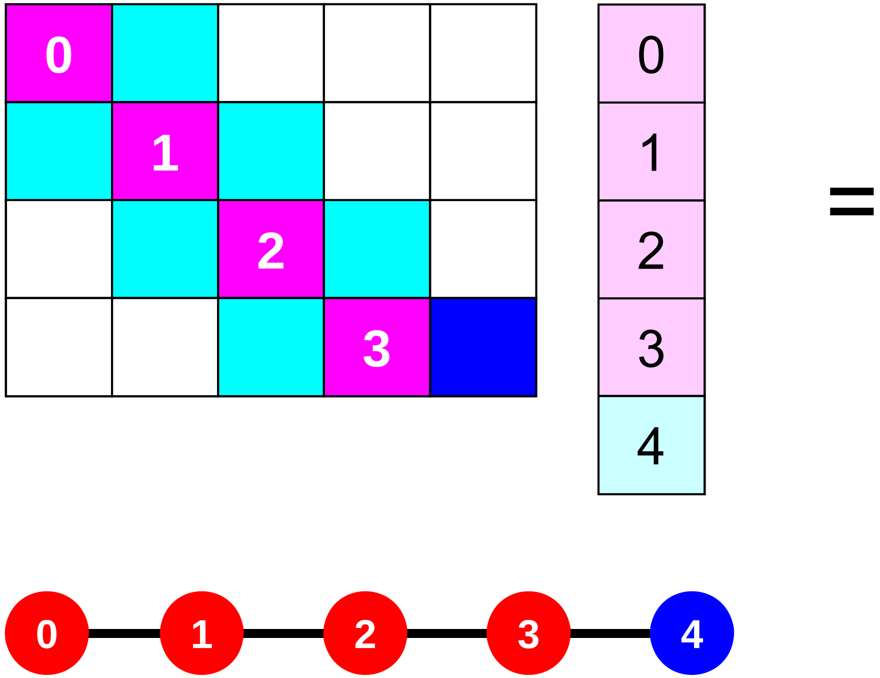
Mat-Vec Products: Local Op. Possible



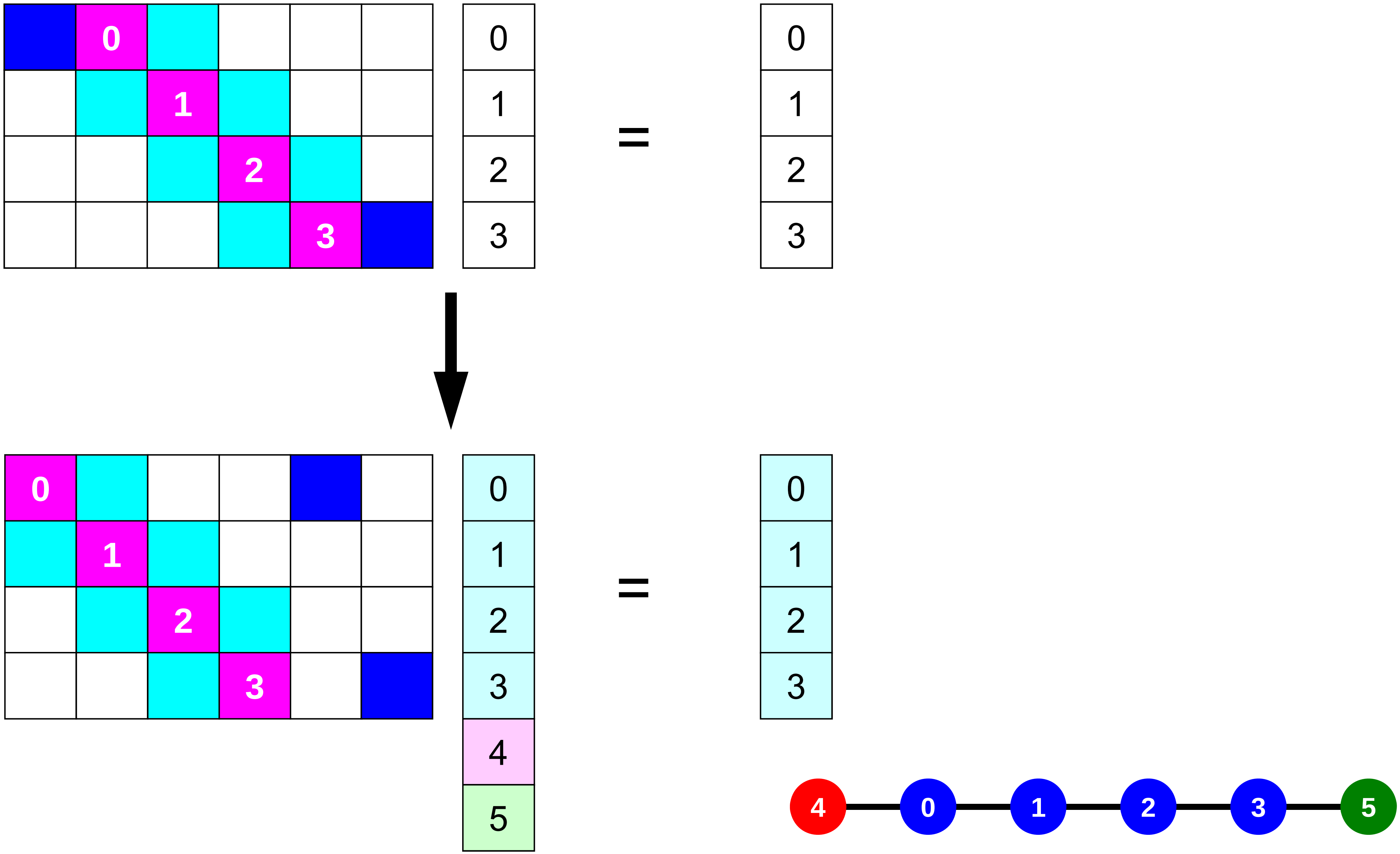
Mat-Vec Products: Local Op. Possible



Mat-Vec Products: Local Op. #0



Mat-Vec Products: Local Op. #1



Mat-Vec Products: Local Op. #2