HEC-MW構造体(1) 分散メッシュ構造体

（PEおよび通信情報）

type hecmwST_local_mesh :: hecMESH

内容）節点・要素・材料情報，PEおよび通信情報
特徴）hecmw_get_mesh によりセット．
HPC-MW ライブラリィのハンドル的役割も

局所データには部分領域間の通信テーブルも含まれている

構造体成分(抜粋)

(2) PEおよび通信情報

```
integer(kind=kint)          :: zero           #0領域か？
integer(kind=kint)          :: my_rank        ランク
integer(kind=kint)          :: PETOT          総領域数
integer(kind=kint)          :: n_subdomain    総領域数（局所分散データからの読み込み）
integer(kind=kint)          :: n_neighbor_pe  隣接領域数
integer(kind=kint),pointer  :: neighbor_pe(:)   隣接領域ID
integer(kind=kint),pointer  :: import_index(:) 受信テーブル用一次元インデックス
integer(kind=kint),pointer  :: import_item(:)   受信テーブル配列
integer(kind=kint),pointer  :: export_index(:) 受信テーブル用一次元インデックス
integer(kind=kint),pointer  :: export_item(:)   受信テーブル配列
integer(kind=kint),pointer  :: shared_index(:) 送受信テーブル用一次元インデックス
integer(kind=kint),pointer  :: shared_item(:)   受信テーブル配列
```

(1) 全体情報 …前出

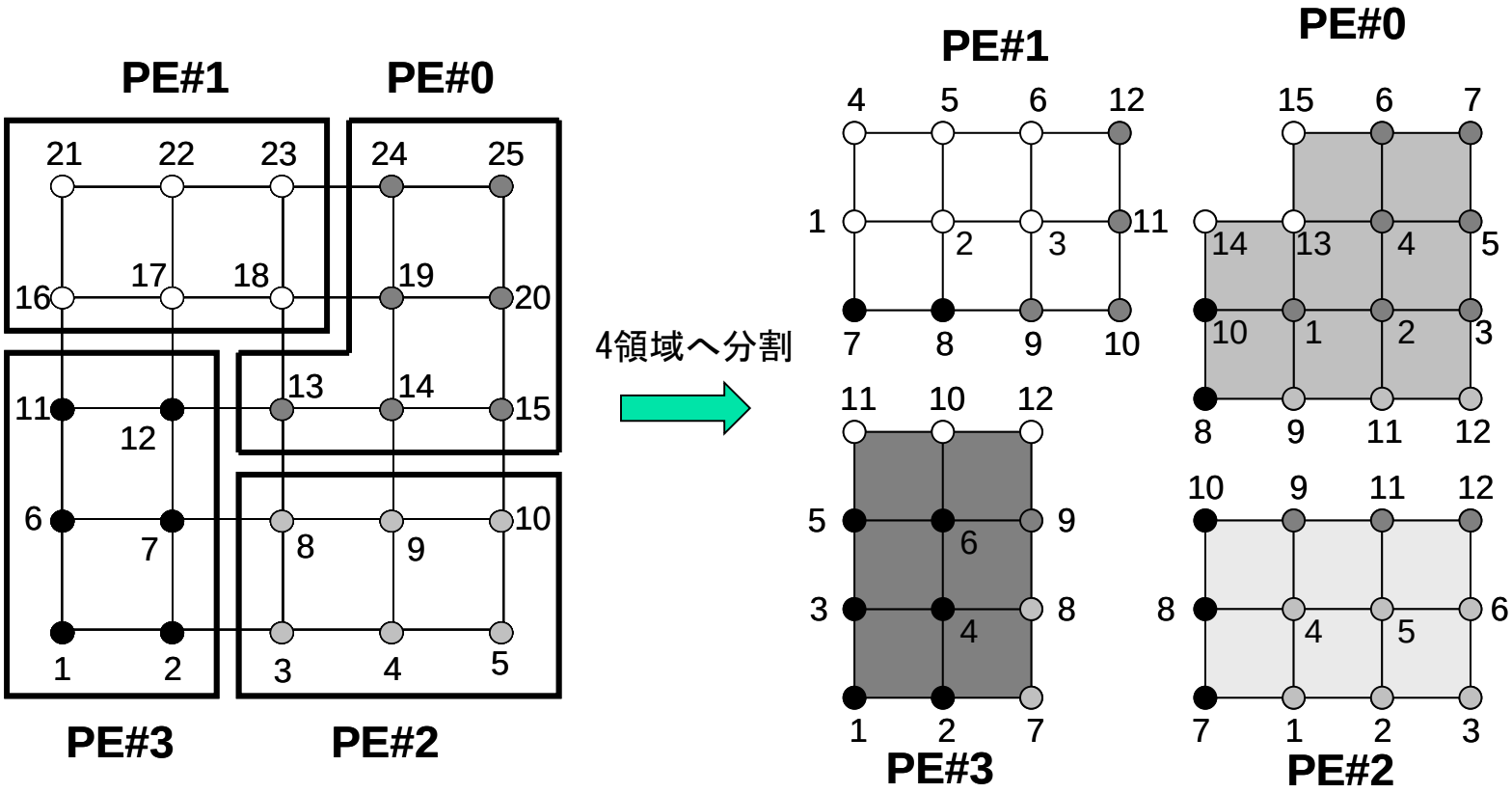
(3) 節点情報 …前出

(4) 要素情報 …前出

(5) 下部構造 …前出

領域間オーバーラップ要素を含む節点ベース領域分割

剛性マトリクスの足し込みなどの処理を**各領域で並列に**実施するためには、
オーバーラップ要素の情報が必要



節点は、通信の観点から以下の3種類に分類される：　　{　} 内は PE#0 の例

・内点（Internal Nodes）：各領域に割り当てられた節点　　{1, 2, 3, 4, 5, 6, 7}

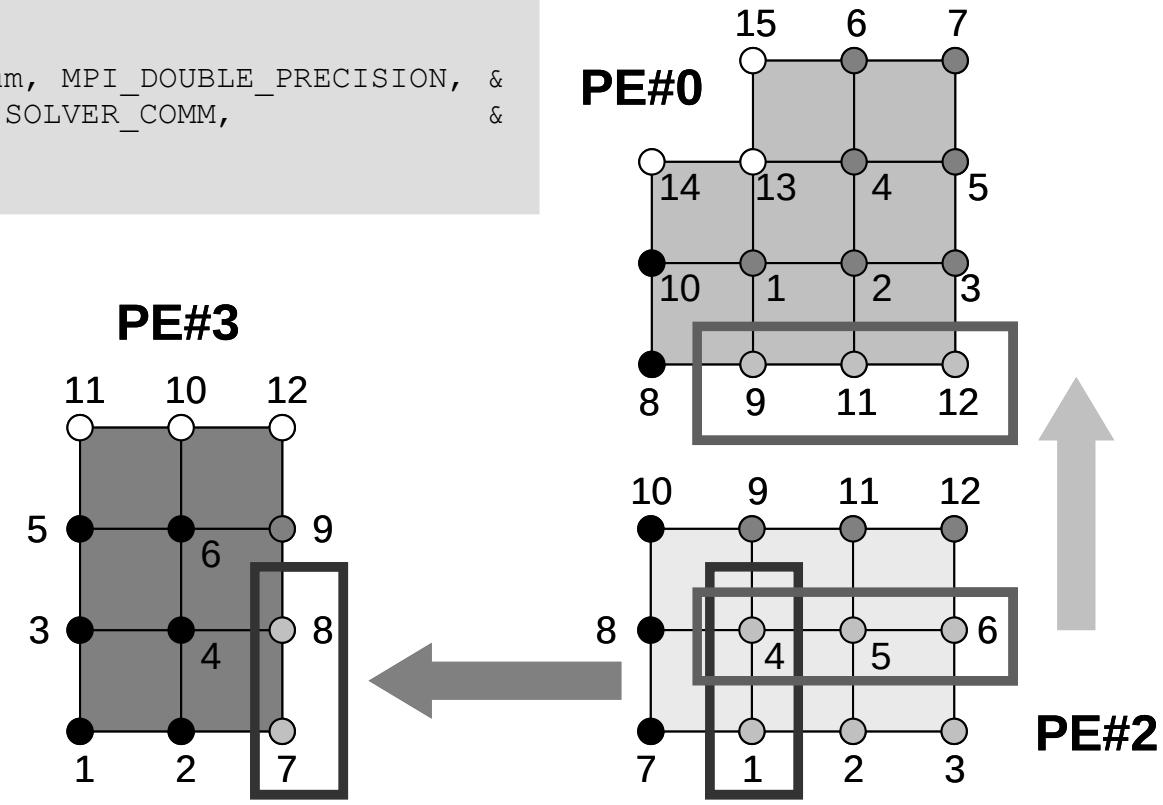
・外点（External Nodes）：他の領域に属しているが、各領域の要素に含まれている節点　　{8. 9. 10, 11, 12, 13, 14, 15}

・境界点（Boundary Nodes）：他の領域の外点となっている内点　　{1, 2, 3, 4, 6}

境界点における値は隣接領域へ「送信（send）」され、送信先では外点として「受信（receive）」される

SEND phase　（PE#2 の例）

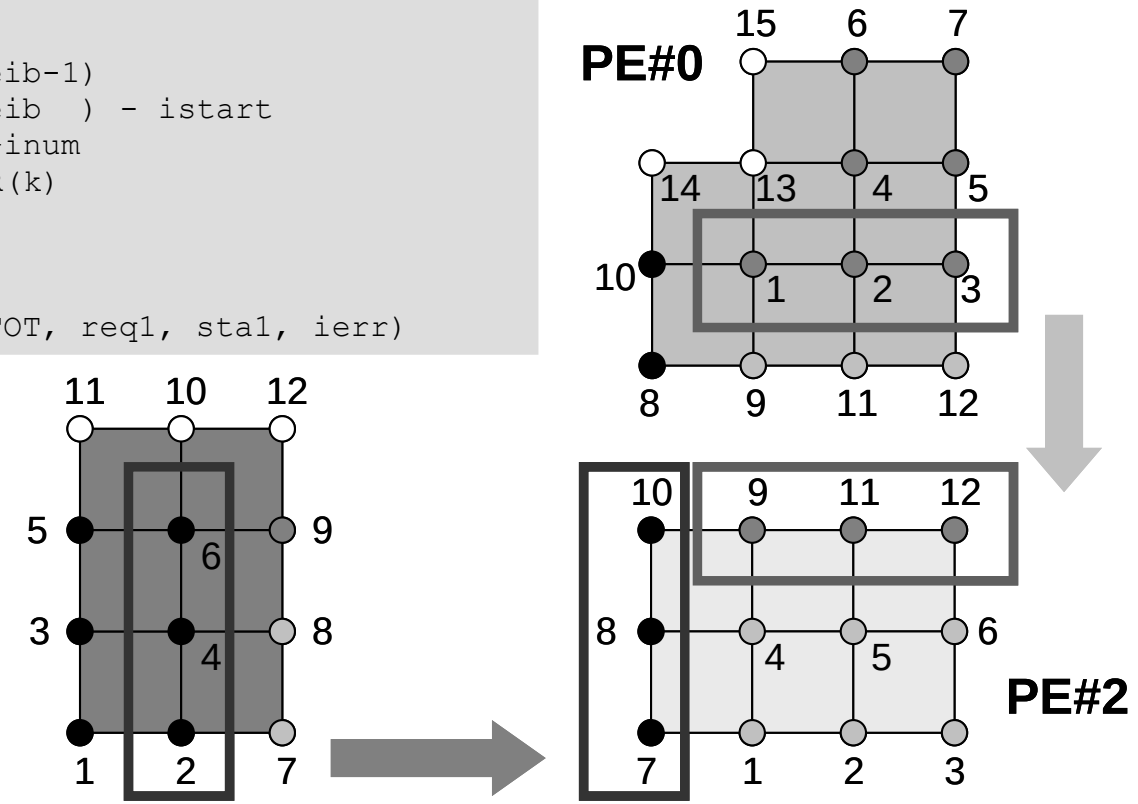
```
do neib= 1, NEIBPETOT
  istart= EXPORT_INDEX(neib-1)
  inum  = EXPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    WS(k)= X(EXPORT_NODE(k))
  enddo
  call MPI_ISEND
       (WS(istart+1), inum, MPI_DOUBLE_PRECISION, &
        NEIBPE(neib), 0, SOLVER_COMM,             &
        req1(neib), ierr)
enddo
```

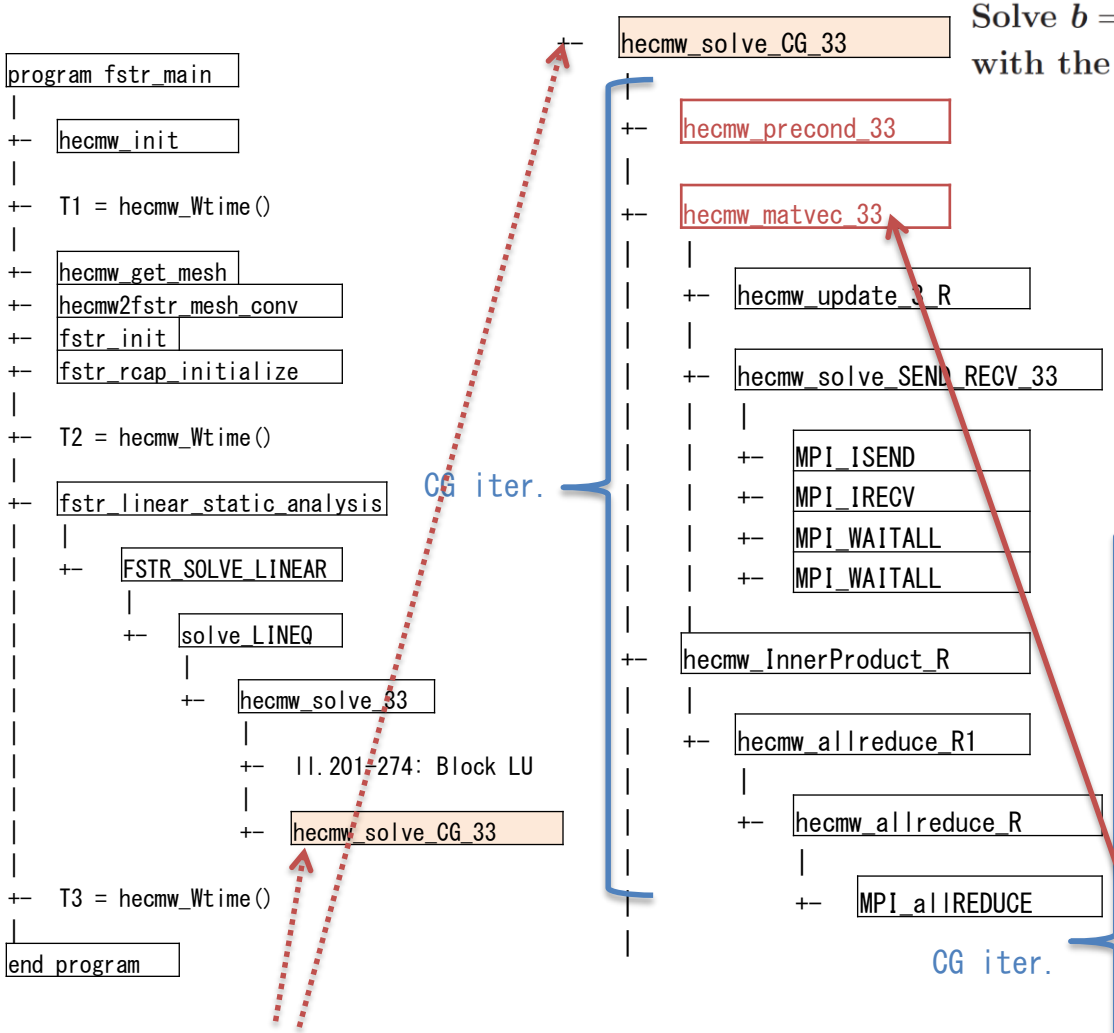


（つづき）

RECEIVE phase　（PE#2 の例）

```
do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  call MPI_IRECV
       (WR(istart+1), inum, MPI_DOUBLE_PRECISION, &
        NEIBPE(neib), 0, SOLVER_COMM,             &
        req2(neib), ierr)
enddo

call MPI_WAITALL (NEIBPETOT, req2, sta2, ierr)

do neib= 1, NEIBPETOT
  istart= IMPORT_INDEX(neib-1)
  inum  = IMPORT_INDEX(neib  ) - istart
  do k= istart+1, istart+inum
    X(IMPORT_NODE(k))= WR(k)
  enddo
enddo

call MPI_WAITALL (NEIBPETOT, req1, sta1, ierr)
```





連立一次方程式の求解（CG法）

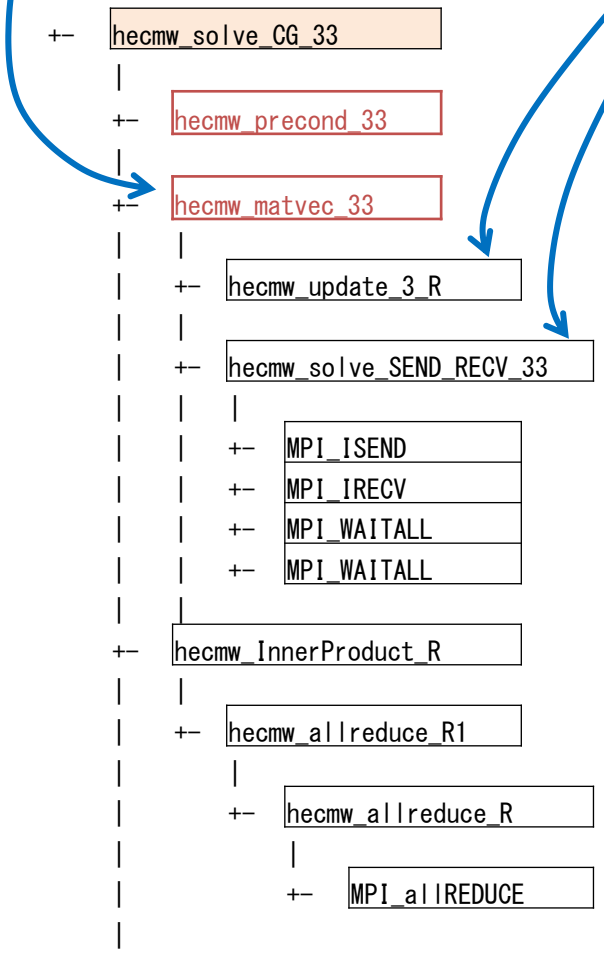
Solve $\boldsymbol{b} = A\boldsymbol{x}$ by the Conjugate Gradient (CG) method with the left preconditioner:

0. Choose the preconditioner M such that $M \simeq A$
   0-a. by Incomplete LU factorization, $M = LU$,
   0-b. by Diagonal block scaling,
   or others
1. Set $\boldsymbol{x} = \boldsymbol{x}_{\rm ini}$: initial guess, and $\boldsymbol{b}$: rhs vector.
2. Set MAXIT: maximum number of iterations, and TOL: tolerance of convergence.
3. $\boldsymbol{r} = \boldsymbol{b} - A\boldsymbol{x}$
4. do iter = 1, MAXIT
5. $\quad \boldsymbol{z} = M^{-1}\boldsymbol{r}$
6. $\quad \rho = (\boldsymbol{r}, \boldsymbol{z})$
7. $\quad$ if (iter .eq. 1) then
8. $\quad\quad \boldsymbol{p} = \boldsymbol{z}$
9. $\quad$ else
10. $\quad\quad \beta = \rho/\rho_1$
11. $\quad\quad \boldsymbol{p} = \boldsymbol{z} + \beta\boldsymbol{p}$
12. $\quad$ endif
13. $\quad \boldsymbol{q} = A\boldsymbol{p}$ 行列ベクトル積
14. $\quad \alpha = \rho/(\boldsymbol{p}, \boldsymbol{q})$
15. $\quad \boldsymbol{x} = \boldsymbol{x} + \alpha\boldsymbol{p}, \quad \boldsymbol{r} = \boldsymbol{r} - \alpha\boldsymbol{q}$
16. $\quad$ Compute the scaled residual norm, RESID = ||r||/||b||
17. $\quad$ if (RESID .le. TOL) exit do-loop
18. $\quad \rho_1 = \rho$
19. enddo

FrontISTRプログラムの構造　-- 並列CG法

疎行列・ベクトル積

hecmw_matvec の中では、前出の通信を行ったのちに、部分領域ごとに並列に以下の行列・ベクトル積を実行する

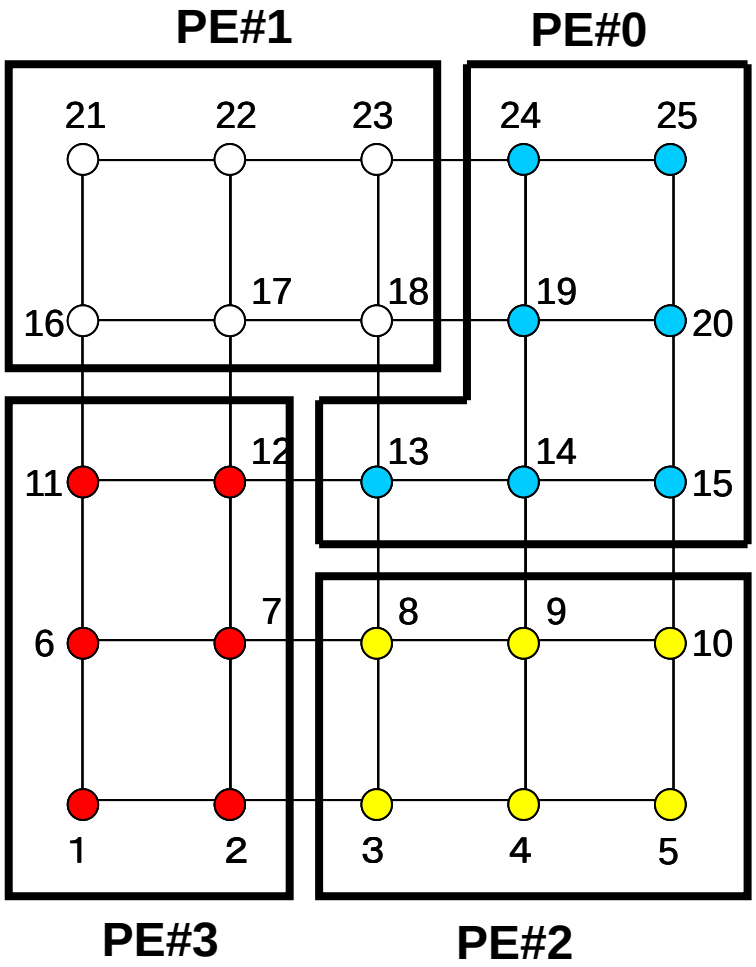


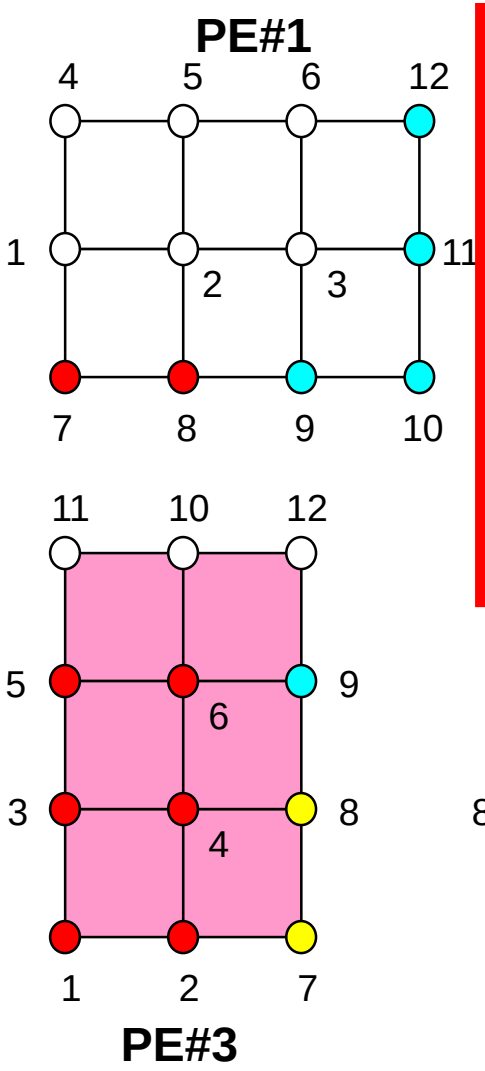

```
do i= 1, N
  isL= INL(i-1) + 1
  ieL= INL(i  )
  WVAL= WW(i,R)
  do j= isL, ieL
    inod = IAL(j)
    WVAL=  WVAL -  AL(j) * WW(inod,Z)
  enddo
  WW(i,Z)= WVAL * DD(i)
enddo

do i= N, 1, -1
  SW = 0.0d0
  isU= INU(i-1) + 1
  ieU= INU(i  )
  do j= isU, ieU
    inod = IAU(j)
    SW= SW + AU(j) * WW(inod,Z)
  enddo
  WW(i,Z)= WW(i,Z) - DD(i) * SW
enddo
```

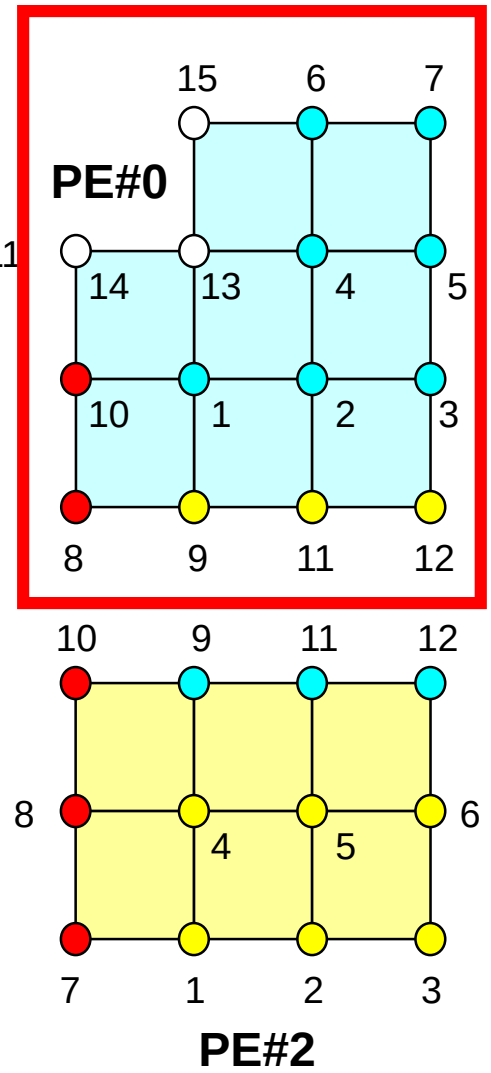
行列・ベクトル積

Local Data Structure

Node-based Partitioning
internal nodes - elements - external nodes

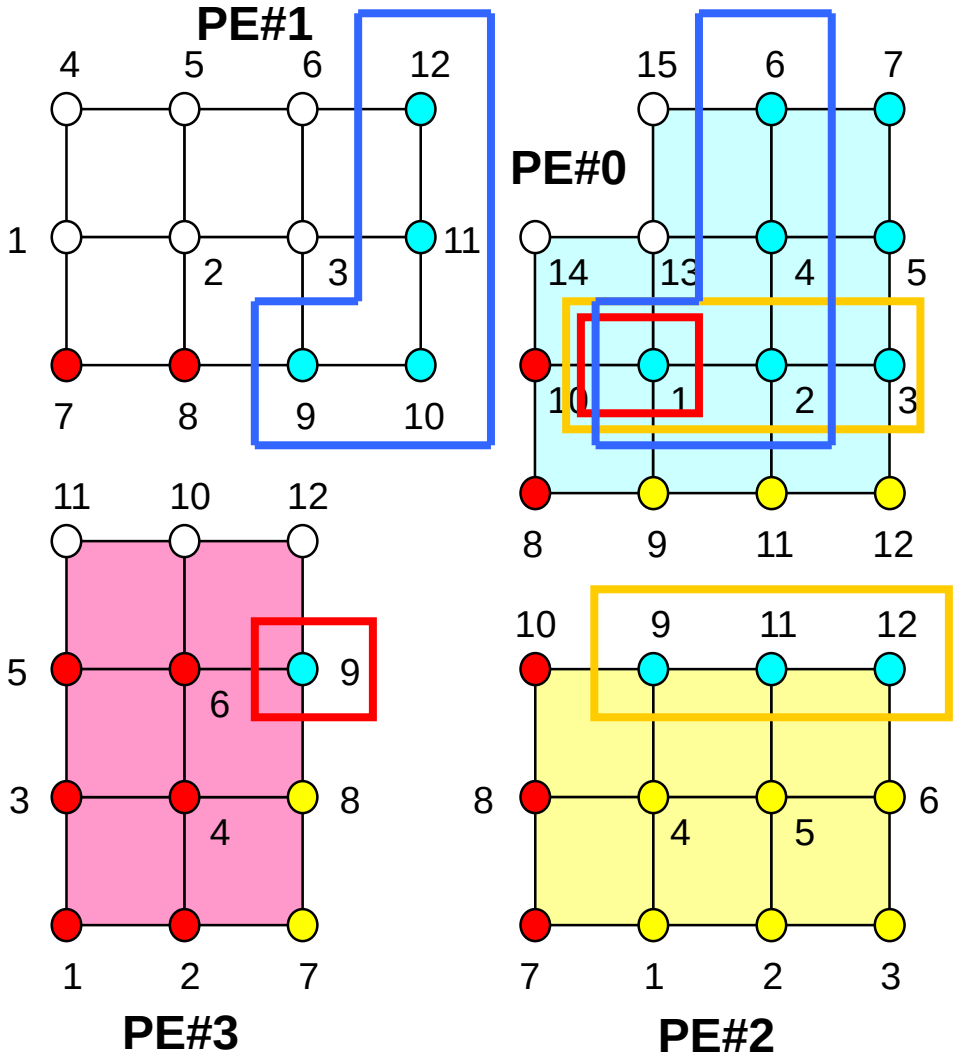






Local Data Structure : PE#0

internal nodes - elements - external nodes



Partitioned nodes themselves
internal nodes

Elements which include “internal nodes”
Provide data locality in order to carry out element-by-element operation in each partition

Nodes included in the elements
external nodes
Numbering : internal -> external

Internal nodes which are “external nodes” for other partitions
boundary nodes

Communication table provides boundary~external node relationship