並列アプリケーション開発法入門（IV）
粒子間熱伝導問題解析コード並列化

2007年7月18日

中島研吾

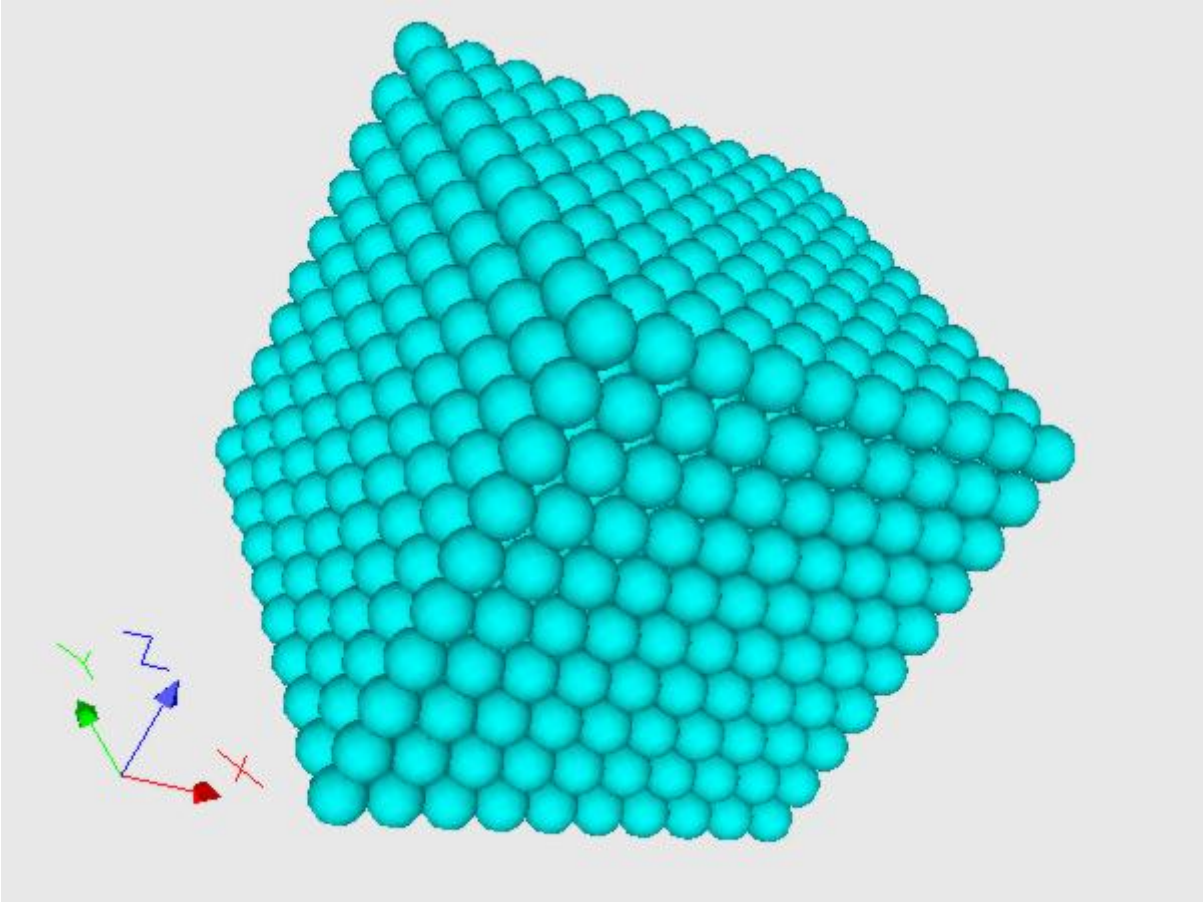
並列計算プログラミング（616-2057）・先端計算機演習I（616-4009）

概　要

- 問題設定
- 1CPU用プログラムの解説
- 並列プログラムの解説

問題設定(1/5)

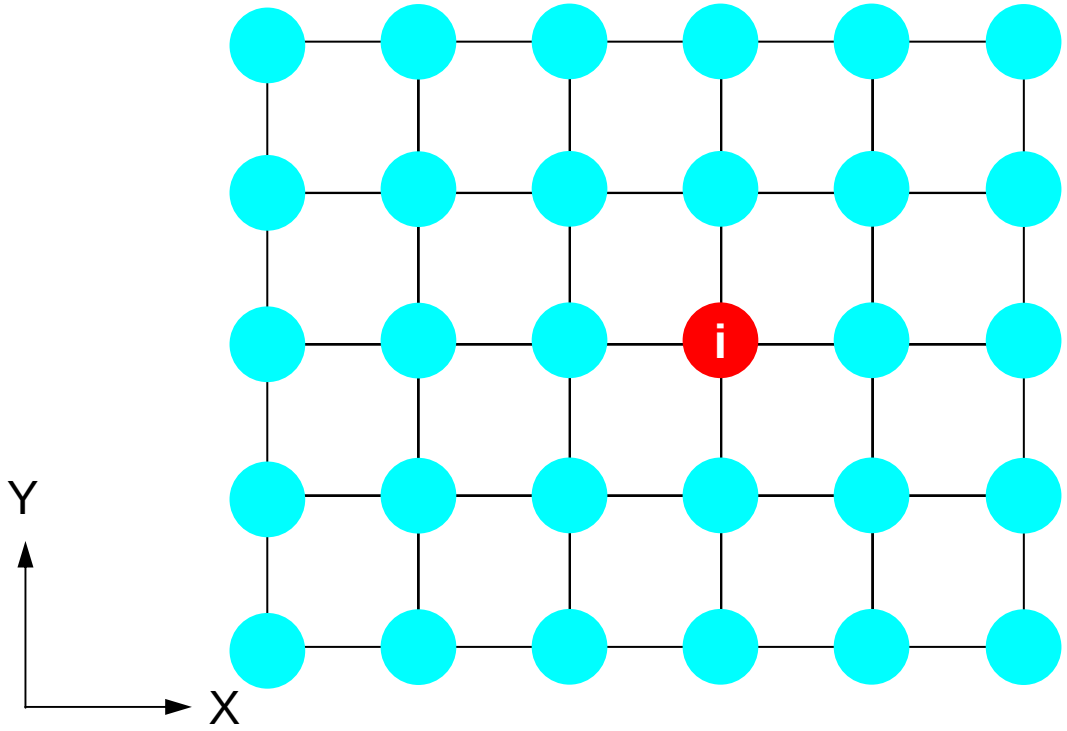
- 空間上に規則正しく，X,Y,Z方向にNX,NY,NZ個ずつ等間隔(DX,DY,DZ)に配置された粒子の集合体を考える。



問題設定(2/5)

粒子iに関する熱流束の/釣り合い

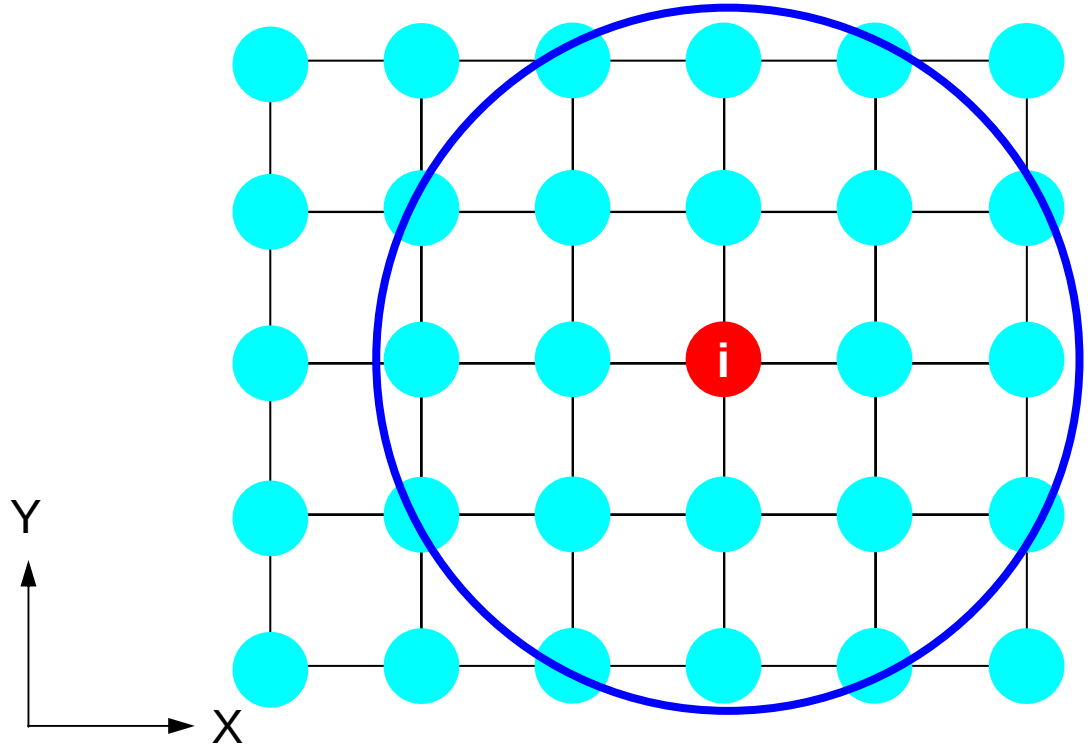
$$\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}}\left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$



問題設定(3/5)

粒子間熱伝導

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}}\left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$

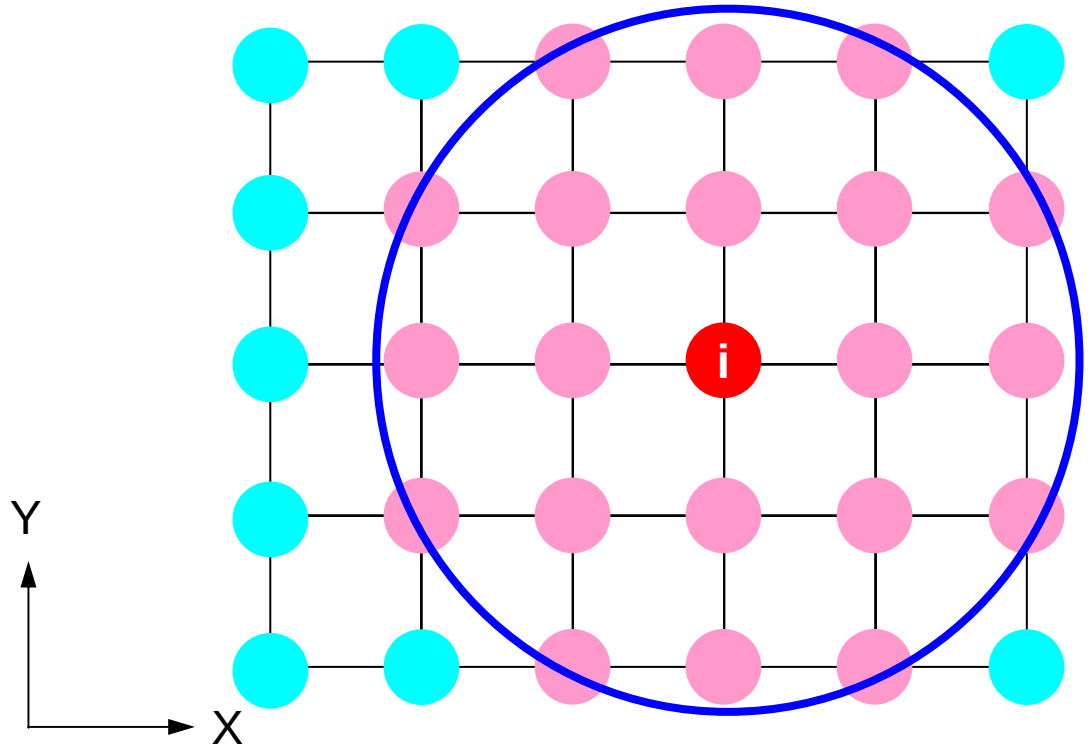


粒子iを中心とする,
半径$RADI_{max}$の球(円)

問題設定(3/5)

粒子間熱伝導

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}}\left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$



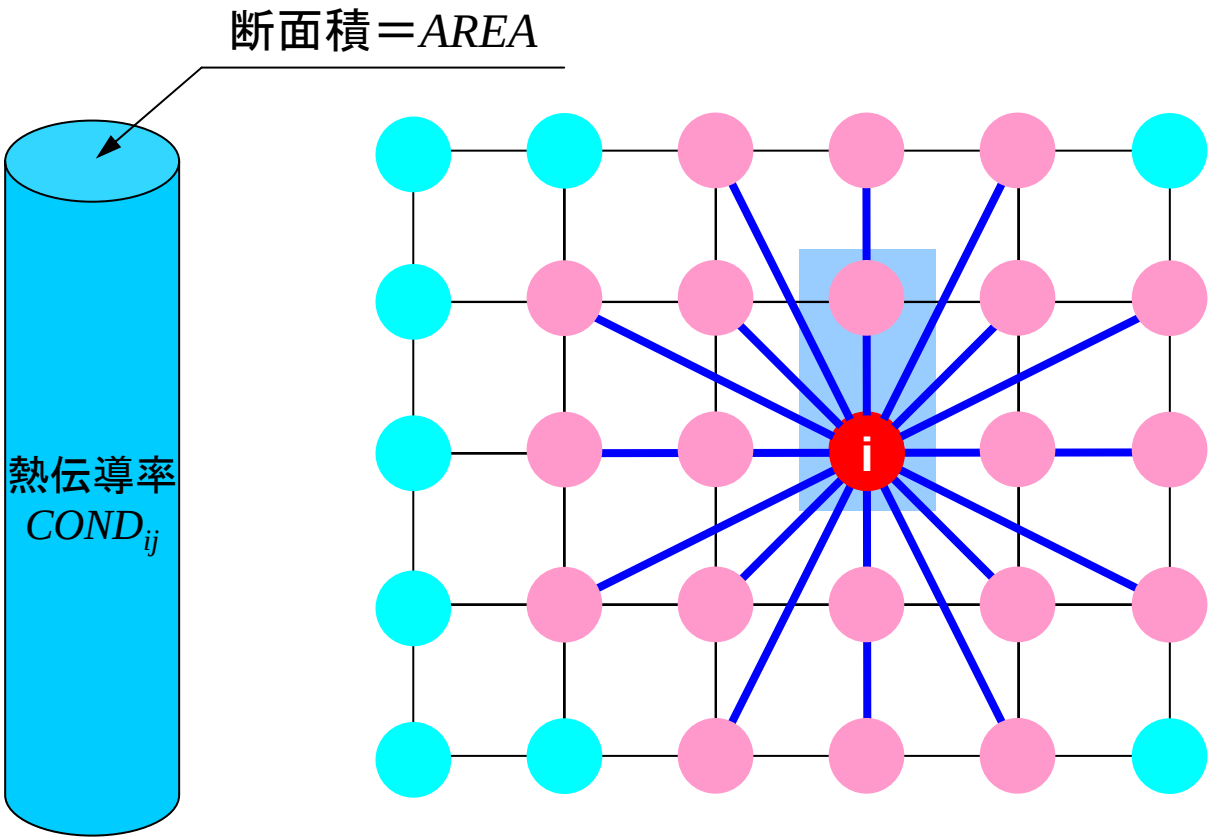
粒子iを中心とする,
半径$RADI_{max}$の球(円)

粒子間の距離DEL_{ij}が$RADI_{max}$のよりも小さい粒子群とのみ熱伝導がある。

問題設定(3/5)

粒子間熱伝導

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}} \left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$



粒子iを中心とする，
半径$RADI_{max}$の球(円)

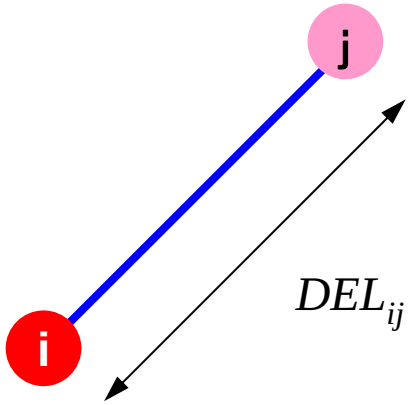
粒子間の距離DEL_{ij}が
$RADI_{max}$のよりも小さい
粒子群とのみ熱伝導が
ある。

長さDEL，断面積$AREA$，
熱伝導率$COND_{ij}$の管で
連結

熱伝導率 : $COND_{ij}$

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}} \left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$

$$COND_{ij} = \frac{COND0}{\min\left(10^{DEL}, 10^{20}\right)}$$

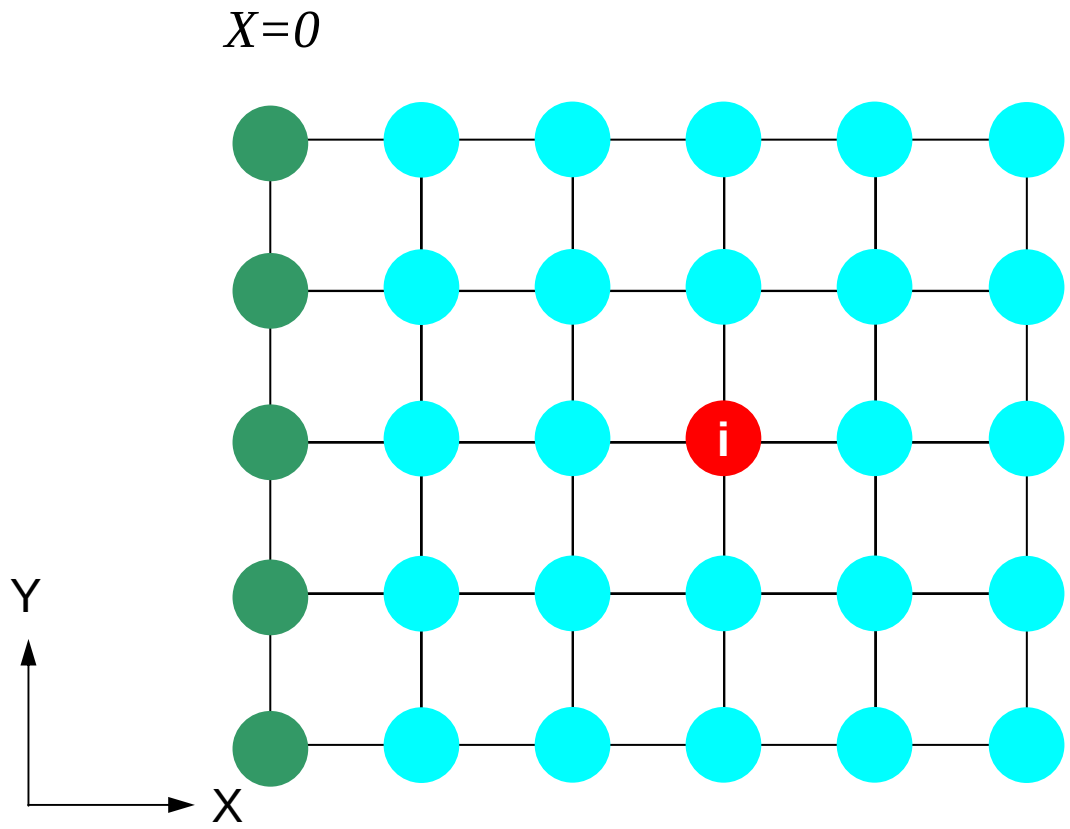


COND0 「基準」熱伝導率

問題設定(4/5)

対流熱伝達

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}} \left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$



対流熱伝達率
$HCONV_i$
$= HCONV$ *if* $X=0$
$= 0$ *otherwise*

熱交換面積
$SURF$

雰囲気温度
T_0

問題設定(5/5)

体積発熱

$$\sum_{i}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij} / COND_{ij}} \left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$

体積発熱量　$QVOL$

発熱体積　$\overline{V}$

$$\overline{V} = C_1 \cdot VOL + C_2 \cdot VOL \cdot DELQ$$

$$DELQ = \sqrt{X_i^{\,2} + Y_i^{\,2} + Z_i^{\,2}}$$

C_1, C_2　定数
VOL　各粒子の「基準」体積
$DELQ$　粒子中心と原点との距離

粒子iに関するつりあい

$$\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}}\left(T_j - T_i\right) + HCONV_i \cdot SURF \cdot \left(T_0 - T_i\right) + QVOL \cdot \overline{V} = 0$$

$$\sum_{j}^{DEL<RADI_{\max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j\right] - \left[\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}}\right] T_i$$

$$+ HCONV_i \cdot SURF \cdot T_0 - HCONV_i \cdot SURF \cdot T_i + QVOL \cdot \overline{V} = 0$$

$$\left[-\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}} - HCONV_i \cdot SURF\right] T_i + \sum_{j}^{DEL<RADI_{\max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j\right]$$

AMAT(i,i)（対角成分）

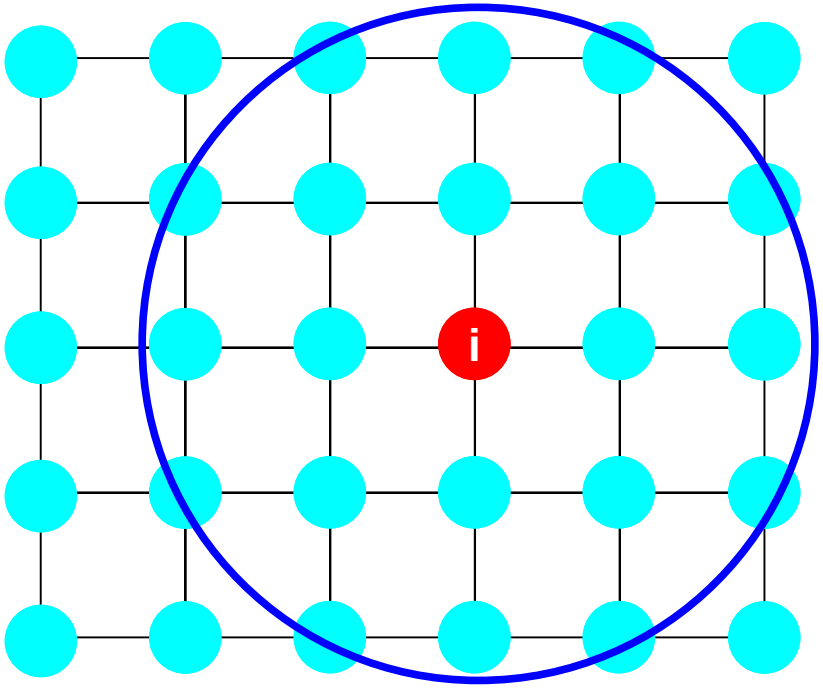
AMAT(i,j)（非対角成分）

$$= -HCONV_i \cdot SURF \cdot T_0 - QVOL \cdot \overline{V}$$

RHS（右辺）

係数行列 AMAT

- 影響範囲 $RADI_{max}$ が小さければ，係数行列は疎
  - 局所的な効果
- 影響範囲が大きければ，係数行列は密
  - 非ゼロ成分の割合が大きくなる



疎行列と密行列：{q}=[A]{p}

疎行列：差分法，有限要素法，有限体積法

```
do i= 1, N
  q(i)= D(i)*p(i)
  do k= index(i-1)+1, index(i)
    q(i)= q(i) + AMATs(k)*p(item(k))
  enddo
enddo
```

密行列：スペクトル法，分子動力学，境界要素法

```
do i= 1, N
  q(i)= 0.d0
  do j= 1, N
    q(i)= q(i) + AMATd(i,j)*p(j)
  enddo
enddo
```

係数行列 AMAT

- 影響範囲 $RADI_{max}$ が小さければ，係数行列は疎
  - 局所的な効果
- 影響範囲が大きければ，係数行列は密
  - 非ゼロ成分の割合が大きくなる

- 影響範囲が大きくなる場合のことを考えて，プログラムの中では係数行列を「密（AMATd(i,j)）」として扱う。

- 問題設定
- **1CPU用プログラムの解説**
- 並列プログラムの解説

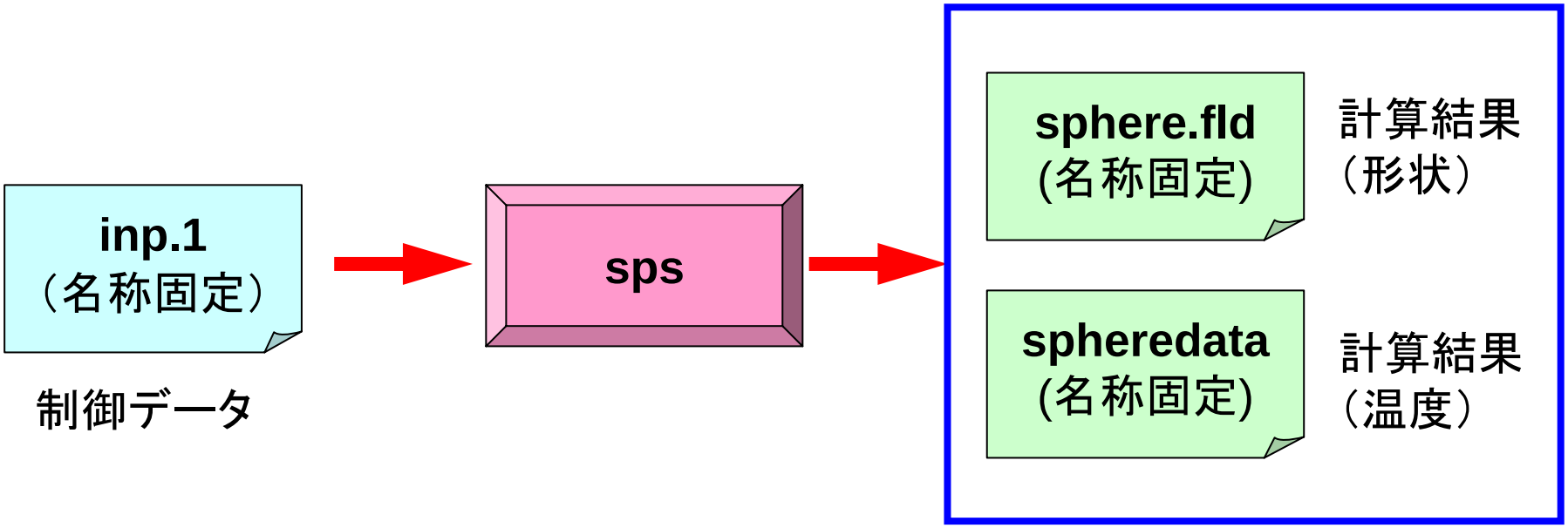
ファイル

```
$> cp /home/nakajima/class/2007summer/sphere.tar .
$> tar xvf sphere.tar
$> ls
      sphere
$> cd sphere        ⇒ <$SPHERE>
$> ls
      C  F
$> cd C
$> ls
     inp1   test.c    testp.c    testp2.c

$> cd ../F
$> ls
     inp1   test.f    testp.f    testp2.f
```

コンパイル，実行

```
$> cd <$SPHERE>/C
$> pgcc -O3 test.c -o sps
$> ./sps
$> ls -l sphere*
   sphere.fld   spheredata

$> cd <$SPHERE>/F
$> pgf90 -O3 test.f -o sps
$> ./sps
$> ls -l sphere*
   sphere.fld   spheredata
```

sps: ファイル，データ


inp.1
（名称固定）
制御データ
sps
sphere.fld
(名称固定)
計算結果
（形状）
spheredata
(名称固定)
計算結果
（温度）

制御データ:inp1

```
10 10 10                        NX, NY, NZ
1.e0 1.e0 1.e0                  DX, DY, DZ
1.e0 10.e0 1.e0 1.e0            VOL, AREA, QVOL, COND0
1.e24 0.e0 10.e0                HCONV, T0, SURF
8.0e0                           RADImax
1.0e0  0.1e0                    C1, C2
```

計算結果（形状）：sphere.fld

*.fldの形式であれば名称は任意

```
# AVS field file                    必須
ndim=    3                         三次元モデルであることを示す
dim1=   10                         NX
dim2=   10                         NY
dim3=   10                         NZ
nspace=    3
veclen=    1
data= float
field= uniform
label= temperature
variable 1 file=./spheredata filetype=ascii   温度ファイル名
```

計算結果（温度）：spheredata

計算結果（形状）からの参照が正しければ名称は任意

```
 1.188402E-24
 5.388751E+00
 9.485117E+00
 1.311638E+01
 1.630137E+01
…
```

```
open (22, file='spheredata', status='unknown')
do i= 1, NX*XY*NZ
  write (22,'(1pe16.6)') PHI(i)
enddo
```

シリアル版の計算手順

- 制御データ入力
- 粒子生成
- 係数マトリクス計算
  - 熱伝導
  - 対流熱伝達
  - 発熱
- CG法による連立一次方程式求解
  - 密行列
  - 点ヤコビ前処理
- 出力
  - MicroAVS用

test.f：シリアル版（1/10）

初期設定

```
      implicit REAL*8 (A-H,O-Z)

      real(kind=8) :: VOL, AREA, QVOL, COND0, COND, RADImax
      real(kind=8) :: HCONV, T0, SURF, DEL, DEL0, coef1, coef2
      real(kind=8), dimension(:)  , allocatable :: XC, YC, ZC
      real(kind=8), dimension(:,:), allocatable :: AMAT
      real(kind=8), dimension(:)  , allocatable :: RHS
      real(kind=8), dimension(:)  , allocatable :: PHI
      real(kind=8), dimension(:,:), allocatable :: W

      integer :: NX, NY, NZ, N
      integer :: R, Z, P, Q, DD
!C
!C +------+
!C | INIT |
!C +------+
!C===
      open (11, file= 'inp1', status='unknown')
        read (11,*) NX, NY, NZ
        read (11,*) DX, DY, DZ
        read (11,*) VOL, AREA, QVOL, COND0
        read (11,*) HCONV, T0, SURF
        read (11,*) RADImax
        read (11,*) coef1, coef2
      close (11)

      N= NX*NY*NZ
```

test.f：シリアル版（2/10）
粒子中心の座標（XC,YC,ZC）

```
      allocate (XC(N), YC(N), ZC(N))

      icou= 0
      do k= 1, NZ
        do j= 1, NY
          do i= 1, NX
            icou= icou + 1
            XC(icou)= dfloat(i-1)*DX
            YC(icou)= dfloat(j-1)*DY
            ZC(icou)= dfloat(k-1)*DZ
          enddo
        enddo
      enddo
!C===
```

粒子の通し番号 ii （1~N）

```
ii=(k-1)*NX*NY + (j-1)*NX + i
```

test.f：シリアル版（3/10）

マトリクス生成：熱伝導部分

```
!C
!C +--------+
!C | MATRIX |
!C +--------+
!C===
      allocate (AMAT(N,N), RHS(N))

      AMAT= 0.d0
      RHS = 0.d0
      do i= 1, N
        do j= 1, N
          if (j.ne.i) then
            DEL= dsqrt((XC(i)-XC(j))**2 + (YC(i)-YC(j))**2                &
     &                                 + (ZC(i)-ZC(j))**2)
            if (DEL.le.RADImax) then
              COND= COND0/(10.d0**dmin1(DEL,20.d0))
              coef= COND*AREA / DEL
              AMAT(i,j)= coef
              AMAT(i,i)= AMAT(i,i) - coef
            endif
          endif
        enddo
      enddo
```

$$\left[-\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}} - HCONV_i \cdot SURF\right] T_i +$$

$$\sum_{j}^{DEL<RADI_{\max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j\right]$$

$$= -HCONV_i \cdot SURF \cdot T_0 - QVOL \cdot \overline{V}$$

$$COND_{ij} = \frac{COND0}{\min\left(10^{DEL}, 10^{20}\right)}$$

test.f：シリアル版（3/10）

マトリクス生成：熱伝導部分

```
!C
!C +--------+
!C | MATRIX |
!C +--------+
!C===
      allocate (AMAT(N,N), RHS(N))

      AMAT= 0.d0
      RHS = 0.d0
      do i= 1, N
        do j= 1, N
          if (j.ne.i) then
            DEL= dsqrt((XC(i)-XC(j))**2 + (YC(i)-YC(j))**2                &
     &                                 + (ZC(i)-ZC(j))**2)
            if (DEL.le.RADImax) then
              COND= COND0/(10.d0**dmin1(DEL,20.d0))
              coef= COND*AREA / DEL
              AMAT(i,j)= coef
              AMAT(i,i)= AMAT(i,i) - coef   対角項
            endif
          endif
        enddo
      enddo
```

$$\left[-\sum_{j}^{DEL<RADI_{max}} \frac{AREA}{DEL_{ij}/COND_{ij}} - HCONV_i \cdot SURF\right] T_i + \sum_{j}^{DEL<RADI_{max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j\right] = -HCONV_i \cdot SURF \cdot T_0 - QVOL \cdot \overline{V}$$

test.f：シリアル版（4/10）

マトリクス生成：体積発熱

```
      do i= 1, N
        DELQ= dsqrt(XC(i)**2 + YC(i)**2 + ZC(i)**2)
        RHS(i)= -QVOL*(coef1*VOL + coef2*DELQ*VOL)
      enddo

      i= 1
      do k= 1, NZ
      do j= 1, NY
        ic= (k-1)*NX*NY + (j-1)*NX + i
        AMAT(ic,ic)= -HCONV*SURF    + AMAT(ic,ic)
        RHS (ic   )= -HCONV*SURF*T0 + RHS (ic)
      enddo
      enddo
!C===
```

$$\left[-\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}} - HCONV_i \cdot SURF \right] T_i + \sum_{j}^{DEL<RADI_{\max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j \right] = -HCONV_i \cdot SURF \cdot T_0 - QVOL \cdot \bar{V}$$

$$\bar{V} = C_1 \cdot VOL + C_2 \cdot VOL \cdot DELQ$$

$$DELQ = \sqrt{X_i^2 + Y_i^2 + Z_i^2}$$

test.f：シリアル版（4/10）

マトリクス生成：対流熱伝達

```
      do i= 1, N
        DELQ= dsqrt(XC(i)**2 + YC(i)**2 + ZC(i)**2)
        RHS(i)= -QVOL*(coef1*VOL + coef2*DELQ*VOL)
      enddo

      i= 1
      do k= 1, NZ
      do j= 1, NY
        ic= (k-1)*NX*NY + (j-1)*NX + i
        AMAT(ic,ic)= -HCONV*SURF    + AMAT(ic,ic)
        RHS (ic   )= -HCONV*SURF*T0 + RHS (ic)
      enddo
      enddo
!C===
```

$$\left[-\sum_{j}^{DEL<RADI_{\max}} \frac{AREA}{DEL_{ij}/COND_{ij}} - HCONV_i \cdot SURF \right] T_i +$$

$$\sum_{j}^{DEL<RADI_{\max}} \left[\frac{AREA}{DEL_{ij}/COND_{ij}} T_j \right]$$

$$= -HCONV_i \cdot SURF \cdot T_0 - QVOL \cdot \overline{V}$$

前処理付き共役勾配法
Preconditioned Conjugate Gradient Method（CG）

Compute $\mathbf{r}^{(0)}= \mathbf{b}-[A]\mathbf{x}^{(0)}$
for i= 1, 2, …
  solve $[M]\mathbf{z}^{(i-1)}= \mathbf{r}^{(i-1)}$
  $\rho_{i-1}= \mathbf{r}^{(i-1)}\ \mathbf{z}^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $\mathbf{q}^{(i)}= [A]\mathbf{p}^{(i)}$
  $\alpha_i = \rho_{i-1}/\mathbf{p}^{(i)}\mathbf{q}^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence $|\mathbf{r}|$
end

前処理：対角スケーリング

$x^{(i)}$ ：ベクトル
α_i ：スカラー

test.f：シリアル版（5/10）

CG法①

```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      EPS= 1.d-08
      allocate (W(N,4), PHI(N))
      W   = 0.d0
      PHI= 0.d0

      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4
      do i= 1, N
        W(i,DD)= 1.0D0 / AMAT(i,i)
      enddo
```

W(i,1)= W(i,R) ⇒ {r}
W(i,2)= W(i,Z) ⇒ {z}
W(i,2)= W(i,Q) ⇒ {q}
W(i,3)= W(i,P) ⇒ {p}
W(i,4)= W(i,DD) ⇒ 1/DIAG

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

test.f：シリアル版（6/10）
CG法②

```
!C
!C-- {r0}= {b} - [A]{xini} |

      do i= 1, N
        W(i,R) = RHS(i)
        do j= 1, N
          W(i,R) = W(i,R) - AMAT(i,j)*PHI(j)
        enddo
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i)**2
      enddo

!C***************************************
      do iter= 1, N
!C
!C-- {z}= [Minv]{r}

      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)} z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1} p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

test.f：シリアル版（7/10）
CG法③

```
!C
!C-- RHO= {r}{z}

      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo

!C
!C-- {p} = {z} if       ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.eq.1 ) then
        do i= 1, N
          W(i,P)= W(i,Z)
        enddo
       else
         BETA= RHO / RHO1
         do i= 1, N
           W(i,P)= W(i,Z) + BETA*W(i,P)
         enddo
      endif
```

Compute $r^{(0)}= b-[A]x^{(0)}$

for i= 1, 2, …

  solve $[M]z^{(i-1)}= r^{(i-1)}$

  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$

  if i=1

    $p^{(1)}= z^{(0)}$

  else

    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$

    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$

  endif

  $q^{(i)}= [A]p^{(i)}$

  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$

  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$

  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$

  check convergence $|r|$

end

test.f：シリアル版（8/10）
CG法④

```
!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q) = 0.d0
        do j= 1, N
          W(i,Q) = W(i,Q) + AMAT(i,j)*W(j,P)
        enddo
      enddo

!C
!C-- ALPHA= RHO / {p}{q}

      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1

!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}

      do i= 1, N
        PHI(i)   = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

test.f：シリアル版（9/10）
CG法⑤

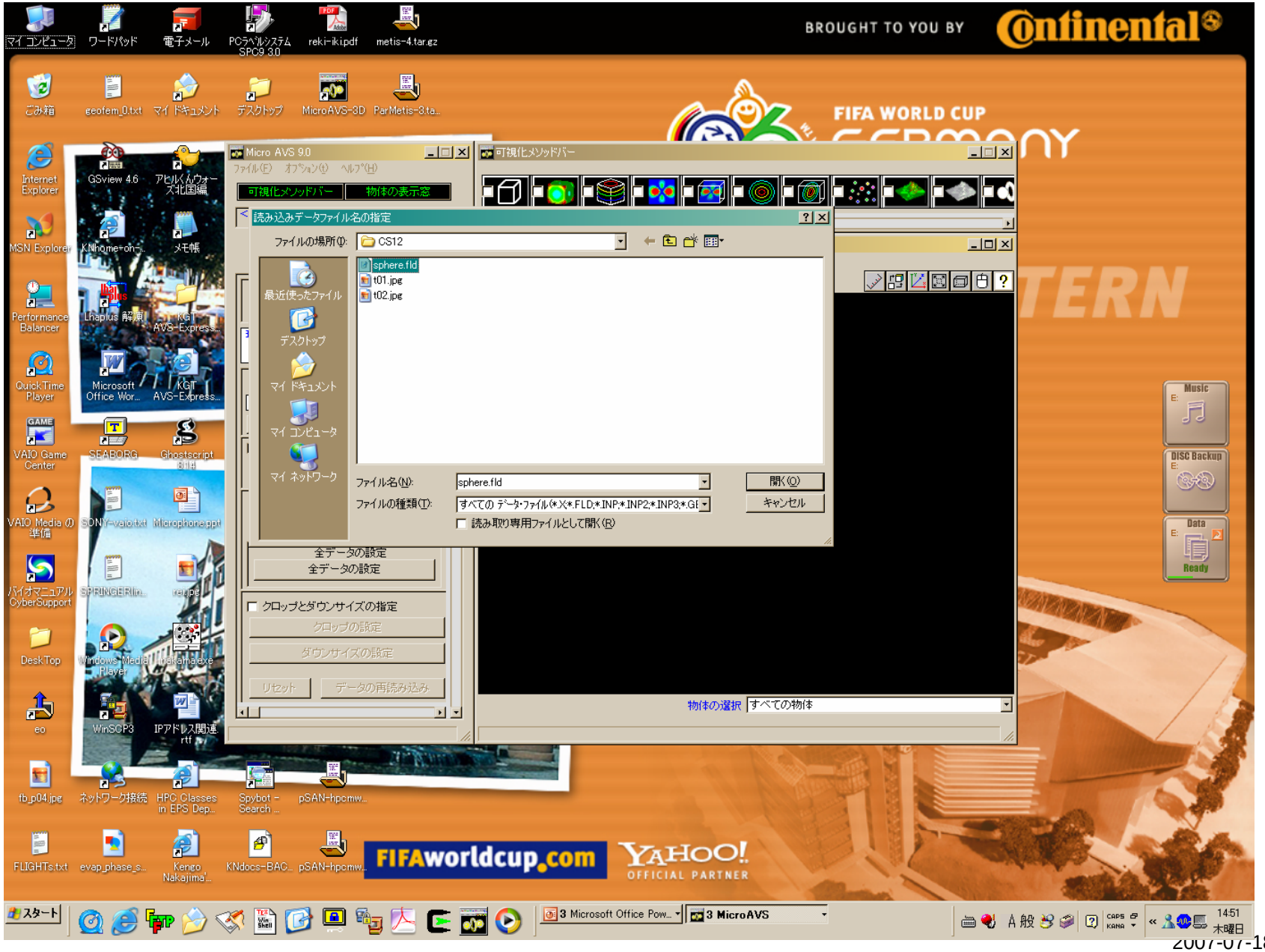
```
      DNRM2 = 0.0
      do i= 1, N
        DNRM2= DNRM2 + W(i,R)**2
      enddo

        RESID= dsqrt(DNRM2/BNRM2)

        write (*, 1000) iter, RESID
 1000   format (i5, 1pe16.6)

        if ( RESID.le.EPS) goto 900
        RHO1 = RHO

      enddo
!C***************************************
      IER = 1

  900 continue
!C===
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

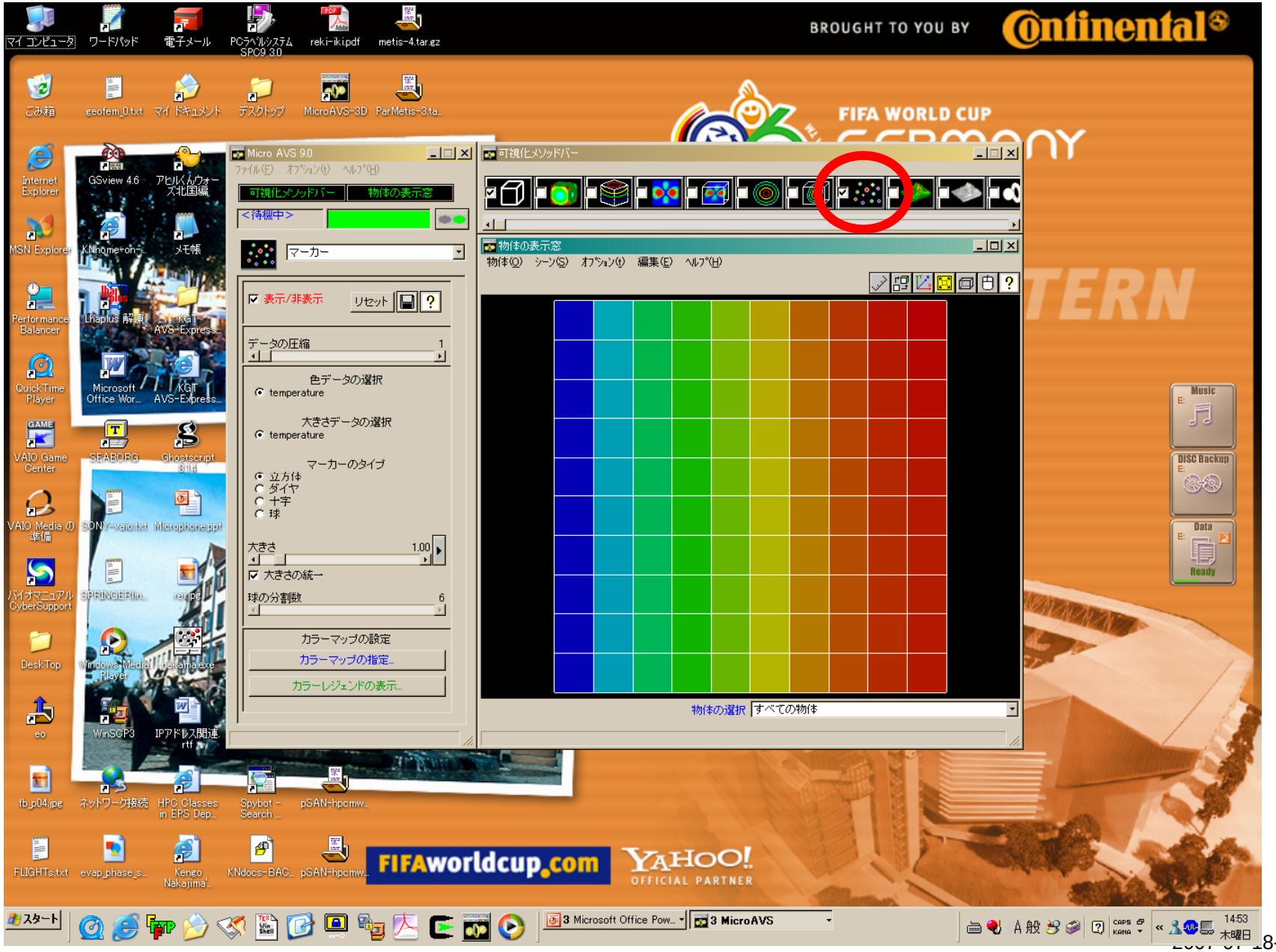
test.f:シリアル版（10/10）
結果出力

```
!C
!C +--------+
!C | OUTPUT |
!C +--------+
!C===
      N1= 1
      N3= 3
!C
!C-- MESH
      open (22, file='sphere.fld', status='unknown')
      write (22,'(a)')    '# AVS field file'
      write (22,'(a,i5)') 'ndim=', N3
      write (22,'(a,i5)') 'dim1=', NX
      write (22,'(a,i5)') 'dim2=', NY
      write (22,'(a,i5)') 'dim3=', NZ
      write (22,'(a,i5)') 'nspace=', N3
      write (22,'(a,i5)') 'veclen=', N1
      write (22,'(a,i5)') 'data= float'
      write (22,'(a,i5)') 'field= uniform'
      write (22,'(a,i5)') 'label= temperature'
      write (22,'(a,i5)') 'variable 1 file=./spheredata filetype=ascii'
      close (22)
!C
!C-- RESULTS
      open (22, file='spheredata', status='unknown')
      do i= 1, N
        write (22,'(1pe16.6)') PHI(i)
      enddo
!C===
```

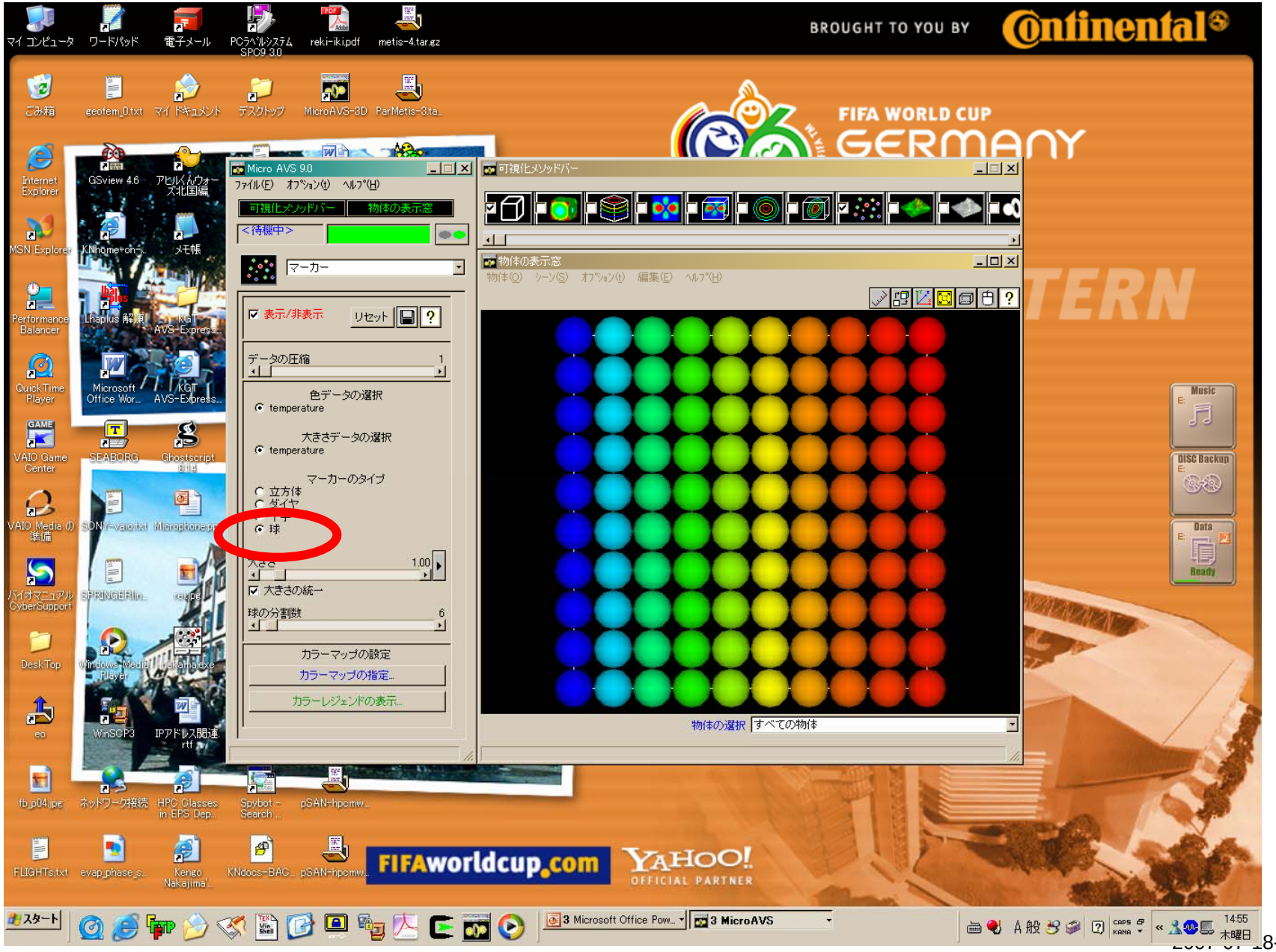
MicroAVSによる処理（1/4）



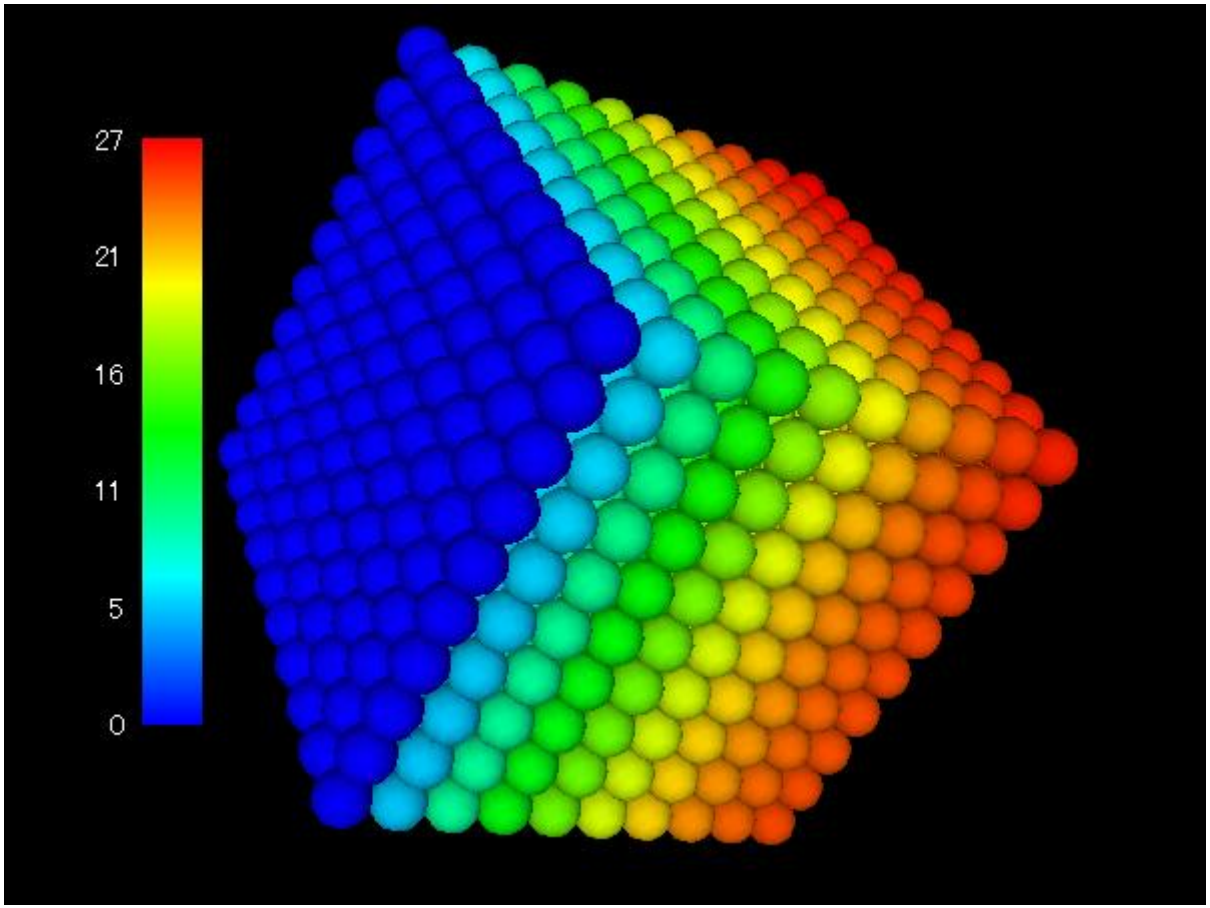
MicroAVSによる処理(2/4)



MicroAVSによる処理(3/4)



MicroAVSによる処理（4/4）

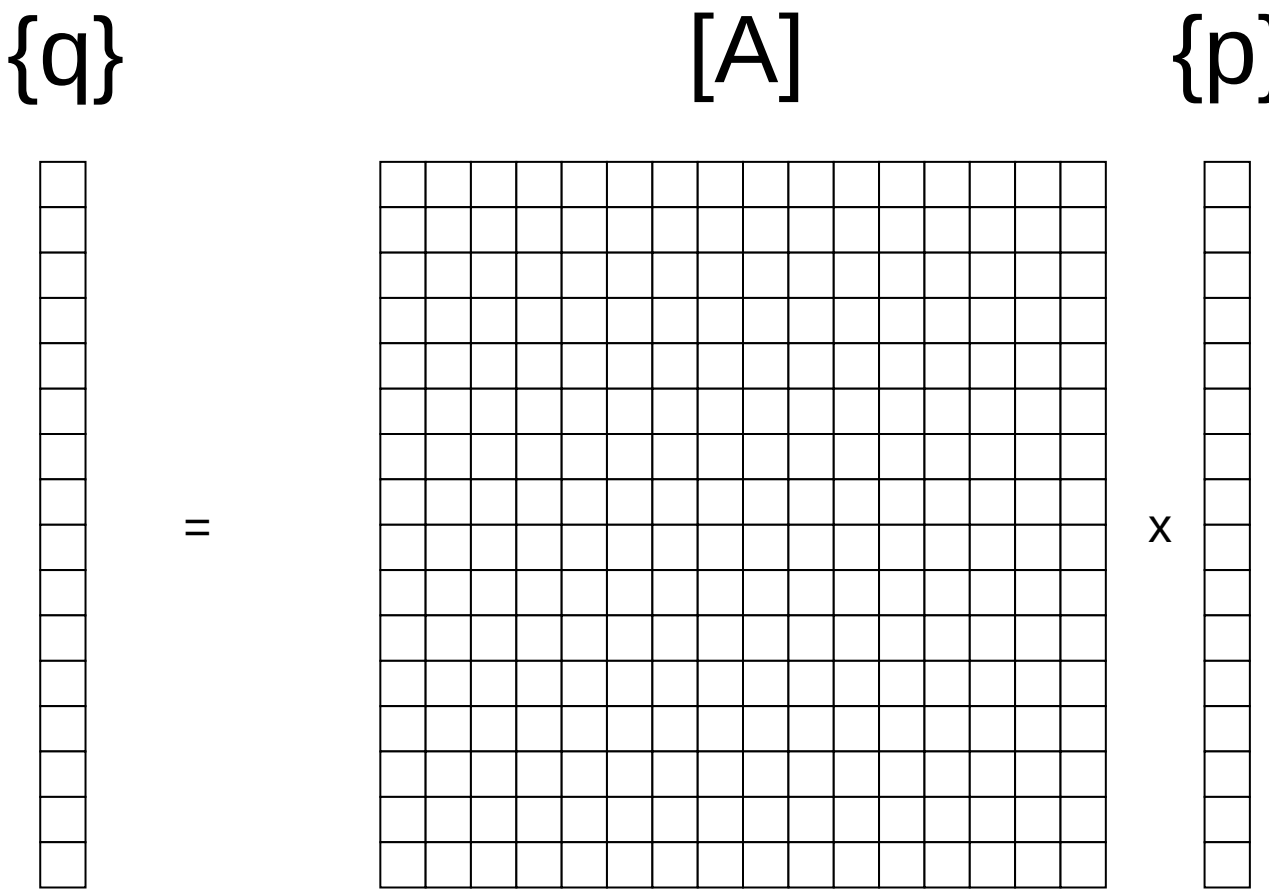


- 問題設定
- 1CPU用プログラムの解説
- **並列プログラムの解説**

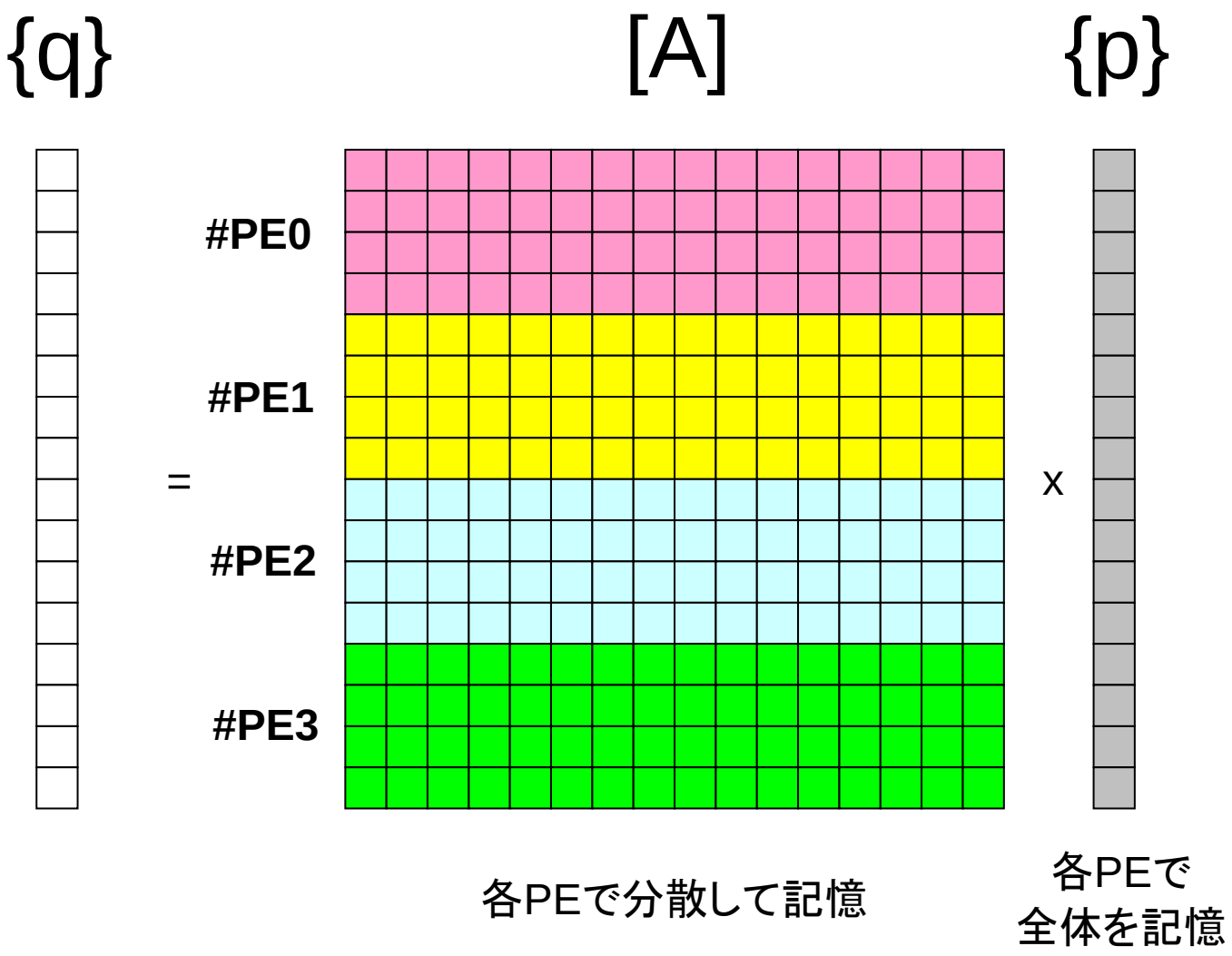
粒子間熱伝導コードの並列化

- 未知数をNとする。
- 有限要素法，有限体積法に必要な記憶容量
  - 未知数ベクトル： O(N)
  - 係数マトリクス： O(N)
- 粒子間熱伝導コードで必要な記憶容量
  - 未知数ベクトル： O(N)
  - 係数マトリクス： $O(N^2)$：Nの値は必然的に小さくなる
- 本手法では，係数マトリクスが必要な記憶容量のほとんどを占めるため：
  - 未知数ベクトル，形状については各PEが全体情報を記憶する：eps_fvmのときのような領域分割は不要。
  - 係数マトリクスを各PEに分散して記憶する。

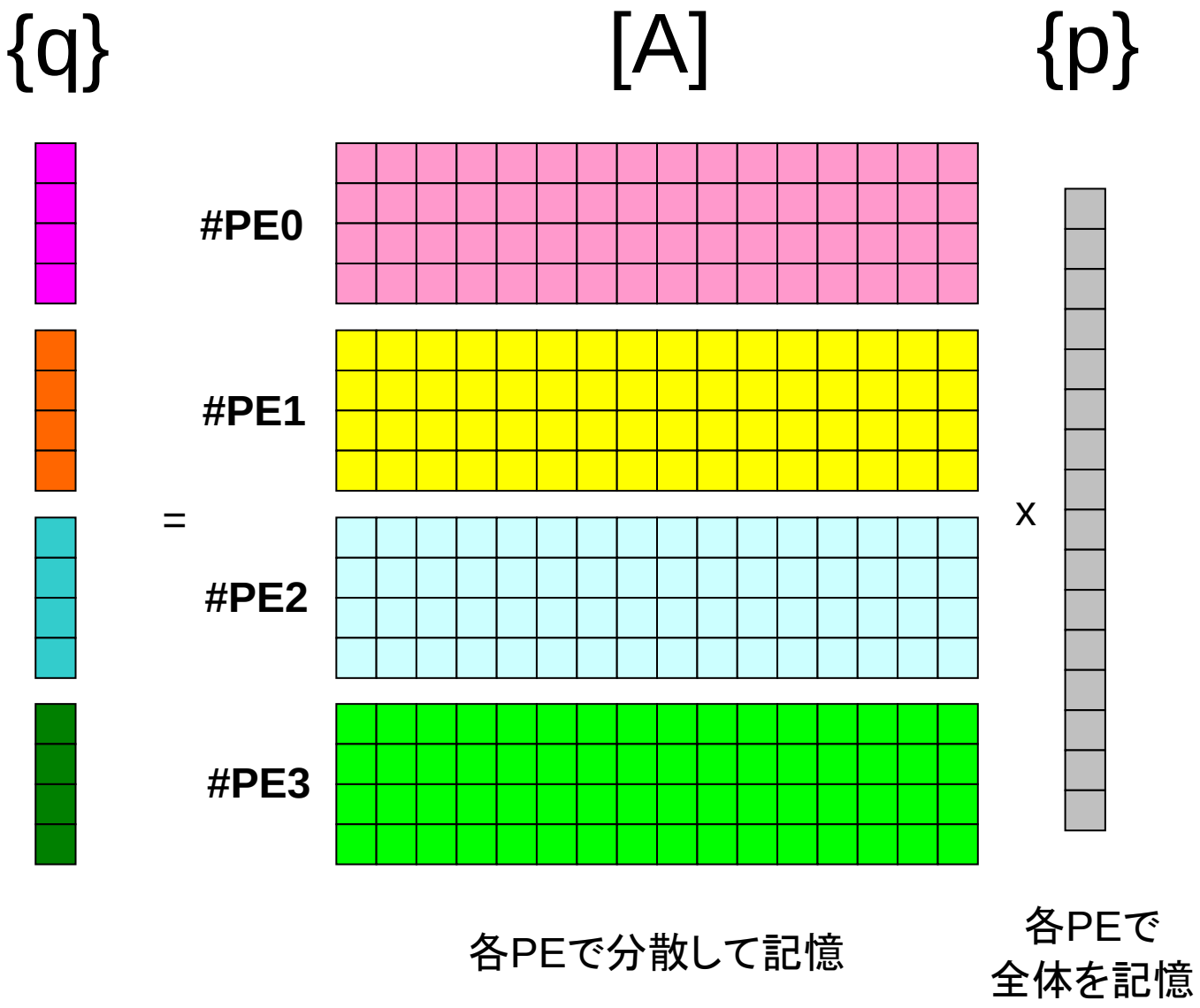
行列ベクトル積の例（1/5）



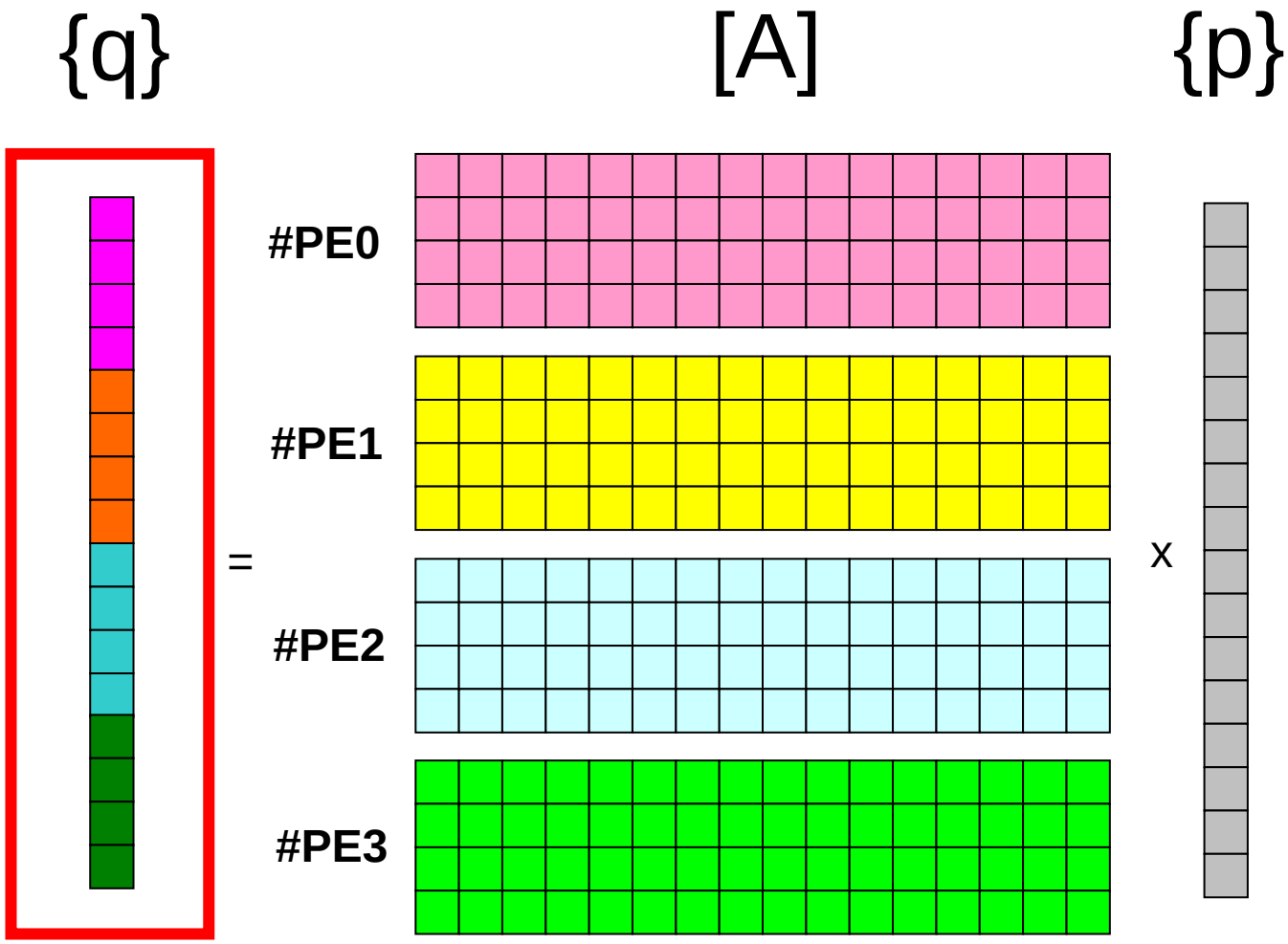
行列ベクトル積の例（2/5）



行列ベクトル積の例（3/5）



行列ベクトル積の例（3/5）



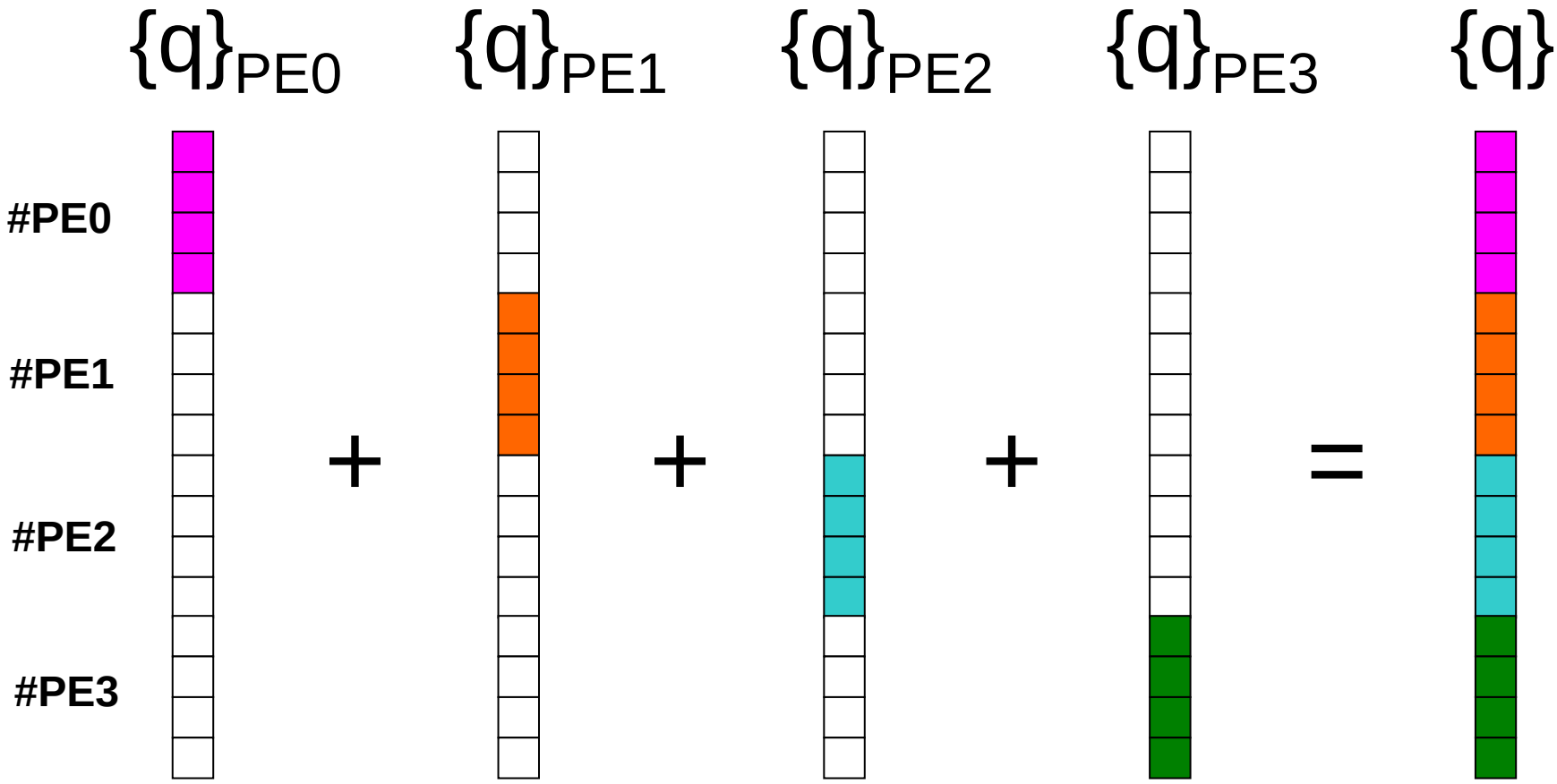
MPI_ALLREDUCE
ALLGATHERVで一つにまとめる

各PEで分散して記憶

各PEで
全体を記憶

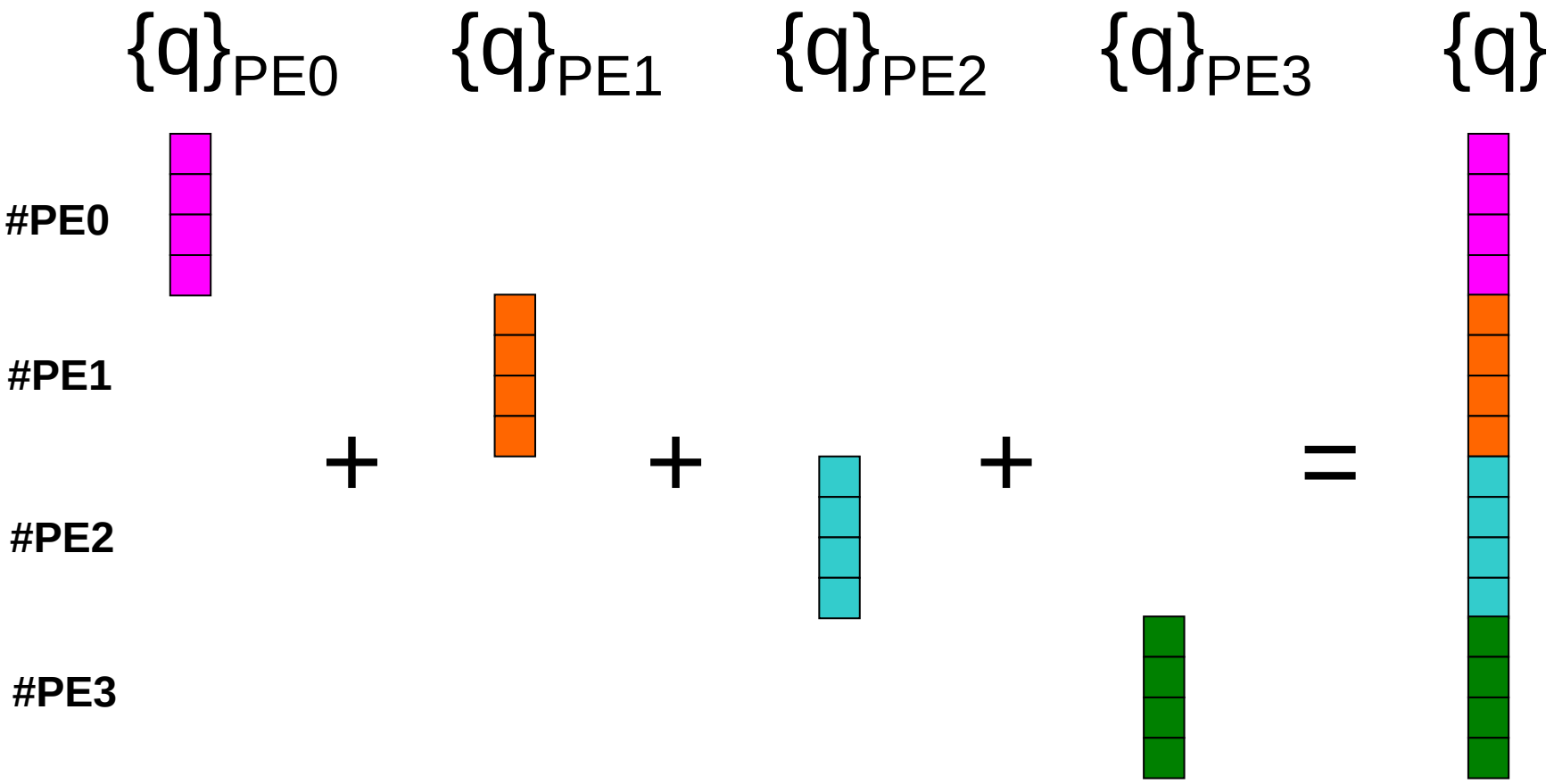
行列ベクトル積の例（4/5）

MPI_ALLREDUCE使用（白い部分は0）

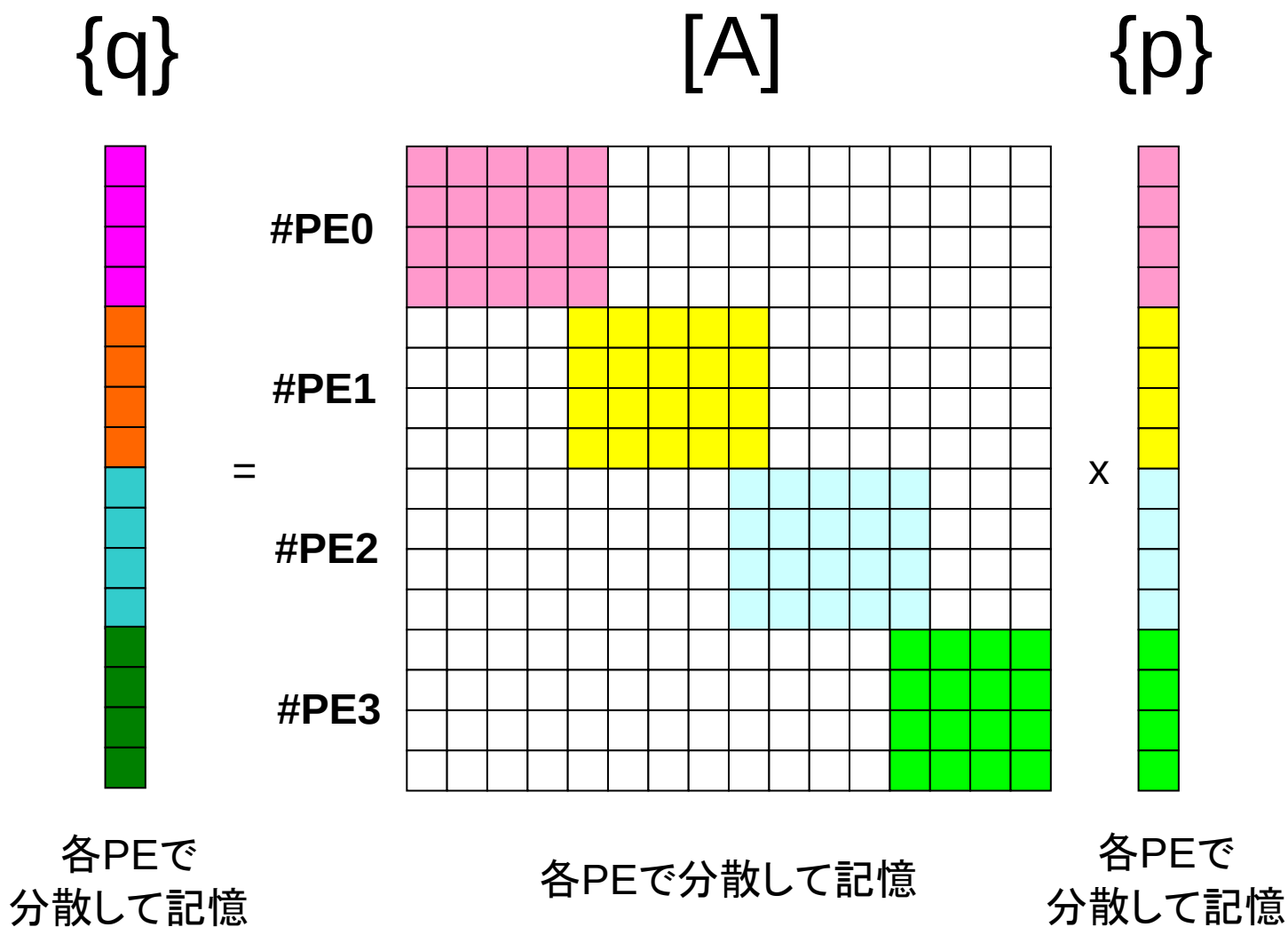


行列ベクトル積の例(5/5)

MPI_ALLGATHERV使用



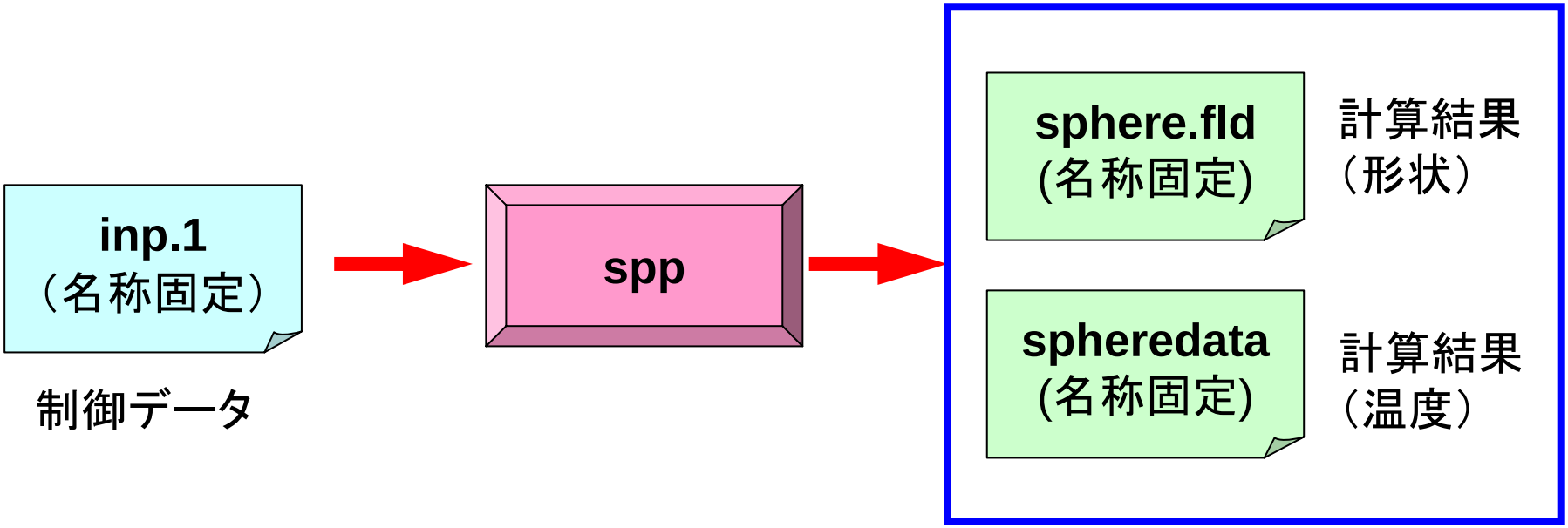
行列ベクトル積：有限体積法等（疎行列）



コンパイル，実行

```
$> cd <$SPHERE>/C
$> mpicc -O3 testp.c -o spp
$> mpirun -np XX spp
$> ls -l sphere*
   sphere.fld   spheredata

$> cd <$SPHERE>/F
$> mpif90 -O3 testp.f -o spp
$> mpirun -np XX spp
$> ls -l sphere*
   sphere.fld   spheredata
```

spp: ファイル，データ


inp.1
（名称固定）
制御データ
spp
sphere.fld
(名称固定)
計算結果
（形状）
spheredata
(名称固定)
計算結果
（温度）

testp.f：パラレル版（1/12）
初期設定①

```
      implicit REAL*8 (A-H,O-Z)

      include 'mpif.h'

      real(kind=8) :: VOL, AREA, QVOL, COND0, COND, RADImax
      real(kind=8) :: HCONV, T0, SURF, DEL, DEL0, coef1, coef2
      real(kind=8), dimension(:)  , allocatable :: XC, YC, ZC
      real(kind=8), dimension(:,:), allocatable :: AMAT
      real(kind=8), dimension(:)  , allocatable :: RHS
      real(kind=8), dimension(:)  , allocatable :: PHI
      real(kind=8), dimension(:,:), allocatable :: W

      integer :: NX, NY, NZ, N
      integer :: R, Z, P, Q, DD, WK

      integer :: PETOT, PEsmpTOT, my_rank, errno
      integer, dimension(:)  , allocatable :: NElocal, AGVindex
      integer, dimension(:)  , allocatable :: PEon
!C
!C +------+
!C | INIT |
!C +------+
!C===
      call MPI_INIT      (ierr)
      call MPI_COMM_SIZE (MPI_COMM_WORLD, PETOT   , ierr)
      call MPI_COMM_RANK (MPI_COMM_WORLD, my_rank , ierr)
```

testp.f：パラレル版（2/12）

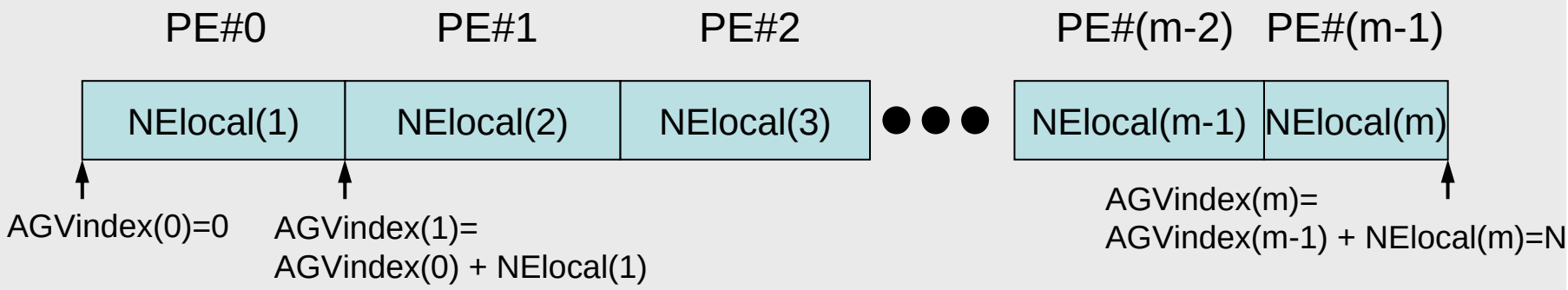
初期設定②：同じ制御データ

```
open (11, file= 'inp1', status='unknown')
  read (11,*) NX, NY, NZ
  read (11,*) DX, DY, DZ
  read (11,*) VOL, AREA, QVOL, COND0
  read (11,*) HCONV, T0, SURF
  read (11,*) RADImax
  read (11,*) coef1, coef2
close (11)

N= NX*NY*NZ

allocate (XC(N), YC(N), ZC(N))

S1_time= MPI_WTIME()

icou= 0
do k= 1, NZ
  do j= 1, NY
    do i= 1, NX
      icou= icou + 1
      XC(icou)= dfloat(i-1)*DX
      YC(icou)= dfloat(j-1)*DY
      ZC(icou)= dfloat(k-1)*DZ
    enddo
  enddo
enddo
```

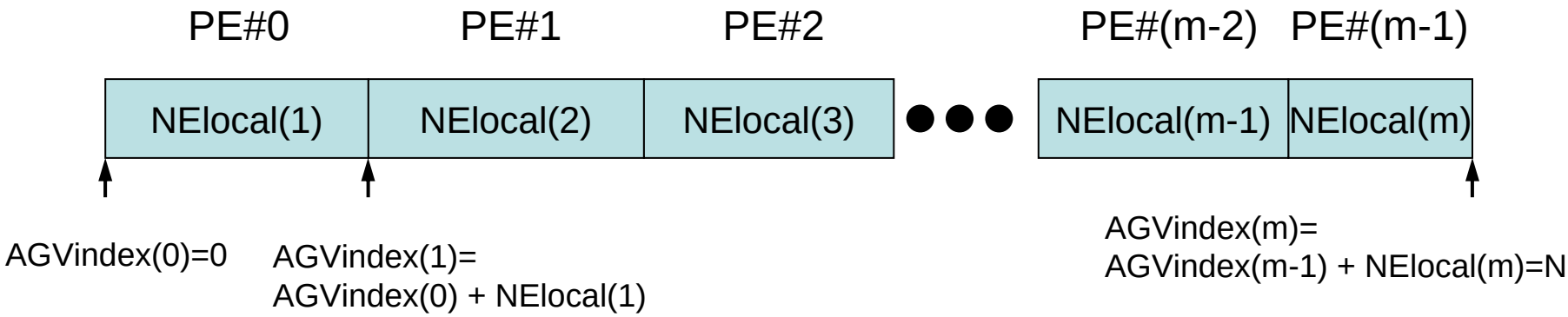
testp.f：パラレル版（3/12）

初期設定③：ローカルデータ設定

```
!C
!C-- LOCAL
      allocate (NElocal(PETOT), AGVindex(0:PETOT))
      allocate (PEon(N))

      NElocal =  0
      AGVindex=  0
      PEon    = -1

      NN= N / PETOT
      NElocal= NN
      NR= N - PETOT*NN
      do ip= 1, PETOT
        if (ip.le.NR) NElocal(ip)= NElocal(ip) + 1
        AGVindex(ip)= AGVindex(ip-1) + NElocal(ip)
      enddo
```



ローカルデータ生成

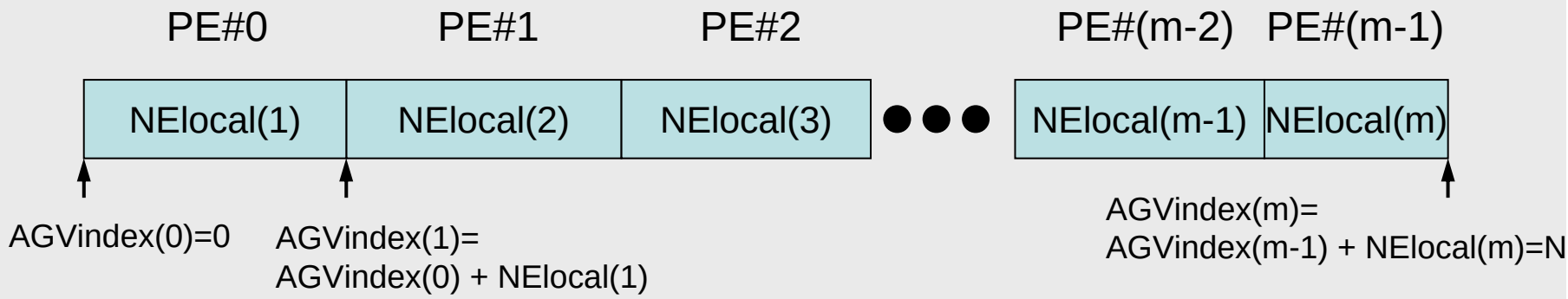


全体ベクトル　　1 ~ N
局所ベクトル　　1 ~ NElocal(my_rank+1)

testp.f：パラレル版（3/12）

初期設定③：ローカルデータ設定

```
!C
!C-- LOCAL
      allocate (NElocal(PETOT), AGVindex(0:PETOT))
      allocate (PEon(N))

      NElocal =  0
      AGVindex=  0
      PEon    = -1
```




```
      do ip= 1, PETOT
       do j= AGVindex(ip-1)+1, AGVindex(ip)
         PEon(j)= ip - 1
        enddo
      enddo
!C===
```

オペレーションの並列化

FEM,FVM

```
do i= 1, N
  do j= 1, ICON(i)
    …
  enddo
enddo
```

Nlocal= N/PETOT

FEM,FVM

```
do i= 1, Nlocal
  do j= 1, ICON(i)
    …
  enddo
enddo
```

本手法

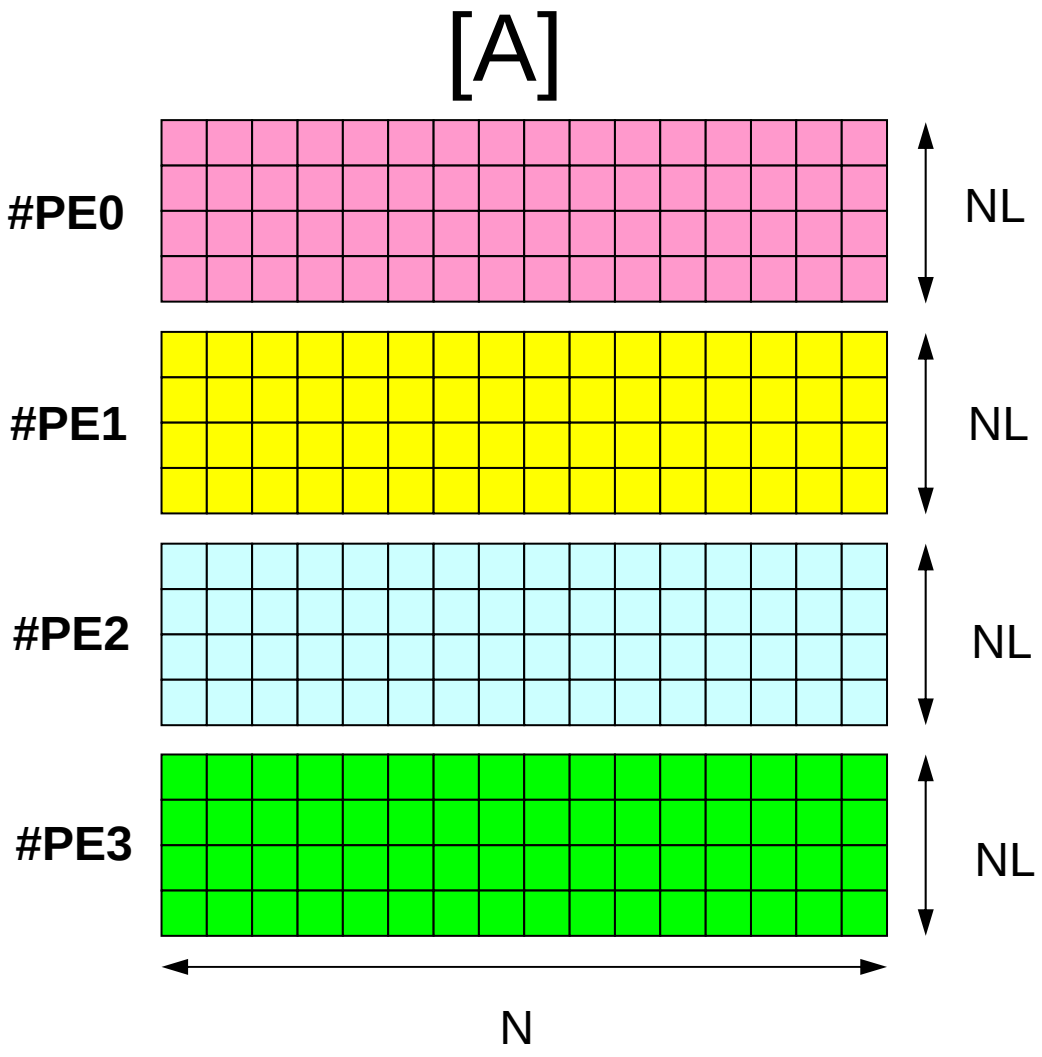
```
do i= 1, N
  do j= 1, N
    …
  enddo
enddo
```

Nlocal= N/PETOT

本手法

```
do i= 1, Nlocal
  do j= 1, N
    …
  enddo
enddo
```

係数行列：各PEで独立に計算，格納

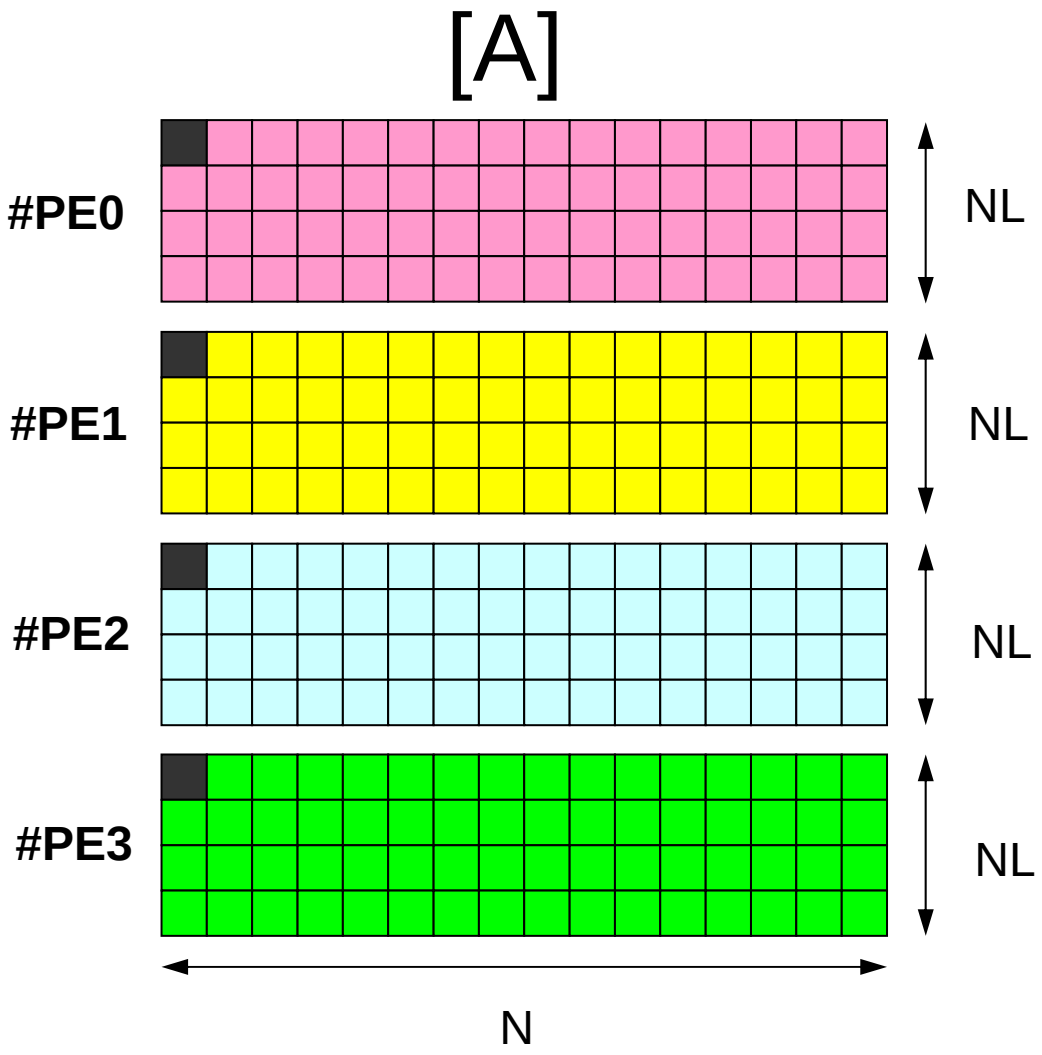


A(NL,N)

NLは各PEで異なる可能性がある。

行に関しては局所番号
列に関しては全体番号

係数行列：各PEで独立に計算，格納

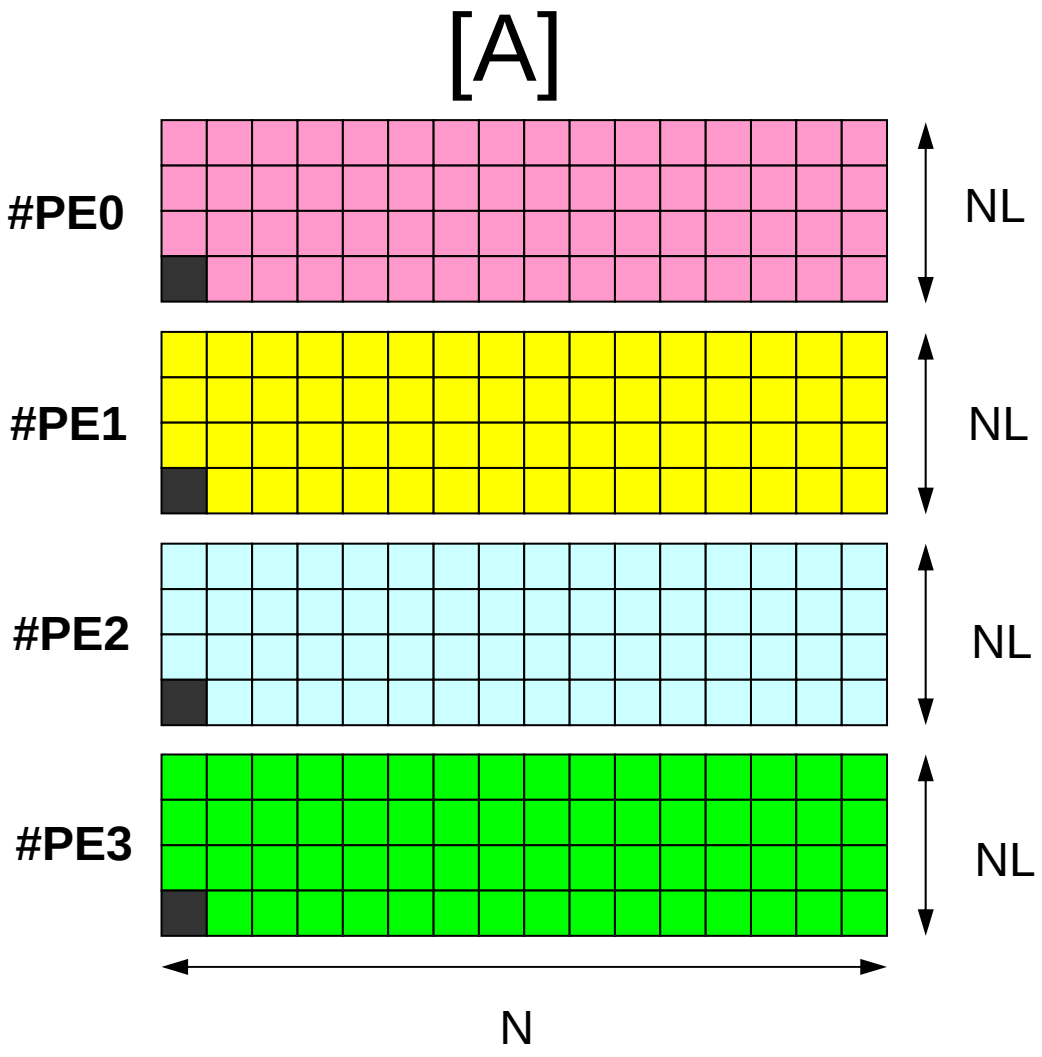


A(NL,N)
NLは各PEで異なる可能性がある。

行に関しては局所番号
列に関しては全体番号

A(1,1)

係数行列：各PEで独立に計算，格納

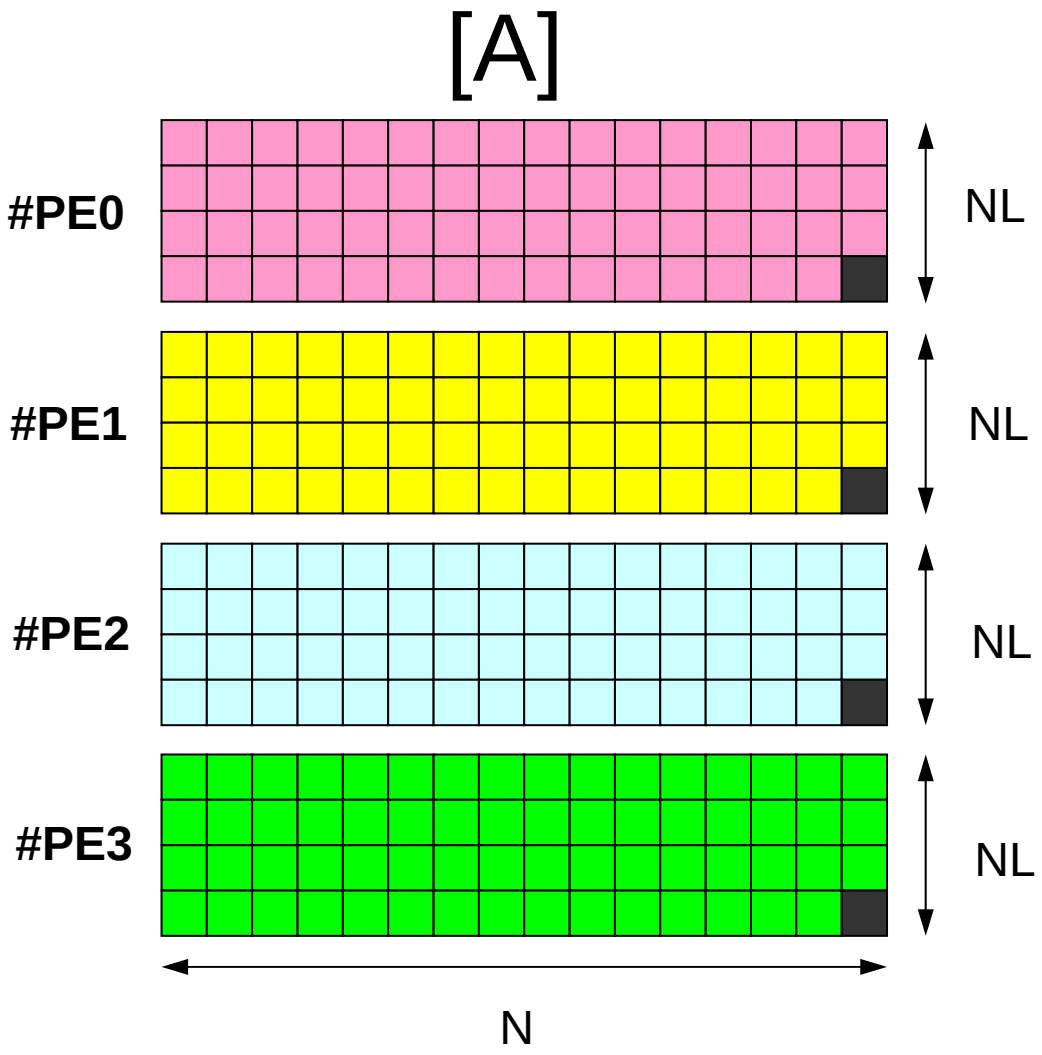


A(NL,N)
NLは各PEで異なる可能性がある。

行に関しては局所番号
列に関しては全体番号

A(NL,1)

係数行列：各PEで独立に計算，格納



A(NL,N)
NLは各PEで異なる可能性がある。

行に関しては局所番号
列に関しては全体番号

A(NL,N)

testp.f：パラレル版（4/12）

マトリクス生成：熱伝導部分

```
!C
!C +--------+
!C | MATRIX |
!C +--------+
!C===
      allocate (AMAT(NElocal(my_rank+1),N), RHS(N))

      AMAT= 0.d0
      RHS = 0.d0
      do i0= 1, NElocal(my_rank+1)
        i= i0 + AGVindex(my_rank)
        do j= 1, N
          if (j.ne.i) then
            DEL= dsqrt((XC(i)-XC(j))**2 + (YC(i)-YC(j))**2                &
     &                                 + (ZC(i)-ZC(j))**2)
            if (DEL.le.RADImax) then
              COND= COND0/(10.d0**dmin1(DEL,20.d0))
              coef= COND*AREA / DEL
              AMAT(i0,j)= coef
              AMAT(i0,i)= AMAT(i0,i) - coef
            endif
          endif
        enddo
      enddo
```

i0：局所番号
i ：全体番号

testp.f：パラレル版（5/12）

マトリクス生成：体積発熱：シリアルと同じ

```
      do i= 1, N
        DEL= dsqrt(XC(i)**2 + YC(i)**2 + ZC(i)**2)
        RHS(i)= -QVOL*(coef1*VOL + coef2*DEL*VOL)
      enddo

      i= 1
      do k= 1, NZ
      do j= 1, NY
        ic= (k-1)*NX*NY + (j-1)*NX + i
        ip= PEon(ic)
        if (my_rank.eq.ip) then
          ic0= ic - AGVindex(ip)
          AMAT(ic0,ic)= -HCONV*SURF + AMAT(ic0,ic)
        endif
        RHS (ic   )= -HCONV*SURF*T0 + RHS (ic)
      enddo
      enddo
!C===
      S2_time= MPI_WTIME()
```

testp.f：パラレル版（5/12）

マトリクス生成：対流熱伝達

```
      do i= 1, N
        DEL= dsqrt(XC(i)**2 + YC(i)**2 + ZC(i)**2)
        RHS(i)= -QVOL*(coef1*VOL + coef2*DEL*VOL)
      enddo

      i= 1
      do k= 1, NZ
      do j= 1, NY
        ic= (k-1)*NX*NY + (j-1)*NX + i
        ip= PEon(ic)
        if (my_rank.eq.ip) then
          ic0= ic - AGVindex(ip)
          AMAT(ic0,ic)= -HCONV*SURF + AMAT(ic0,ic)
        endif
        RHS (ic   )= -HCONV*SURF*T0 + RHS (ic)
      enddo
      enddo
!C===
      S2_time= MPI_WTIME()
```

ic0：　局所番号
ic：　全体番号

testp.f：パラレル版（6/12）

CG法①

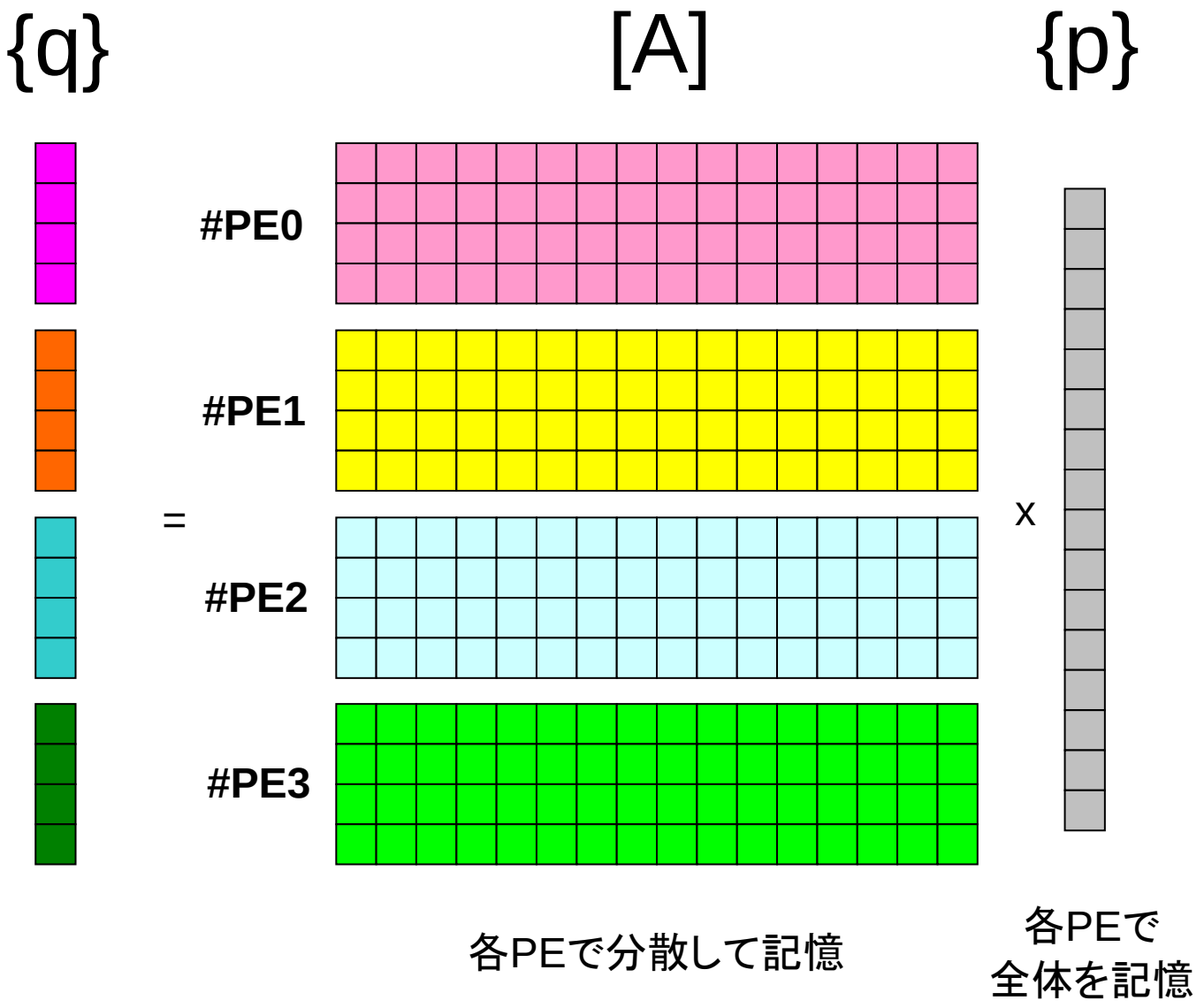
```
!C
!C +---------------+
!C | CG iterations |
!C +---------------+
!C===
      EPS= 1.d-08
      allocate (W(N,5), PHI(N))

      NEL= NElocal(my_rank+1)

      W   = 0.d0
      PHI= 0.d0

      R = 1
      Z = 2
      Q = 2
      P = 3
      DD= 4
      WK= 5
```

Compute $r^{(0)}= b-[A]x^{(0)}$
for i= 1, 2, …
  solve $[M]z^{(i-1)}= r^{(i-1)}$
  $\rho_{i-1}= r^{(i-1)}\ z^{(i-1)}$
  if i=1
    $p^{(1)}= z^{(0)}$
  else
    $\beta_{i-1}= \rho_{i-1}/\rho_{i-2}$
    $p^{(i)}= z^{(i-1)} + \beta_{i-1}\ p^{(i)}$
  endif
  $q^{(i)}= [A]p^{(i)}$
  $\alpha_i = \rho_{i-1}/p^{(i)}q^{(i)}$
  $x^{(i)}= x^{(i-1)} + \alpha_i p^{(i)}$
  $r^{(i)}= r^{(i-1)} - \alpha_i q^{(i)}$
  check convergence |r|
end

CG法（密行列）の並列化

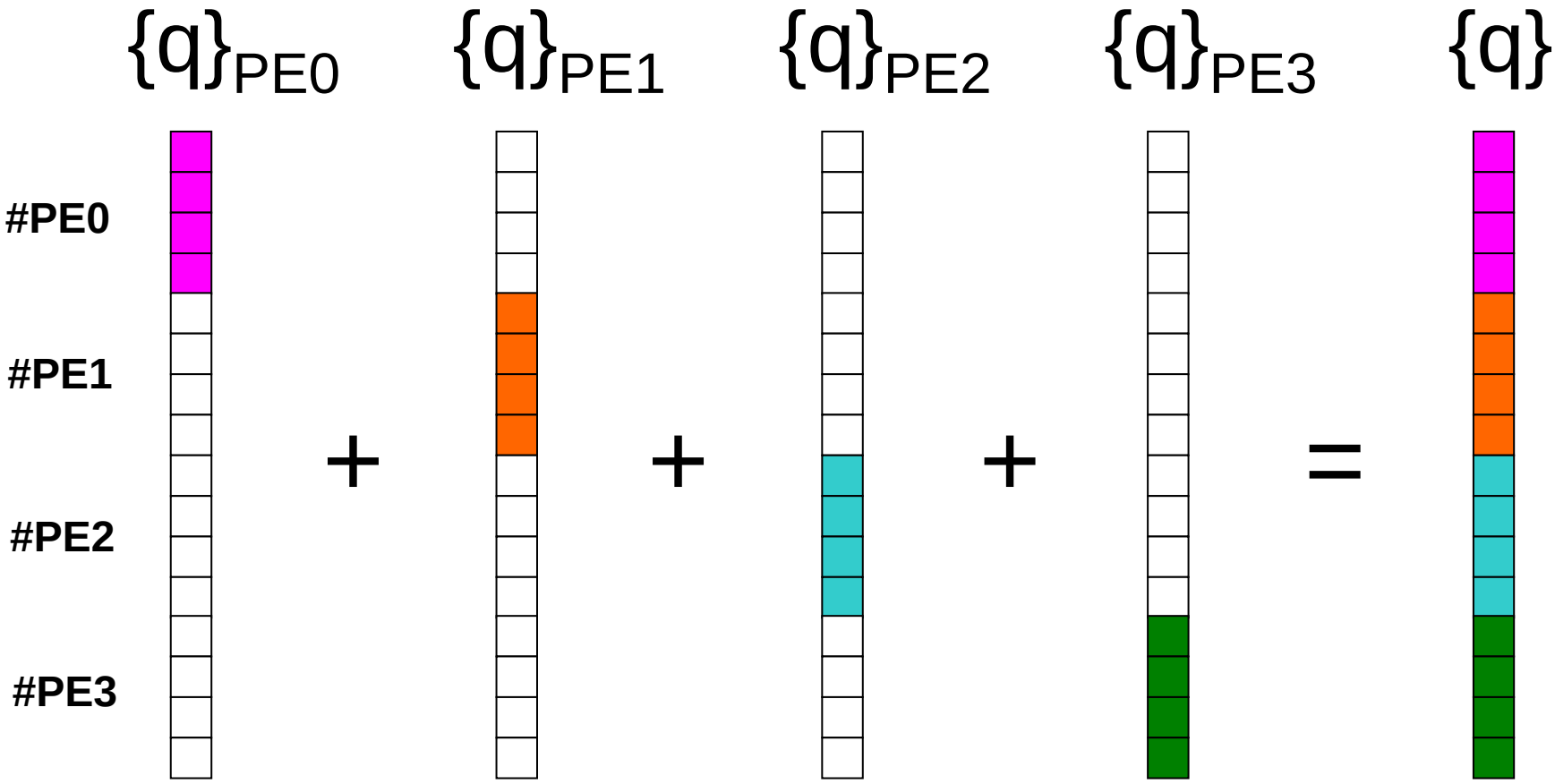
- 計算量のオーダー
  - 行列ベクトル積 N^2
  - 前処理（対角スケーリング） N
  - 内積 N
  - DAXPY（{y}= a{x}+{y}） N
- Nがある程度大きくなると，行列ベクトル積が占める量が圧倒的になる
  - 行列ベクトル積の部分のみ並列化
  - 他は並列化しない（各PEで同じ計算を1~Nで実施）
    - 局所データを使用して並列に計算することも可能であるが，計算量に比べて通信オーバーヘッドの方が概して大きくなる。

行列ベクトル積（1/2）



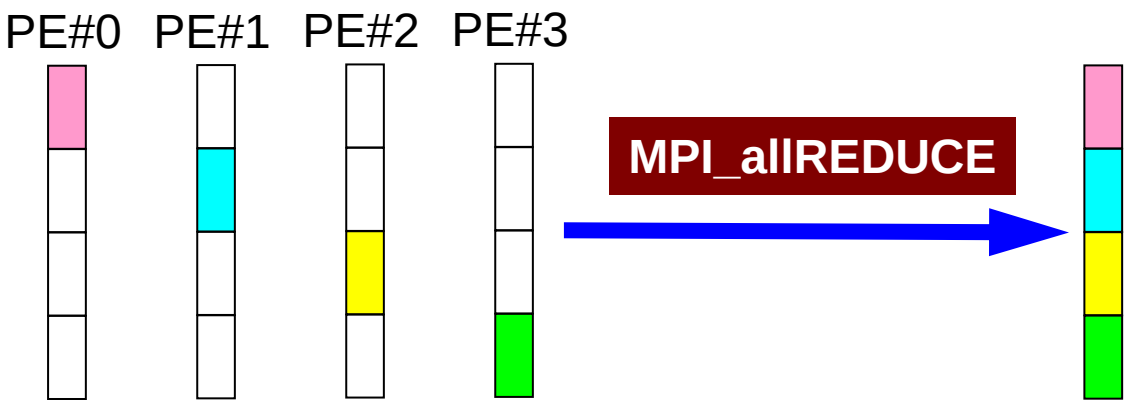
行列ベクトル積(2/2)

MPI_ALLREDUCE使用(白い部分は0)



testp.f：パラレル版（7/12）
CG法②：行列の対角成分の逆数の計算

```
      do i0= 1, NEL
        i= i0 + AGVindex(my_rank)
        W(i,DD)= 1.d0/AMAT(i0,i)
      enddo

      do i= 1, N
        W(i,WK)= 0.d0
      enddo

      call MPI_allREDUCE                                                &
     &       (W(1,DD), W(1,WK), N, MPI_DOUBLE_PRECISION, MPI_SUM,       &
     &        MPI_COMM_WORLD, ierr)

      do i= 1, N
        W(i,DD)= W(i,WK)
      enddo
```



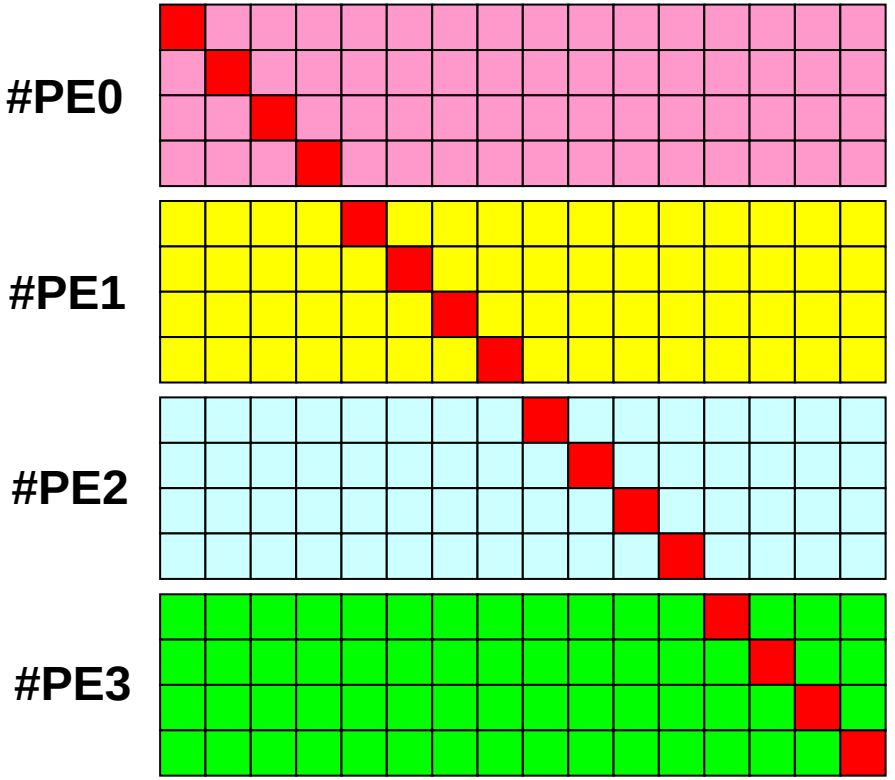
各PEで担当部分計算（白い部分0）

行列の対角成分の逆数

- 前処理については各PEで以下のような計算を実施したい。

```
do i= 1, N
  W(i,Z)= W(i,DD) * W(i,R)
enddo
```

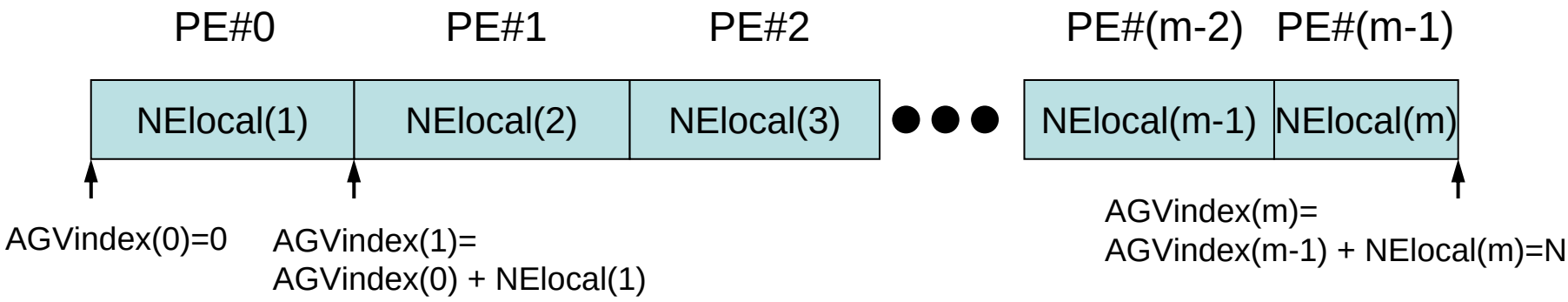
- しかし，係数行列を右のように分散して覚えているので，「対角成分」は各PEに分散して記憶されている。
  - 従って，各PEで計算したW(i,DD)をMPI_ALLREDUCEしている。



testp2.f：MPI_Allgathervを使用

行列の対角成分の逆数の計算

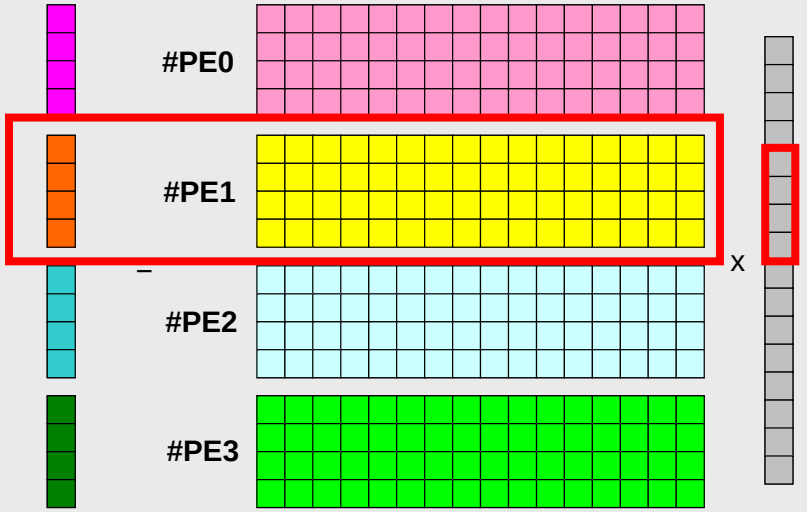
```
      do i0= 1, NEL
        i= i0 + AGVindex(my_rank)
        W(i0,WK)= 1.d0/AMAT(i0,i)
      enddo

      call MPI_allGATHERv                                          &
     &    (W(1,WK), NEL,                    MPI_DOUBLE_PRECISION,  &
     &     W(1,DD), NElocal, AGVindex(0), MPI_DOUBLE_PRECISION,    &
     &     MPI_COMM_WORLD, ierr)
```



全体ベクトル　　1 ~ N
局所ベクトル　　1 ~ NEL

testp.f：パラレル版（8/12）
CG法③

```
!C
!C-- {r0}= {b} - [A]{xini} |
      do i0= 1, NEL
        i= i0 + AGVindex(my_rank)
        W(i,R)= RHS(i)
      enddo

      do j= 1, N
        XX= PHI(j)
        W(j,WK)= 0.d0
        do i0= 1, NEL
          i= i0 + AGVindex(my_rank)
          W(i,R)= W(i,R) - AMAT(i0,j) * XX
        enddo
      enddo

      call MPI_allREDUCE                                              &
     &        (W(1,R), W(1,WK), N, MPI_DOUBLE_PRECISION, MPI_SUM,      &
     &         MPI_COMM_WORLD, ierr)

      do i= 1, N
        W(i,R)= W(i,WK)
      enddo

      BNRM2= 0.0D0
      do i= 1, N
        BNRM2 = BNRM2  + RHS(i)**2
      enddo
```



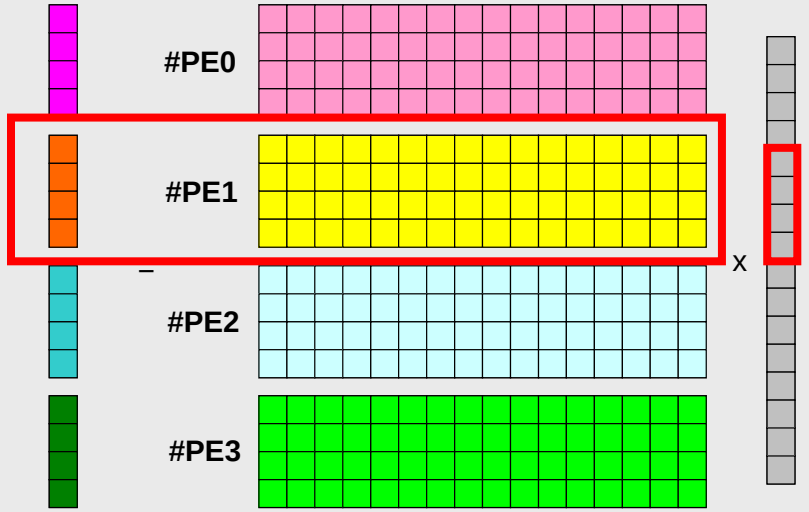
testp.f：パラレル版（9/12）
CG法④

```
!C***********************************************************************
      do iter= 1, N
!C
!C-- {z}= [Minv]{r}
      do i= 1, N
        W(i,Z)= W(i,DD) * W(i,R)
      enddo
!C
!C-- RHO= {r}{z}
      RHO= 0.d0
      do i= 1, N
        RHO= RHO + W(i,R)*W(i,Z)
      enddo
!C
!C-- {p} = {z} if      ITER=1
!C   BETA= RHO / RHO1  otherwise

      if ( iter.eq.1 ) then
        do i= 1, N
          W(i,P)= W(i,Z)
        enddo
       else
         BETA= RHO / RHO1
         do i= 1, N
           W(i,P)= W(i,Z) + BETA*W(i,P)
         enddo
      endif
```

testp.f：パラレル版（10/12）

CG法⑤

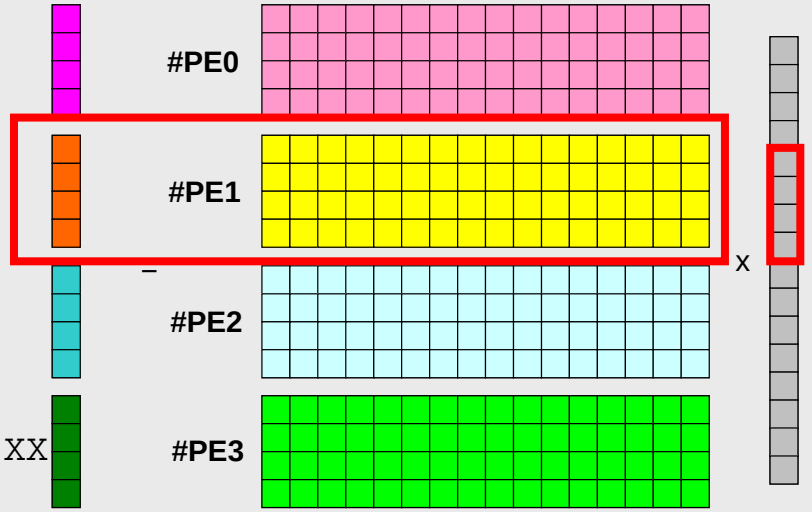
```
!C
!C-- {q}= [A]{p}

      do i= 1, N
        W(i,Q)= 0.d0
      enddo

      do j= 1, N
        XX= W(j,P)
        W(j,WK)= 0.d0
        do i0= 1, NEL
          i= i0 + AGVindex(my_rank)
          W(i,Q)= W(i,Q) + AMAT(i0,j) * XX
        enddo
      enddo

      call MPI_allREDUCE                                                &
     &        (W(1,Q), W(1,WK), N, MPI_DOUBLE_PRECISION, MPI_SUM,        &
     &         MPI_COMM_WORLD, ierr)

      do i= 1, N
        W(i,Q)= W(i,WK)
      enddo
```



testp2.f：MPI_Allgatherv 使用

{q}= [A]{p}：少し速い：ゼロクリア少ない

```
!C
!C-- {q}= [A]{p}



      do i0= 1, NEL
        W(i0,WK)= 0.d0
      enddo

      do j= 1, N
        XX= W(j,P)
        do i0= 1, NEL
          W(i0,WK)= W(i0,WK) + AMAT(i0,j) * XX
        enddo
      enddo

      call MPI_allGATHERv                                              &
     &    (W(1,WK), NEL,                    MPI_DOUBLE_PRECISION,      &
     &     W(1,Q ), NElocal, AGVindex(0), MPI_DOUBLE_PRECISION,        &
     &     MPI_COMM_WORLD, ierr)
```



testp.f：パラレル版（11/12）
CG法⑥

```
!C
!C-- ALPHA= RHO / {p}{q}
      C1= 0.d0
      do i= 1, N
        C1= C1 + W(i,P)*W(i,Q)
      enddo
      ALPHA= RHO / C1
!C
!C-- {x}= {x} + ALPHA*{p}
!C   {r}= {r} - ALPHA*{q}
      do i= 1, N
        PHI(i)  = PHI(i) + ALPHA * W(i,P)
        W  (i,R)= W(i,R) - ALPHA * W(i,Q)
      enddo

      DNRM2 = 0.0
      do i= 1, N
        DNRM2= DNRM2 + W(i,R)**2
      enddo

        RESID= dsqrt(DNRM2/BNRM2)

        if ( RESID.le.EPS) goto 900
        RHO1 = RHO
      enddo
!C*****************************************************************
  900 continue
!C===
```

testp.f: パラレル版（12/12）

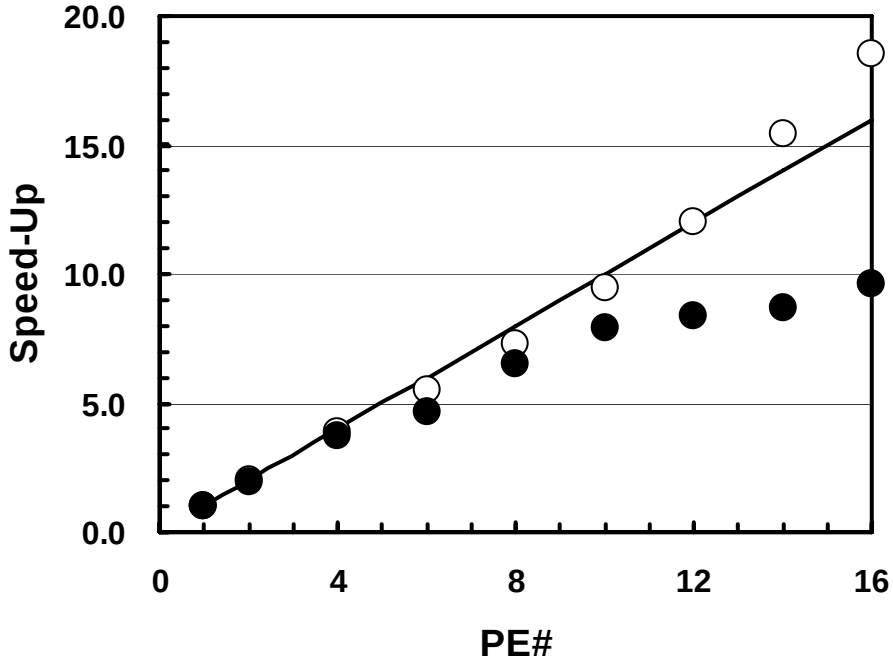
結果出力

```
      if (my_rank.eq.0) then
!C
!C +--------+
!C | OUTPUT |
!C +--------+
!C===
      N1= 1
      N3= 3

      open (22, file='sphere.fld', status='unknown')
      write (22,'(a)')    '# AVS field file'
      write (22,'(a,i5)') 'ndim=', N3
      write (22,'(a,i5)') 'dim1=', NX
      write (22,'(a,i5)') 'dim2=', NY
      write (22,'(a,i5)') 'dim3=', NZ
      write (22,'(a,i5)') 'nspace=', N3
      write (22,'(a,i5)') 'veclen=', N1
      write (22,'(a,i5)') 'data= float'
      write (22,'(a,i5)') 'field= uniform'
      write (22,'(a,i5)') 'label= temperature'
      write (22,'(a,i5)') 'variable 1 file=./spheredata filetype=ascii'
      close (22)

      open (22, file='spheredata', status='unknown')
      do i= 1, N
        write (22,'(1pe16.6)') PHI(i)
      enddo
!C===
     endif
```

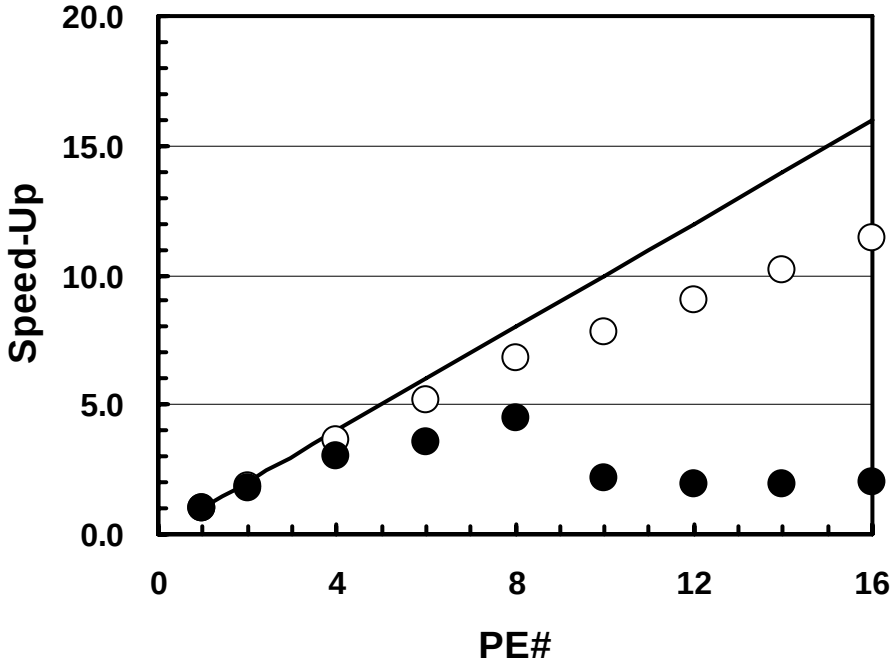
計算例: testp.f：遅い方

○：$N=20^3=8,000$, ●：$N=10^3=1,000$

IBM p5（九州大学）



cenju

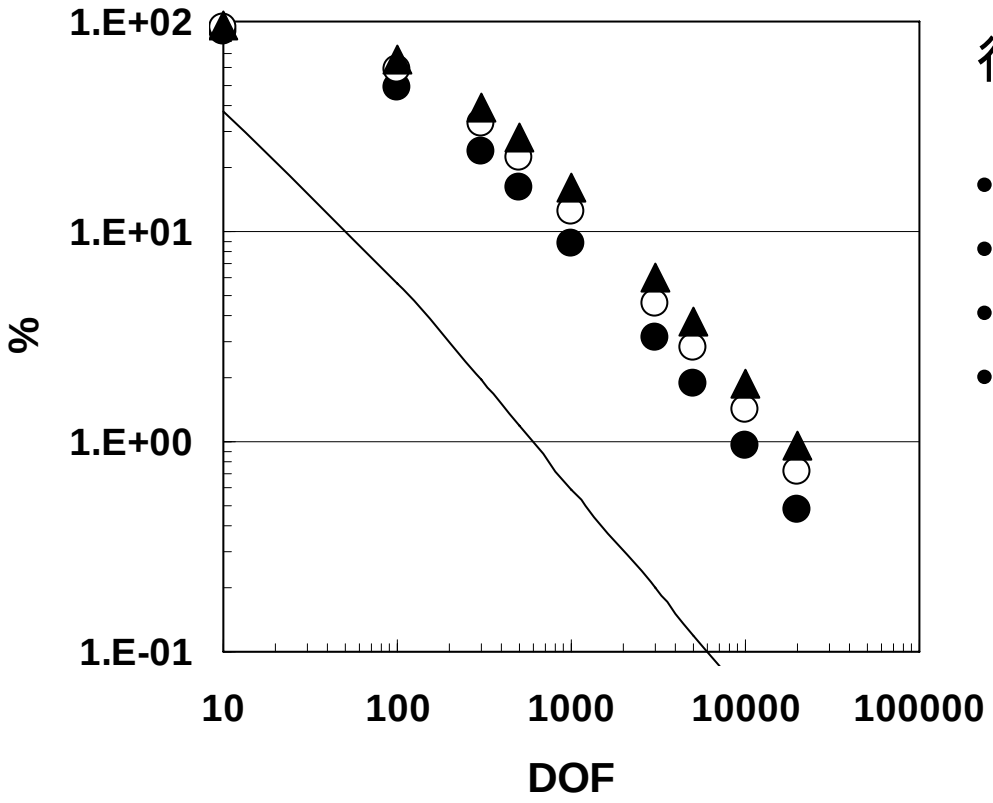


$N=20^3$

	4PE's	8PE's
testp:	9.54	5.13
testp2:	9.13	4.82

自習課題

- CG法の全てのプロセスを並列に計算するように書き換えて見よ。現在の「行列ベクトル積」のみを並列に計算する場合と比較して見よ。



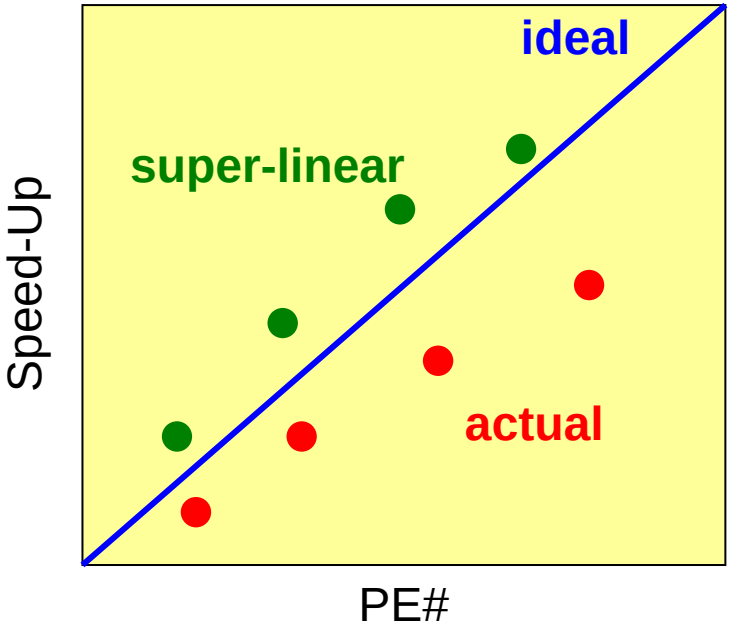
行列ベクトル積「以外」の計算量の比率

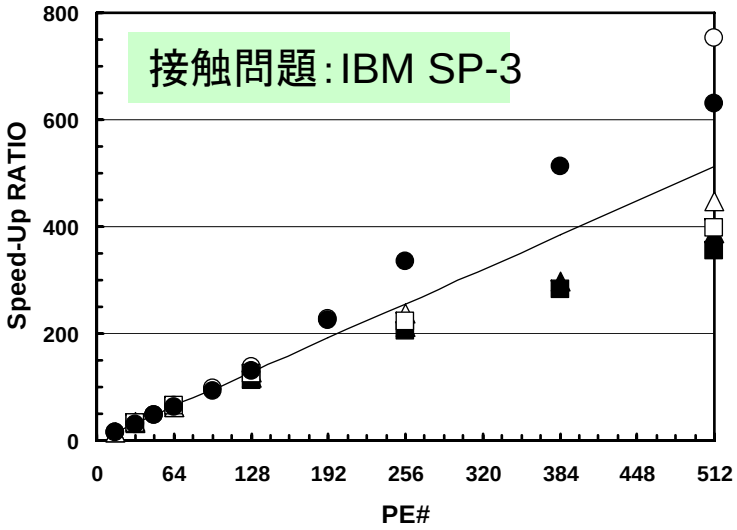
- 実線　完全に並列にした場合
- ●　「行列ベクトル積」のみ，16PE
- ○　「行列ベクトル積」のみ，24PE
- ▲　「行列ベクトル積」のみ，32PE

並列分散データ構造

- FVM型
  - 隣接領域
  - 内点，外点，境界点
  - 通信テーブル（送信，受信）
- 本手法
  - 内点
  - 基本的にはグローバル情報中心
  - ベクトルをローカルに扱う場合とグローバルに扱う場合がある。

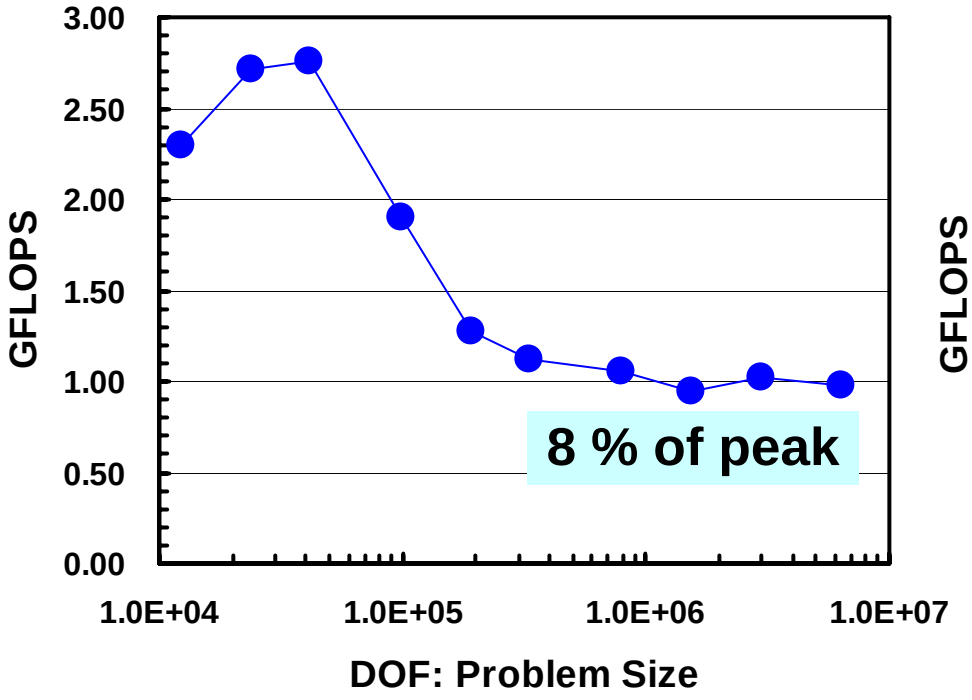
Strong-Scalingにおける「Super-Linear」

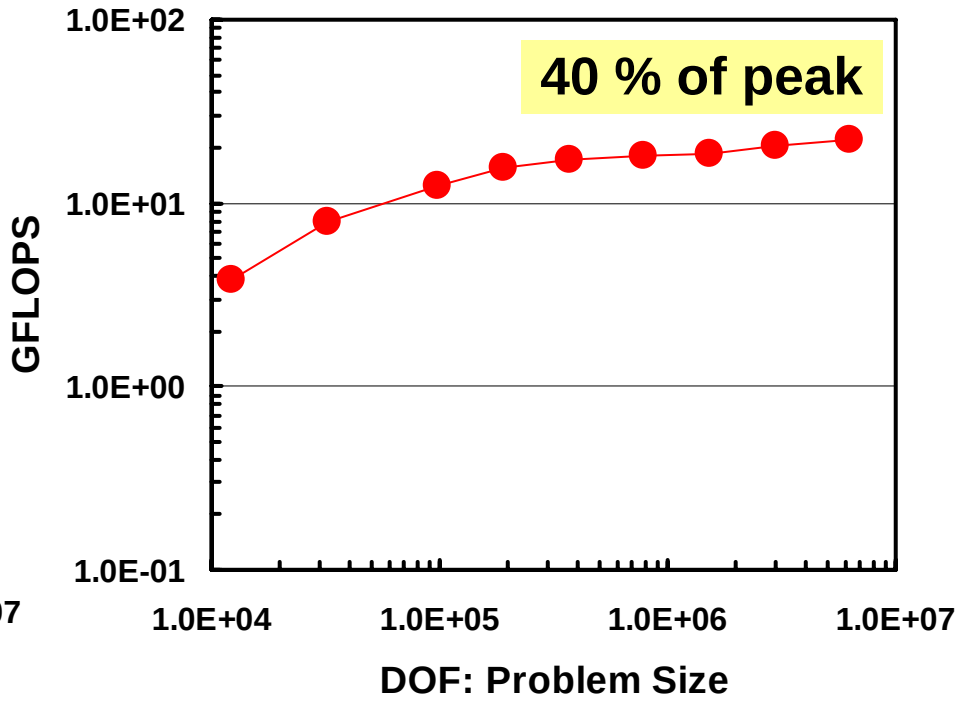




- 問題規模を固定して，使用PE数を増加させて行った場合，通常は通信の影響のために，効率は理想値（m個のPEを使用した場合，理想的にはm倍の性能になる）よりも低くなるのが普通である。
- しかし，スカラープロセッサ（PC等）の場合，逆に理想値よりも，高い性能が出る場合がある。このような現象を「Super-Linear」と呼ぶ。
  - ベクトル計算機では起こらない。

典型的な挙動

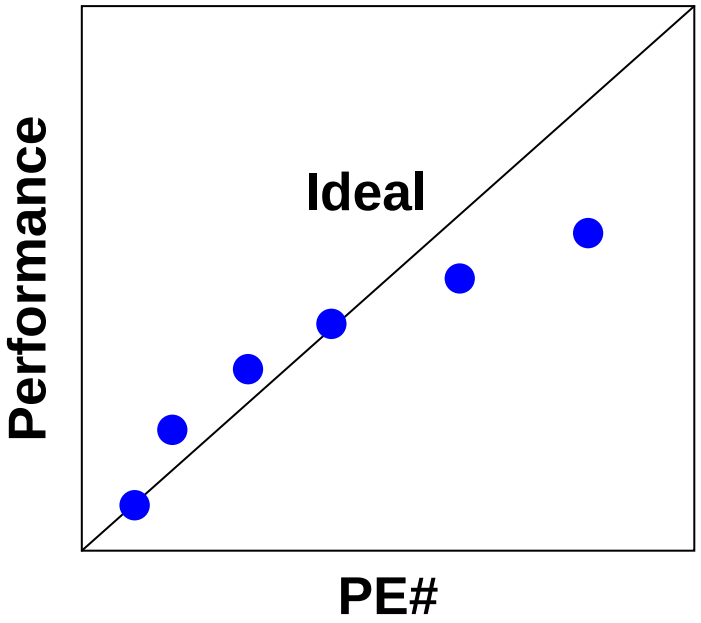


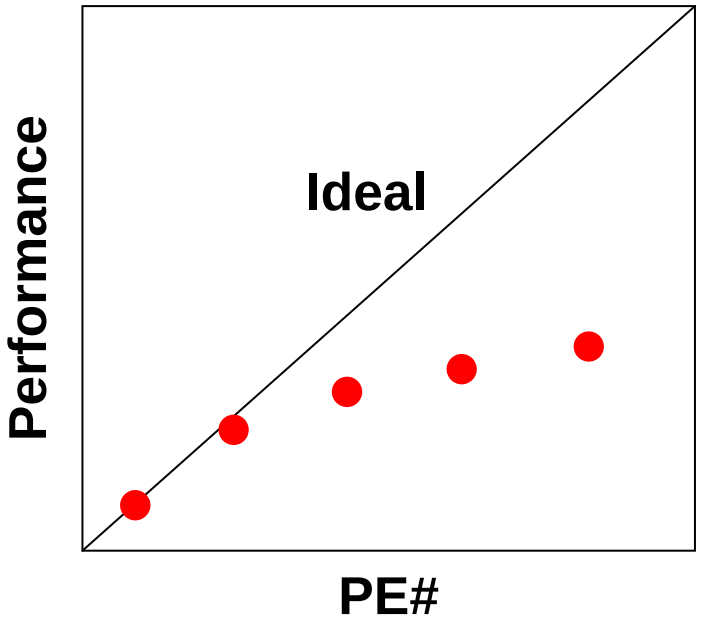


IBM-SP3:
問題サイズが小さい場合はキャッシュの影響のため性能が良い

Earth Simulator:
大規模な問題ほどベクトル長が長くなり，性能が高い

並列計算
Strong Scaling（全体問題規模固定）



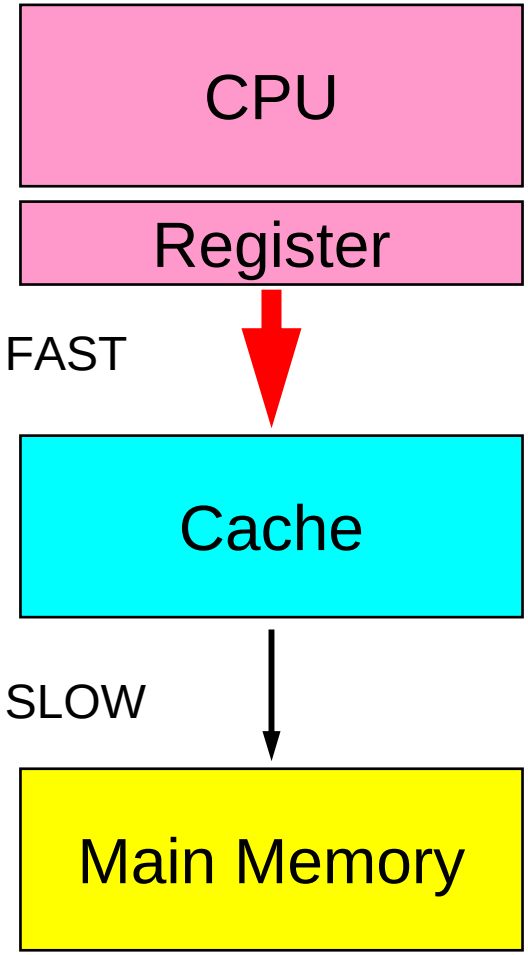


IBM-SP3:
PE（Processing Element）数が少ない場合はいわゆるスーパースカラー。
PE数が増加すると通信オーバーヘッドのため性能低下。

Earth Simulator:
PE数が増加すると，通信オーバーヘッドに加え，PEあたりの問題規模が小さくなるため性能低下。

Super-Linearの生じる理由

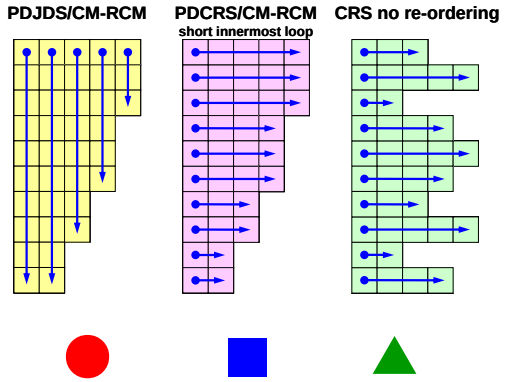
- キャッシュの影響
- スカラープロセッサでは，全般に問題規模が小さいほど性能が高い。
  - キャッシュの有効利用

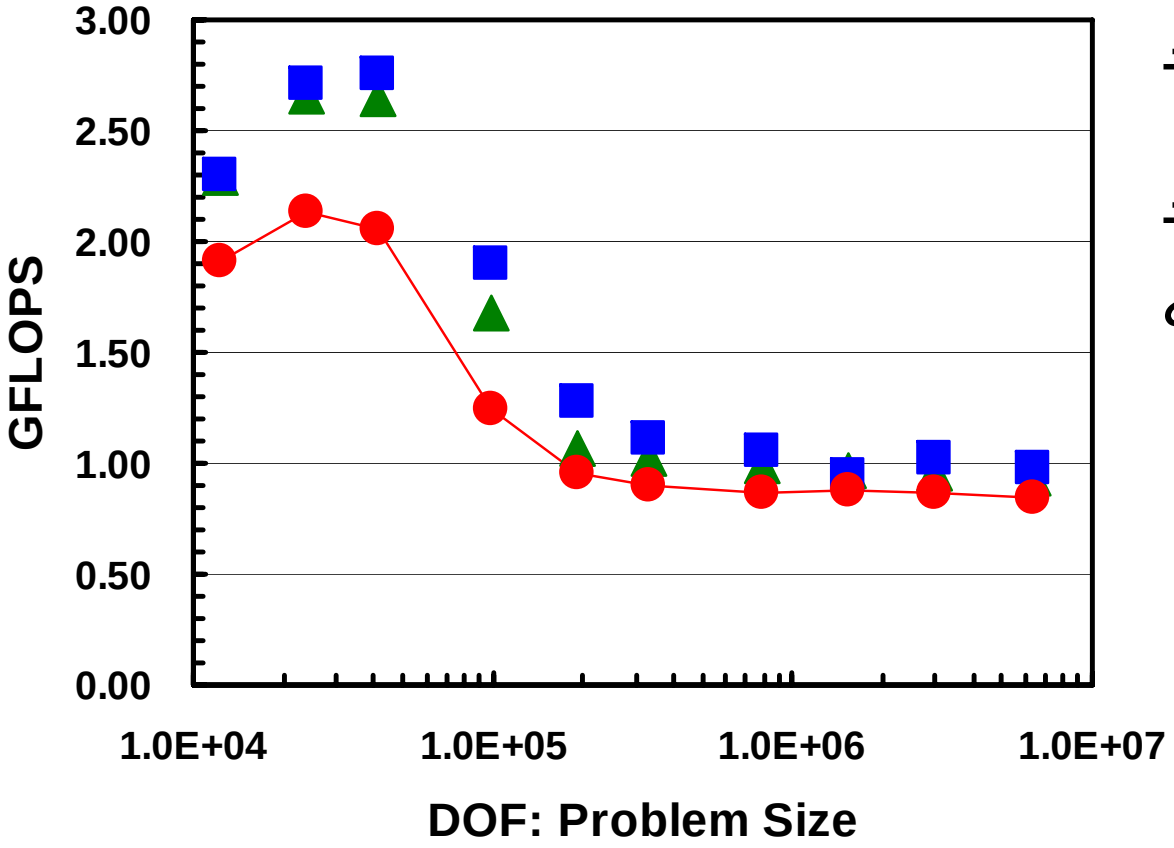


Effect of Re-Ordering

Results on 8 PEs

IBM-SP3 (Seaborg@NERSC)



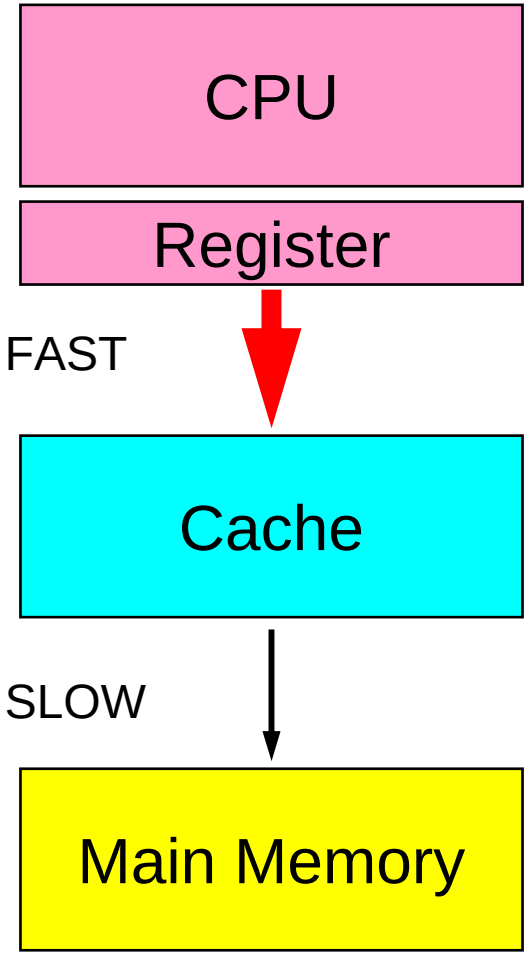


データ格納法の影響小

データサイズが小さいほど性能が高い

Super-Linearの生じる理由

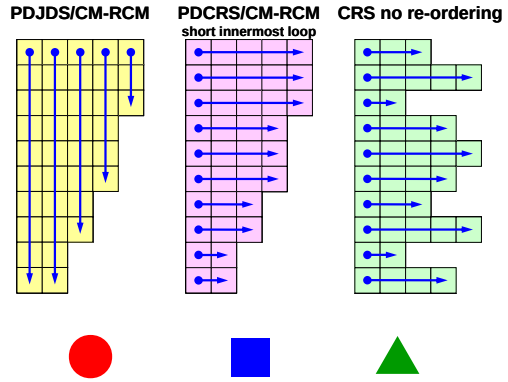
- キャッシュの影響
- スカラープロセッサでは，キ
全般に問題規模が小さいほ
ど性能が高い。
  - キャッシュの有効利用
- プロセッサ数の増加によって，
1プロセッサあたりの問題規
模が小さくなると，その分
キャッシュを有効利用できる
可能性がある。
  - PE数が更に増加すると通信
の影響が出る。

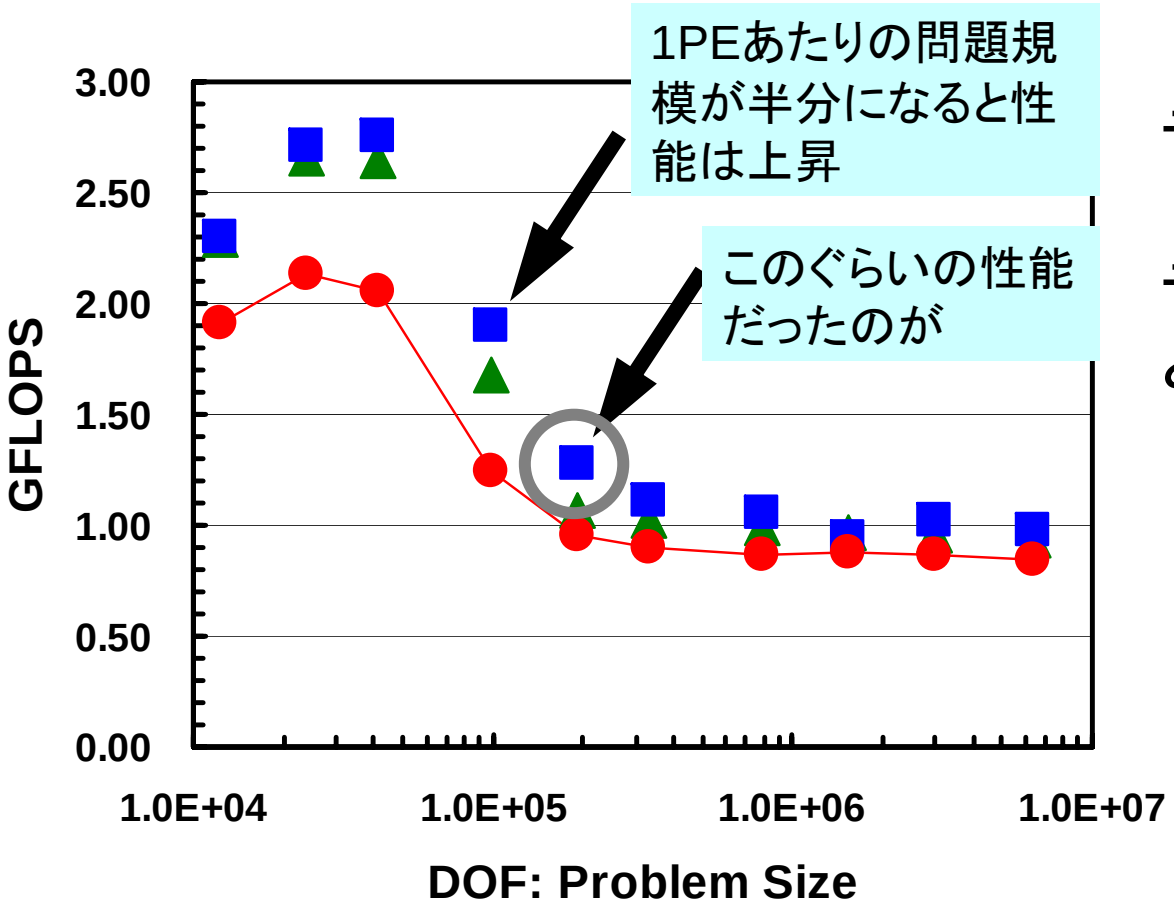


Effect of Re-Ordering

Results on 8 PEs

IBM-SP3 (Seaborg@NERSC)



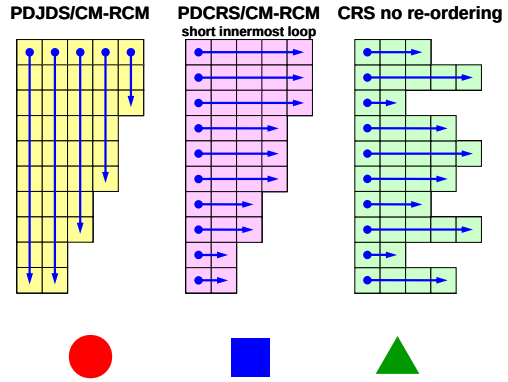


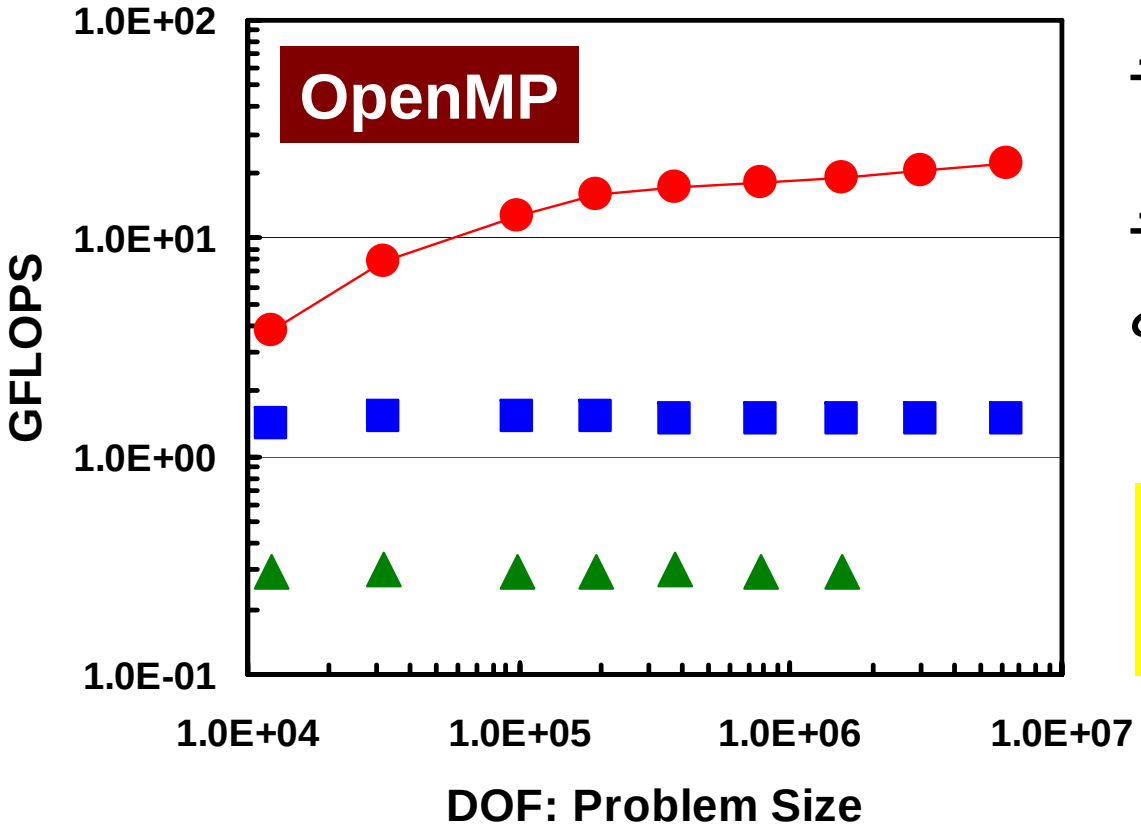
データ格納法の影響小

データサイズが小さいほど性能が高い

Effect of Re-Ordering

Results on 1 SMP node
Color #: 99 (fixed)
Earth Simulator





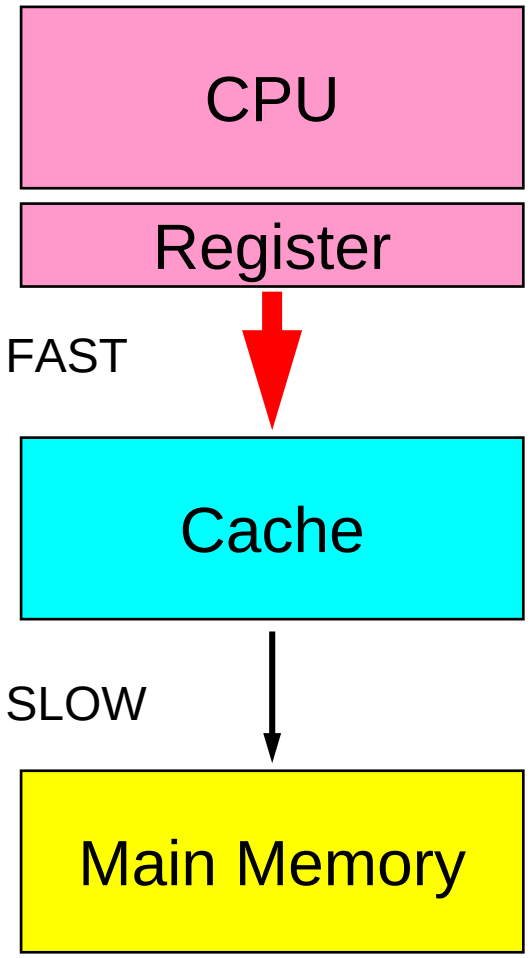
データ格納法の影響大

データサイズが大きいほど性能が高い

ベクトル機では原理的にそのようなこと（Super-Linear）は起こらない。

Super-Linearの「起こりやすさ」

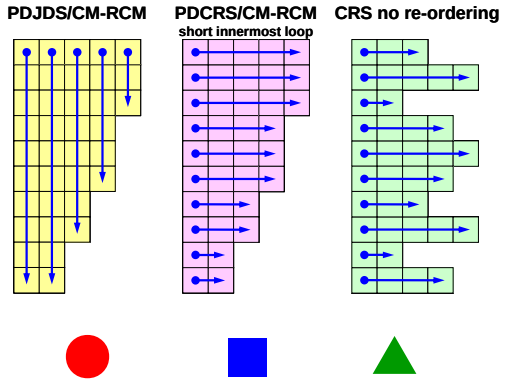
- キャッシュサイズが比較的大きい
  - IBM-SP3：8MB/PE
  - cenju：1MB/PE
- メインメモリーへのアクセススピードが遅い
  - 特にレイテンシ（立ち上がり）
- PE間通信が速い
- キャッシュが小さく，メモリのアクセススピードが速いOpteronではSuper-Linearは顕著ではない。

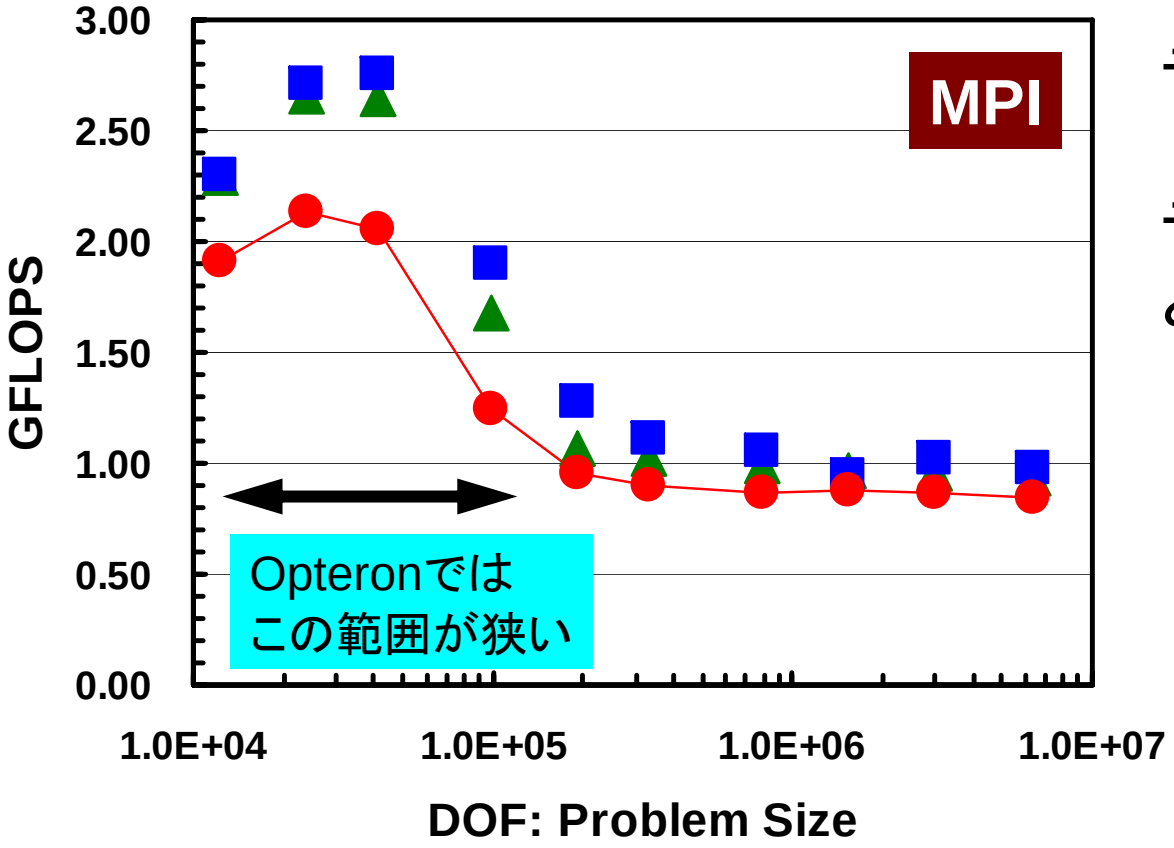


Effect of Re-Ordering

Results on 8 PEs

IBM-SP3 (Seaborg@NERSC)





データ格納法の影響小

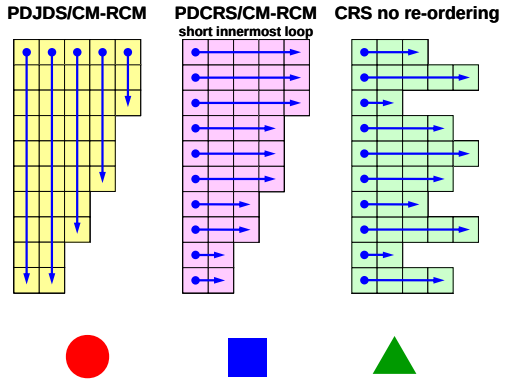
データサイズが小さいほど性能が高い

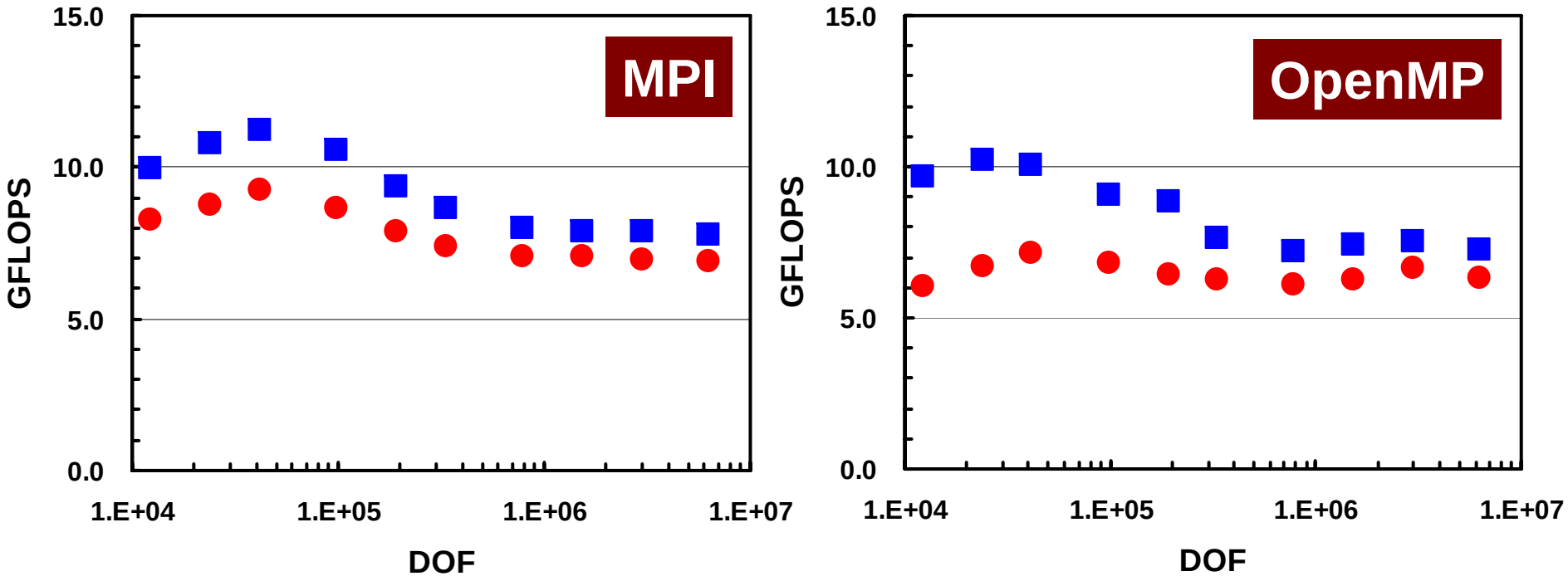
Effect of Re-Ordering

Results on 8 PEs

Hitachi SR11000/UTYO

今はもうちょっと速い



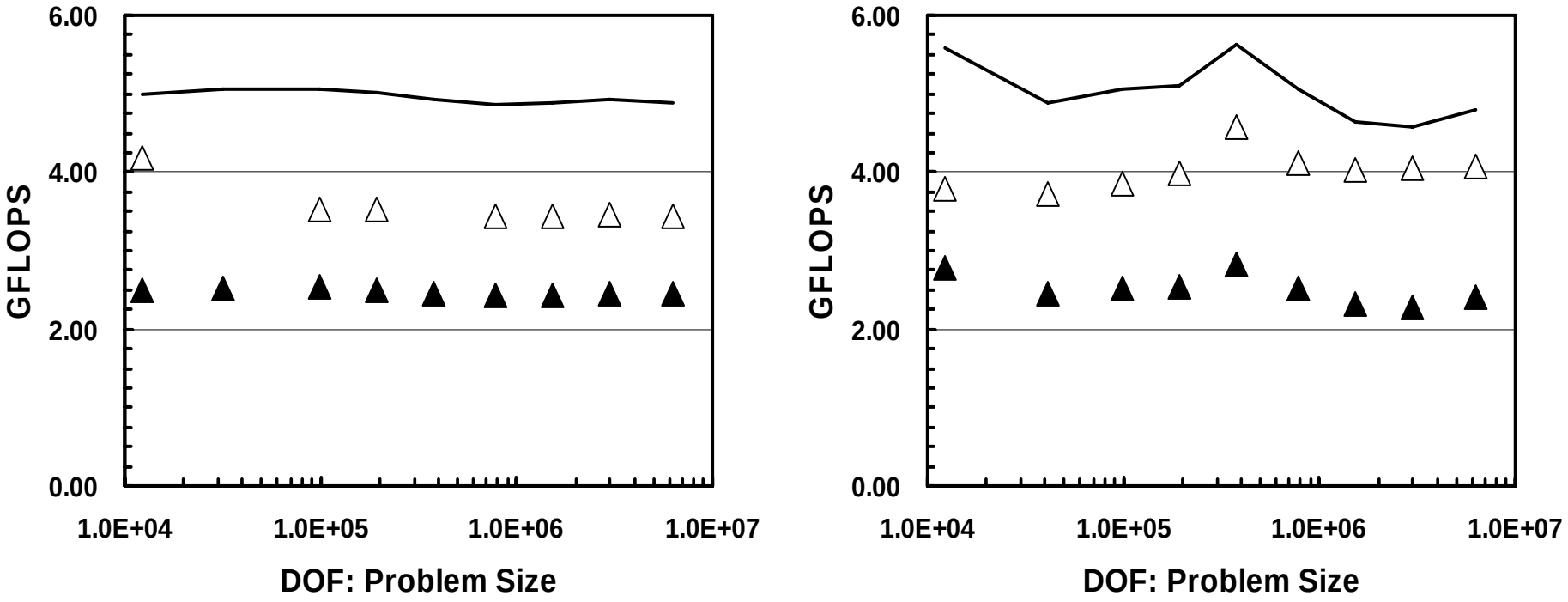


教育用PCクラスタの性能評価

Xeon 2.8 GHz
5.6 GFLOPS/PE

AMD Opteron 1.8 GHz
3.6 GFLOPS/PE

▲ 1-PE/node
△ 2-PE/node
（実線は▲を2倍した値）



最近の傾向：キャッシュサイズ小，メモリレイテンシ小，メモリバンド幅大
キャッシュからはあふれるがその影響は小

終わりに （1/2）

- 今学期の授業で並列計算に関わる一通りのことは教えたつもり。当然ではあるが，実際の計算，シミュレーションに使ってこそ意味がある。
  - 自分でプログラミングをやりながら復習してもらいたい。
- 一言で「並列化」と言っても対象とするアプリケーションによって様々なアプローチがある。
  - データ構造＝アルゴリズム　の重要性
  - 「要所」さえ押さえれば決して難しいものではない
- 諸兄の今後の研究に役立つことがあれば幸いである。

終わりに （2/2）

- 後期（先端計算機実習II）
  - 水曜 1445－1615
  - OpenMP，オーダリング，マルチグリッド
  - Cenju，Hitachi SR11000
  - 個別指導