



並列プログラミング言語XcalableMP 実習2

Coarray機能について

中尾昌広

目次

- Coarray機能について（復習）
- 【演習1】 スカラ変数のPut/Get（実行のみ）
- 【演習2】 配列のPut/Get（実行のみ）
- 【演習3】 行列積の作成（プログラミング演習+実行）

Coarray機能 (XMP/Fortran)

- ローカルデータに対する片側通信 (Get/Put) の記述

ノード2が持つb(3:5)のデータを
node1がa(1:3)にGetする

```
integer :: a(10)
integer :: b(10)[*] ! b is a coarray
:
if(me .eq. 1) then
  a(1:3) = b(3:5)[2] ! Get
```

[]はノード番号を表す

node 1



a(10)

b(10)

node 2



a(10)



3

5

Coarray機能 (XMP/C)

- ローカルデータに対する片側通信 (Get/Put) の記述

ノード2が持つb[3:3]のデータを
node1がa[0:3]にGetする

```
int a[10];  
int b[10]:[*]; // b is a coarray  
:  
if(me == 1)  
    a[0:3] = b[3:3]:[2]; // Get
```

コロンの後の[]はノード番号を表す

配列名[開始要素：転送要素数]

node 1



a[10]

b[10]

node 2



a[10]



3

5

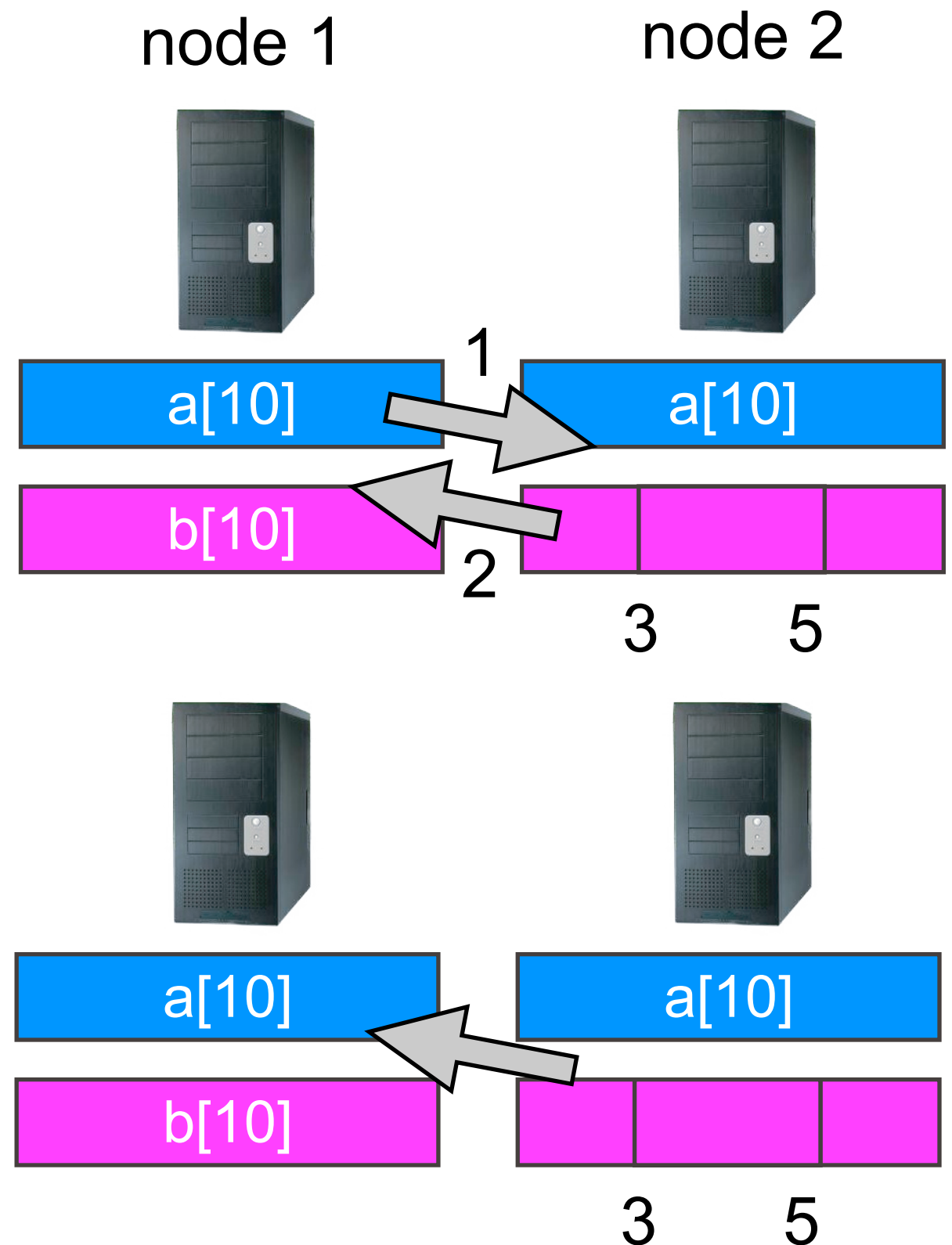
Get/Put in XMP/Fortran

● Get

```
integer :: a(10), b(10)[*]  
:  
if(me .eq. 1)  
  a(1:3) = b(3:5)[2]    ! Get
```

● Put (一般にGetよりも高速)

```
integer :: a(10)[*], b(10)  
:  
if(me .eq. 2)  
  a(1:3)[1] = b(3:5)    ! Put
```



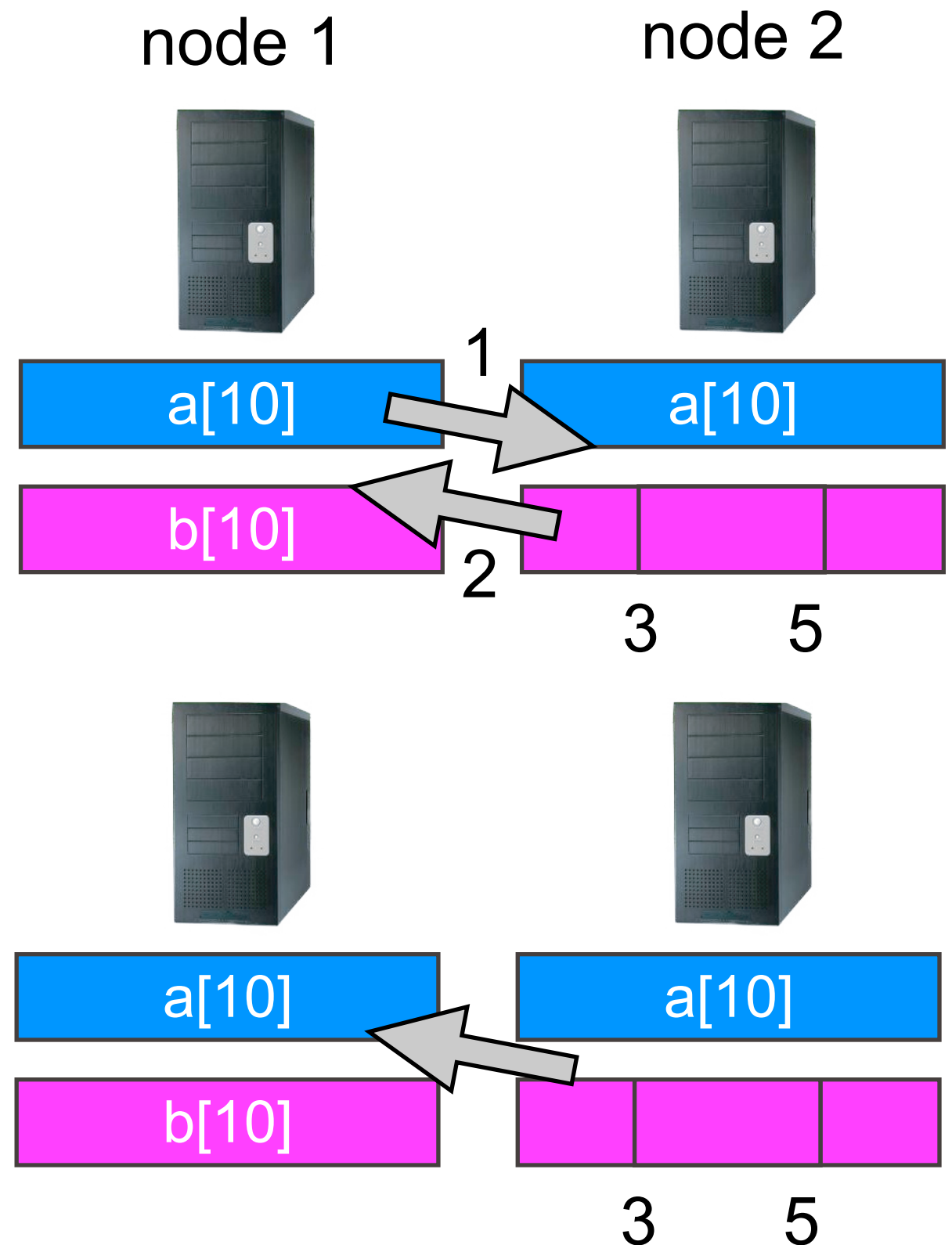
Get/Put in XMP/C

● Get

```
int a[10], b[10]:[*];  
:  
if(me == 1)  
    a[0:3] = b[3:3]:[2];    // Get
```

● Put (一般にGetよりも高速)

```
int a[10]:[*], b[10];  
:  
if(me == 2)  
    a[0:3]:[1] = b[3:3];    // Put
```



同期 : sync all

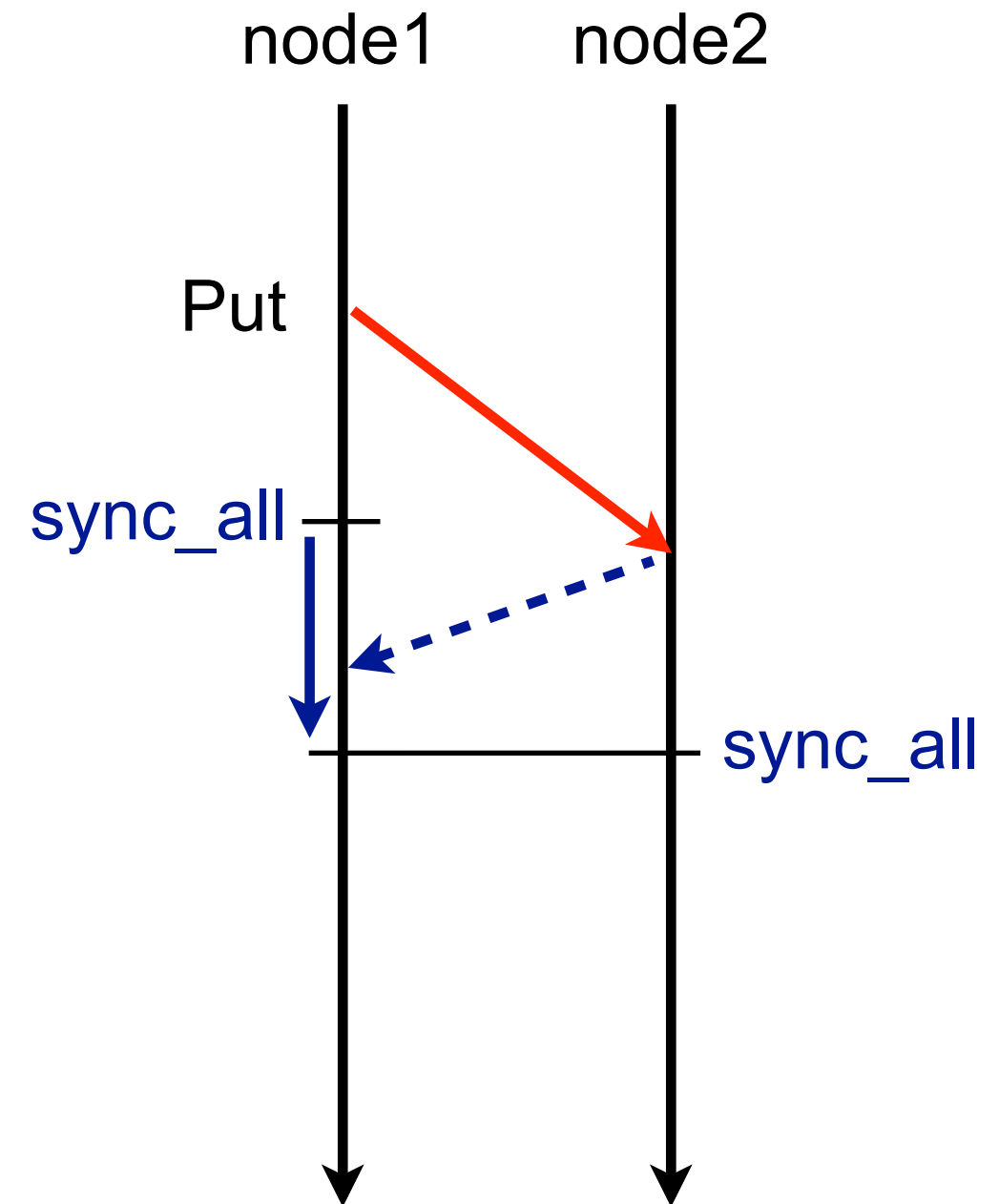
- XMP/Fortran

```
sync all
```

- XMP/C

```
void xmp_sync_all(int *status)
```

今までに発行した全片側通信が終了し、
かつバリア同期を行う



実習1

- スカラ変数のPut/Get
 - xmpcc coarray_scalar.c or xmpf90 coarray_scalar.f90
 - 2プロセスで実行して下さい

XMP/Fortran

```
if(me == 1){  
    tmp = val[2] ! Get  
    val[2] = val ! Put  
}  
sync all
```

XMP/C

```
if(me == 1){  
    tmp = val:[2]; // Get  
    val:[2] = val; // Put  
}  
xmp_sync_all(&status);
```

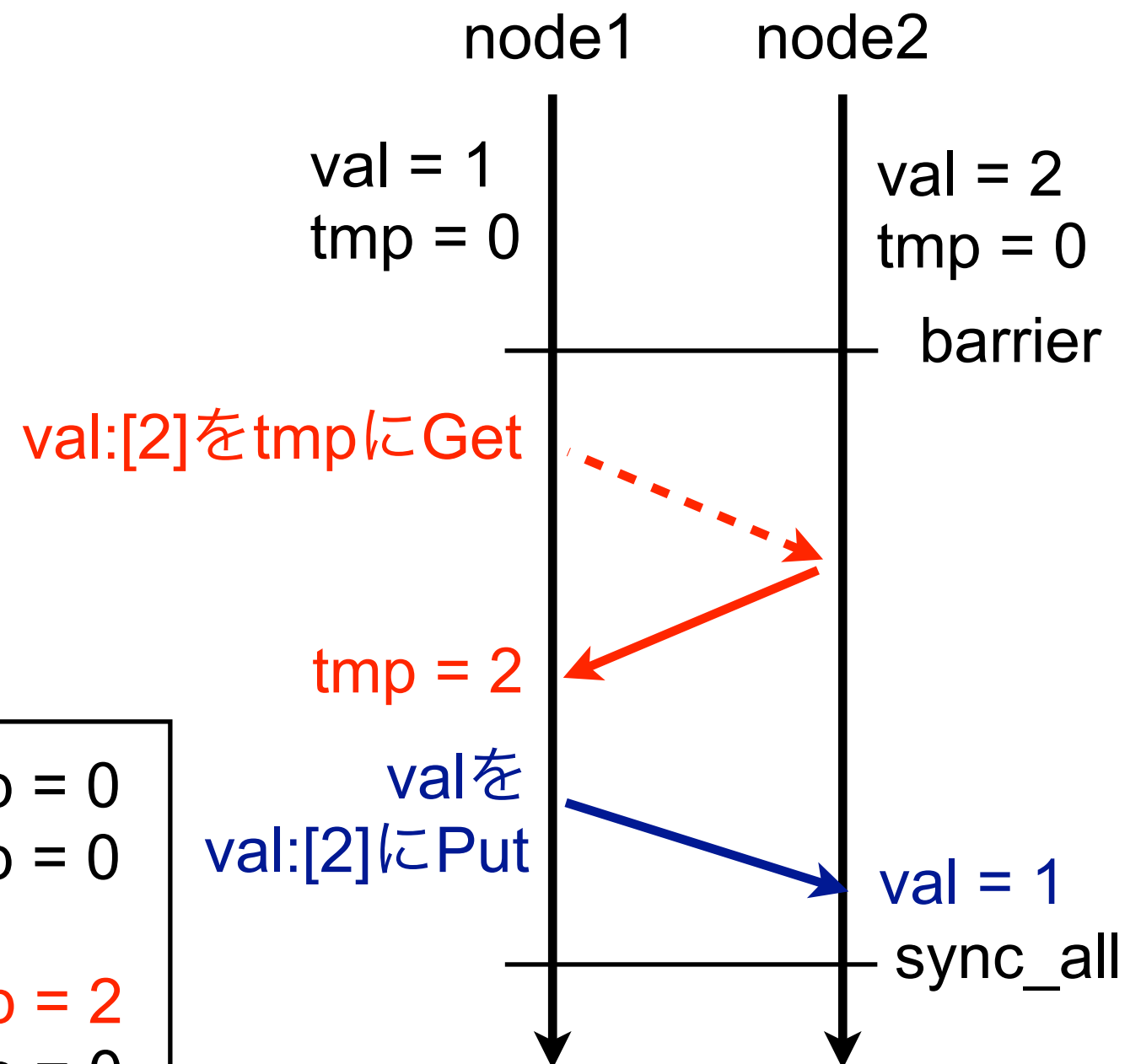

実習1 (結果)

```
#pragma xmp barrier
if(me == 1){
    tmp = val:[2]; // Get
    val:[2] = val; // Put
}
xmp_sync_all(&status);
```

出力結果：

```
[START] My node is 1, val = 1 tmp = 0
[START] My node is 2, val = 2 tmp = 0

[END]    My node is 1, val = 1 tmp = 2
[END]    My node is 2, val = 1 tmp = 0
```



実習2

- 配列のPut/Get
 - xmpcc coarray_vector.c or xmpf90 coarray_vector.f90
 - 2プロセスで実行して下さい
 - 配列の初期値は下記の通り

node 1

```
a[10] = {0, 1, .., 9};  
b[10] = {0, 1, .., 9};  
c[10][10] = {{0, 1, .., 9},  
              {10, 11, .., 19},  
              ...  
              {90, 91, .., 99}};
```

node 2

```
a[10] = {10, 11, .., 19};  
b[10] = {10, 11, .., 19};  
c[10][10] = {{100, 101, .., 109},  
              {110, 111, .., 119},  
              ...  
              {190, 191, .., 199}};
```

実習2 (結果1)

XMP/Fortran

```
if(me .eq. 1) then
  a(1:3) = a(6:8)[2]  ! Get
end if
```

XMP/C

```
if(me == 1){
  a[0:3] = a[5:3]:[2];  // Get
}
```

My node is 1:

a[0] = 15

a[1] = 16

a[2] = 17

a[3] = 3

a[4] = 4

a[5] = 5

a[6] = 6

a[7] = 7

a[8] = 8

a[9] = 9

実習2 (結果2)

XMP/Fortran

```
if(me .eq. 1) then  
    b(1:10:2) = b(1:10:2)[2]    ! Get  
end if
```

XMP/C

```
if(me == 1){  
    b[0:5:2] = b[0:5:2]:[2];    // Get  
}
```

My node is 1:

b[0] = 10

b[1] = 1

b[2] = 12

b[3] = 3

b[4] = 14

b[5] = 5

b[6] = 16

b[7] = 7

b[8] = 18

b[9] = 9

実習2 (結果3)

XMP/Fortran

```
if(me .eq. 1) then  
    c(1:5,1:5)[2] = c(1:5,1:5)  // Put  
end if
```

XMP/C

```
if(me == 1){  
    c[0:5][0:5]:[2] = c[0:5][0:5];  // Put  
}
```

My node is 2:

0	1	2	3	4	105	106	107	108	109
10	11	12	13	14	115	116	117	118	119
20	21	22	23	24	125	126	127	128	129
30	31	32	33	34	135	136	137	138	139
40	41	42	43	44	145	146	147	148	149
150	151	152	153	154	155	156	157	158	159
160	161	162	163	164	165	166	167	168	169
170	171	172	173	174	175	176	177	178	179
180	181	182	183	184	185	186	187	188	189
190	191	192	193	194	195	196	197	198	199

行列積（演習）

- $N \times N$ の行列積

- $C = A \times B$

- 並列化の方針

- 4ノードに分割して実行する
- ノード1がAとBのデータのすべてを持っている
- 最終的な解はノード1が出力する
- Coarrayを用いてデータの転送を行う

- 発展課題（今回はしません）

- Cannonとfoxのアルゴリズム
 - ・ <http://www.kata-lab.itc.u-tokyo.ac.jp/OpenLecture/SP20101221.pdf>
- 第三回 XMP Challenge
 - ・ <http://xcalablemp.org/ja/challenge-3rd.html>

```
for(i=0;i<N;i++)  
    for(j=0;j<N;j++)  
        for(k=0;k<N;k++)  
            c[i][k] += a[i][j] * b[j][k];
```

行列積（演習）

- 部分行列に分けて計算する（行と列に2分割ずつ。Nは偶数）

C_{00}	C_{01}
C_{10}	C_{11}

 $=$

A_{00}	A_{01}
A_{10}	A_{11}

 \times

B_{00}	B_{01}
B_{10}	B_{11}

$$C_{00} = A_{00} \times B_{00} + A_{01} \times B_{10} \longleftarrow \text{node 1}$$

$$C_{01} = A_{00} \times B_{01} + A_{01} \times B_{11} \longleftarrow \text{node 2}$$

$$C_{10} = A_{10} \times B_{00} + A_{11} \times B_{10} \longleftarrow \text{node 3}$$

$$C_{11} = A_{10} \times B_{01} + A_{11} \times B_{11} \longleftarrow \text{node 4} \quad \text{が担当する}$$

行列積（演習）

matmul.c or matmul.f90を元に作成してください。

実装箇所は、下記の赤の部分のみです。

1. （比較のための）逐次の行列積
2. 配列の初期化（関数init_dmat()）
3. Coarrayを用いた配列の転送（関数move_data()）
 - 部分配列AとBを、ノード1から各ノードに転送する
 - XMP/Cにおいてnode2が必要なのは、 $A[0:N/2][0:N]$ と $B[0:N][N/2:N/2]$
 - XMP/Fortranにおいてnode2が必要なのは、 $A(1:N, 1:N/2)$ と $B(N/2+1:N, 1:N)$
4. 行列積の実行（ $C = A \times B$ ，関数mul_dmat()）
5. 各ノードからCの結果を集める（関数gather_data()）
6. 妥当性の検証（関数verify()）
 - verifyの値が逐次版と同じ結果ならOK，計測時間を較べてみて下さい