GIẢI THUẬT VÀ CÀI ĐẶT
CHƯƠNG TRÌNH SONG SONG

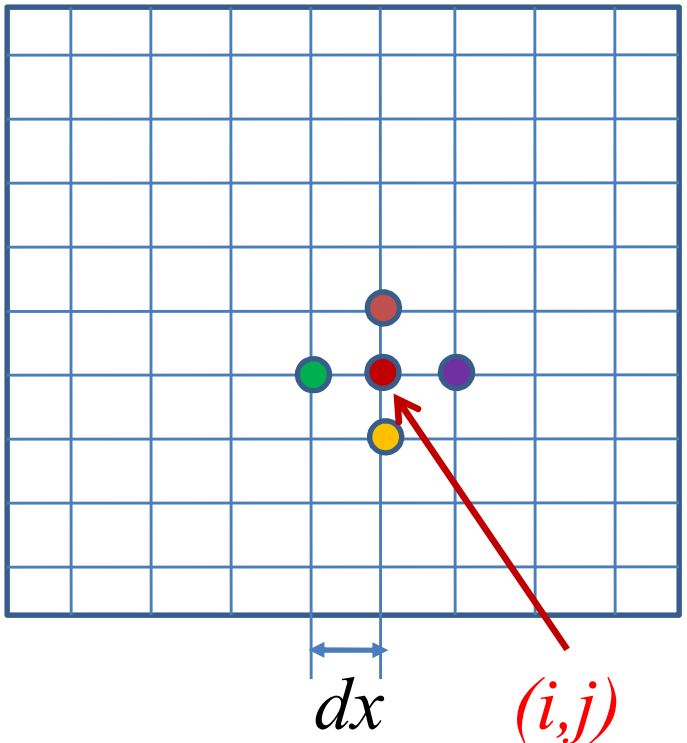
BÀI TOÁN CÓ SỰ PHỤ THUỘC DỮ LIỆU
PHƯƠNG TRÌNH NHIỆT

Mô hình toán học và Phương pháp giải

- Phương trình nhiệt (PDE):

$$\frac{\partial C}{\partial t} = D\nabla^2 C$$

$$\nabla^2 C = \frac{\partial^2 C}{\partial x^2} + \frac{\partial^2 C}{\partial y^2}$$



- Công thức giải
  - Khởi tạo giá trị ban đầu: $C^0_{i,j}$
  - Tại bước n+1:

$$\nabla^2 C^{tn}_{i,j} = FD^{tn}_{i,j} = \frac{\left(C^{tn}_{i+1,j} + C^{tn}_{i-1,j} + C^{tn}_{i,j+1} + C^{tn}_{i,j-1} - 4C^{tn}_{i,j}\right)}{dx^2}$$

$$C^{tn+1}_{i,j} = C^{tn}_{i,j} + dt * D * FD^{tn}_{i,j}$$

- Sự phụ thuộc dữ liệu?

Sự phụ thuộc dữ liệu

- Việc tính toán tại một điểm lưới (i,j) cần thông tin tại những điểm lưới xung quanh: (i-1,j), (i+1,j) , (i,j-1) , (i,j+1) gọi là sự phụ thuộc dữ liệu.
  - Đối với chương trình tuần tự: Cần điều kiện biên
  - Đối với chương trình song song (sử dụng mô hình bộ nhớ phân tán): Cần trao đổi thông tin (truyền thông) giữa các CPU, cần đồng bộ trong tính toán.

Mô hình toán học và Phương pháp giải

- Ký hiệu:

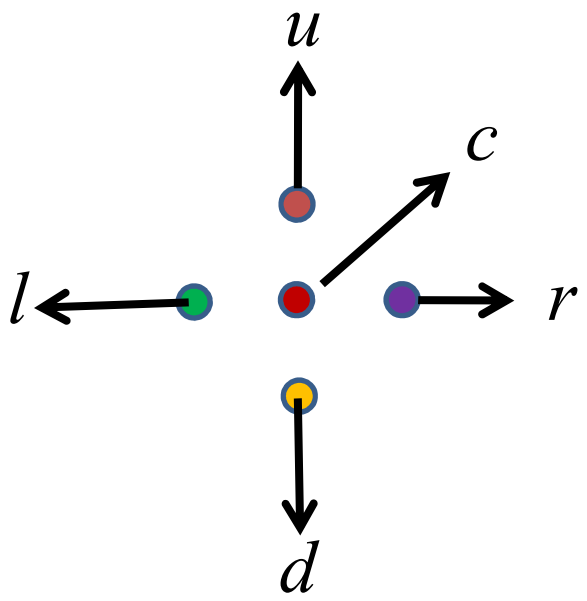
c : $C_{i,j}$,

u : $C_{i-1,j}$,

d : $C_{i+1,j}$,

l : $C_{i,j-1}$,

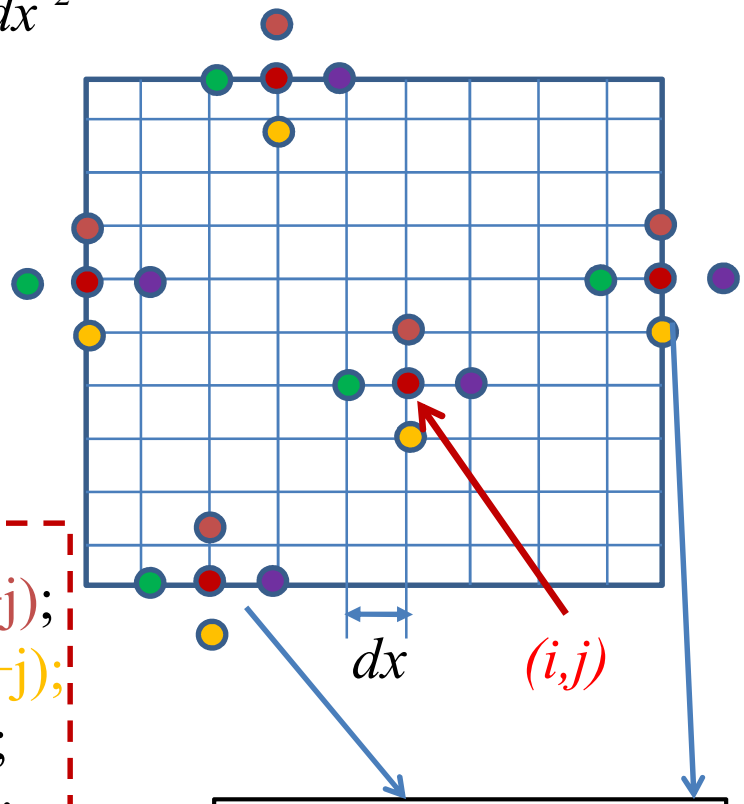
r : $C_{i,j+1}$.



Cài đặt hàm rời rạc hóa theo không gian: FD

$$\nabla^2 C_{i,j}^{tn} = FD_{i,j}^{tn} = \frac{\left(C_{i+1,j}^{tn} + C_{i-1,j}^{tn} + C_{i,j+1}^{tn} + C_{i,j-1}^{tn} - 4C_{i,j}^{tn}\right)}{dx^2}$$

```
void FD(float *C, float *dC) {
int i, j;
float c,u,d,l,r;
for (  i = 0 ; i < m ; i++ )
  for ( j = 0 ; j < n ; j++ )
    {
      c = *(C+i*n+j);
      u = (i==0)     ? *(C+i*n+j) : *(C+(i-1)*n+j);
      d = (i==m-1) ? *(C+i*n+j) : *(C+(i+1)*n+j);
      l = (j==0)     ? *(C+i*n+j) : *(C+i*n+j-1);
      r = (j==n-1)  ? *(C+i*n+j) : *(C+i*n+j+1);
      *(dC+i*n+j) = (1/(dx*dx))*(u+d+l+r-4*c);
    }
}
```



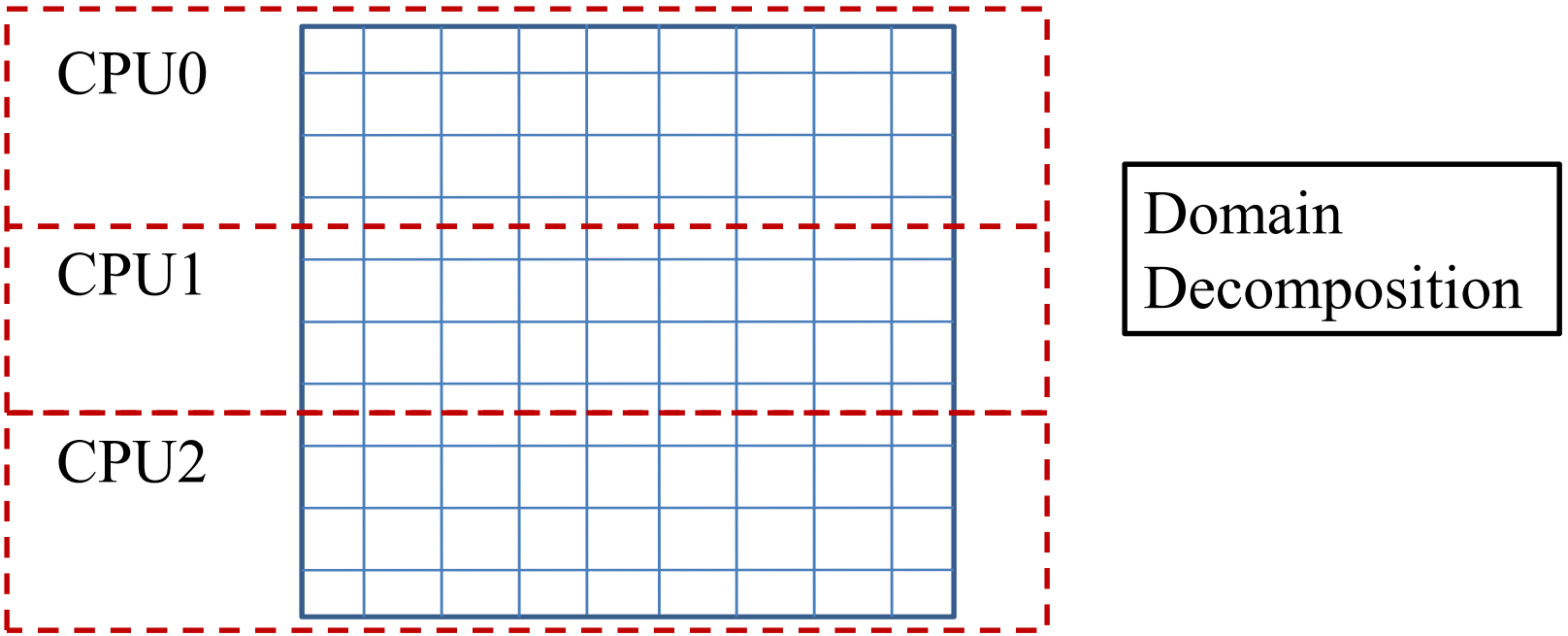
Cài đặt hàm tích hợp theo thời gian

$$C_{i,j}^{tn+1} = C_{i,j}^{tn} + dt * D * FD_{i,j}^{tn}$$

```
while (t<=T)
{
    FD(C, dC);
    for (  i = 0 ; i < m ; i++ )
        for ( j = 0 ; j < n ; j++ )
            *(C+i*n+j) = *(C+i*n+j) + dt*(*(dC+i*n+j));
     t=t+dt;
}
```

Giải thuật song song SPMD 1

- SPMD: Single Program Multiple Data



Giải thuật song song SPMD 2

- B1: Khởi tạo/nhập giá trị ban đầu (Input data)
  - Thông thường tại CPU0, gọi là Root
- B2: Chia miền tính toán
- B3: Gửi Input từ Root đến tất cả các CPU
- B4: Tính toán (Mỗi CPU tính toán trên subdomain tương ứng)
- B5: Các CPU gửi kết quả tính toán (Output) về Root

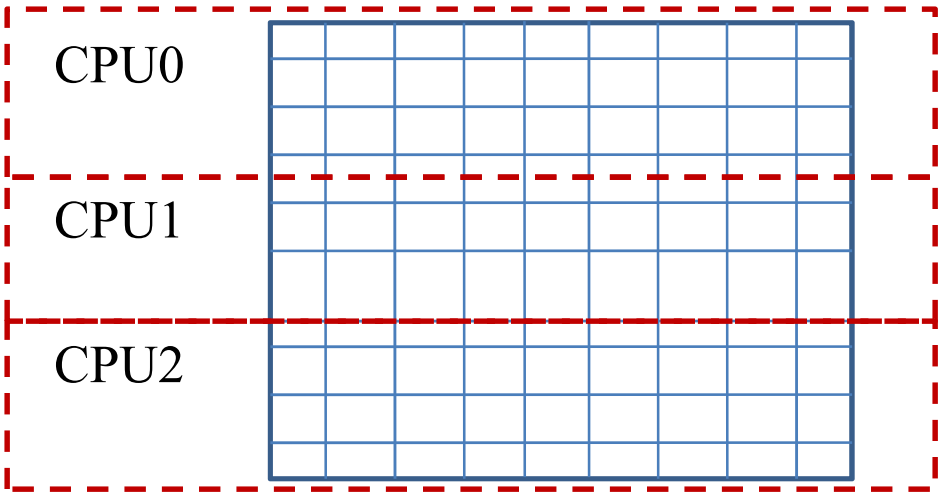
B3 và B5: Truyền thông (Input và Output)

Giải thuật song song SPMD 3

- B1: Khởi tạo giá trị ban đầu
  - Theo yêu cầu của từng bài toán
  - Ảnh hưởng đến kết quả đầu ra

Giải thuật song song SPMD 4

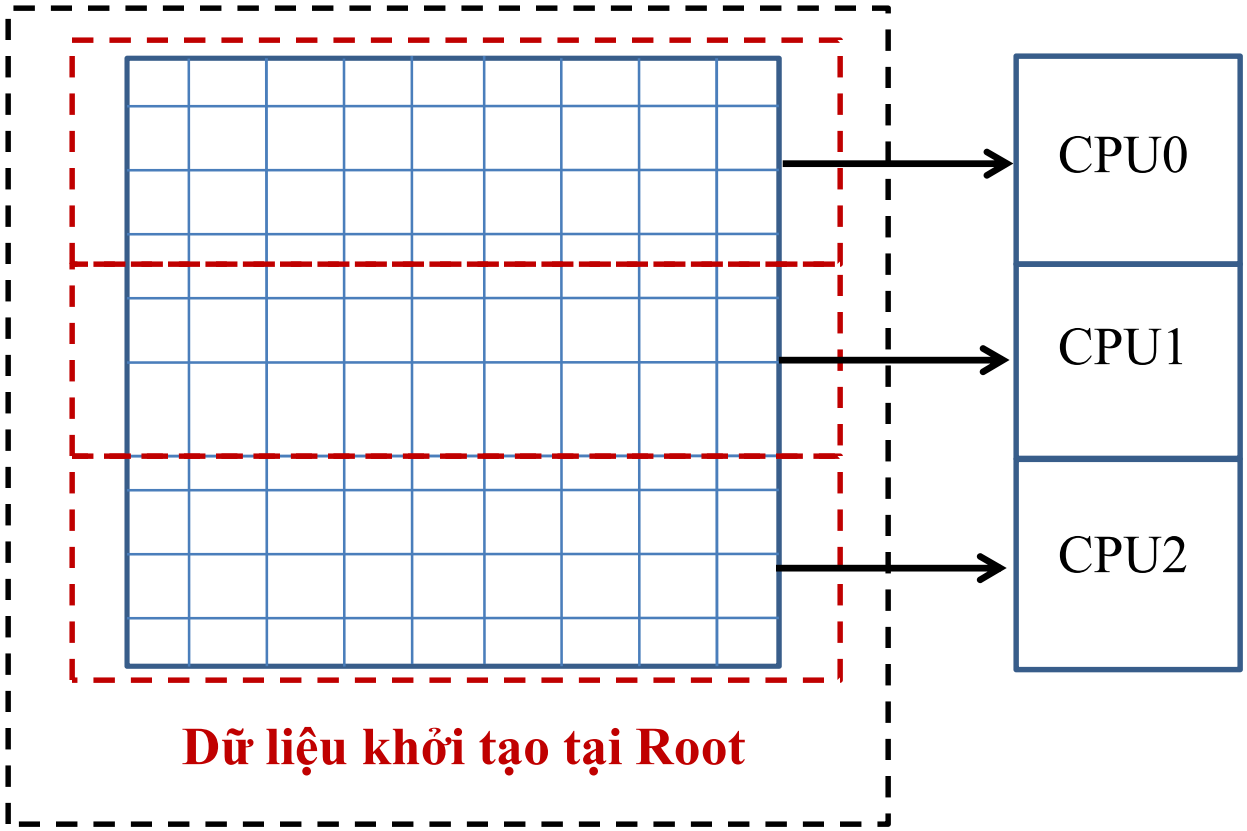
- B2: Chia miền tính toán
  - Có rất nhiều cách chia
  - Chọn cách chia đơn giản nhất là chia theo hàng
  - Giả sử kích thước toàn miền tính toán là: $m \times n$
  - Miền tính toán trên mỗi CPU là: $mc_x n$, trong đó: **mc=m/NP** với **NP** là số lượng CPU



Giải thuật song song SPMD 5

- B3: Gửi Input từ Root đến tất cả các CPU

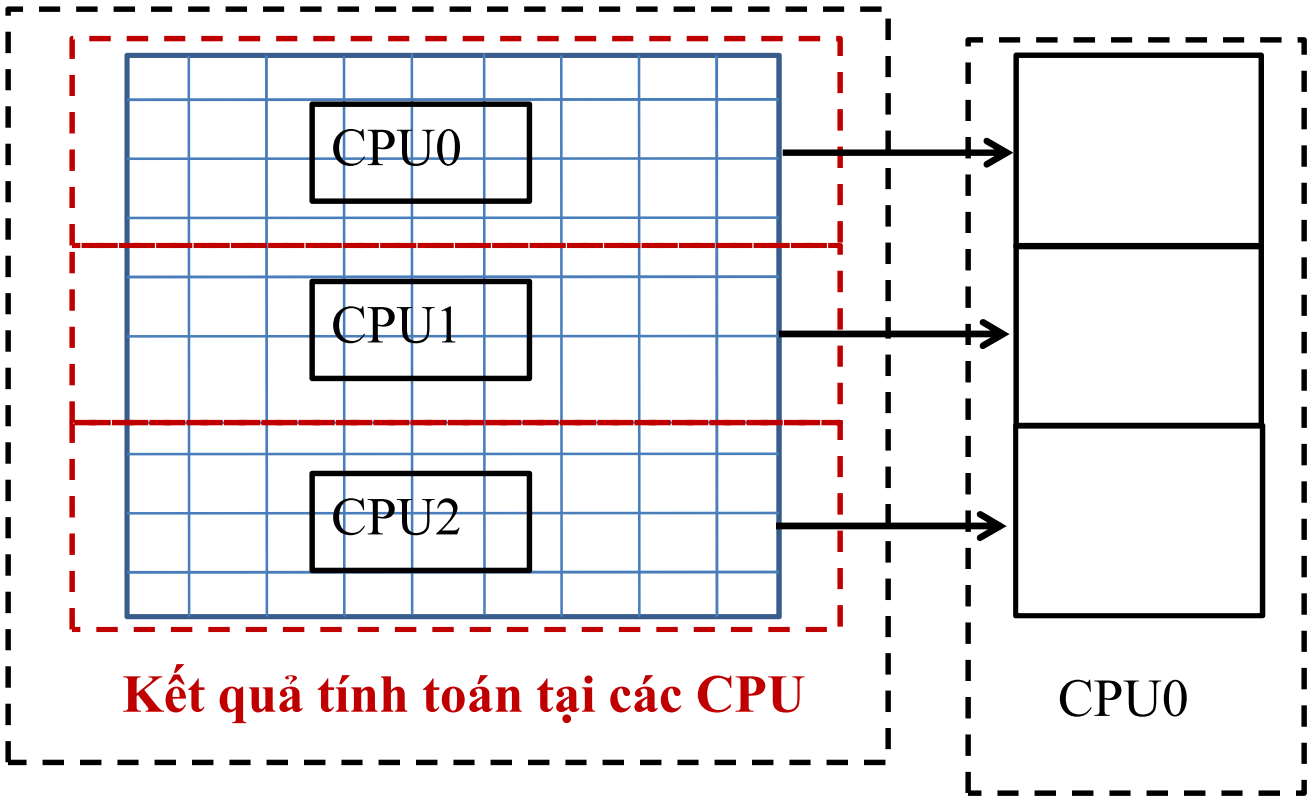
```
MPI_Scatter (C, mc*n, MPI_FLOAT,
             Cs, mc*n, MPI_FLOAT, 0,
             MPI_COMM_WORLD);
```



Giải thuật song song SPMD 6

- B5: Các CPU gửi kết quả tính toán (Output) về Root

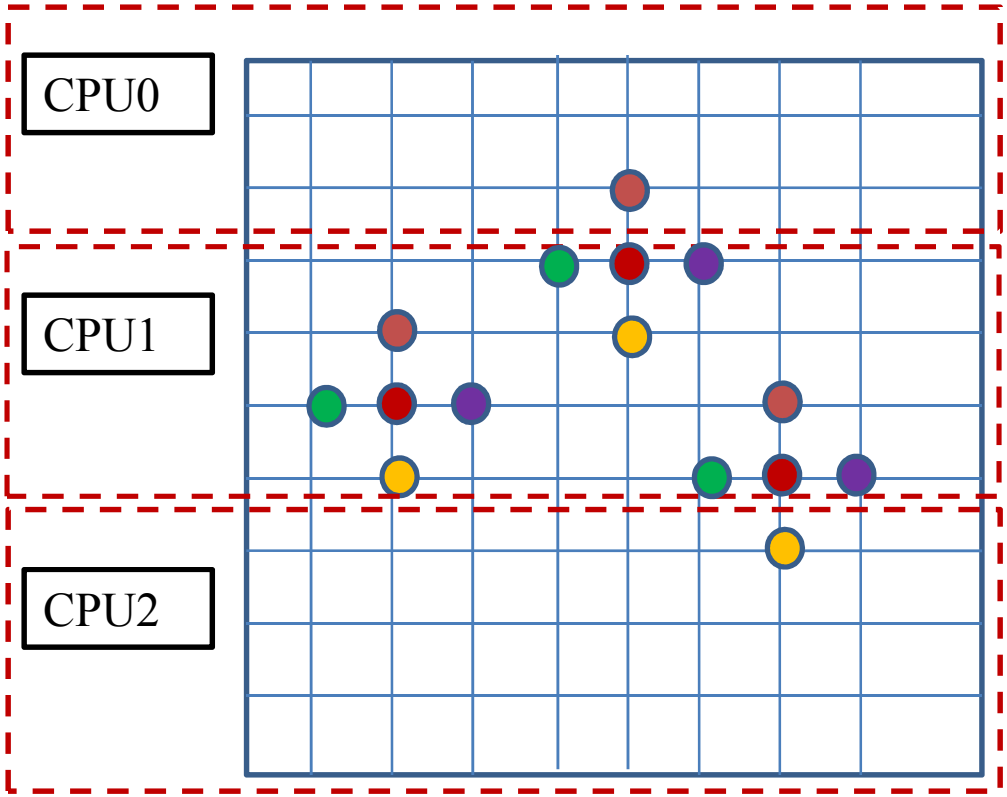
```
MPI_Gather ( Cs, mc*n, MPI_FLOAT,
             C, mc*n, MPI_FLOAT, 0,
             MPI_COMM_WORLD);
```



Giải thuật song song SPMD 7

- B4: Tính toán

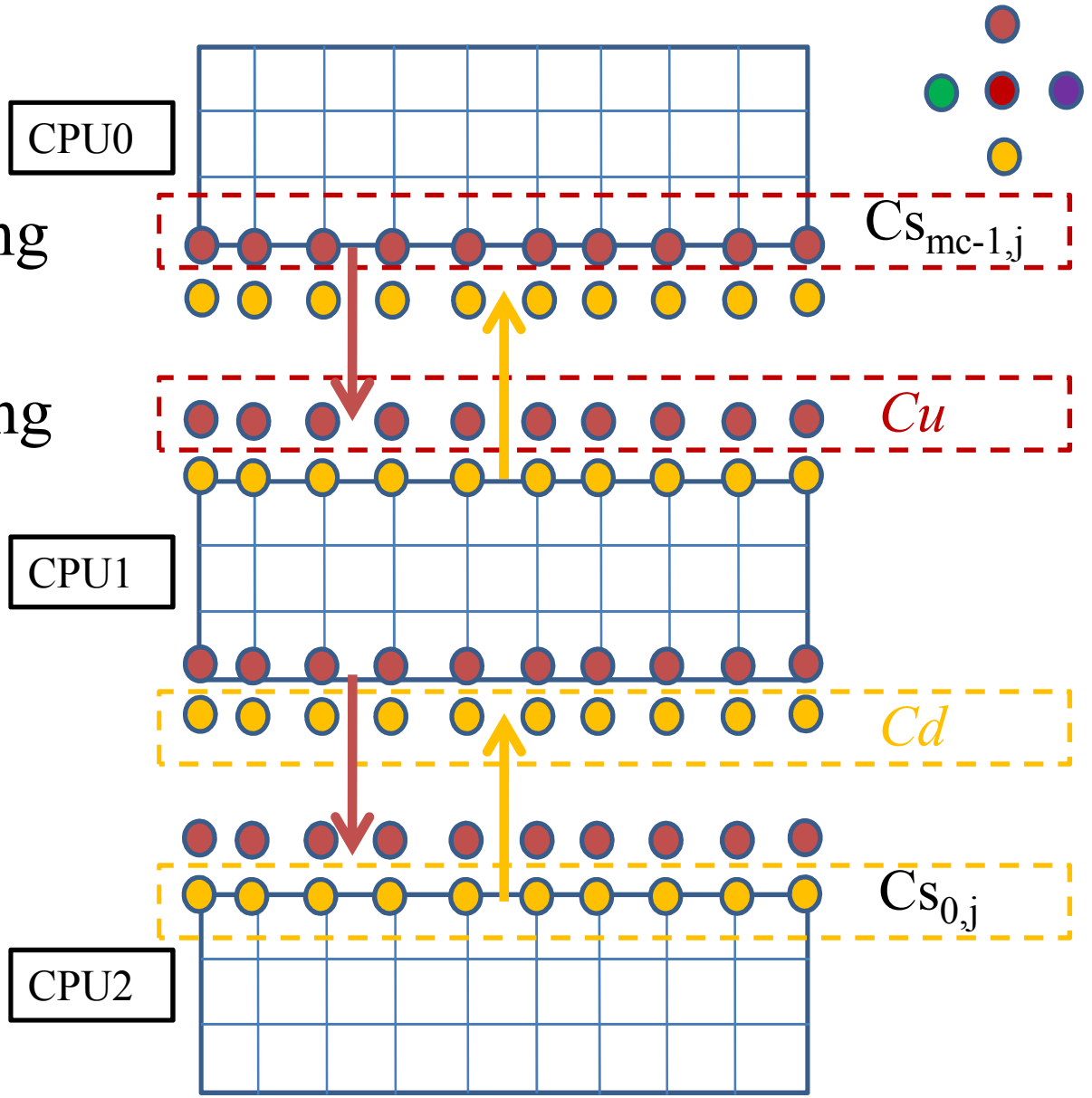
- *B4.1: Truyền thông*
- *B4.2: Tính toán*



Giải thuật song song SPMD 8

- B4.1: Truyền thông

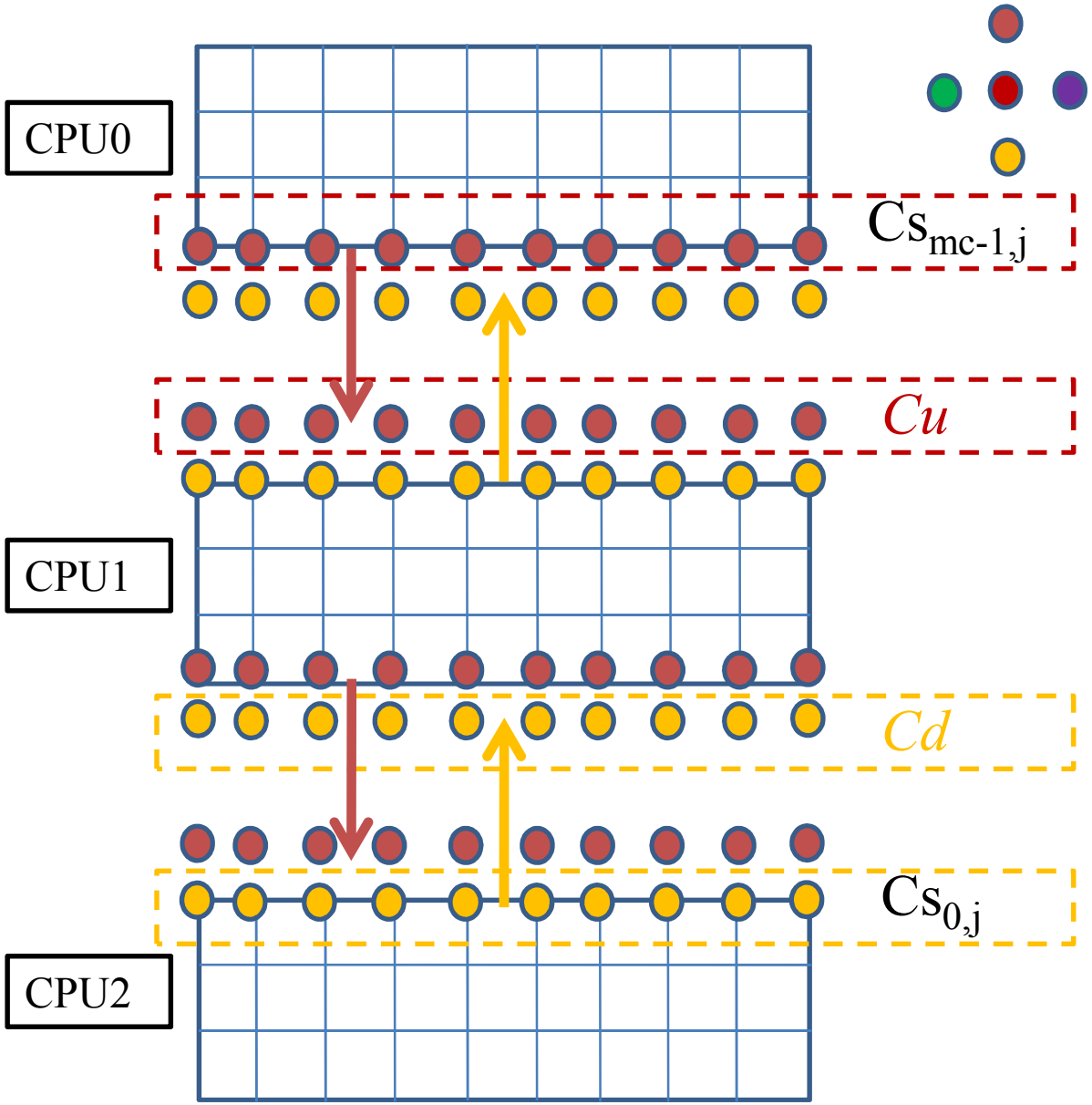
- B4.1a): Truyền thông mảng ***Cu***
- B4.1b): Truyền thông mảng ***Cd***



Giải thuật song song SPMD 9

- B4.1a): Truyền thông mảng ***Cu***

```
if (rank==0){
   for (j=0; j<n; j++)  *(Cu+j) = *(Cs+0*n+j);
   MPI_Send (Cs+(mc- )*n, n, MPI_FLOAT, rank+1, rank, …);
 } else if (rank==NP-1) {
   MPI_Recv (Cu, n, MPI_FLOAT, rank-1, rank-1, …);
 } else {
   MPI_Send (Cs+(mc-1)*n, n, MPI_FLOAT, rank+1, rank,…);
   MPI_Recv(Cu, n, MPI_FLOAT, rank-1, rank-1, …);
 }
```


CPU0
CPU1
CPU2
$Cs_{mc-1,j}$
Cu
Cd
$Cs_{0,j}$

Giải thuật song song SPMD 10

- B4.1b): Truyền thông mảng ***Cd***

```
if (rank==NP-1){
   for (j=0; j<n; j++)  *(Cd+j) = *(Cs+(mc-1)*n+j);
   MPI_Send (Cs, n, MPI_FLOAT, rank-1, rank, …);
 } else if (rank==0) {
   MPI_Recv (Cd, n, MPI_FLOAT, rank+1, rank+1, …);
 } else {
   MPI_Send (Cs, n, MPI_FLOAT, rank-1, rank, …);
   MPI_Recv (Cd, n, MPI_FLOAT, rank+1, rank+1, …);
 }
```

Giải thuật song song SPMD 11

- B4.2: Tính toán

```
void FD(float *Cs, float *Cu, float *Cd, float *dCs ,int ms) {
int i, j;
float c,u,d,l,r;
for (  i = 0 ; i < ms ; i++ )
  for ( j = 0 ; j < n ; j++ )
    {
     c = *(Cs+i*n+j);
     u = (i==0)          ?  *(Cu+j)      : *(Cs+(i-1)*n+j);
     d = (i==ms-1)       ?  *(Cd+j)      : *(Cs+(i+1)*n+j);
     l = (j==0)          ?  *(Cs+i*n+j) : *(Cs+i*n+j-1);
     r = (j==n-1)        ?  *(Cs+i*n+j) : *(Cs+i*n+j+1);
     *(dCs+i*n+j) = (D/(dx*dx))*(u+d+l+r-4*c);
    }
}
```

Một số ví dụ ứng dụng