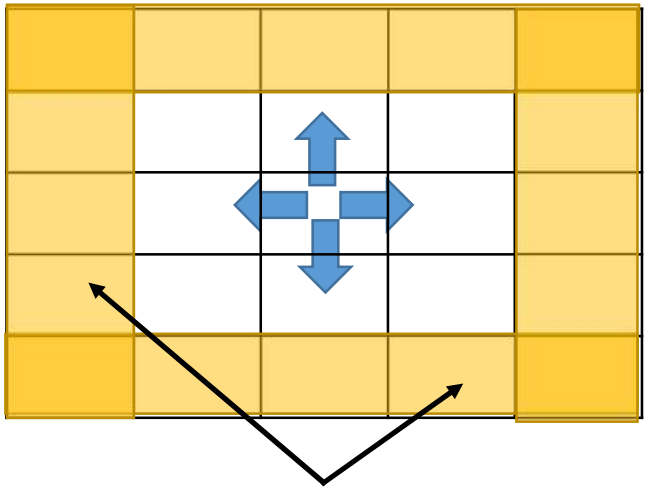
Jacobi MPI –example

Also known as 2D stencil or Laplace 2D operator

Jacobi Solver

- The Jacobi method is an iterative algorithm for solving a system of linear equations.
- In the 2D model an approximation can be made by taking the average of the 4 neighbouring values (4 point stencil).



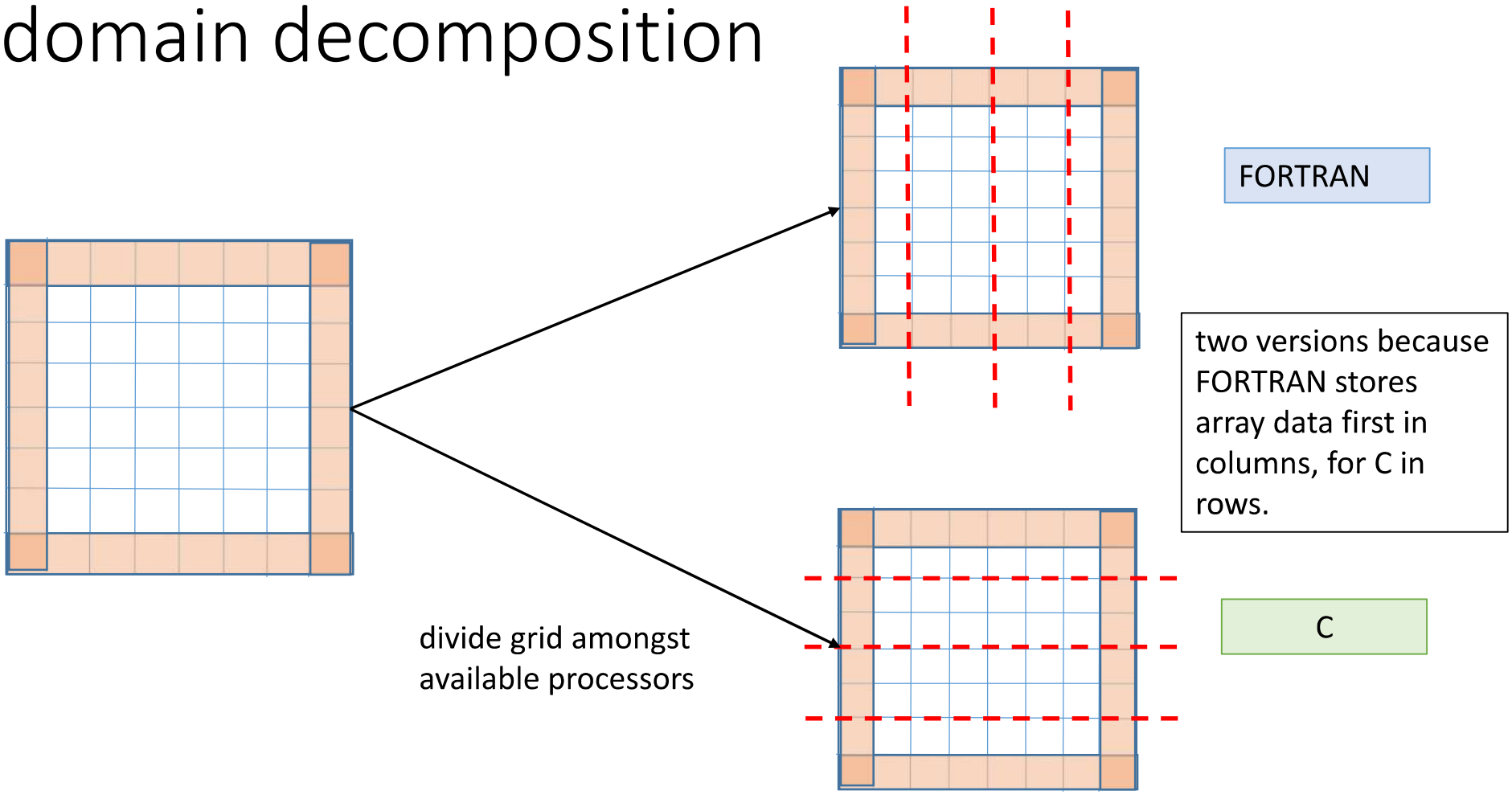

```
REAL A(0:n+1,0:n+1), B(1:n,1:n)
...
!Main Loop
DO WHILE(.NOT.converged)
   ! perform 4 point stencil
   DO j=1, n
      DO i=1, n
         B(i,j)=0.25*(A(i-1,j)+A(i+1,j)+A(i,j-
1)+A(i,j+1))
      END DO
   END DO

   ! copy result back into array A
   DO j=1, n
      DO i=1, n
         A(i,j) = B(i,j)
      END DO
   END DO
...
! convergence test
END DO
```

Jacobi in Parallel

- For simplicity, we will use 1D domain decomposition, dividing rows or columns of the grid among the MPI ranks.
- Since each rank will need data from neighbouring domains need to set up halo regions.
- For efficiency, exchange columns with Fortran and rows with C.
- As a first version, can use MPI_Send and MPI_Recv to exchange the data.

1D-domain decomposition



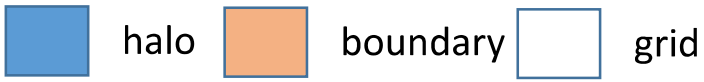
Halo exchange -Fortran Version

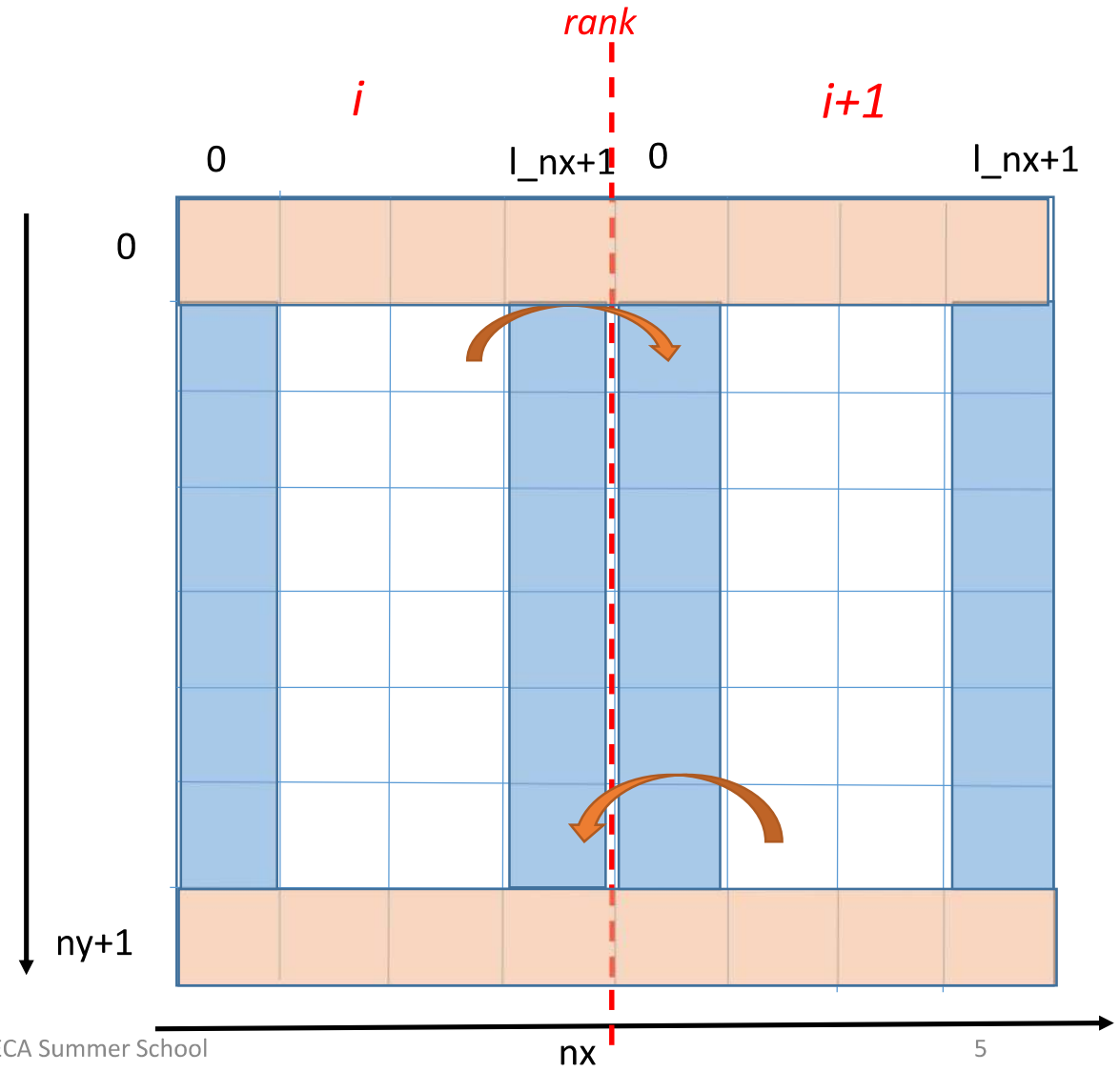
We start with a data grid(ny,nx) which becomes grid(0:ny+1,0:nx+1) when we include the boundaries.

Each local domain is (1..ny,1..l_{nx}) but with a halo region + boundaries we have

(0..ny+1,0..l_{nx}+1).

Ranks 0 and size-1 need only 1 halo region, since they must store the left and right border conditions.





Halo exchange C Version

For C we exchange rows instead of columns, such that the rows are divided among the MPI ranks.

MPI ranks 0 and size-1 need only 1 halo region (and only 2 transfers) because they contain the top and bottom boundary conditions.

Note we have inverted the nx and ny axes with respect to the FORTRAN version.