Parallelizing The Laplace Solver

Writing a parallel program step by step

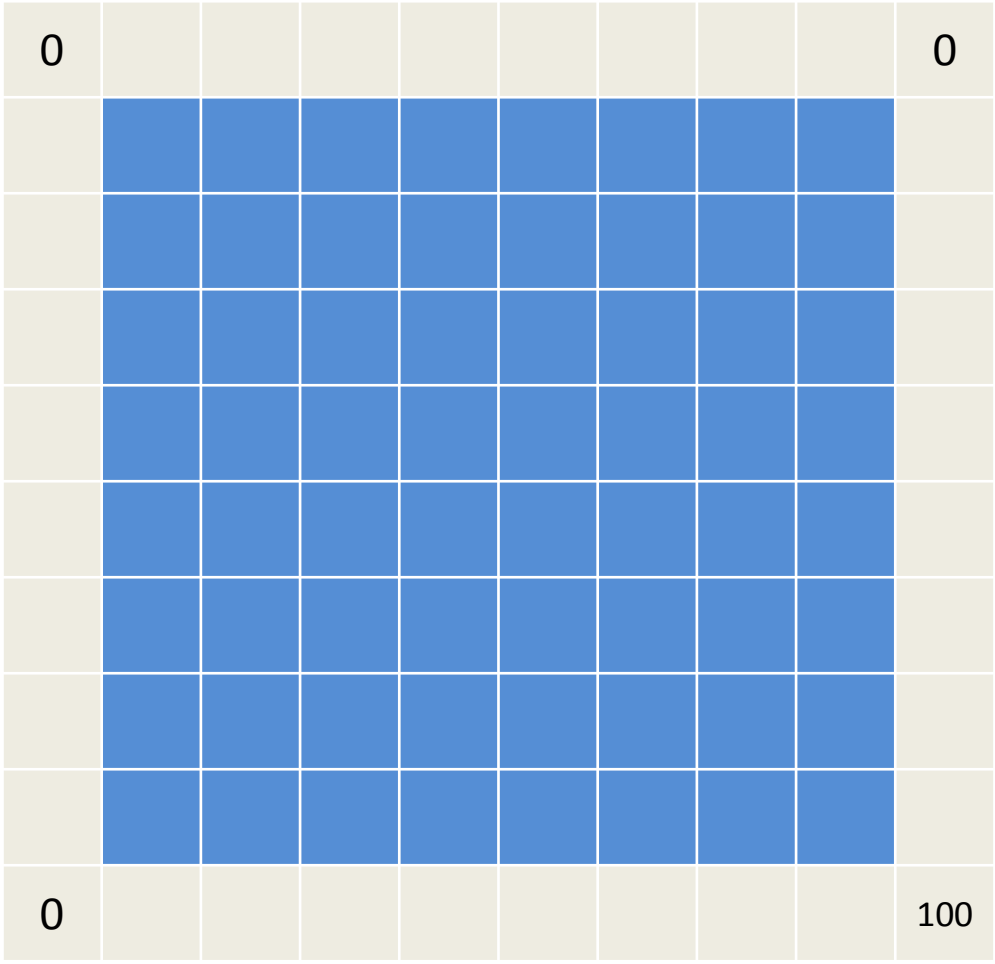
- Step 1. Start from serial programs as a baseline
  - Something to check correctness and efficiency against
- Step 2. Analyze and profile the serial program
  - Identify the “hotspot”
  - Identify the parts that can be parallelized
- Step 3. Parallelize code incrementally
- Step 4. Check correctness of the parallel code
- Step 5. Iterate step 3 and 4

```
//Read and validate command line arguments

//Initialize the arrays
For each element
   value=0

//Set up boundary conditions
For each boundary element
   value=something

// Main loop
For each iteration
   For each element
      Update the value with the 5-point stencil
      Calculate convergence error
   Find maximum convergence error
   If the convergence criterion is met
      Break from the loop
```

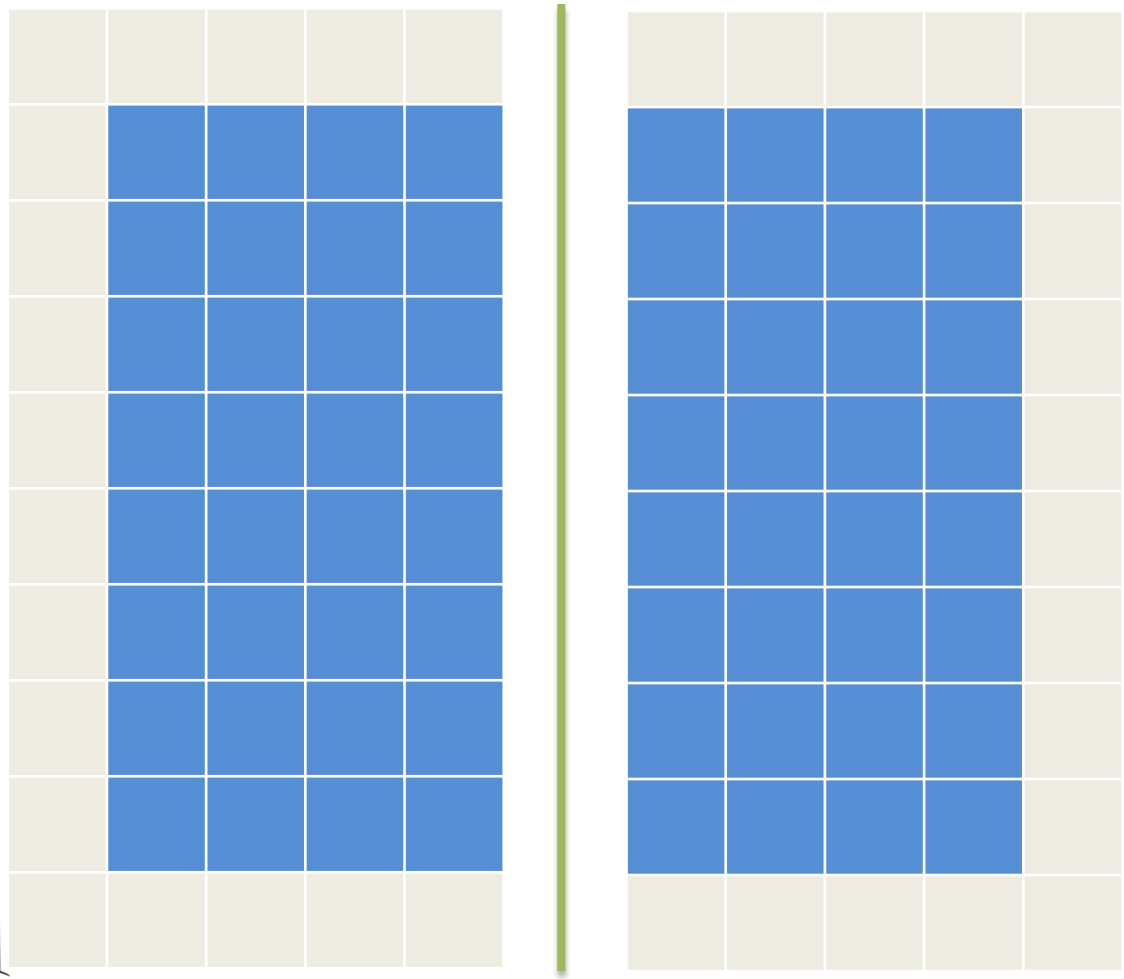
Which parts can and should be parallelized?

```
//Read and validate command line arguments

//Initialize the arrays
For each element
   value=0

//Set up boundary conditions
For each boundary element
   value=something

// Main loop
For each iteration
   For each element
      Update the value with the 5-point stencil
      Calculate convergence error
   Find maximum convergence error
   If the convergence criterion is met
      Break from the loop
```

These are the parallelizable parts

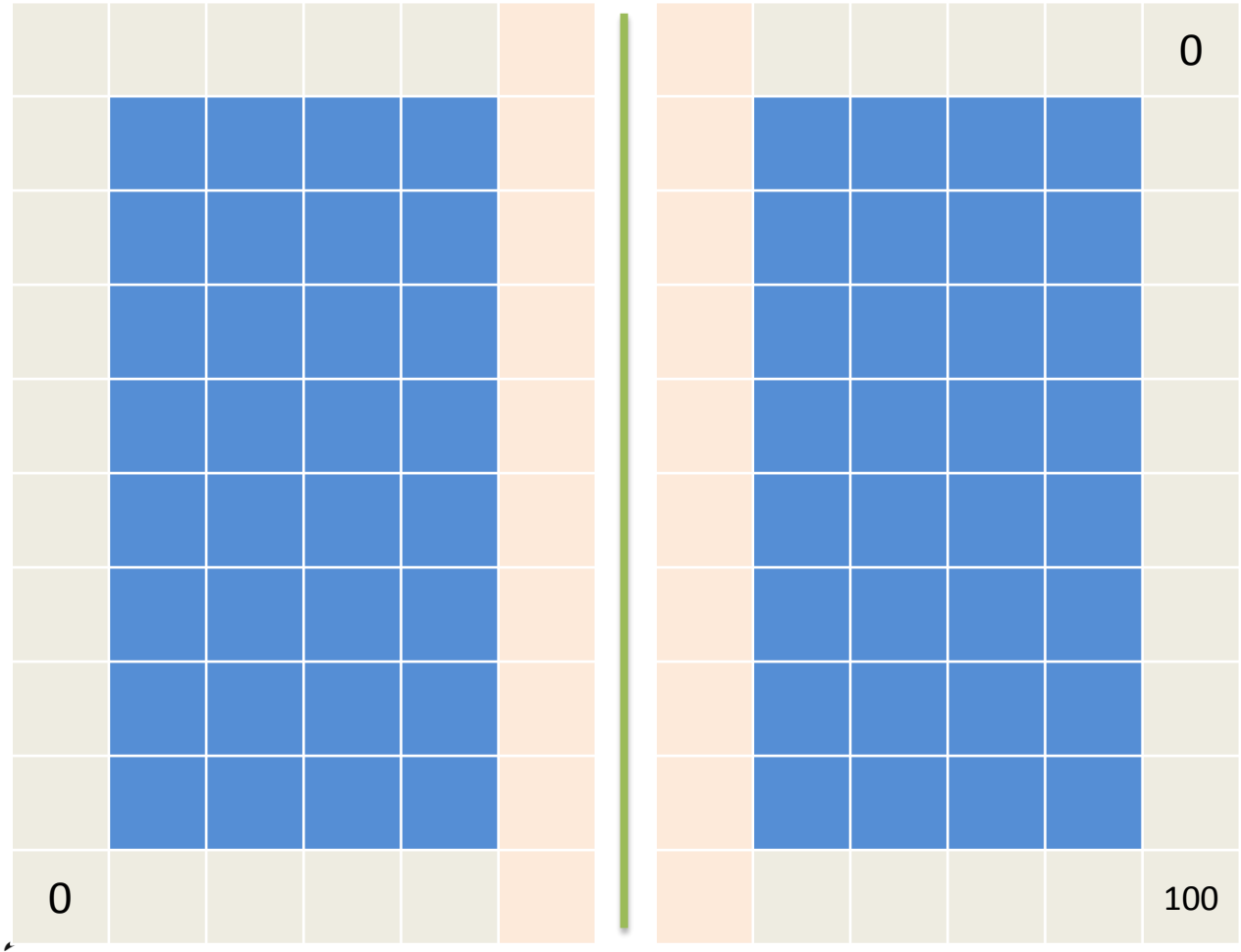


What if we have two workers (assuming distributed memory model)?



We can divide the domain into two parts and assign one to each worker

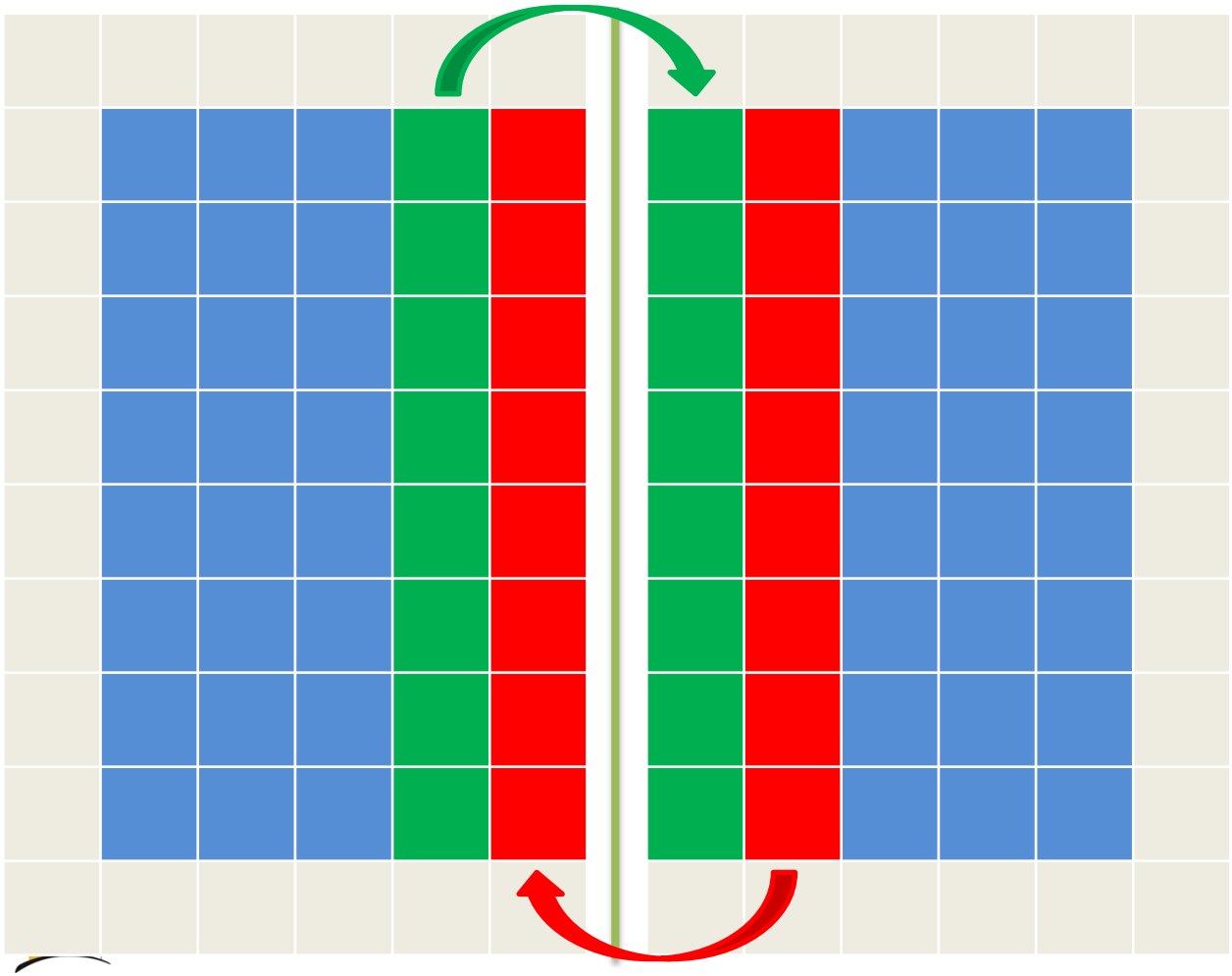
Anything wrong with this decomposition?



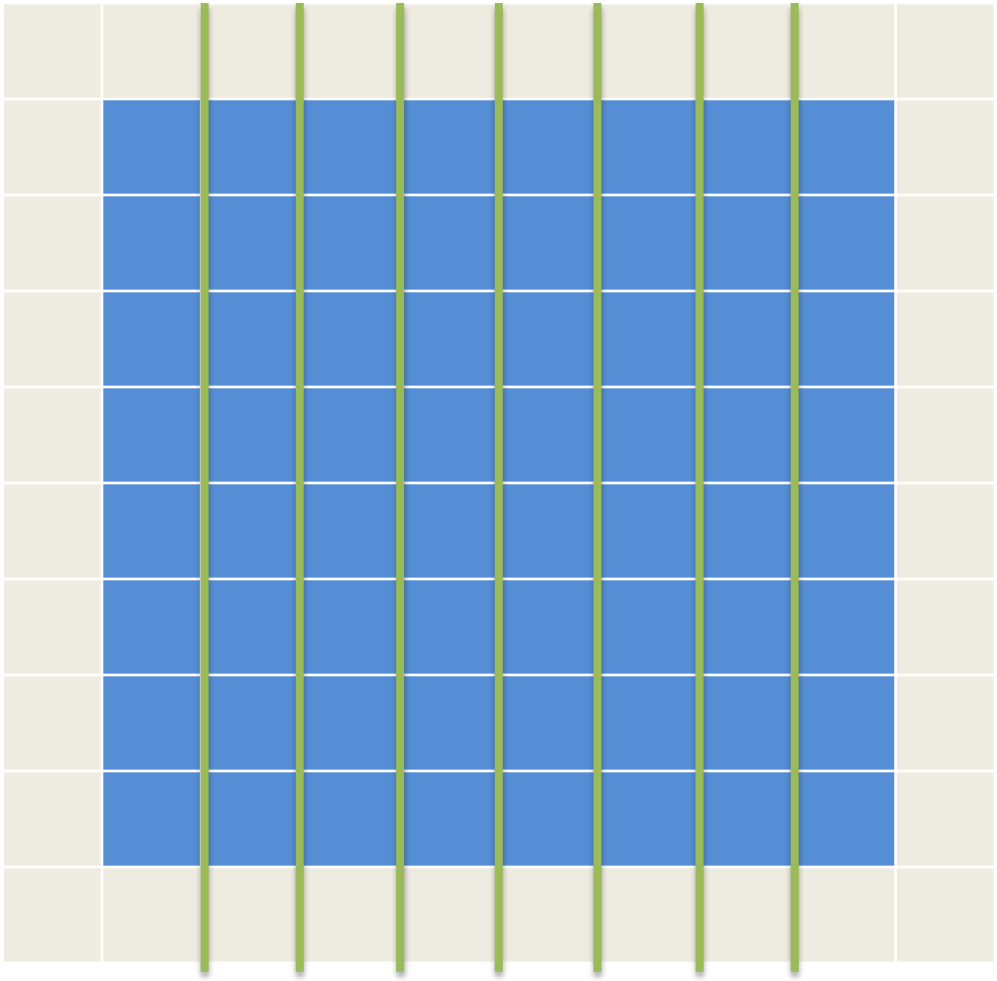
Each will need their own ghost zone

Independently, They can initialize the array, set the boundary condition and update values using the 5-point stencil.

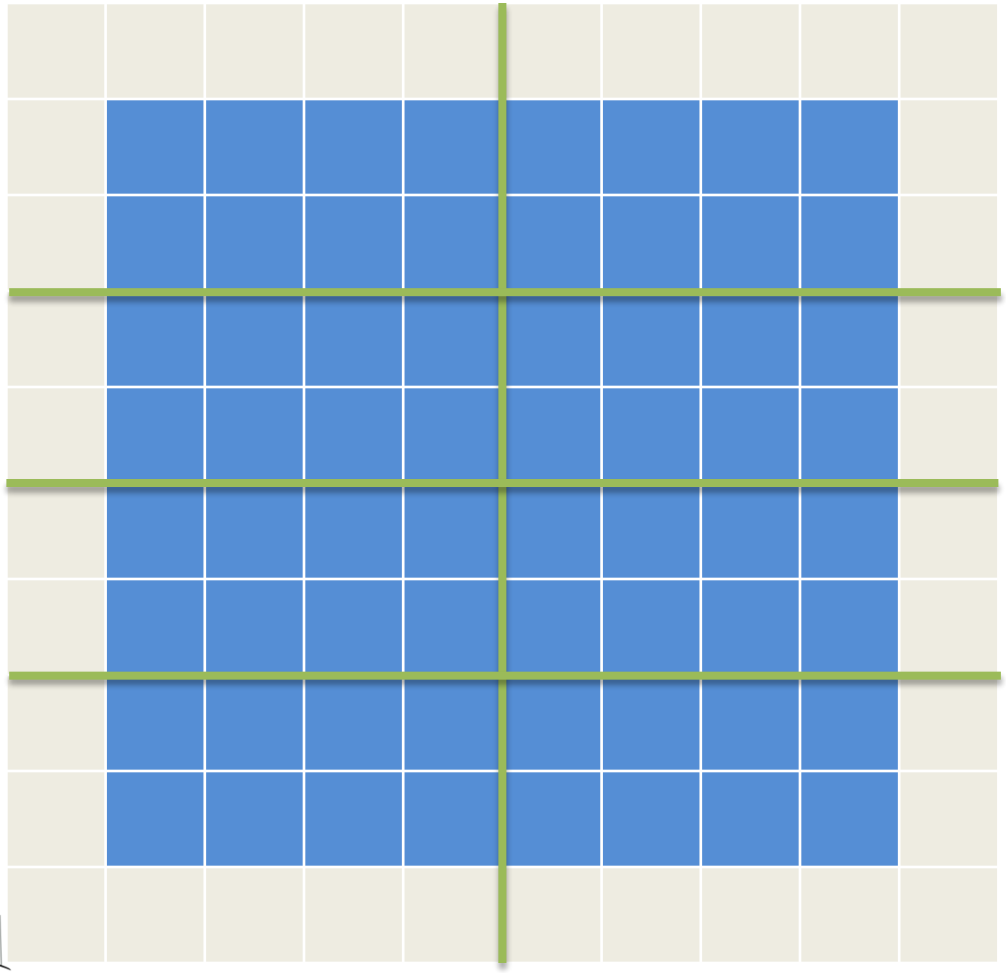
Is communication necessary?



After each iteration, they need to exchange boundary data with neighbors. Also, they need to exchange the max convergence error with all other workers so the global error can be found.



What if we have more workers? Can we still do 1-d decomposition? And does the direction (row or column) of decomposition matter?



1-d decomposition should still work. But how about 2-d decomposition?

```
//Read and validate command line arguments

//Initialize the arrays
Decide the array size for each process
For each element
   value=0

//Set up boundary conditions
For each boundary element
   value=something

// Main loop
For each iteration
   For each element
      Update the value with the 5-point stencil
      Calculate convergence error
   Find maximum convergence error
   Exchange boundary data with neighbors
   Exchange the max convergence error with all others
   If the convergence criterion is met
      Break from the loop
```

Parallel Laplace solver

Laplace v0

1. MPI_Init etc.
2. Calculate subdomain size: $N_{sub}=N/N_{procs}$
3. Allocate the array as either $(N_{sub}+2,N+2)$ or $(N+2,N_{sub}+2)$

Laplace v1

1. Set boundary condition correctly (using rank and nprocs)
2. In each iteration, after updating all element, exchange boundary data:
   if (rank != 0) exchange with above or left
   if (rank != nprocs-1) exchange with below or right
3. Find the global max convergence error and distribute to all processes

Laplace v2