Image Processing

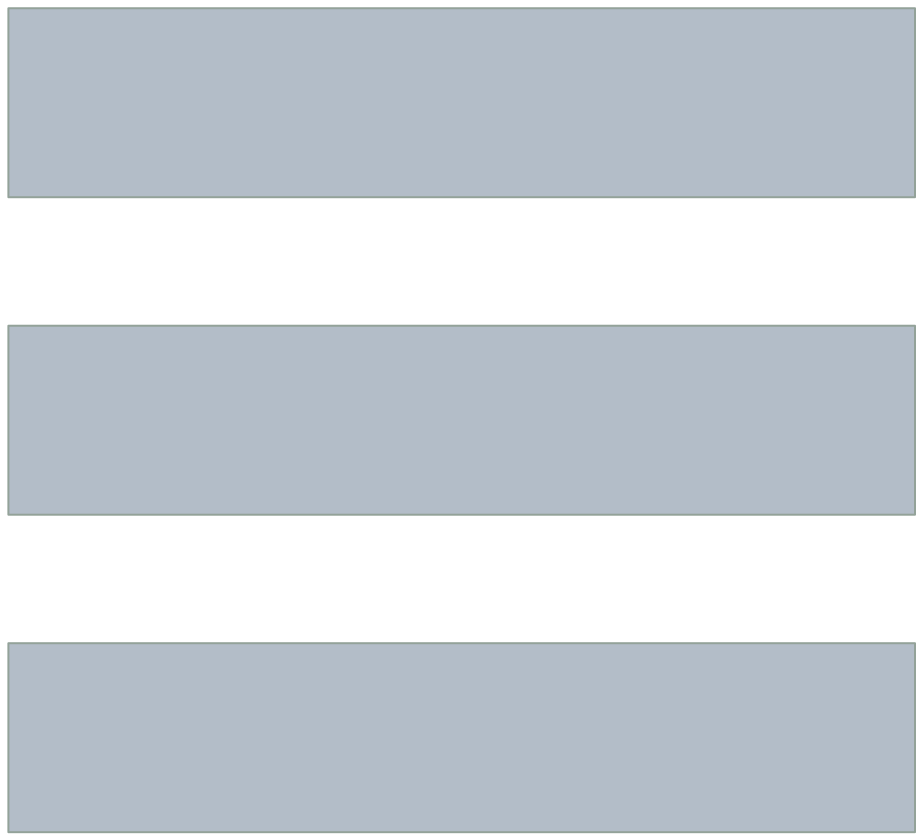
A case study for a domain decomposed MPI code

Domain Decomposition 1

- Starting with a big array:

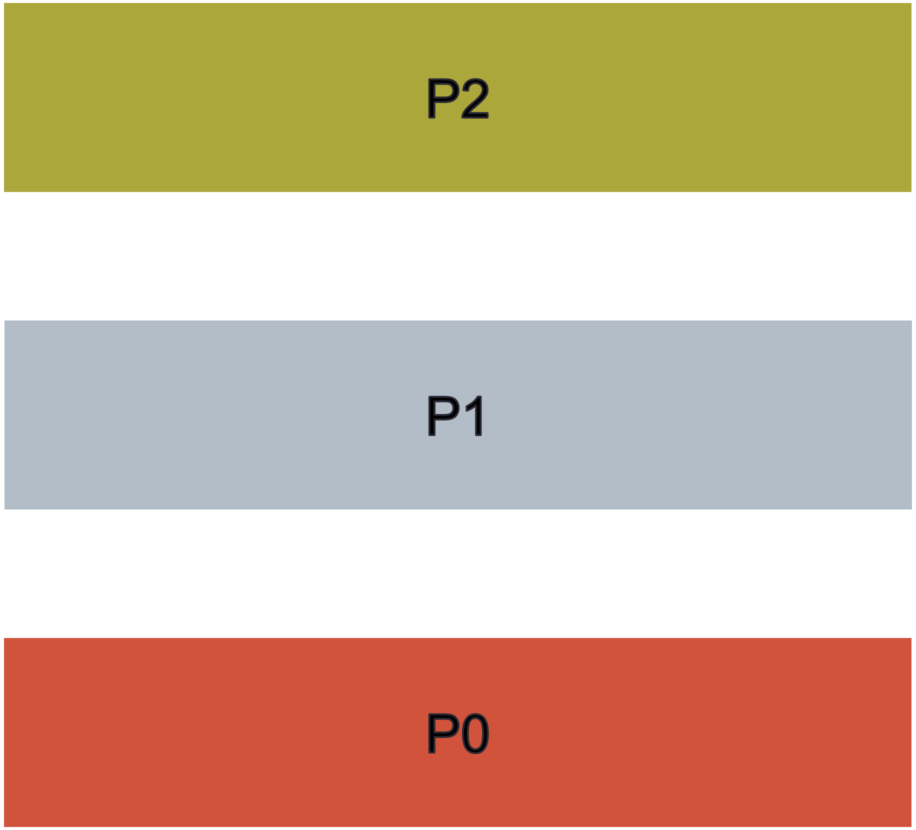
Domain Decomposition 2

- Split it into pieces:



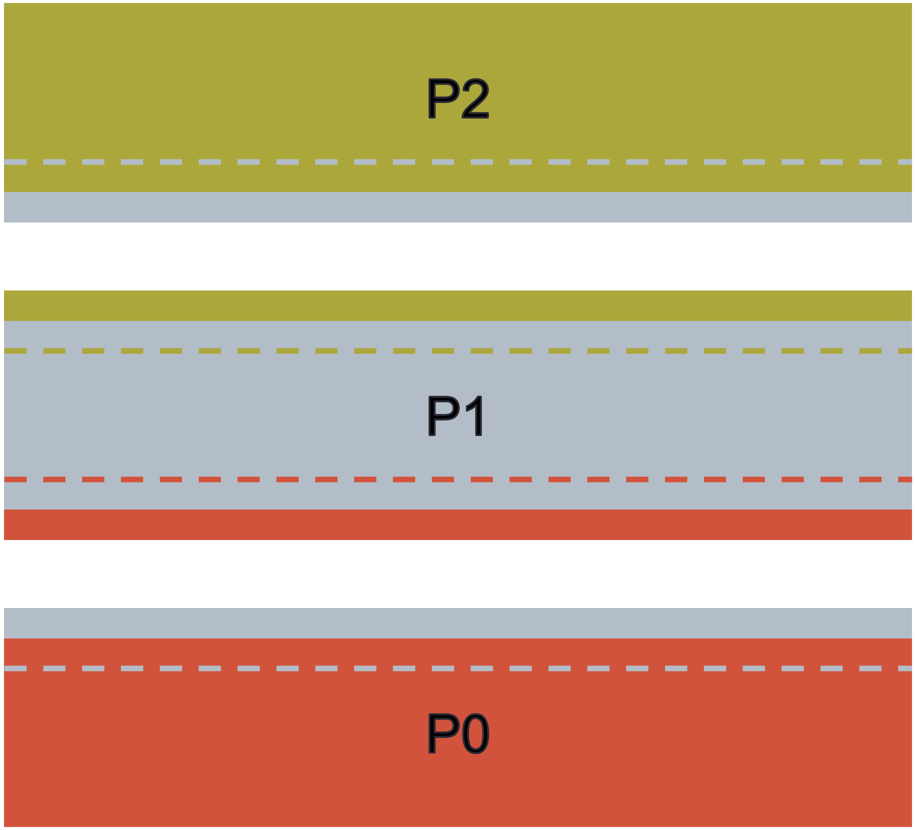
Domain Decomposition 3

- Assign pieces to processors:

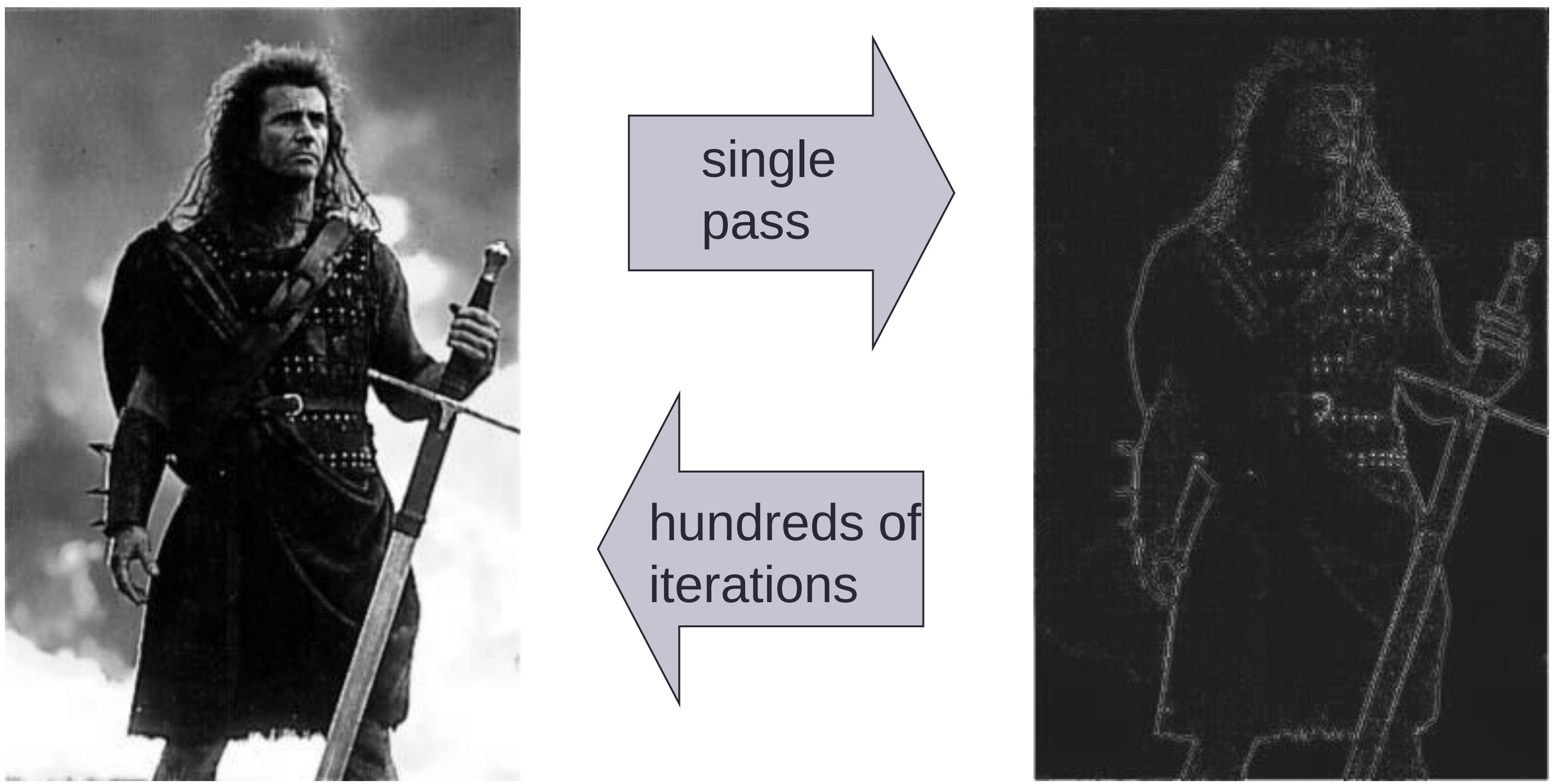


Domain Decomposition 4

- Use Halos to deal with interactions



Edge detection / image reconstruction



Edge detection

- Compare pixel to its four nearest neighbours
  - pixel values are from 0 (black) to 255 (white)

$$edge_{i,j} = image_{i+1,j} + image_{1,j+1} + image_{i-1,j} + image_{i,j-1} - 4\, image_{i,j}$$

- Pad 2D arrays with halos
  - in serial code, halo values set to white (i.e. 255)

Image reconstruction

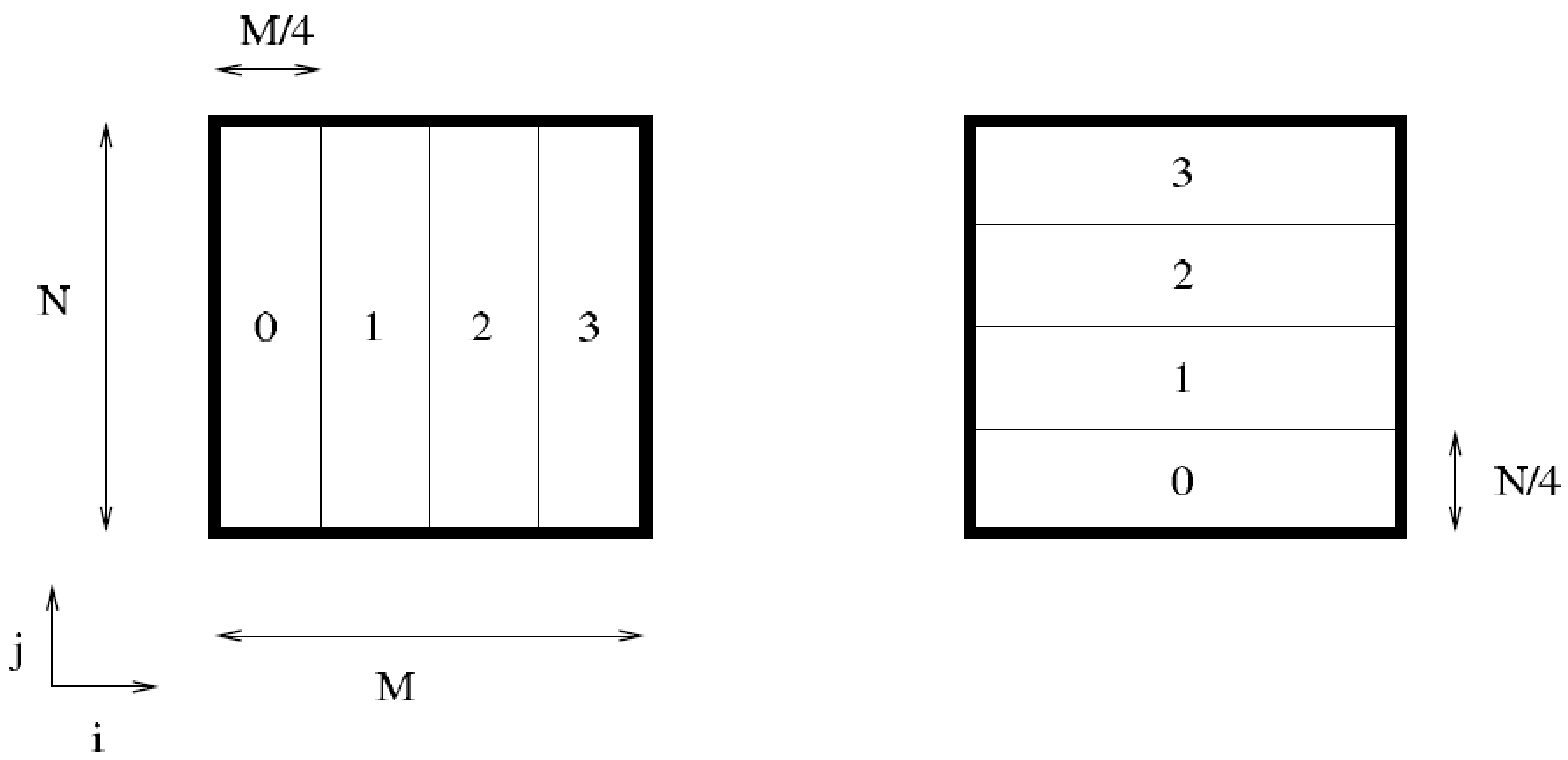
- Jacobi Solver to undo the simple edge detection algorithm (a five-point stencil)
  - simple example of discretised partial differential equation with nearest-neighbour interactions
  - actually solving $\nabla^2 image = edge$

$$new_{i,j} = \frac{1}{4}\left(old_{i+1,j} + old_{1,j+1} + old_{i-1,j} + old_{i,j-1} - edge_{i,j}\right)$$

- Repeat many times
  - in parallel, must update halo values from neighbours every iteration

Domain Decomposition

- Different choices in C and Fortran



The case study

- I provide you with:
  - More detailed printed instruction
  - Tar-ball (Choice of C or Fortran)
    - Input routine
    - Output routine
    - Couple of input files
- Tasks
  - Write a serial code (with halos for fixed boundary conditions)
    - ***check that the serial code works!!***
  - Distribute the work onto the processors; separate reconstructions
  - Get the halos exchanged; single reconstruction, identical to serial
  - Further suggestions on the instruction sheet