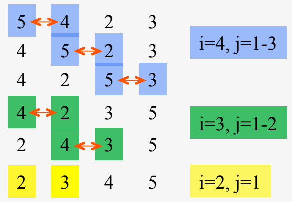
Chapter 5. Cases Study

- Organization
  - Parallel sort
  - Matrix – matrix multiplication
  - Image processing
  - Dynamical system simulation

Bubble sort odd-even

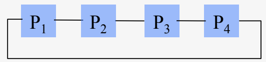
- Sequential bubble sort

  for i=n to 2
    for j=1 to i-1
    else compare-exchange(j, j+1)

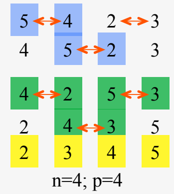


- Bubble sort odd-even
  - 1 element per processor

    for k=1 to n/2
      $P_{i,}$ i odd compare-exchange with P_{i+1}
      $P_{i,}$ i even compare-exchange with P_{i+1}
  - on a ring



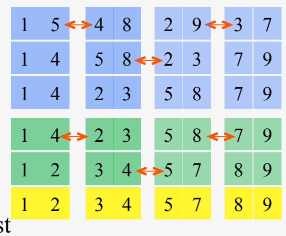
  - n stages of comparison



Bubble sort odd-even

- Bubble sort odd-even
  - n/p elements per processor

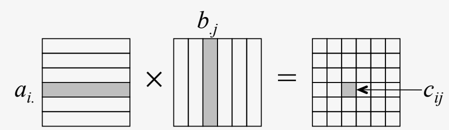
    Each processor sorts its elements
    for k=1 to p/2
      $P_{i,}$ i odd with P_{i+1}:
        - exchange of their sub-lists
        - gather-sort of 2 sub-lists
        - P_i keeps the lower half of the list
        - P_{i+1} keeps the upper half of the list
      $P_{i,}$ i even with P_{i+1}
        - exchange of their sub-lists
        - gather-sort of 2 sub-lists
        - P_i keeps the lower half of the list
        - P_{i+1} keeps the upper half of the list



Matrix-Matrix Multiplication

- Problem
  - Compute $C=A\times B$, A, B: matrix nxn on a torus of pxp process

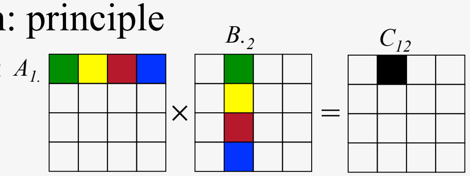
$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}$$



- Parallel algorithm: principle

  Process (s,t) calculates:

$$C_{st} = \sum_{k=0}^{p-1} A_{sk} B_{kt}$$

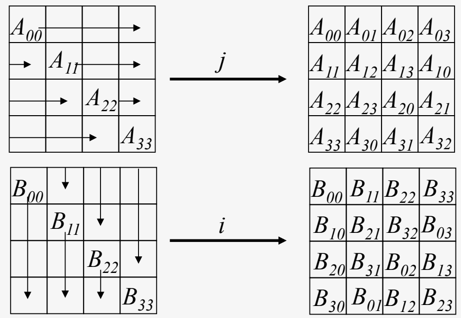


Matrix-Matrix Multiplication

- Parallel algorithm 1: each process computes one bloc of C matrix
  - **First stage**
    1. Cut the matrix A, B, C into blocs
    2. Distribute the blocs A_{ij} and B_{ij} without replication
    3. Compute a $A_{ik}*B_{kj}$
  - **Second stage**
    1. Circulating of the blocs A_{ij} and B_{ij} (For example, send the A_{ij} to the left, and the B_{ij} to the upper)
    2. Compute a $A_{ik}*B_{kj}$
    3. Repeat p-1 times ($N=p*p$)

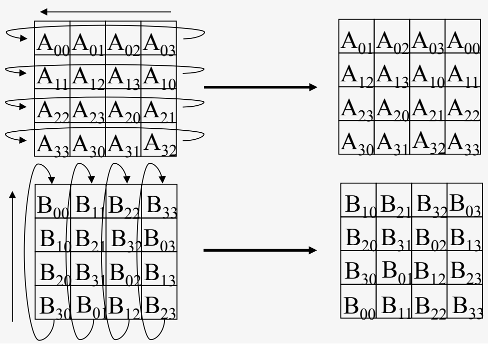
Matrix-Matrix Multiplication

- Parallel algorithm 1: example
  - Initial distribution of A_{ij} and B_{ij}



Matrix-Matrix Multiplication

- Parallel algorithm 1: example
  - Circulating of the blocs A_{ij} and B_{ij}: *p-1* times



Matrix-Matrix Multiplication

- Performance evaluation
  - Computation time:

$$Tcomp_{ij}^{0} = \left[t*t\left(t*t_{mult}+(t-1)*t_{add}\right)\right],\ t=\frac{n}{p}$$

$$Tcomp_{ij}^{k} = \left[t*t\left(t*t_{mult}+(t-1)*t_{add}\right)\right]+\left[t*t*t_{add}\right],\ \ k=1,\ldots,p-1$$

$$Tcomp_{ij} = p*\left[t*t\left(t*t_{mult}+(t-1)*t_{add}\right)\right]+(p-1)*(t*t*t_{add})$$

  - Communication time:

$$Tdist = 2*\left[(?)\beta+(?)t*t*sizeof(component)\tau\right]$$

$$Tcirc = (p-1)*\left[\beta+t*t*sizeof(component)\tau\right]$$

$$Tgather = \left[(?)\beta+(?)t*t*sizeof(component)\tau\right]$$

Image processing – local linear operations

- Problem
  - Replacing each pixel of an image by the average of its neighbors

$$C'(x,y) = w_0 C(x-1,y-1) + w_1 C(x-1,y) + \ldots + w_8 C(x+1,y+1)$$

w_0	w_1	w_2
w_3	w_4	w_5
w_6	w_7	w_8

Mask of 3x3

- Decomposition

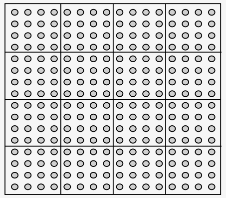
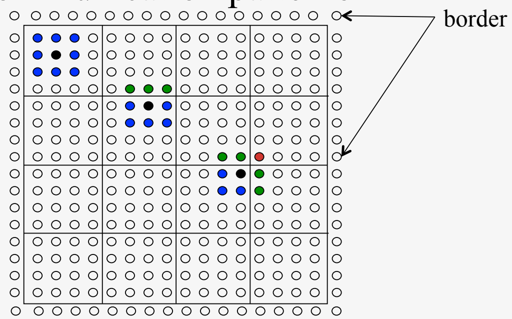


Image processing

- Communication patterns



w_0	w_1	w_2
w_3	w_4	w_5
w_6	w_7	w_8

Mask of 3x3

- Data duplication – No
- Communication at each step

Case Study: Dynamical system simulation

- Dynamical system of particles
  - N particles in a square domain
  - Interactions between the particles
    - If the particles are close enough they repel each other
    - If the particles are far enough they attracted to each other
    - If the particles are more than some distance, r_0 apart, they do not influence each other

Case Study: Dynamical system simulation

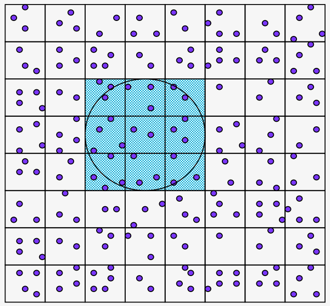
- Dynamical system simulation
  - Given data: masses, initial positions and velocities of the particles
  - Simulation: movement of the particles at a series of discrete time steps
- Force on particle i

$$F_i = \sum_{j=0}^{n-1} f_{ij} \qquad f_{ij} = G\frac{m_i m_j}{r_{ij}^2}$$

  - f_{ij}: force exerted on particle i by particle j

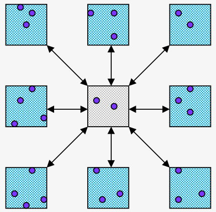
Case Study: Dynamical system simulation

- Partitioning of the problem
  - Divide the domain into cells of size $r_0 \times r_0$



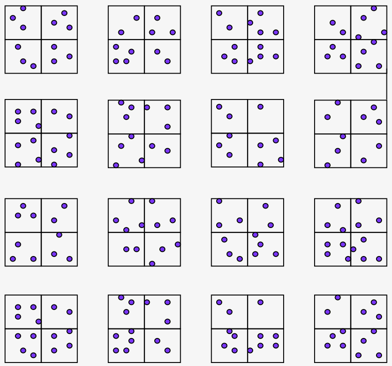
Case Study: Dynamical system simulation

- Communication pattern



Case Study: Dynamical system simulation

- Agglomerate and mapping of tasks



Case Study: Dynamical system simulation

- Data structures
  - Structure for particles
    - Position
    - Velocity
  - Structure for cells
    - List of particles
  - Structure for process
    - Table/List of cells

Case Study:
Dynamical system simulation

❑ Sequential code

1. Initialize arrays
   (a) initialize particle's positions and velocities
   (b) initialize the 2D array of cell lists by inserting particles into the linked lists
2. Update loop
   For each time step
   (a) find force on each particle, update particle's position and velocity
   (b) move particle under influence of the force exerted on it

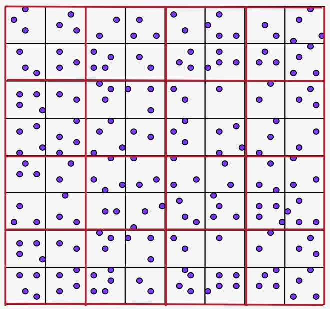
Case Study:
Dynamical system simulation

❑ Parallel code

1. Arrays initialization
2. Data Distribution
3. Parallel loop:
   For each time step
   (a) communicate particle data for boundary cells
   (b) find force on each particle of the process
   (c) update particle's positions and velocities
   (d) migrate particles with communication

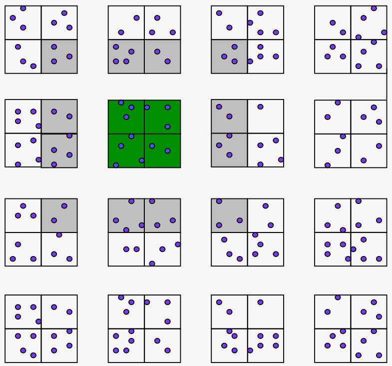
Case Study:
Dynamical system simulation

❑ Data distribution



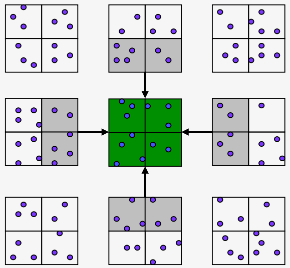
Case Study:
Dynamical system simulation

❑ Data dependencies



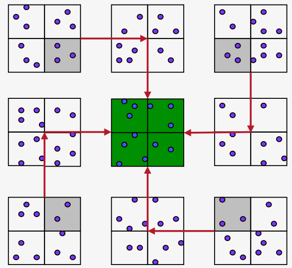
Case Study: Dynamical system simulation

- Communication of boundary cell data
  - Direct communication with direct neighbors



Case Study: Dynamical system simulation

- Communication of boundary cell data
  - Communication with diagonal neighbors



Case Study: Dynamical system simulation

- Particles migration

1. For each particle p
   - update position and velocity
   - determine which cell p is in
   - if p has moved to new cell
     - delete p from list of old cell
     - if p has moved to different process
       - put p into appropriate communication buffer
       - remove p from particle array
     - else add p to list of new cell

Case Study: Dynamical system simulation

- Particles migration

2. Communication of buffers
3. If the particle received from another process belongs to it
   - add the particle to the list of its new cell

   else
   - put the particle to appropriate buffer to be passed to another process

- Notes
  - Irregular communication