Designing Parallel Programs

Abhinav Bhatele, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

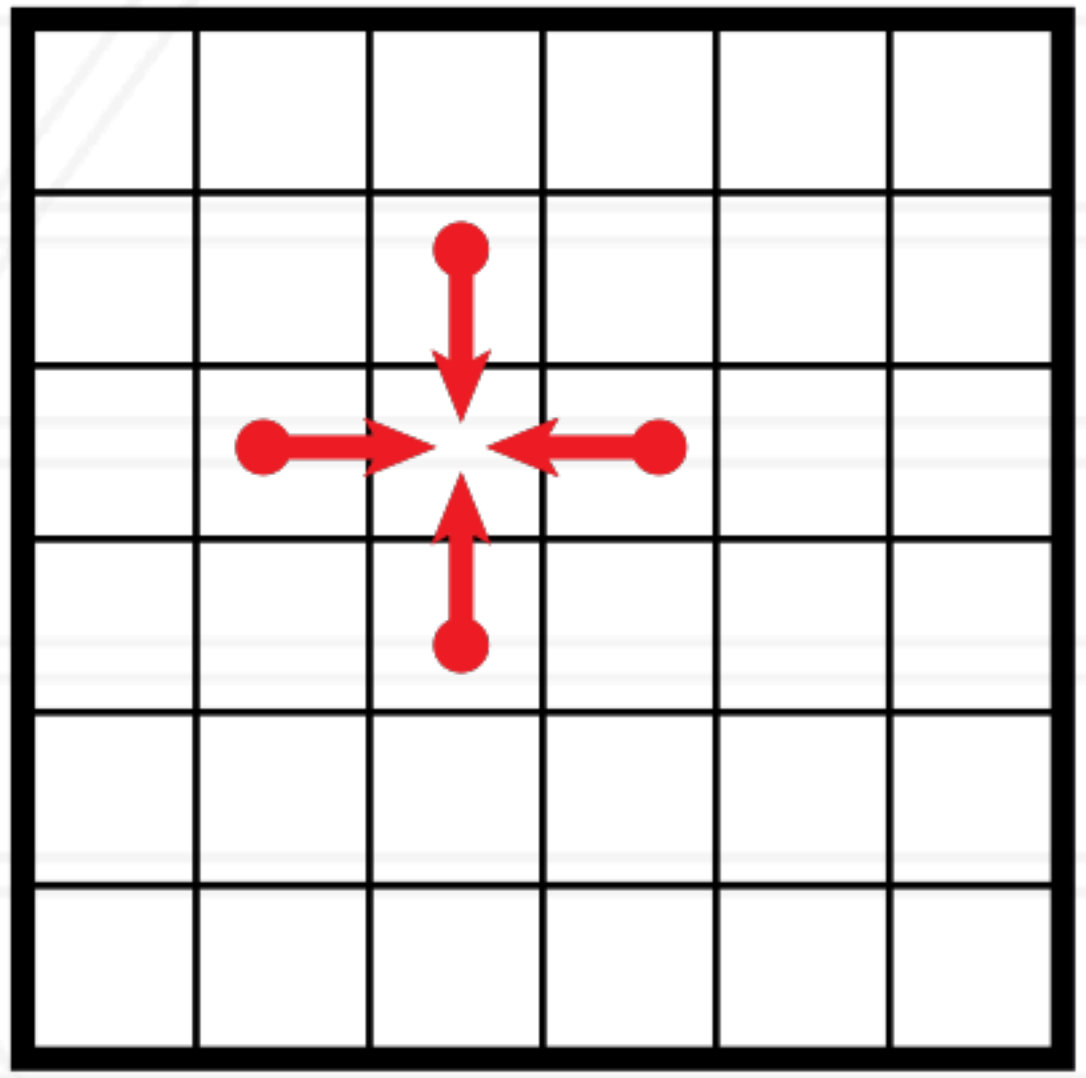
- Deepthought2 (dt2) accounts have been mailed to everyone
- Please cc the TAs also when emailing me
- Prefix [CMSC416] or [CMSC818X] in your email subject

Writing parallel programs

- Decide the serial algorithm first
- Data: how to distribute data among threads/processes?
  - Data locality: assignment of data to specific processes to minimize data movement
- Computation: how to divide work among threads/processes?
- Figure out how often communication will be needed

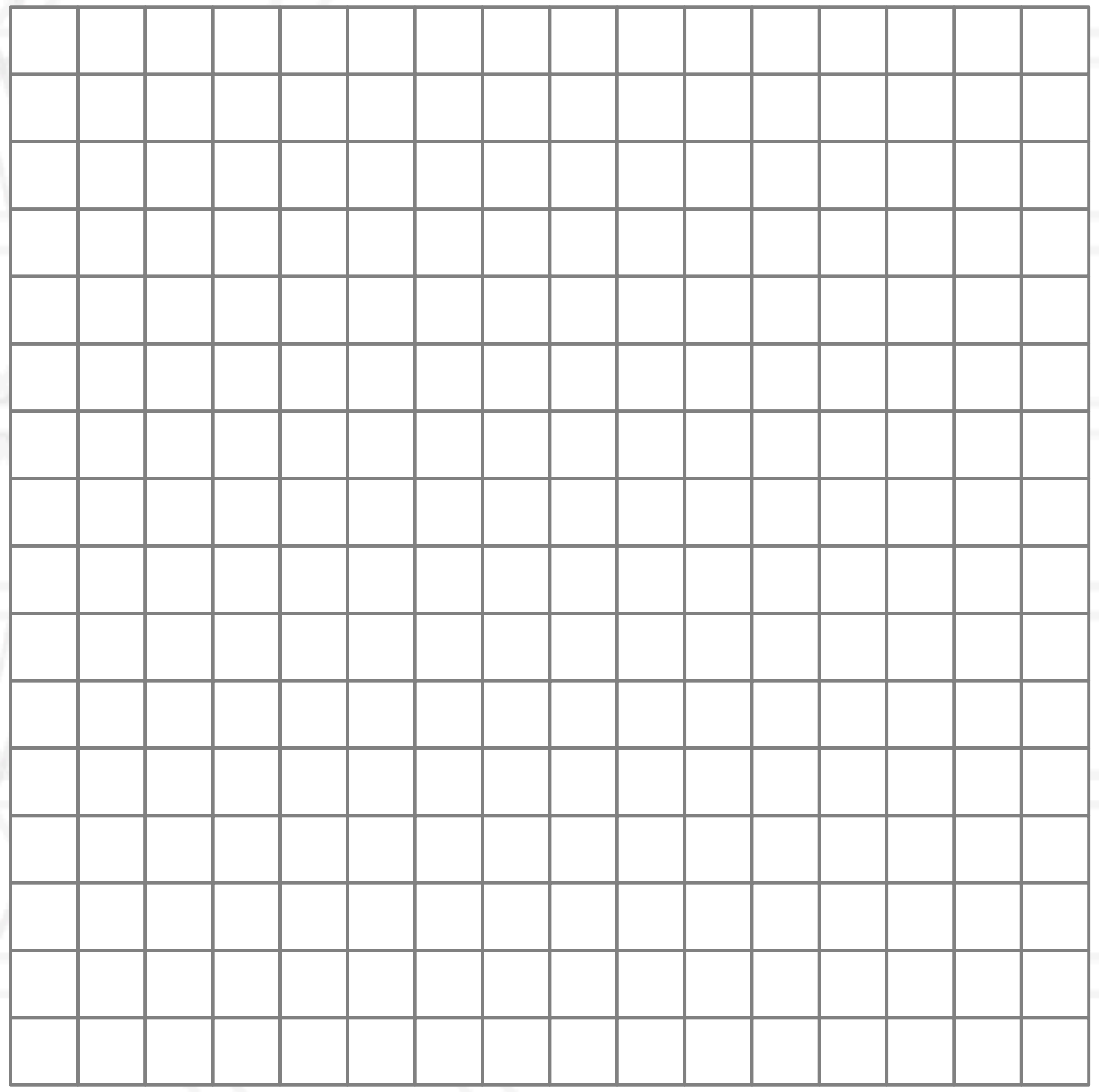
Two-dimensional stencil computation

- Commonly found kernel in computational codes
- Heat diffusion, Jacobi method, Gauss-Seidel method



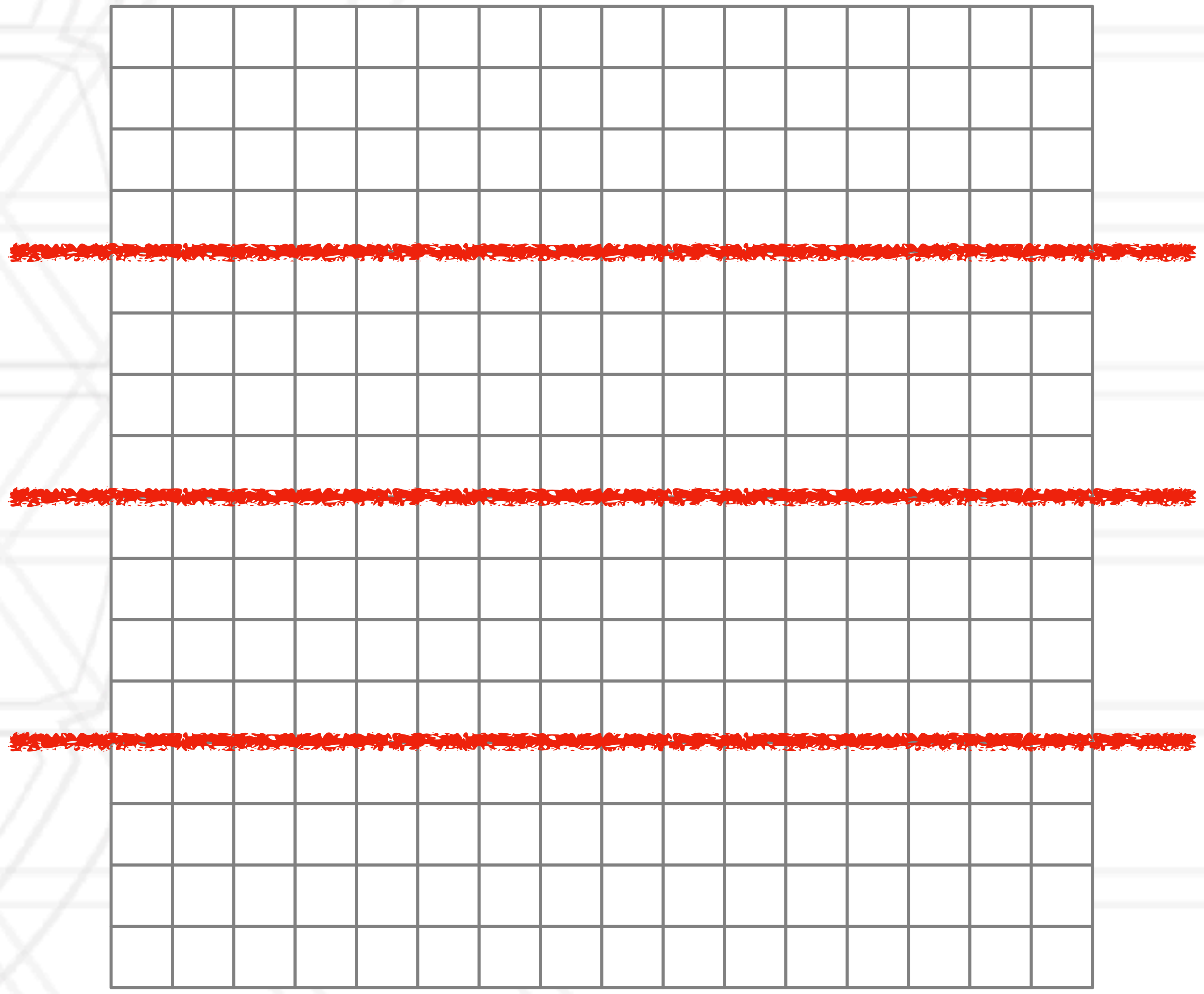
$$A[i,j] = \frac{A[i,j] + A[i-1,j] + A[i+1,j] + A[i,j-1] + A[i,j+1]}{5}$$

2D stencil computation in parallel



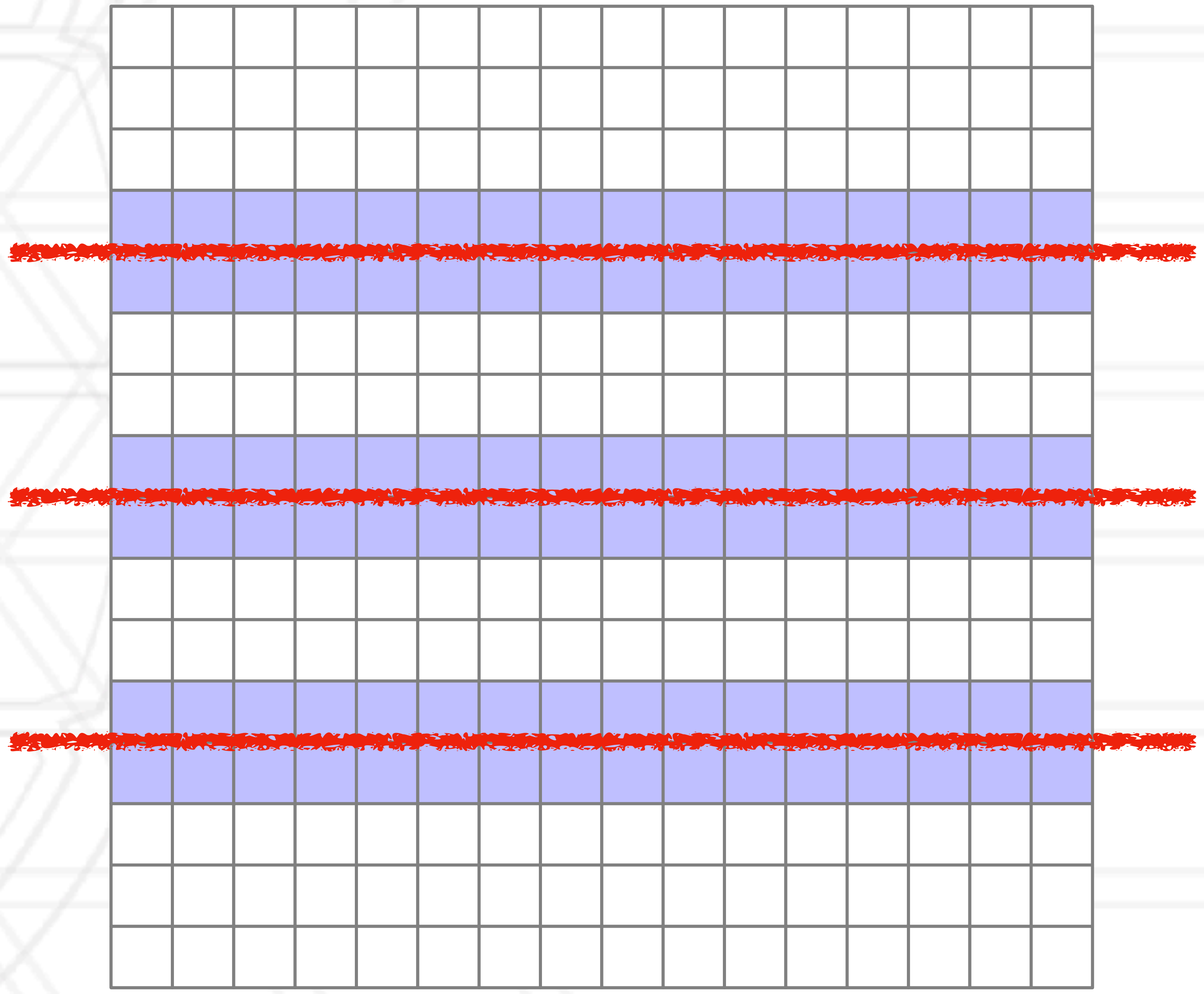
2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes



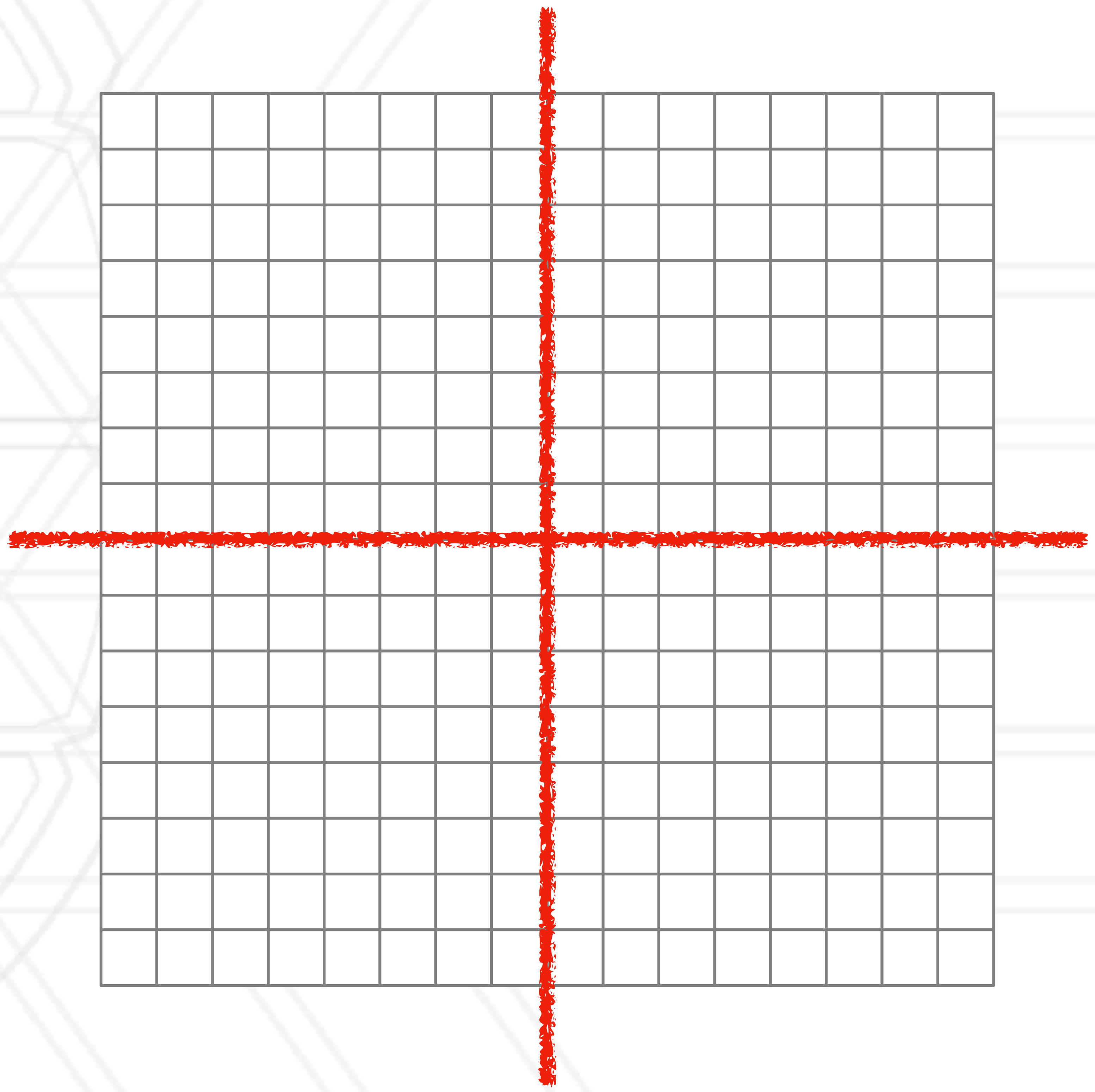
2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes



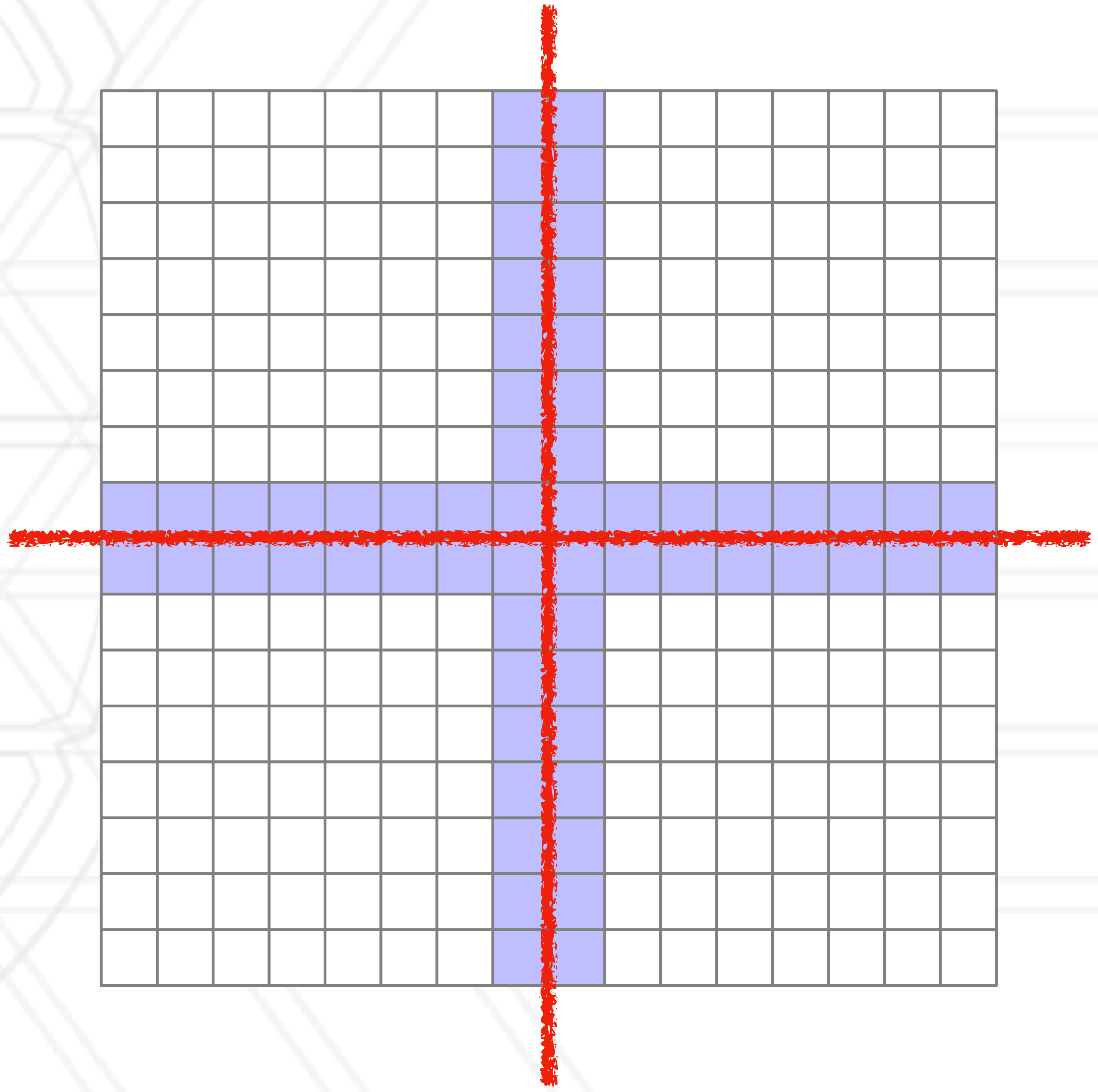
2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes
- 2D decomposition
  - Divide both rows and columns (2d blocks) among processes



2D stencil computation in parallel

- 1D decomposition
  - Divide rows (or columns) among processes
- 2D decomposition
  - Divide both rows and columns (2d blocks) among processes



Prefix sum

- Calculate partial sums of elements in array
- Also called a "scan" sometimes

```
pSum[0] = A[0]

for(i=1; i<N; i++) {
    pSum[i] = pSum[i-1] + A[i]
```

A	1	2	3	4	5	6	…
pSum	1	3	6	10	15	21	…

Parallel prefix sum

2	8	3	5	7	4	1	6

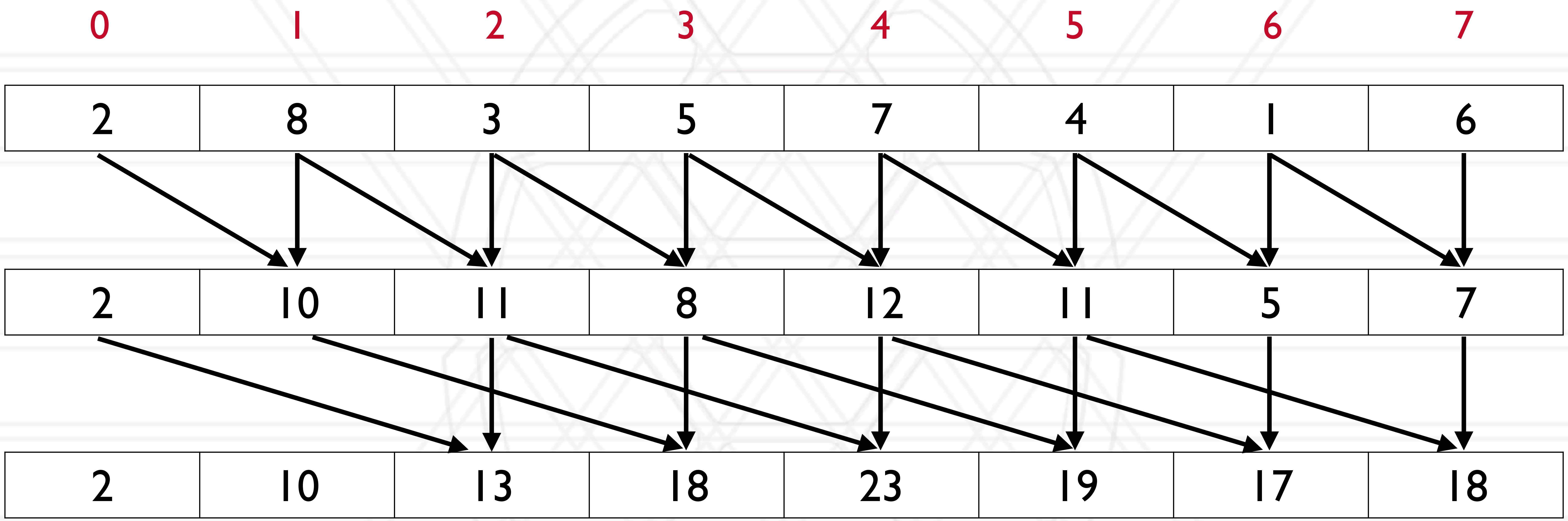
Parallel prefix sum

0	1	2	3	4	5	6	7
2	8	3	5	7	4	1	6

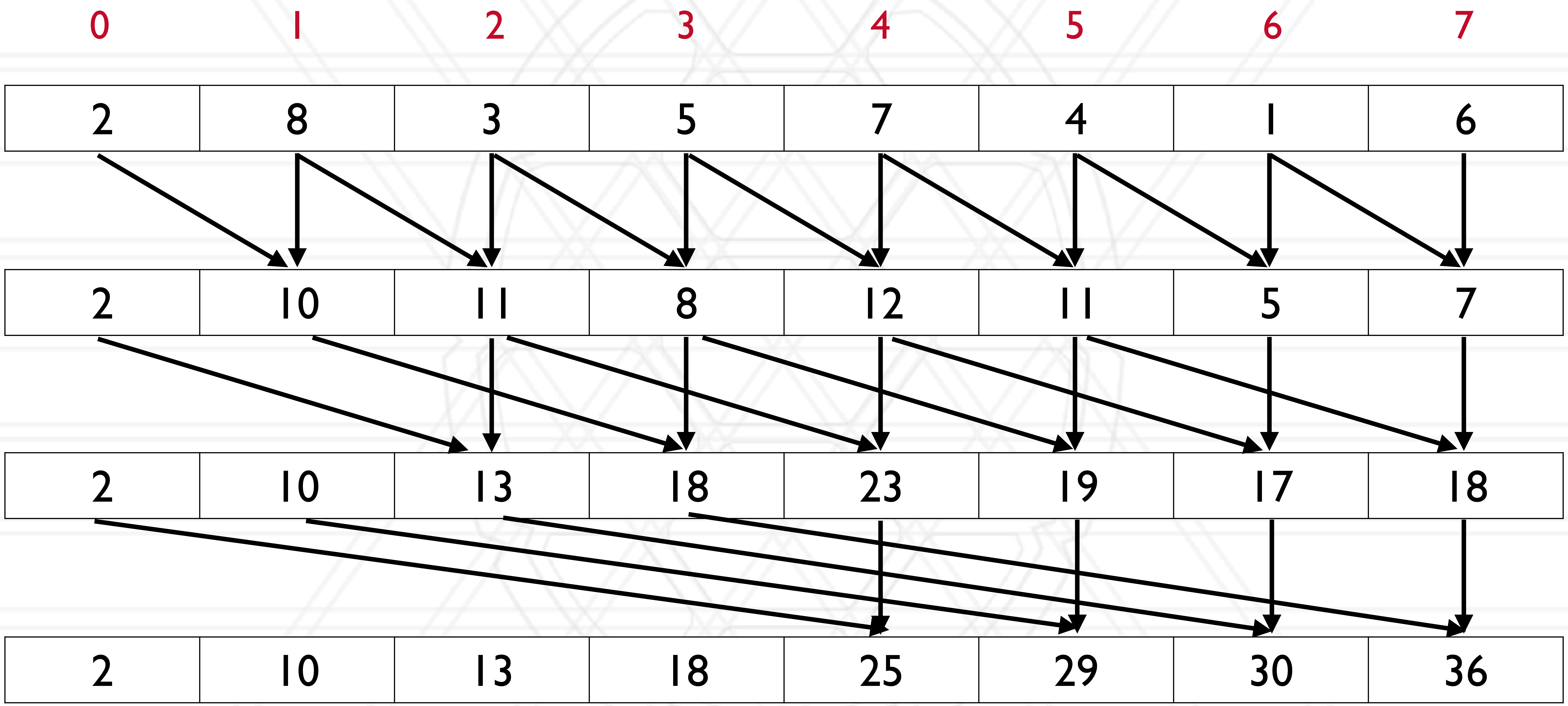
Parallel prefix sum

0	1	2	3	4	5	6	7
2	8	3	5	7	4	1	6
2	10	11	8	12	11	5	7

Parallel prefix sum



Parallel prefix sum



In practice

In practice

- You have N numbers and P processes, N >> P

In practice

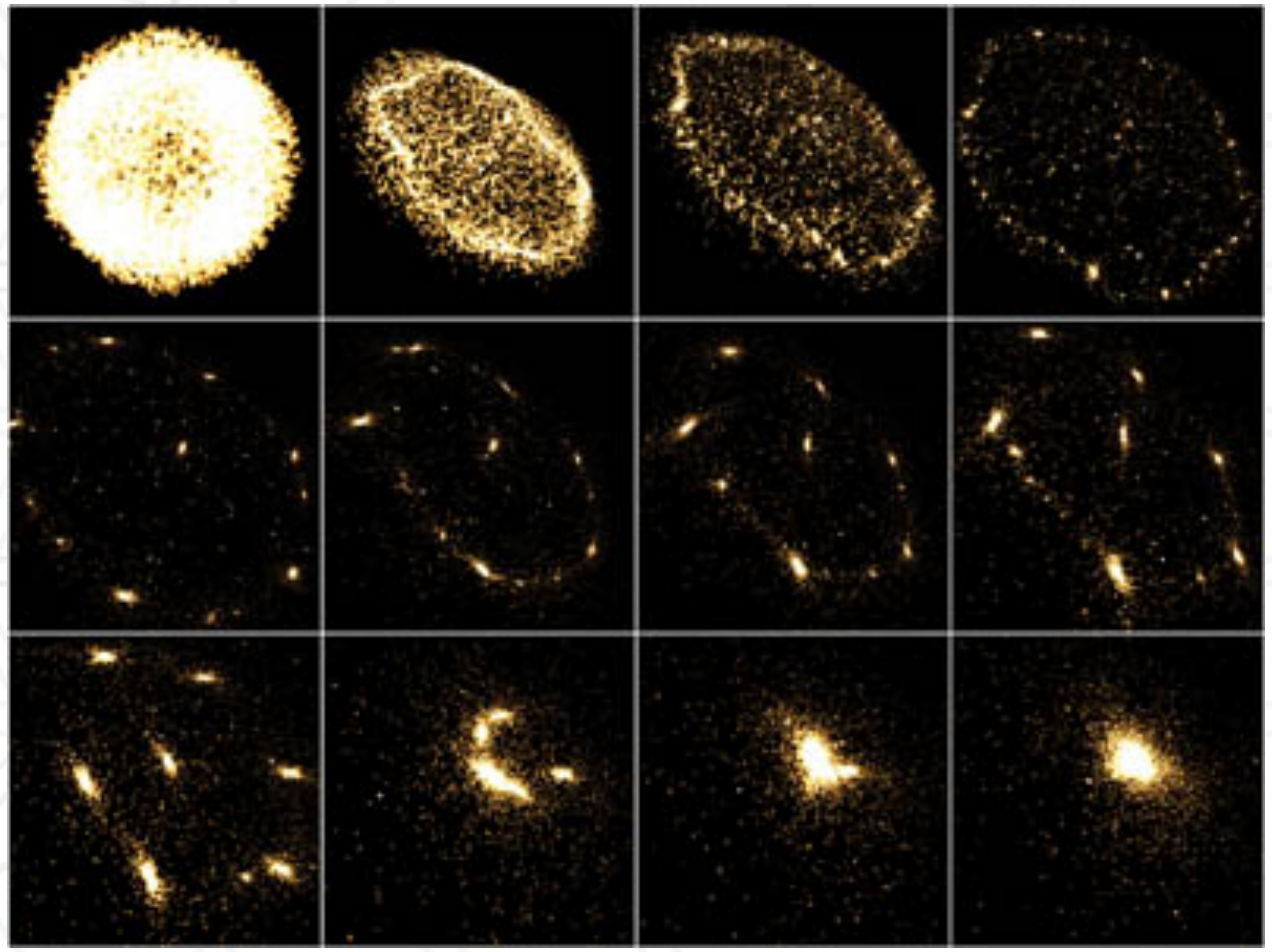
- You have N numbers and P processes, N >> P
- Assign a N/P block to each process
  - Do calculation for the blocks on each process locally

In practice

- You have N numbers and P processes, N >> P
- Assign a N/P block to each process
  - Do calculation for the blocks on each process locally
- Then do parallel algorithm with partial prefix sums

The *n*-body problem

- Simulate the motion of celestial objects interacting with one another due to gravitational forces
- Naive algorithm: $O(n^2)$
  - Every body calculates forces pair-wise with every other body (particle)



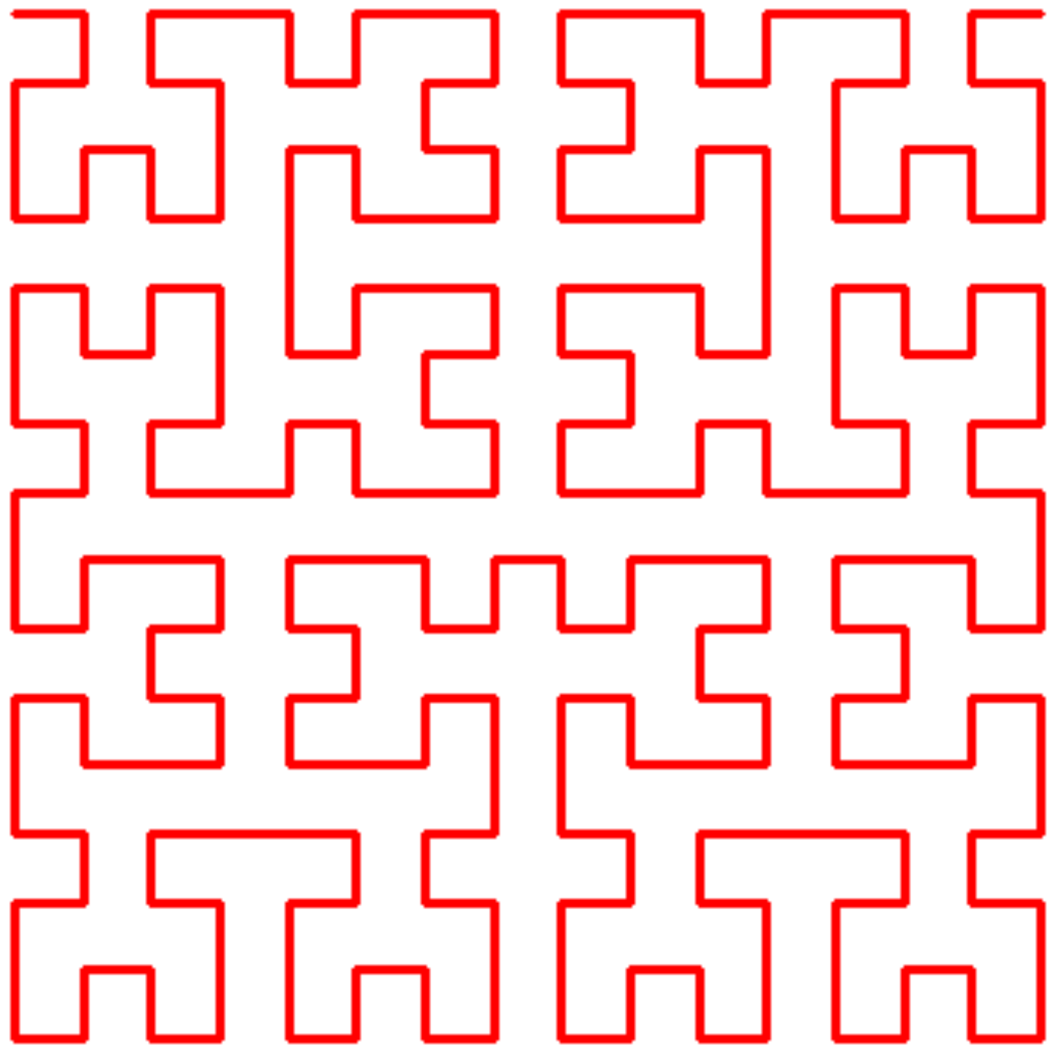
https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

Data distribution in *n*-body problems

- Naive approach: Assign n/p particles to each process
- Other approaches?

Data distribution in *n*-body problems

- Naive approach: Assign n/p particles to each process
- Other approaches?



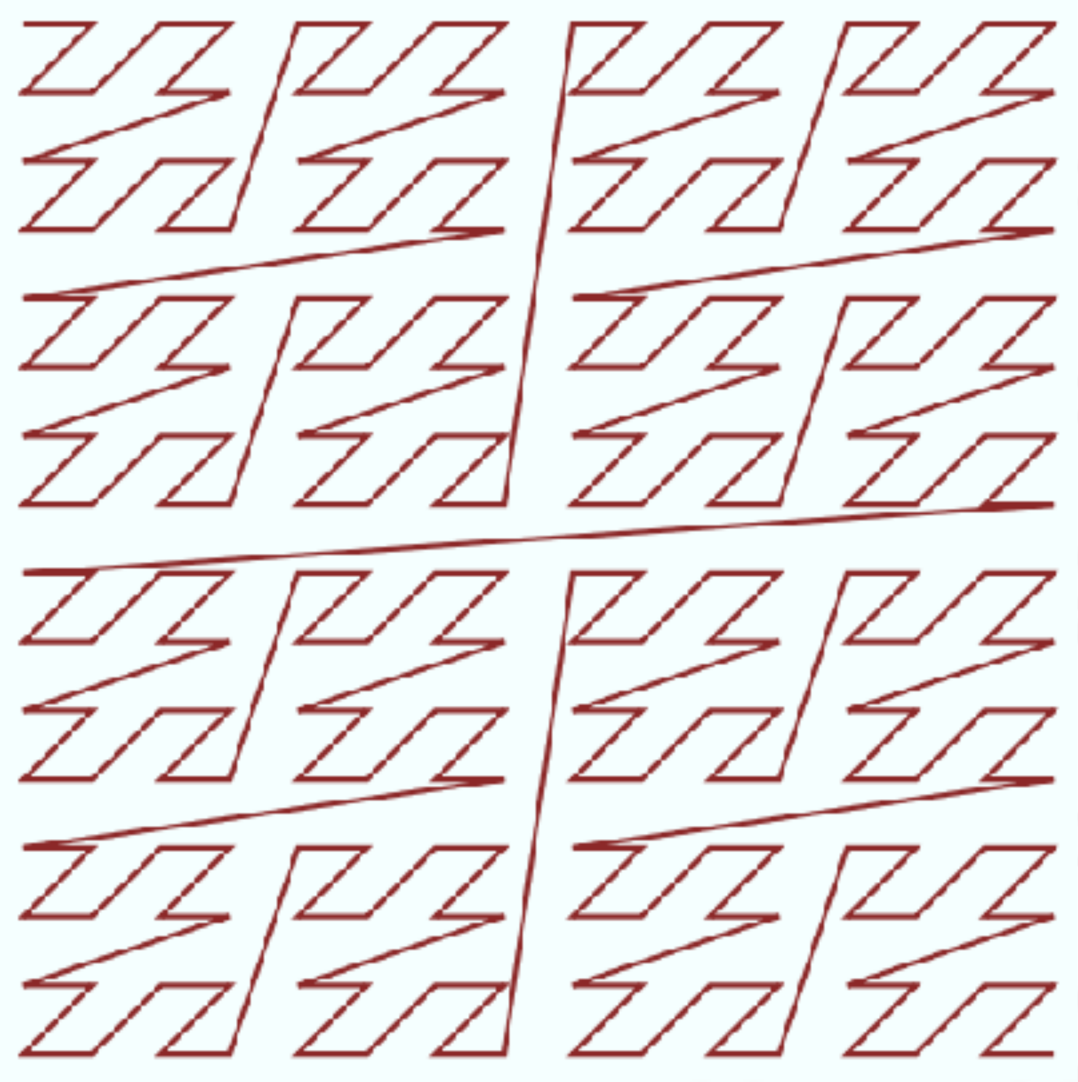
Space-filling curves

http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve

Data distribution in *n*-body problems

- Naive approach: Assign n/p particles to each process
- Other approaches?



Space-filling curves

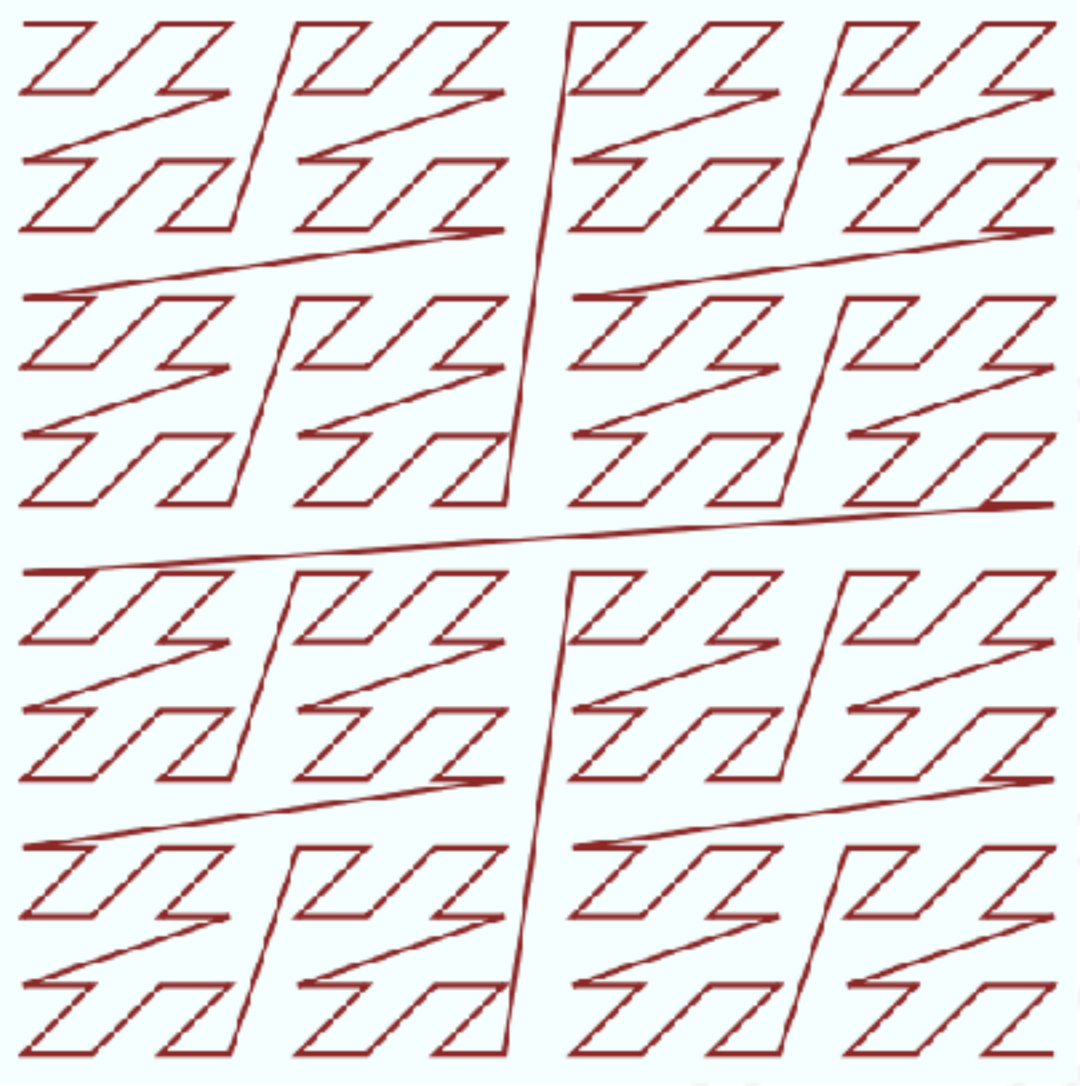
http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve

Data distribution in *n*-body problems

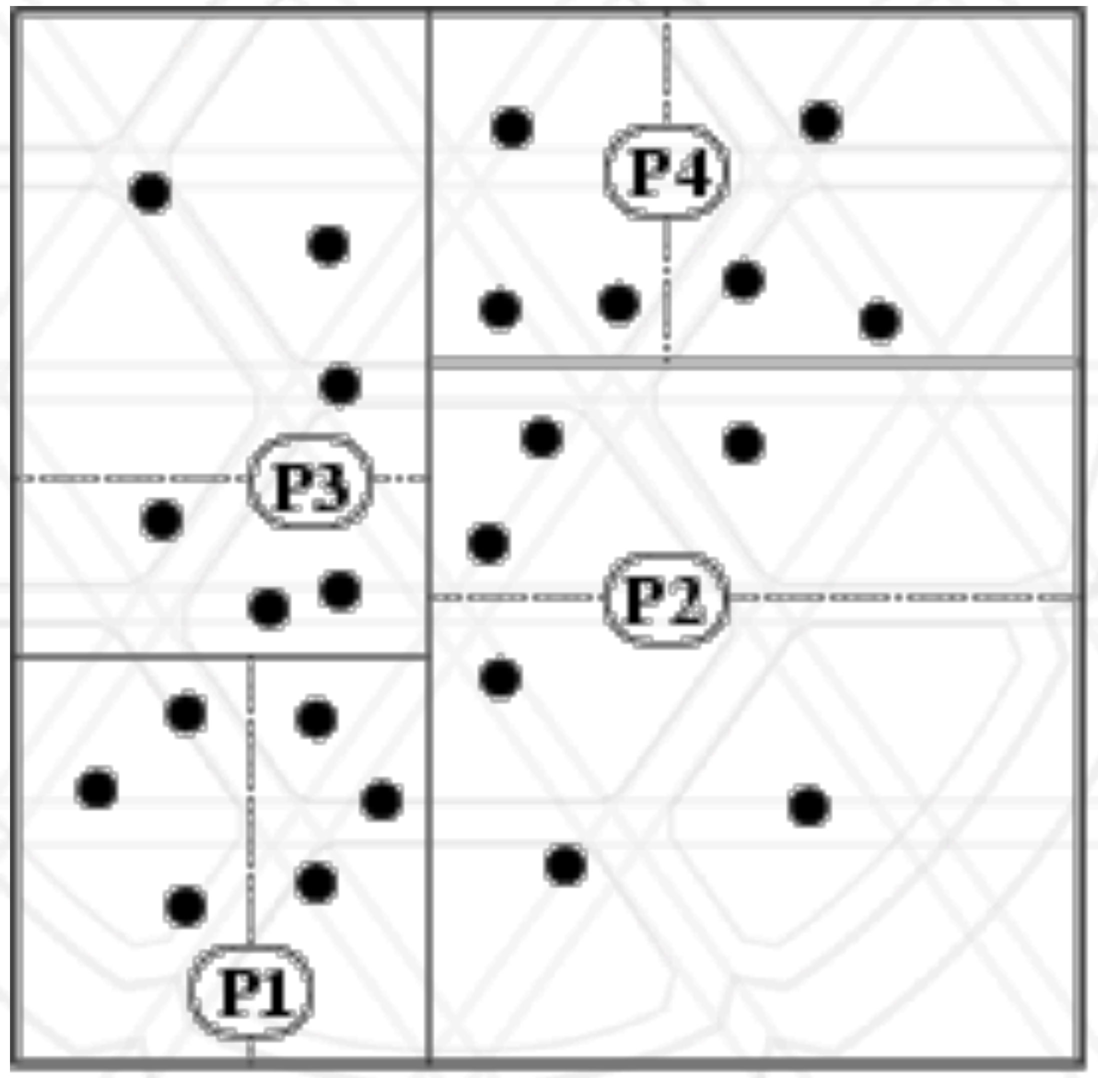
- Naive approach: Assign n/p particles to each process
- Other approaches?

Space-filling curves

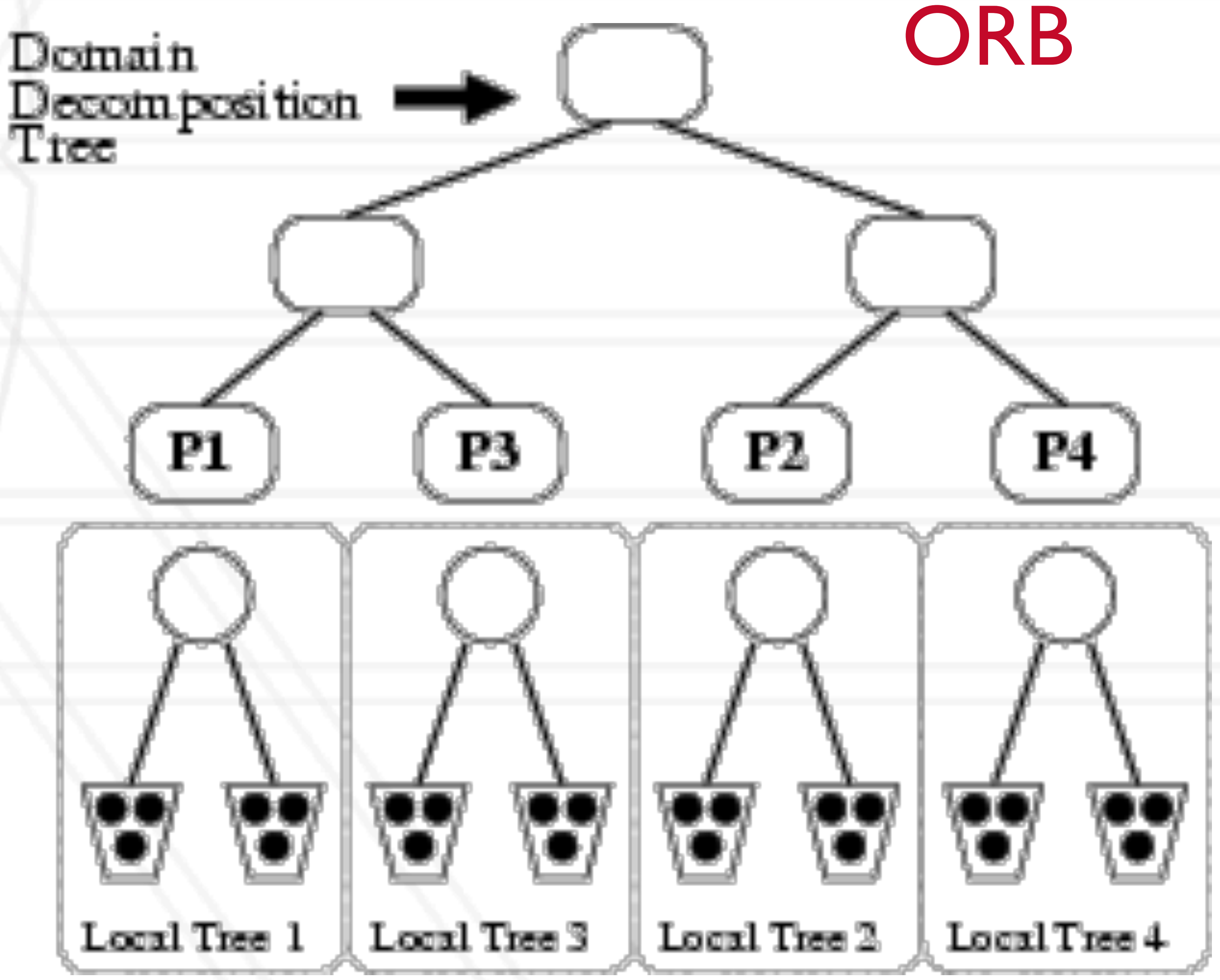


http://datagenetics.com/blog/march22013/

https://en.wikipedia.org/wiki/Z-order_curve



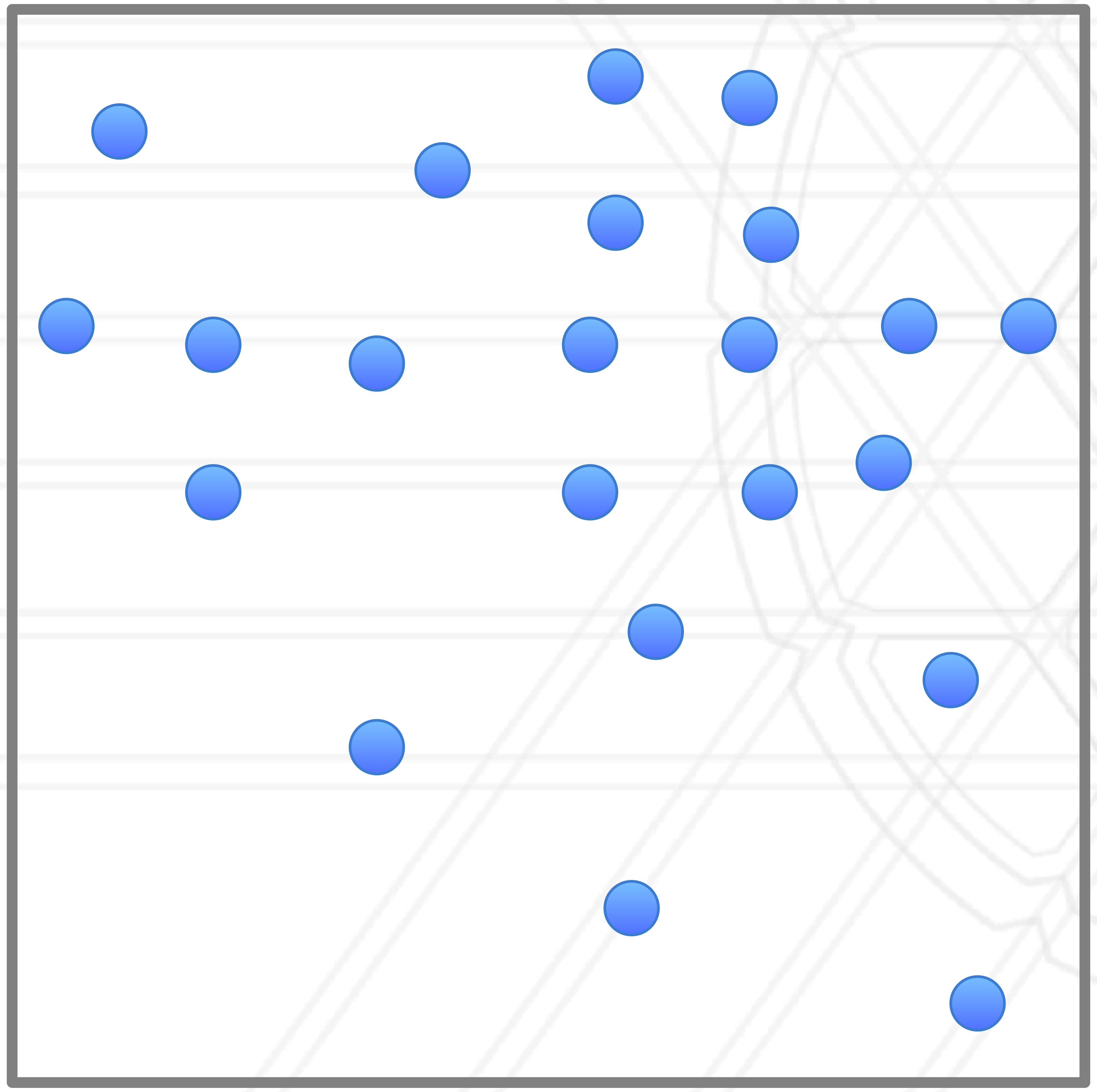
ORB



http://charm.cs.uiuc.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_apps_ChaNGa.pdf

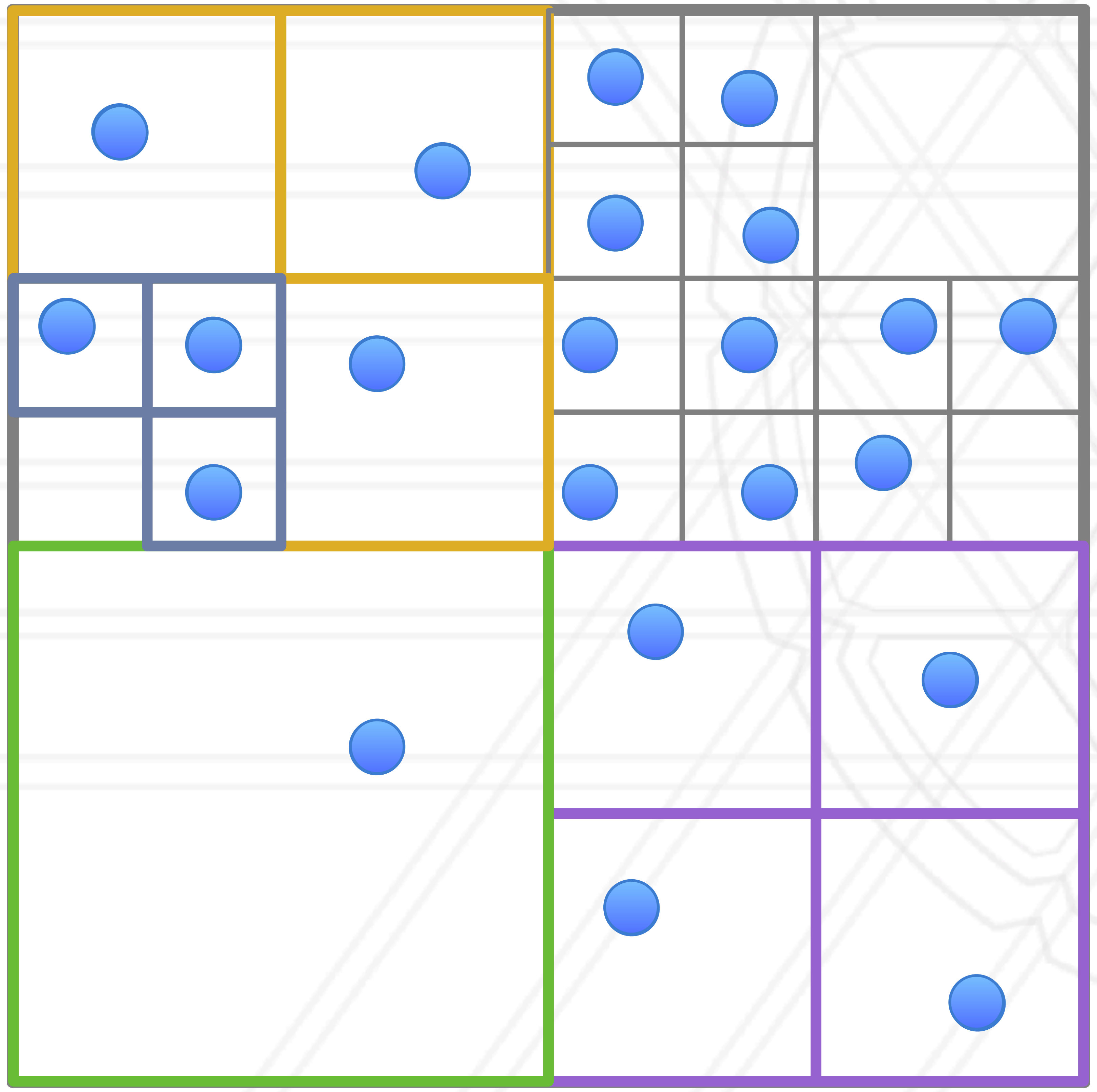
Data distribution in *n*-body problems

- Let us consider a two-dimensional space with bodies/particles in it



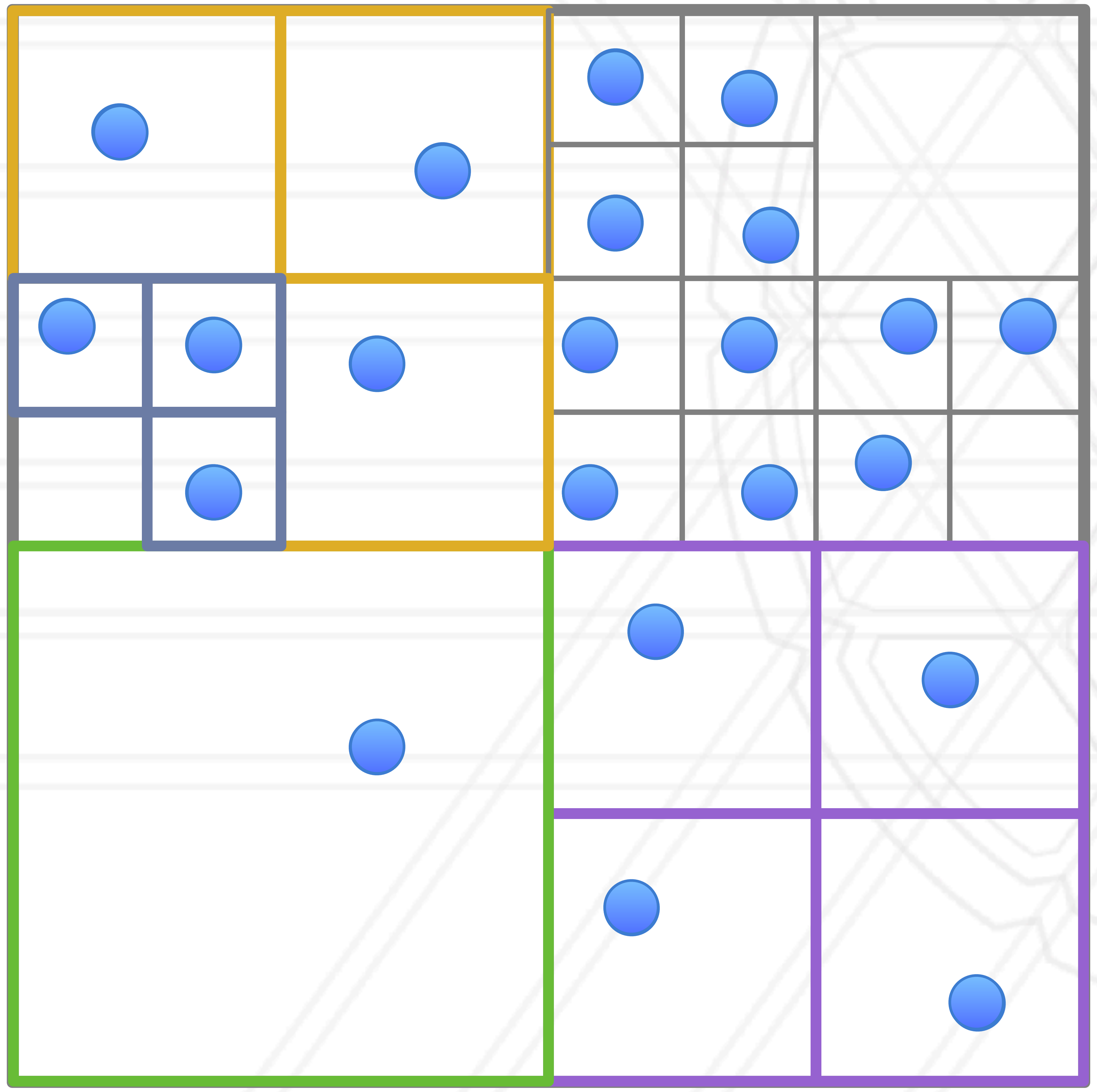
Data distribution in *n*-body problems

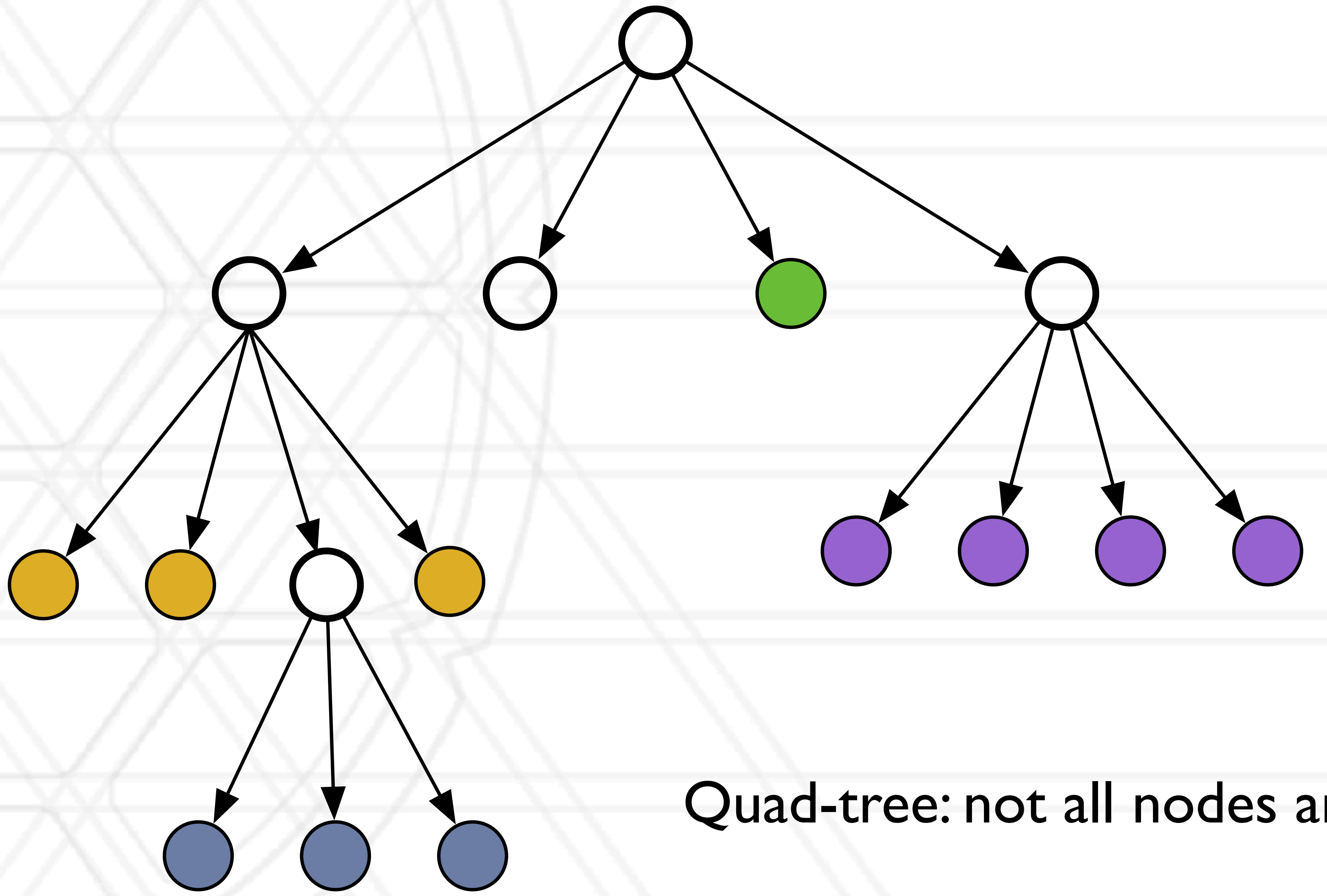
- Let us consider a two-dimensional space with bodies/particles in it



Data distribution in *n*-body problems

- Let us consider a two-dimensional space with bodies/particles in it





Quad-tree: not all nodes are shown

Load balance and grain size

- Load balance: try to balance the amount of work (computation) assigned to different threads/ processes
  - Bring ratio of maximum to average load as close to 1 as possible
  - Secondary consideration: also load balance amount of communication
- Grain size: ratio of computation-to-communication
  - Coarse-grained (more computation) vs. fine-grained (more communication)

UNIVERSITY OF MARYLAND

Abhinav Bhatele

5218 Brendan Iribe Center (IRB) / College Park, MD 20742

phone: 301.405.4507 / e-mail: bhatele@cs.umd.edu