Designing Parallel Programs

Alan Sussman, Department of Computer Science

UNIVERSITY OF MARYLAND

Announcements

- Zaratan accounts have been created for everyone
  - Some of you will have to activate your TERPConnect account, and should have received an email on how to do that
- ELMS should be visible to everyone now – so far only Announcements
- When emailing me, please cc the TAs also
  - Emails are on the class website: https://www.cs.umd.edu/class/spring2023/cmsc416/index.shtml
  - Prefix [CMSC416] to your email subject
- Office hours posted on class website – Zoom info for online hours in ELMS
- Assignment 0 will (likely) be posted on Feb. 9 and will be due a week later
  - Not graded, 0 points

Getting started with zaratan

- 380 nodes with AMD "Milan" processors (128 cores/node)
- 20 nodes with four NVIDIA A100 GPUs (in addition to 128 cores/node)

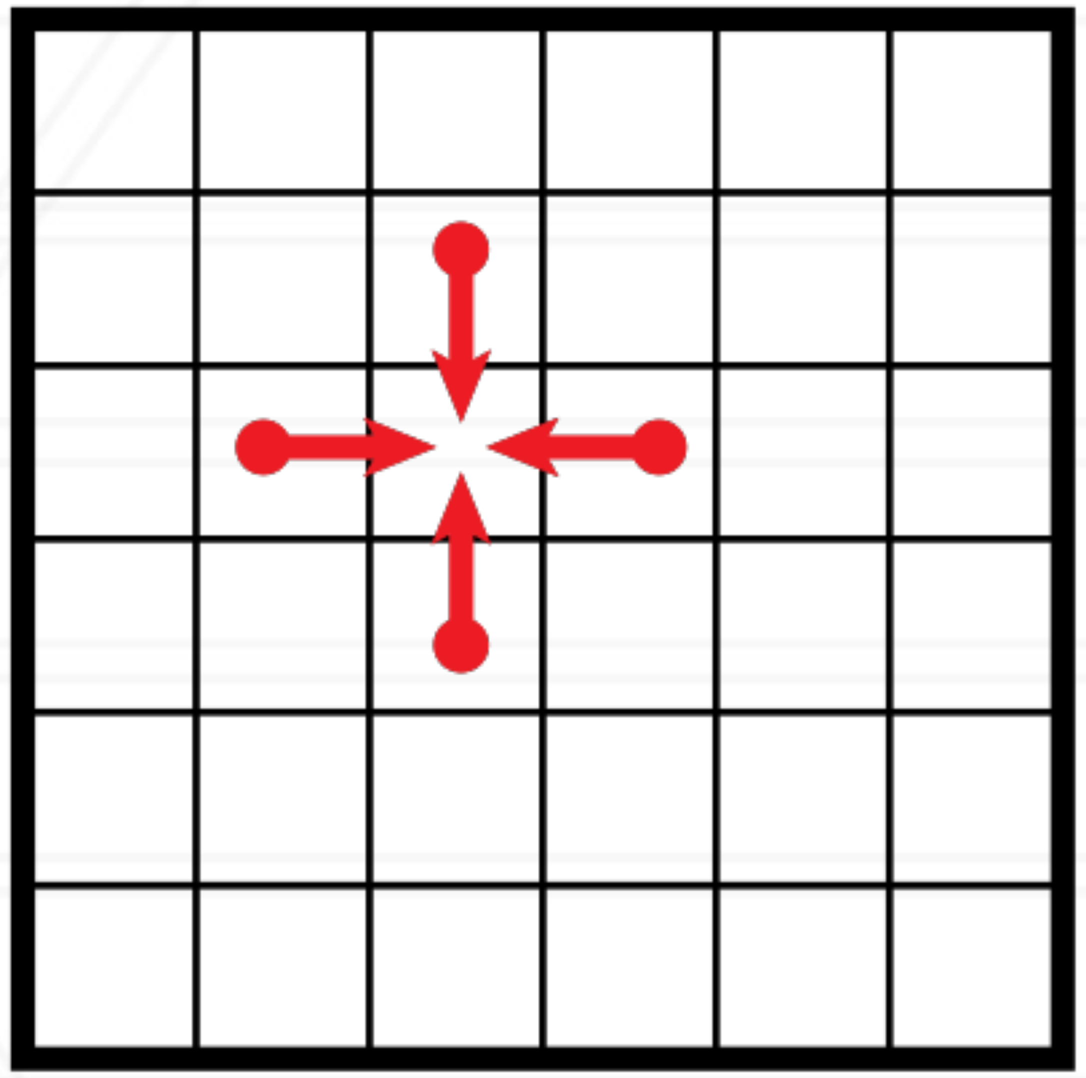
```
ssh username@login.zaratan.umd.edu
```

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



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

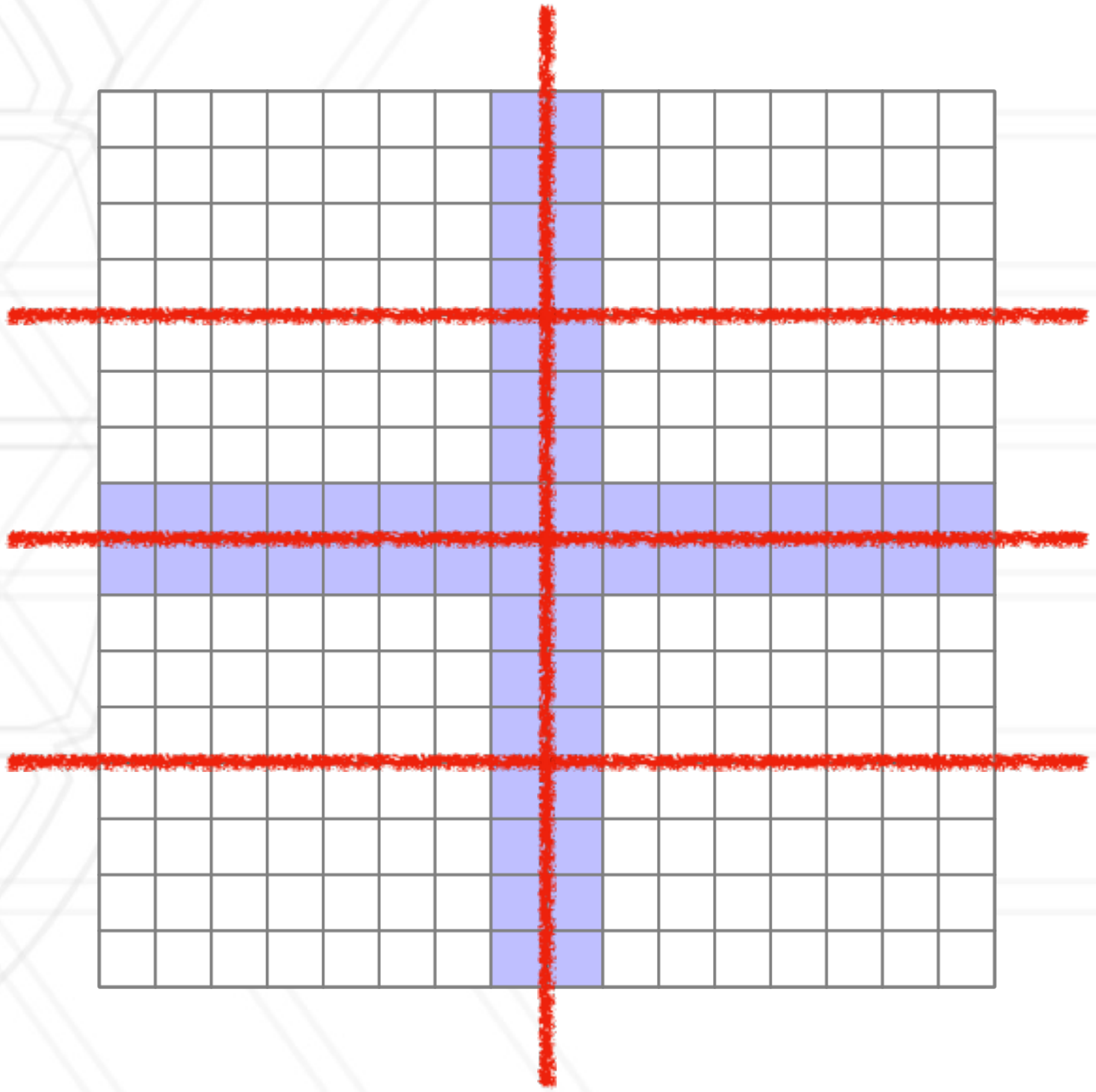
Serial code

```
for(int t=0; t<num_steps; t++) {
  ...

  for(i ...)
    for(j ...)
      A_new[i, j] = (A[I, j] + A[i-1, j] + A[i+1, j] + A[i, j-1] + A[i, j+1]) * 0.2


  // copy contents of A_new into A
  ...
}
```

2D stencil computation in parallel

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

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



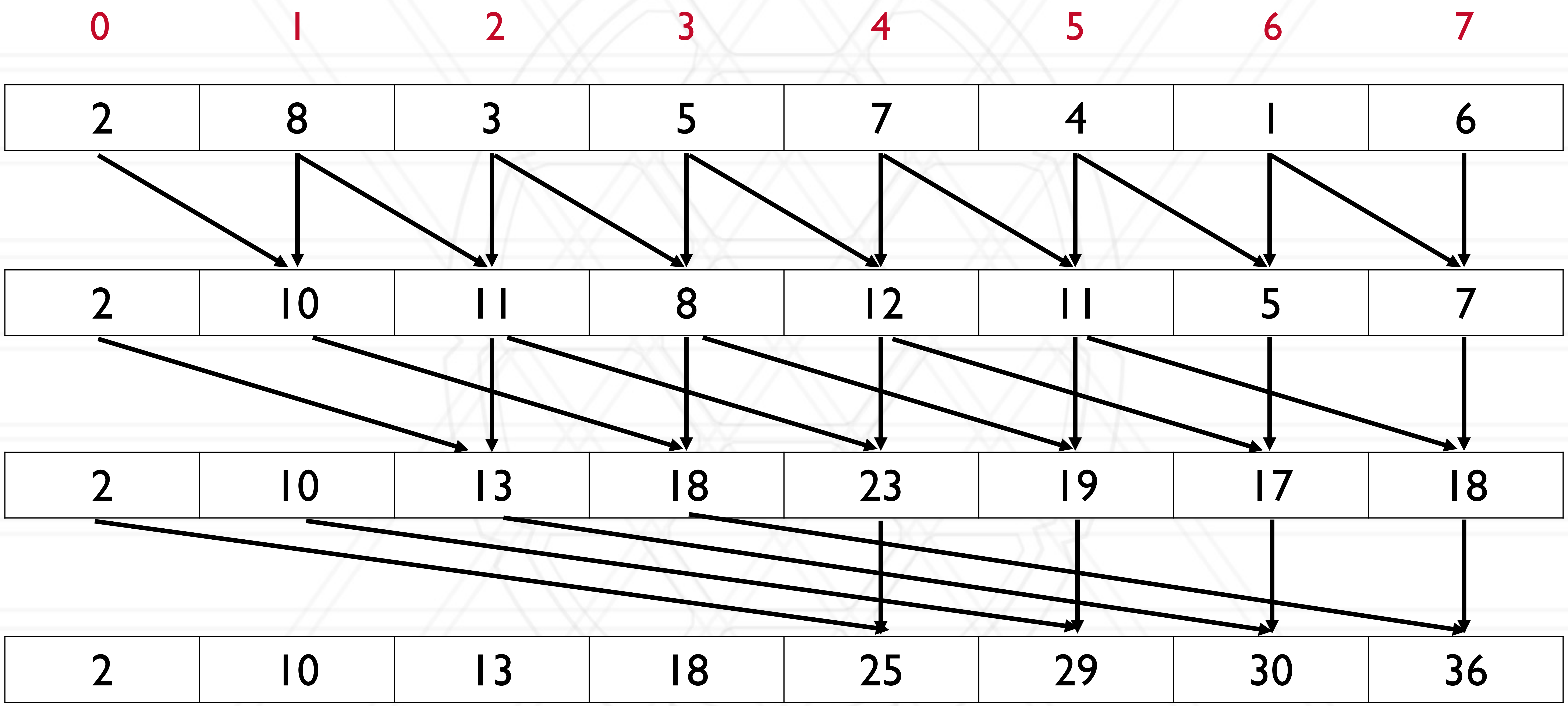
Prefix sum

- 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

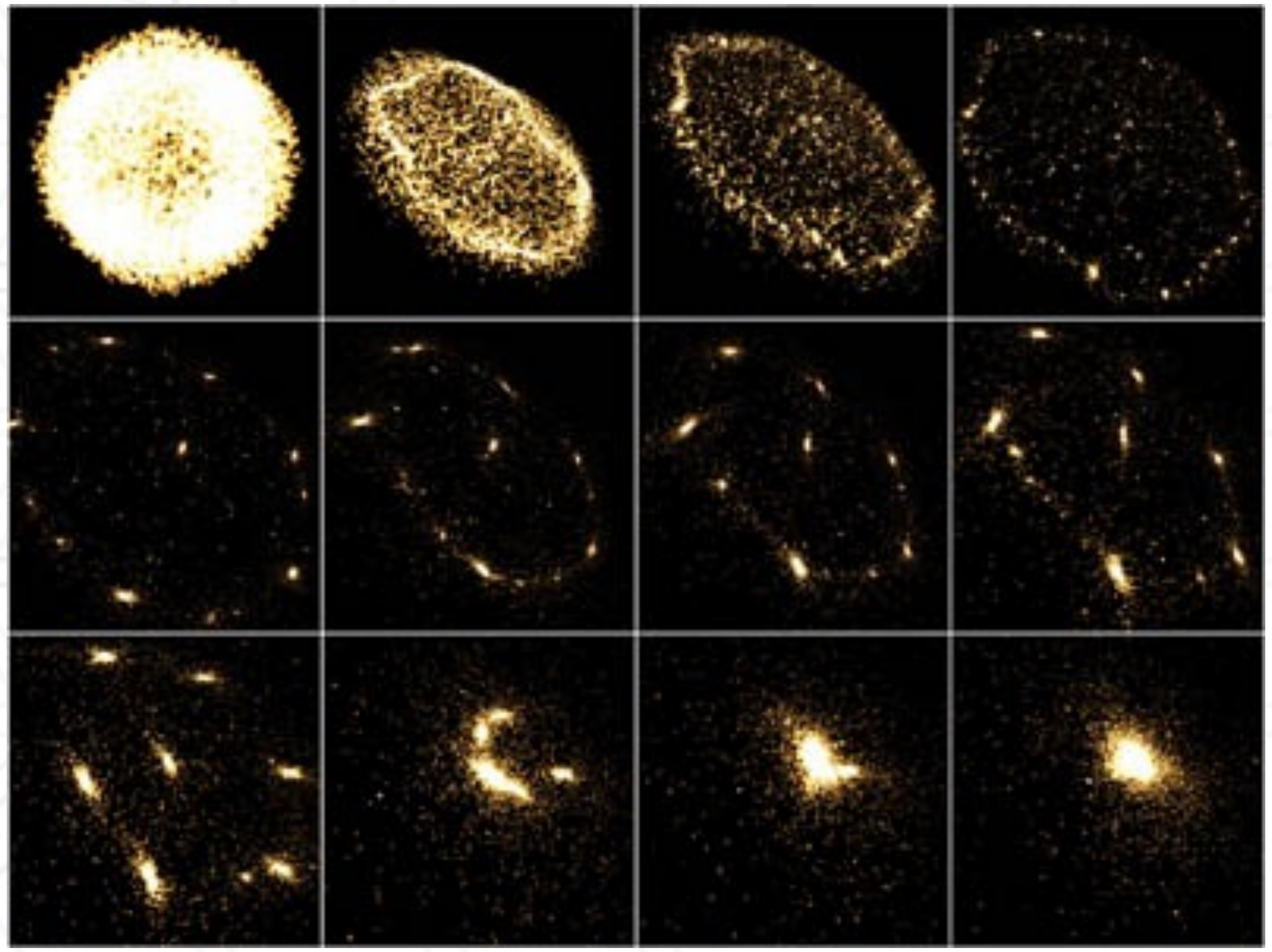


In practice

- 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)

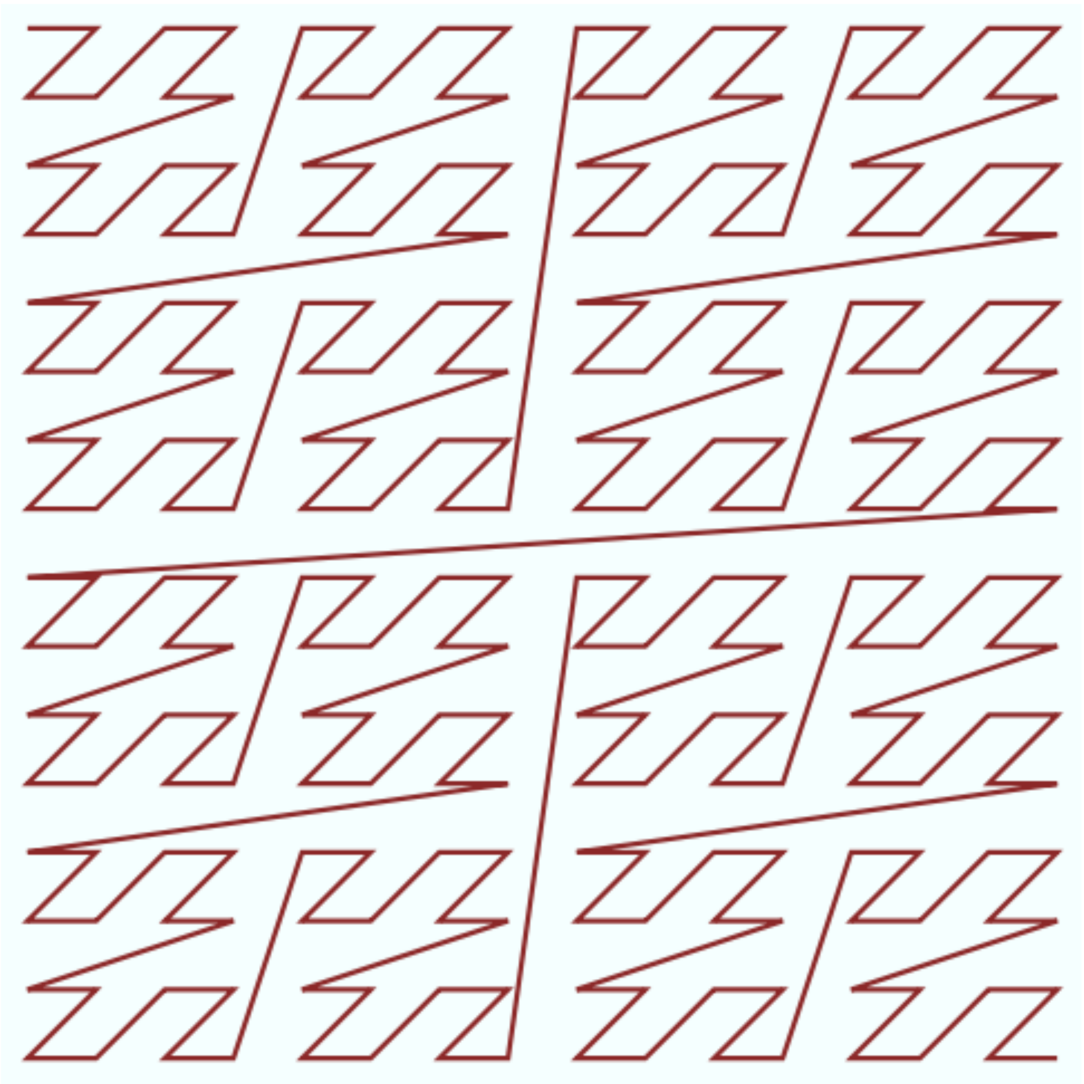


https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

Data distribution in *n*-body problems

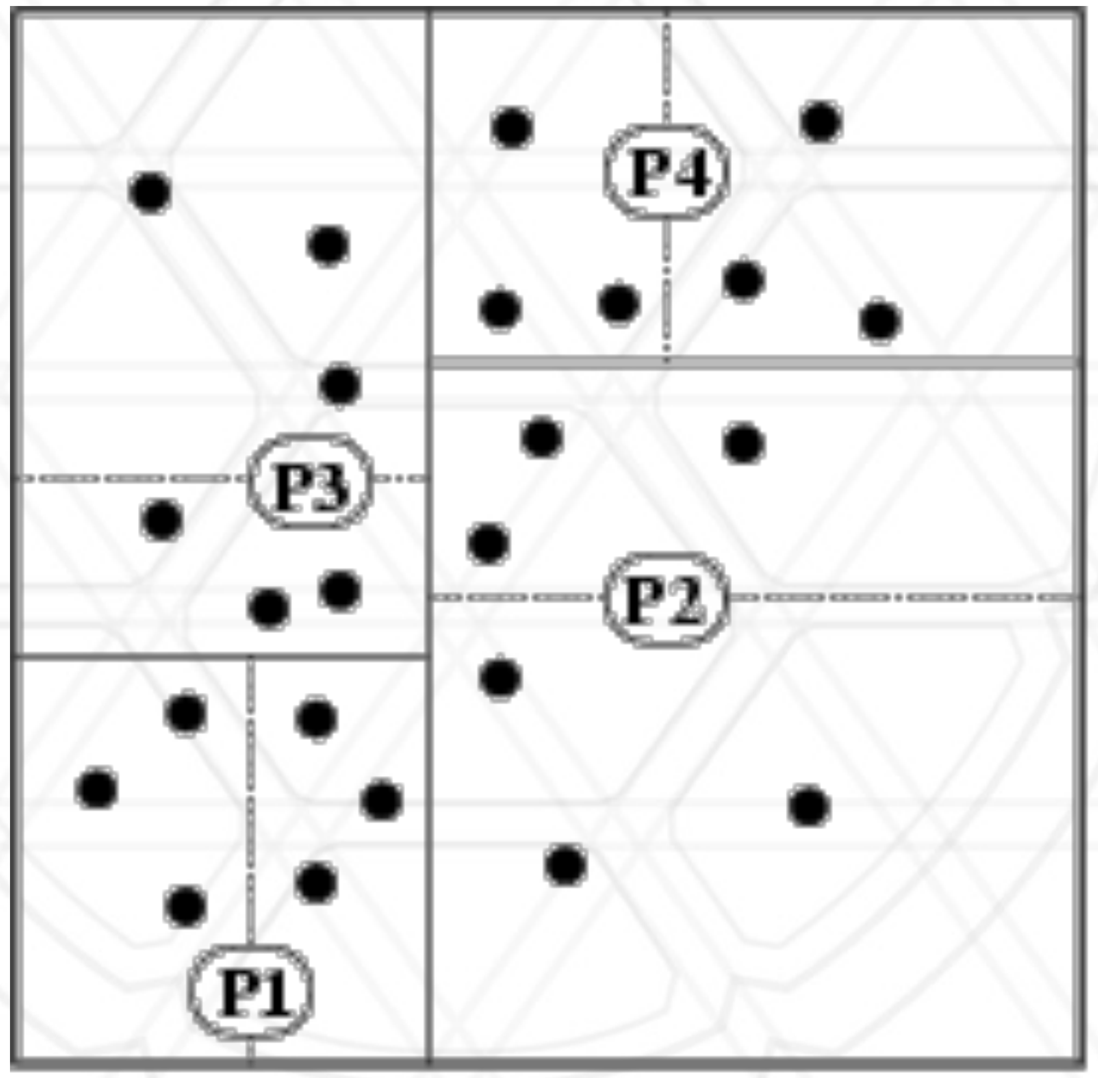
- Naive approach: Assign n/p particles to each process
- Other approaches?

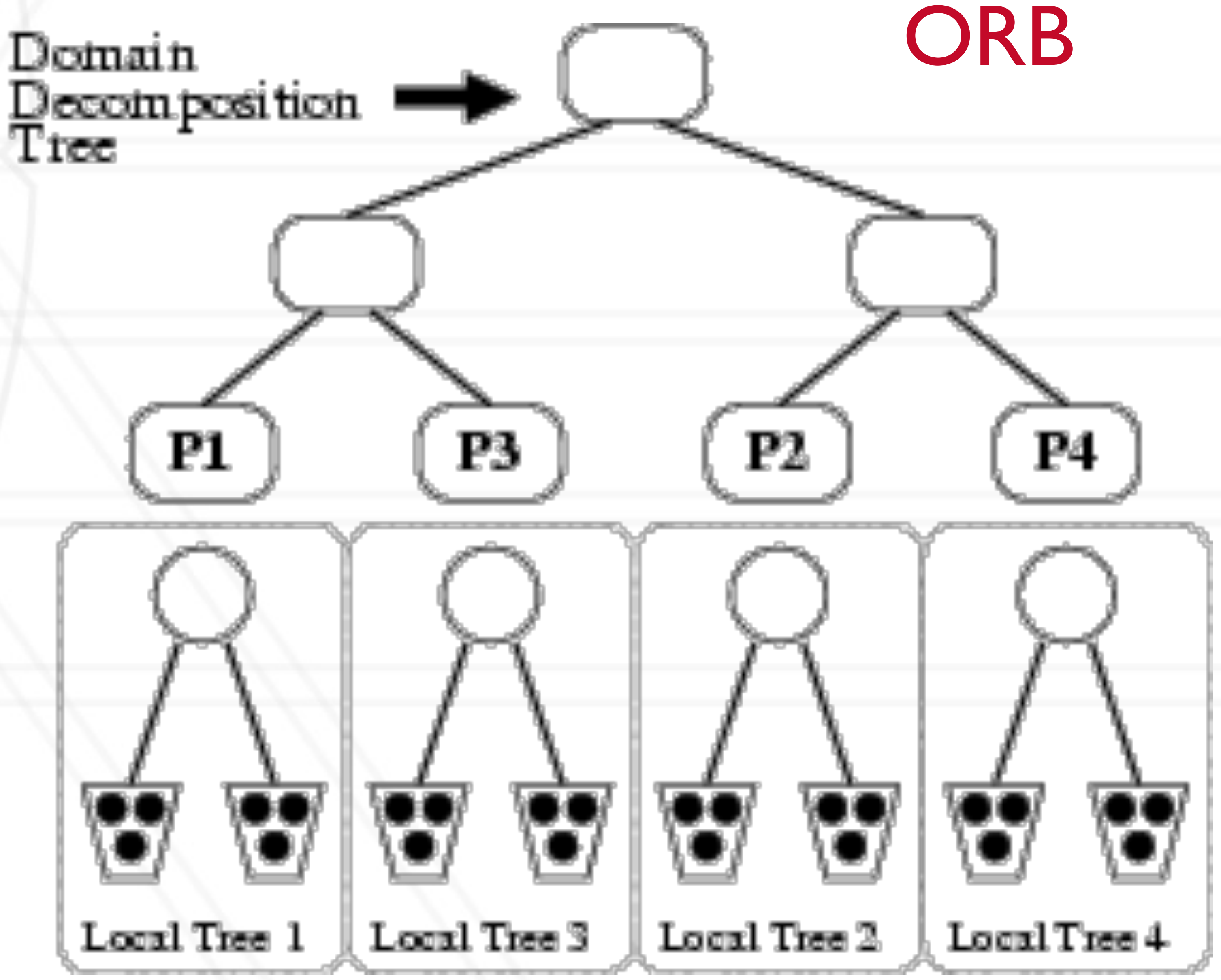
Space-filling curves



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

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

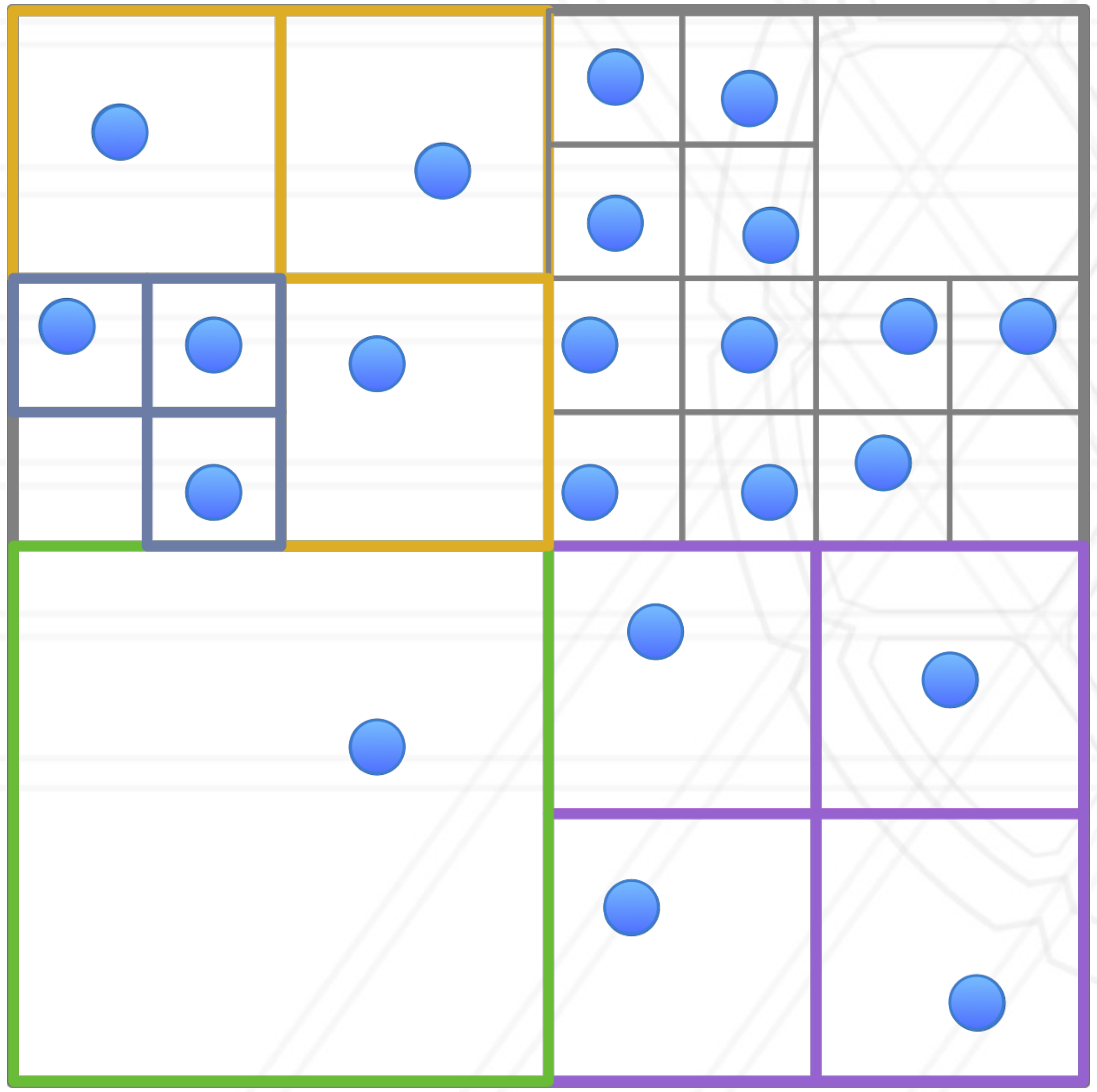


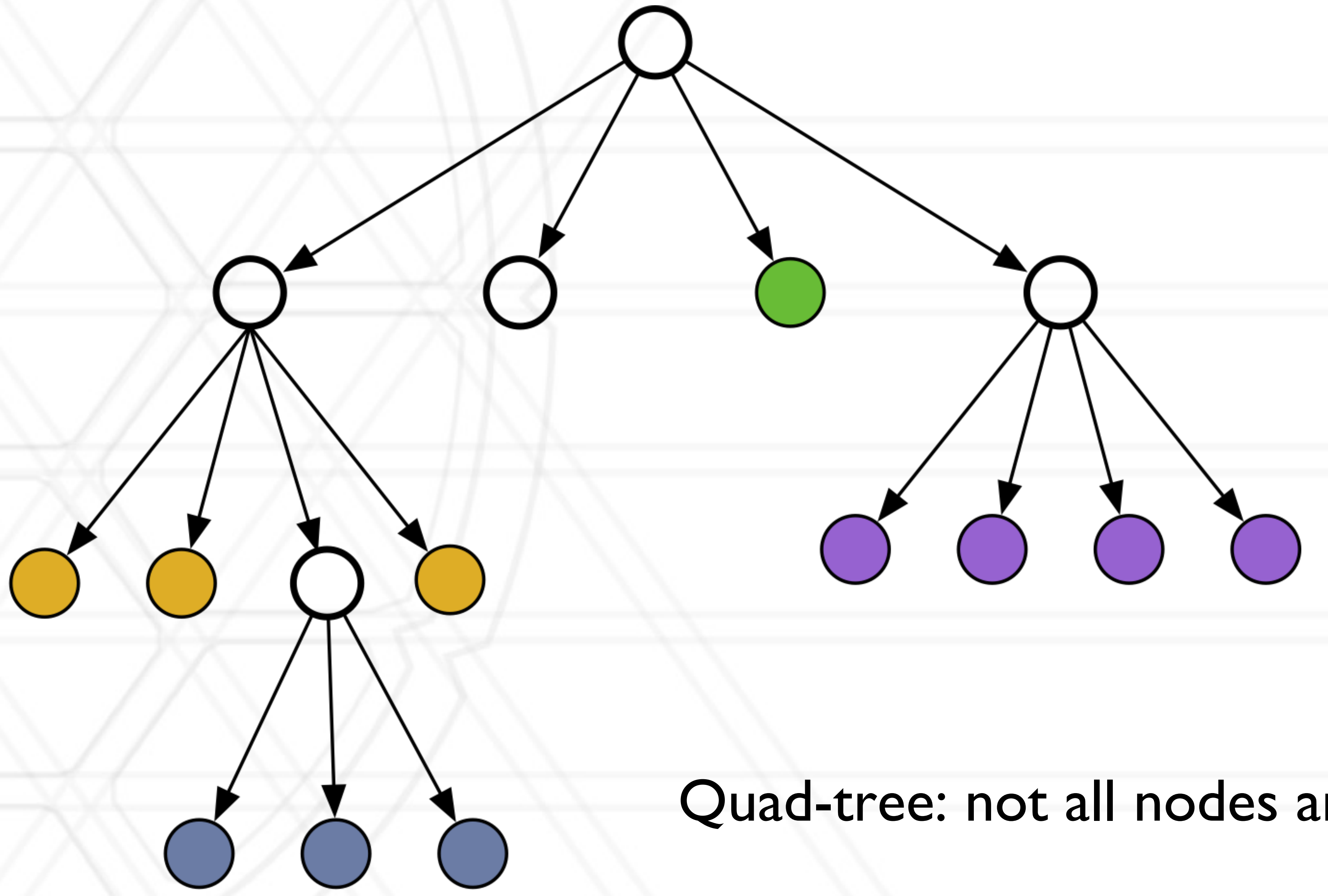


http://charm.cs.uiuc.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_apps_ChaNGa.pdf

Data distribution in *n*-body problems

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





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