Introduction to Parallel Computing (CMSC416)

# Parallel CSE Applications

Alan Sussman, Department of Computer Science

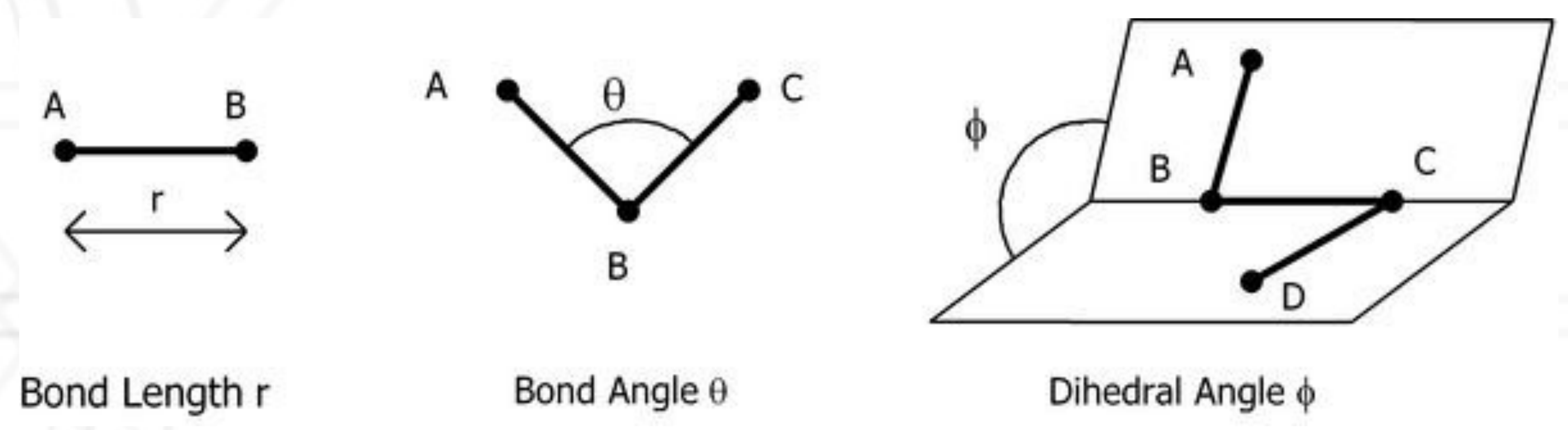
UNIVERSITY OF MARYLAND

# Announcements

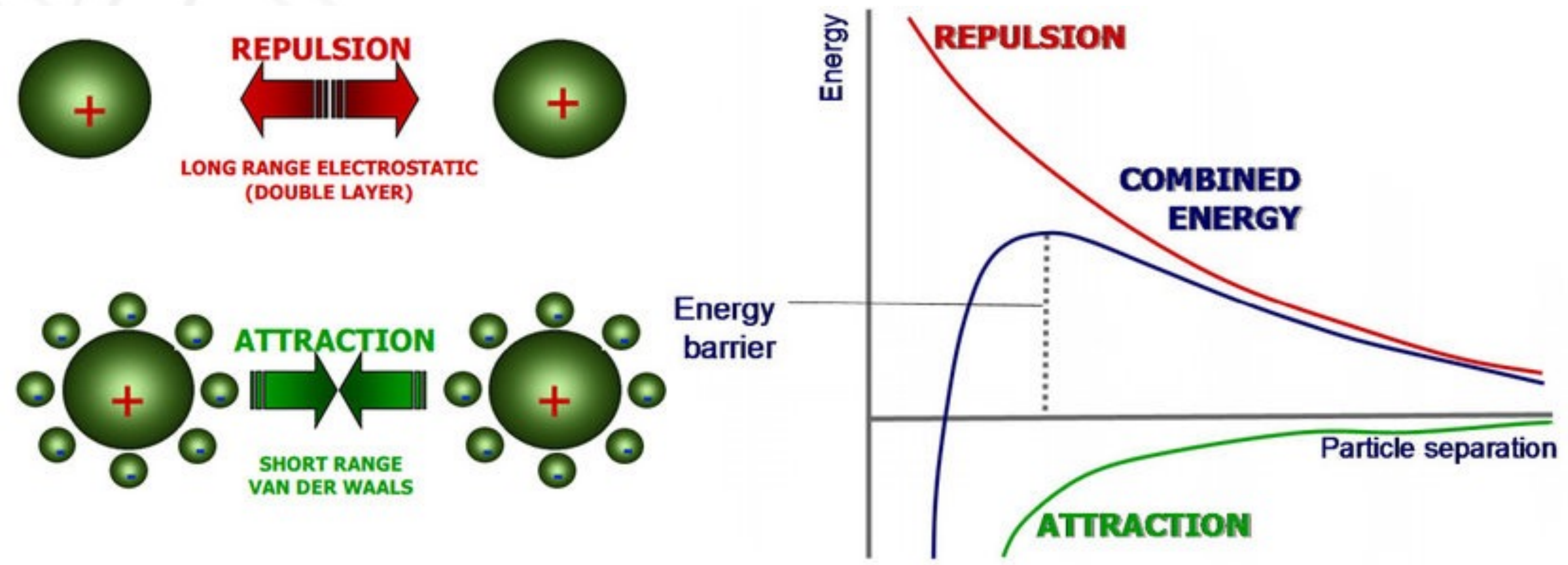
- Assignment 4 is due today, late deadline tomorrow
- Quiz 3 will be posted next Wednesday, May 10
  - In ELMS, for 24 hours
  - Mainly on topics since last quiz

# Molecular Dynamics

- Calculate trajectories of atoms and molecules by solving Newton's equations of motions
- Force calculations
  - Bonded interactions: bonds, angles, dihedrals
  - Non-bonded interactions: van der Waal's and electrostatic forces




- Number of atoms: thousands to millions
- Simulation step: ~1 femtosecond ($10^{-15}$ s)
- Applications to drug design, materials design, etc.

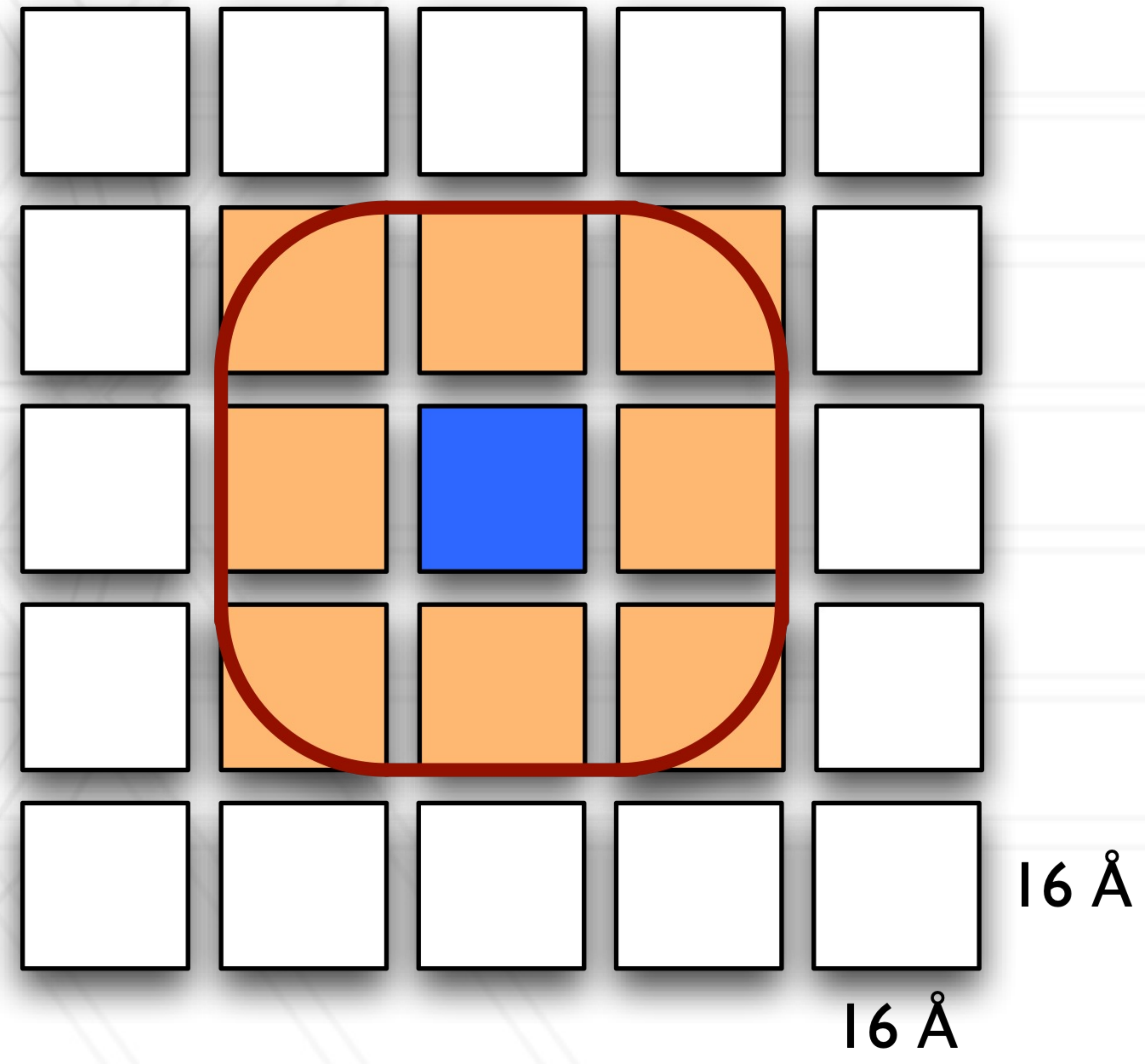


# Sequential Algorithm

- At every step, calculate forces on each atom
  - Calculate bonded and short-range forces every step
  - Calculate long-range non-bonded forces every few time steps (using PME, $P^3M$ methods, etc.)
- Particle mesh Ewald (PME) summation:
  - Calculate short-range forces in real space
  - Calculate long-range interactions in Fourier space –
    - Create a 3D mesh/grid representing charge densities of atoms
    - Use FFTs to reduce complexity from $O(N^2)$ to $O(N \log N)$
- Calculate velocities and new positions
- Repeat …

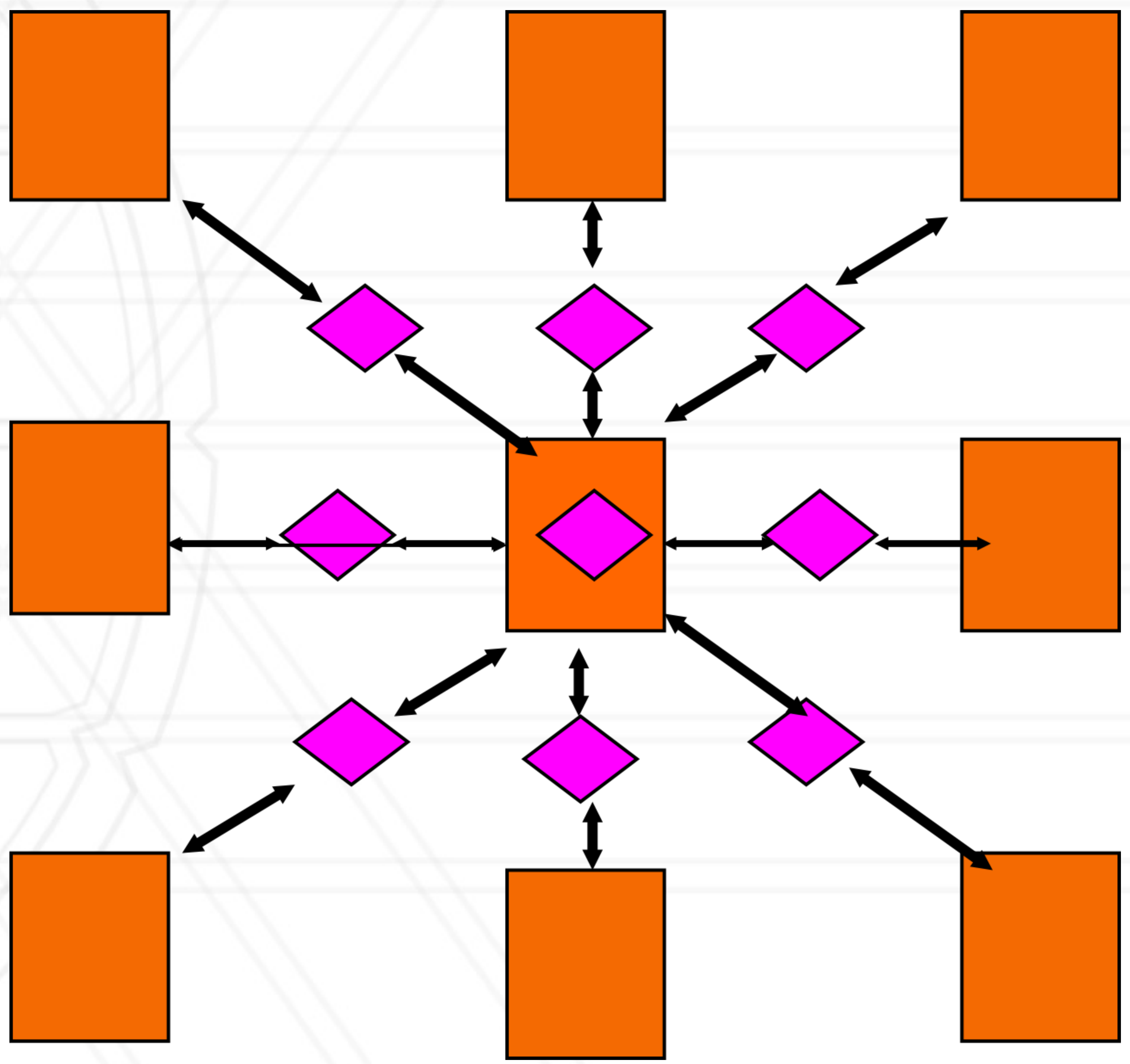
# Traditional approaches to parallelization

- Atom decomposition:
  - Partition the atoms across processes
- Force decomposition:
  - Distribute the force matrix to processes
  - Matrix is sparse and non-uniform
- Spatial decomposition:
  - Assign a region of the 3D simulation space to each process



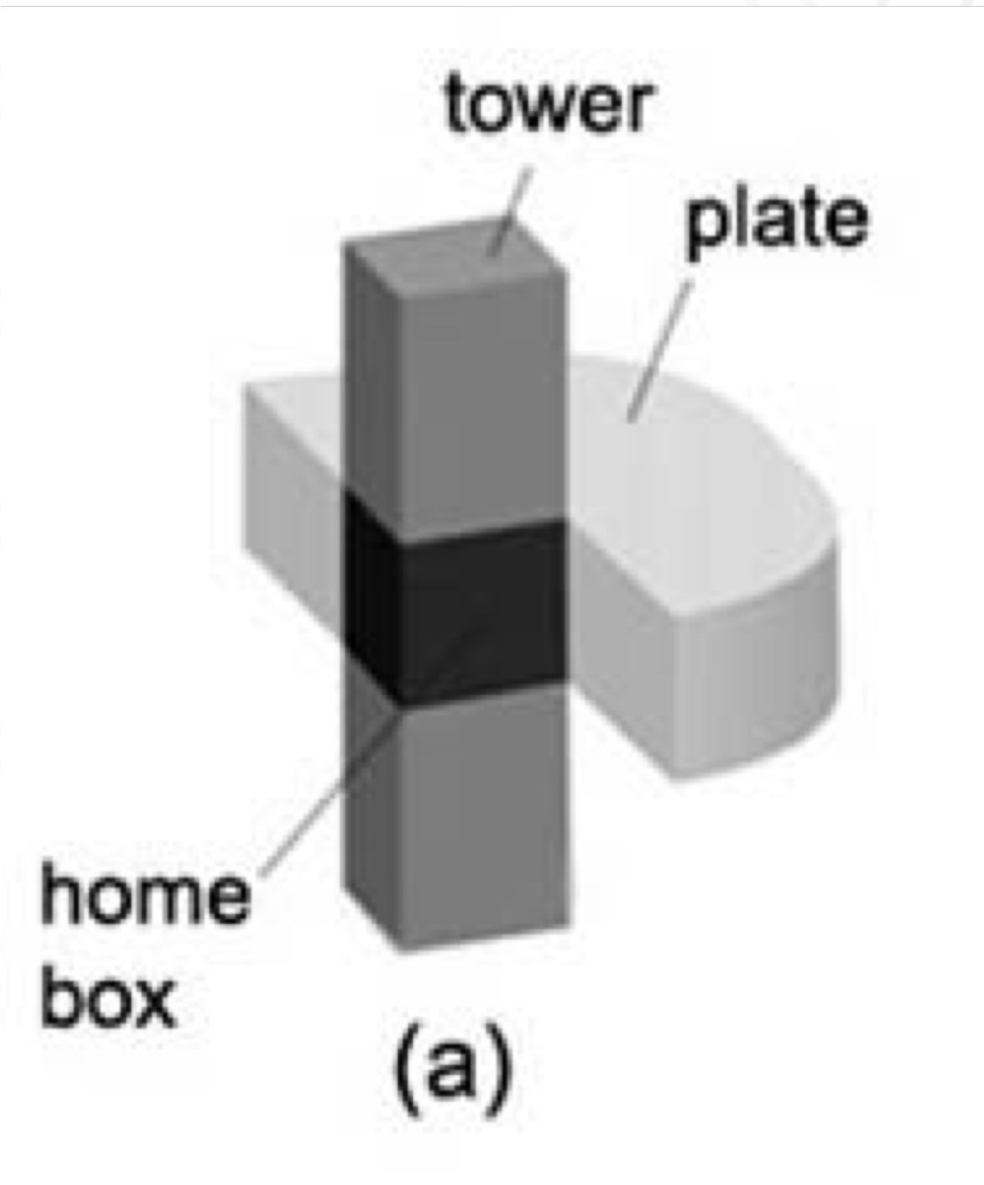
# Hybrid parallelization

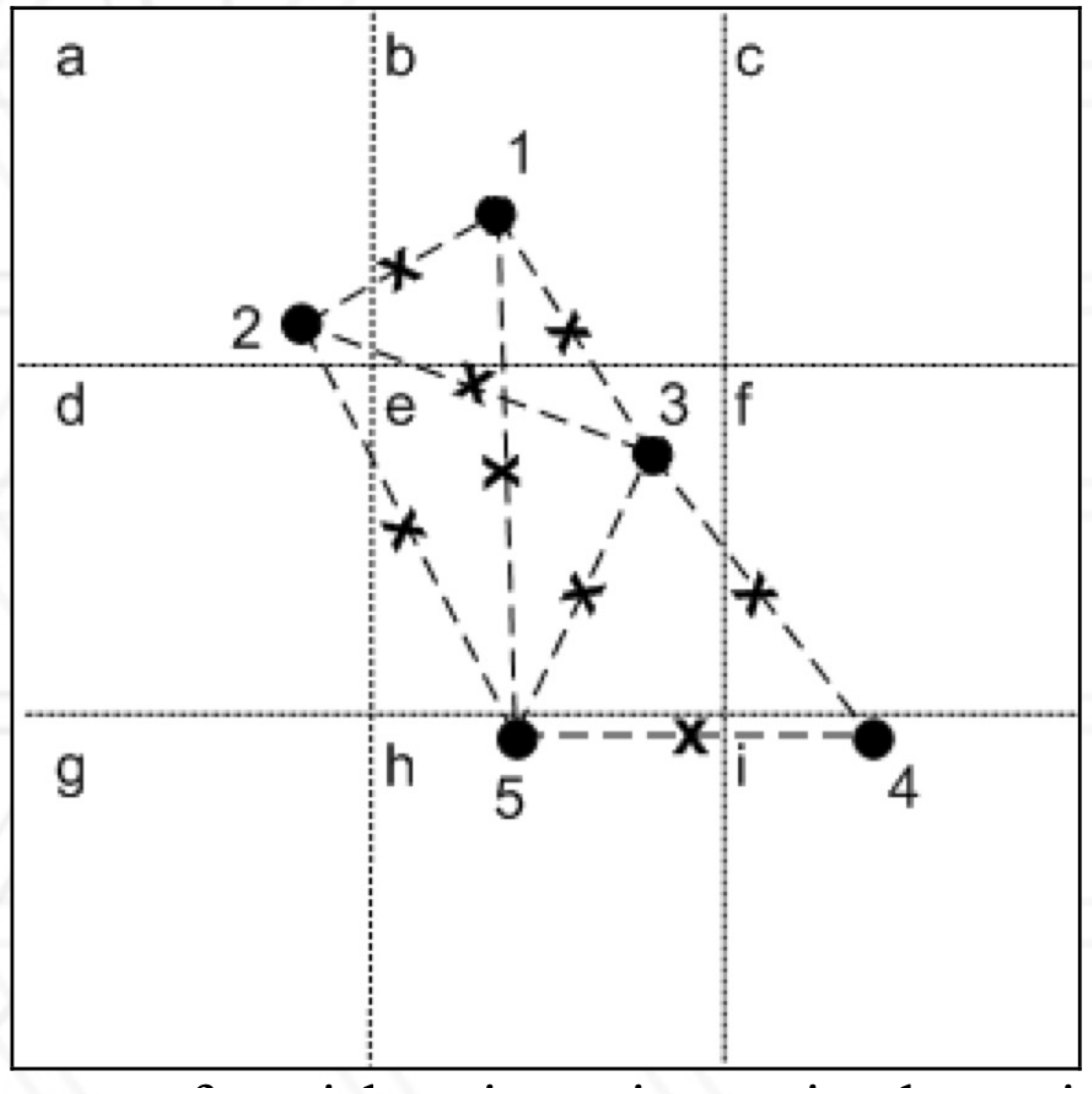
- Hybrid of spatial and force decomposition
- Decouple assignment of data and work to processes
- Distribute both atoms and the force calculations to different processes



# Neutral territory (NT) methods

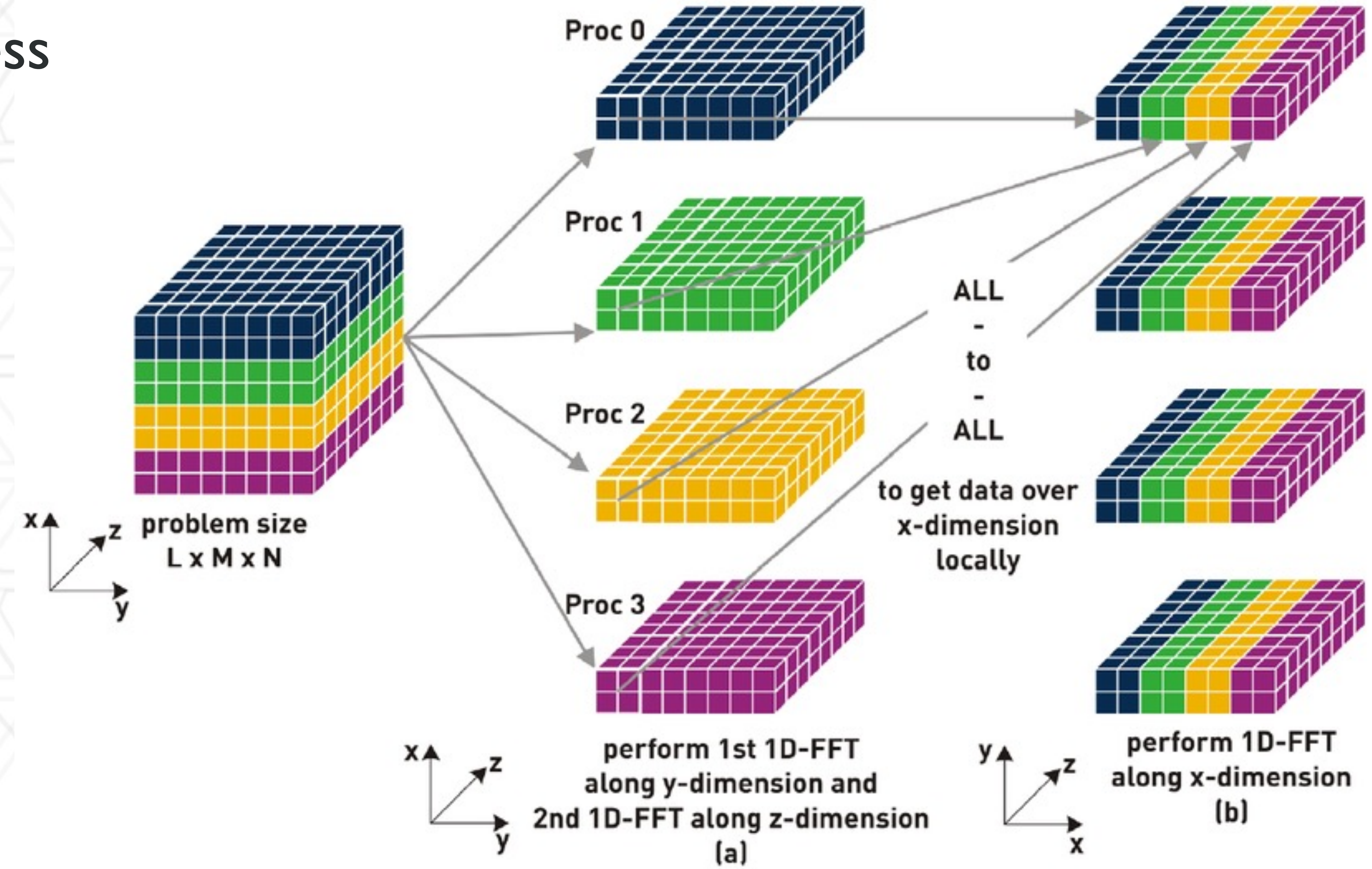
- Desmond's mid-point method



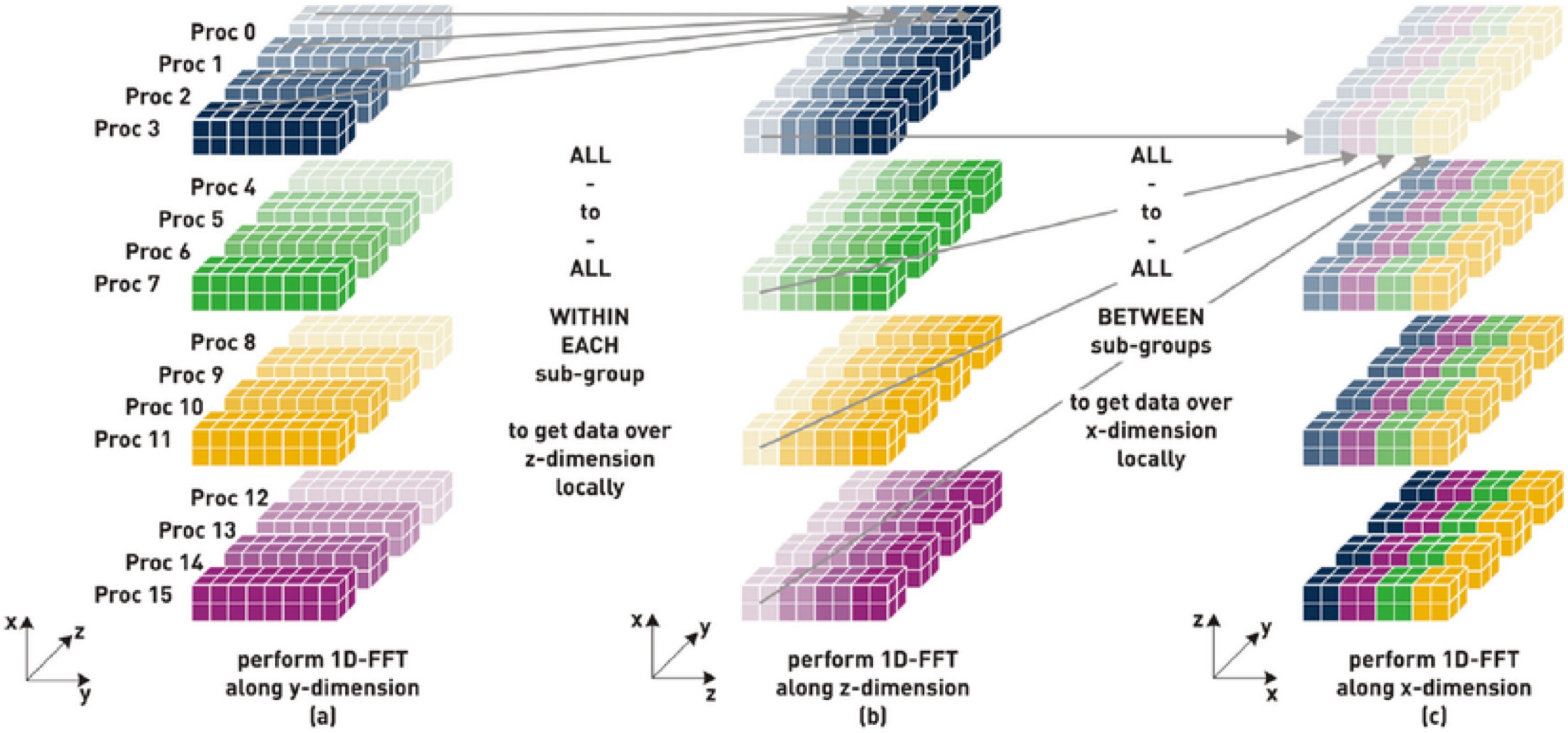
# Parallelization of PME (3D FFT)

- Bring all the data to one process
- 1D or slab decomposition

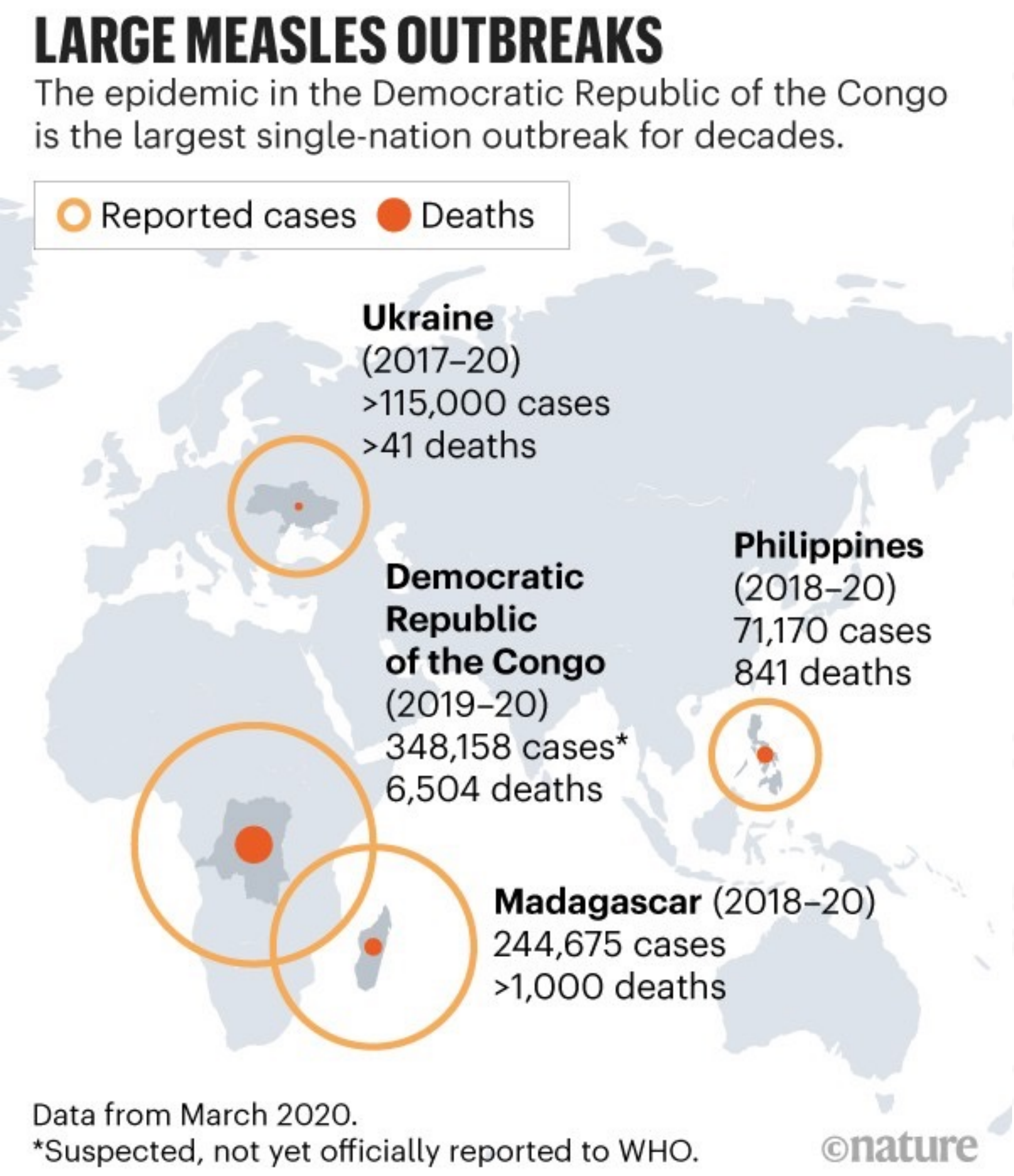


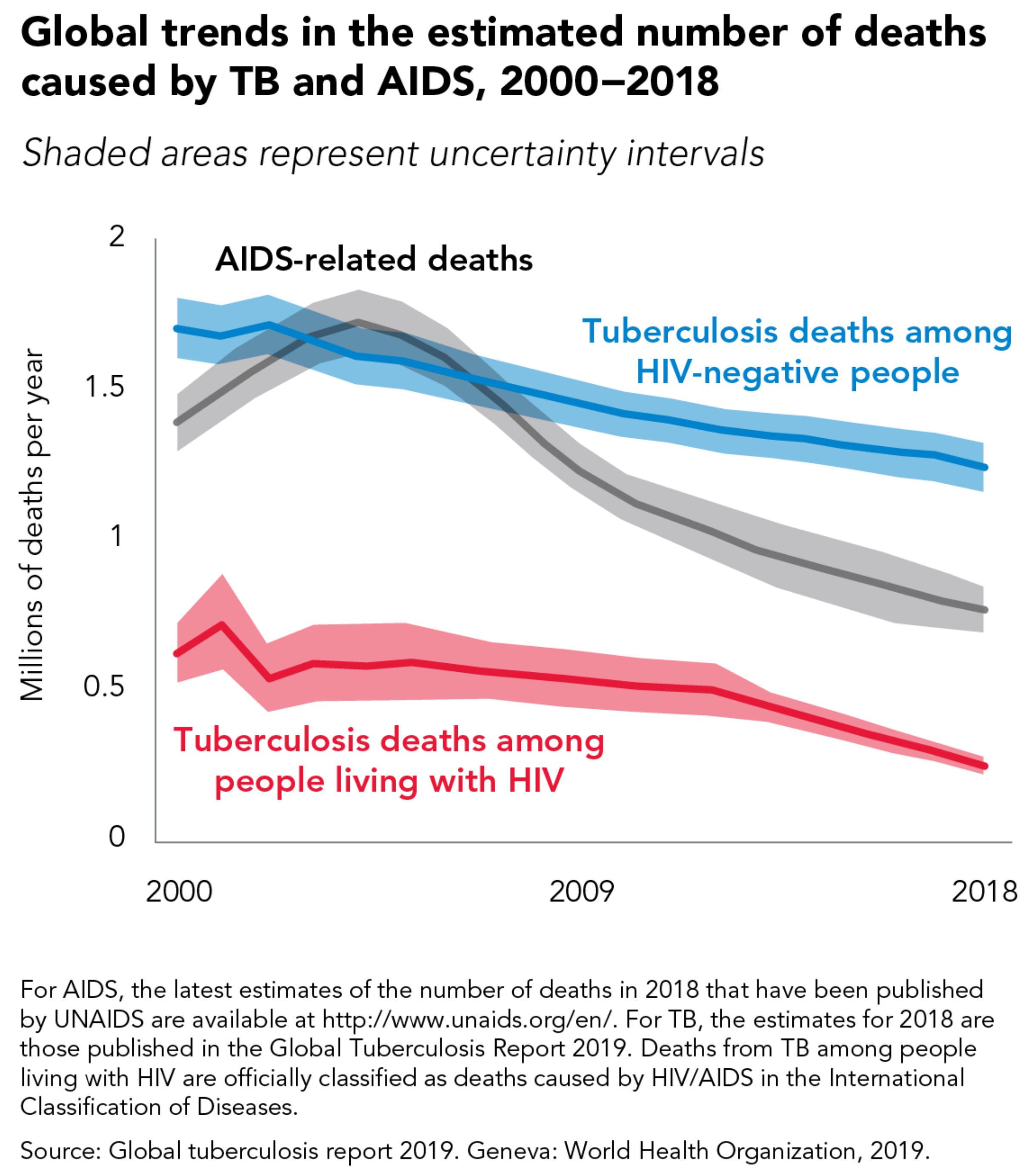
# Parallelization of PME (3D FFT)

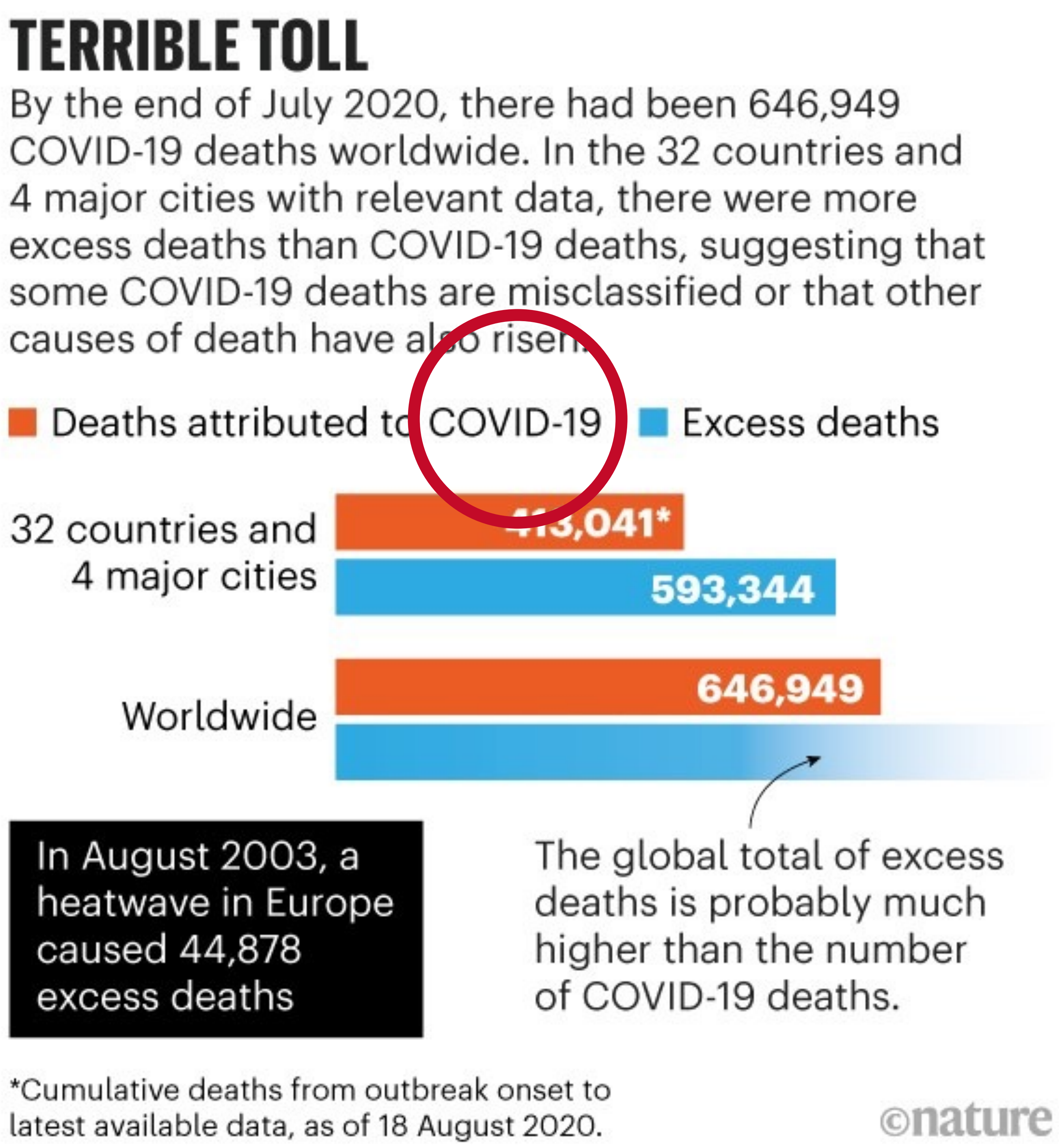
- 2D or pencil decomposition



# Measles killed 200,000 in 2020 alone!




Predictions said that 1.66 million people died of tuberculosis in 2020

https://www.nature.com/articles/d41586-020-01011-6

https://www.nature.com/articles/d41586-020-02497-w

# Societal challenge

- Controlling the spread of infectious diseases is important
- Computational and mathematical modeling of epidemics important to assist governments in responding to outbreaks
- Made challenging due to:
  - increased and denser urbanization
  - increased local and global travel
  - increasingly immuno-compromised population

# Approach: individual-based simulation

- Agent-based modeling to simulate epidemic diffusion
- Models agents (people) and interactions between them
- People interact when they visit the same location at the same time
- These "interactions" between pairs of people are represented as "visits" to locations
- Use a bi-partite graph of people and locations or a people-people interactivity graph

# Serial algorithm

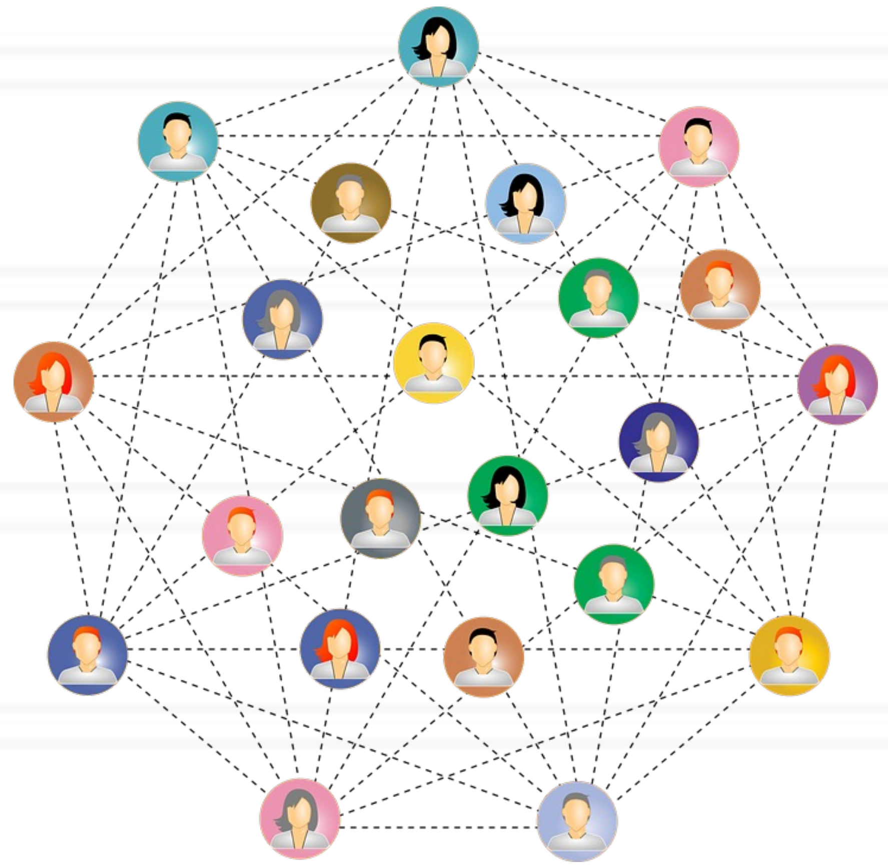
- At each timestep (typically a day):
  - Determine which people visit which locations
  - "Send" people to those locations
  - At each location "interactions" happen and transmission happens
  - Update people's states at the end of the day and continue
- Interventions (vaccinations, school closures) can be added on certain days to change people's susceptibility, movements etc.

# Combination of network theory and discrete-event simulations

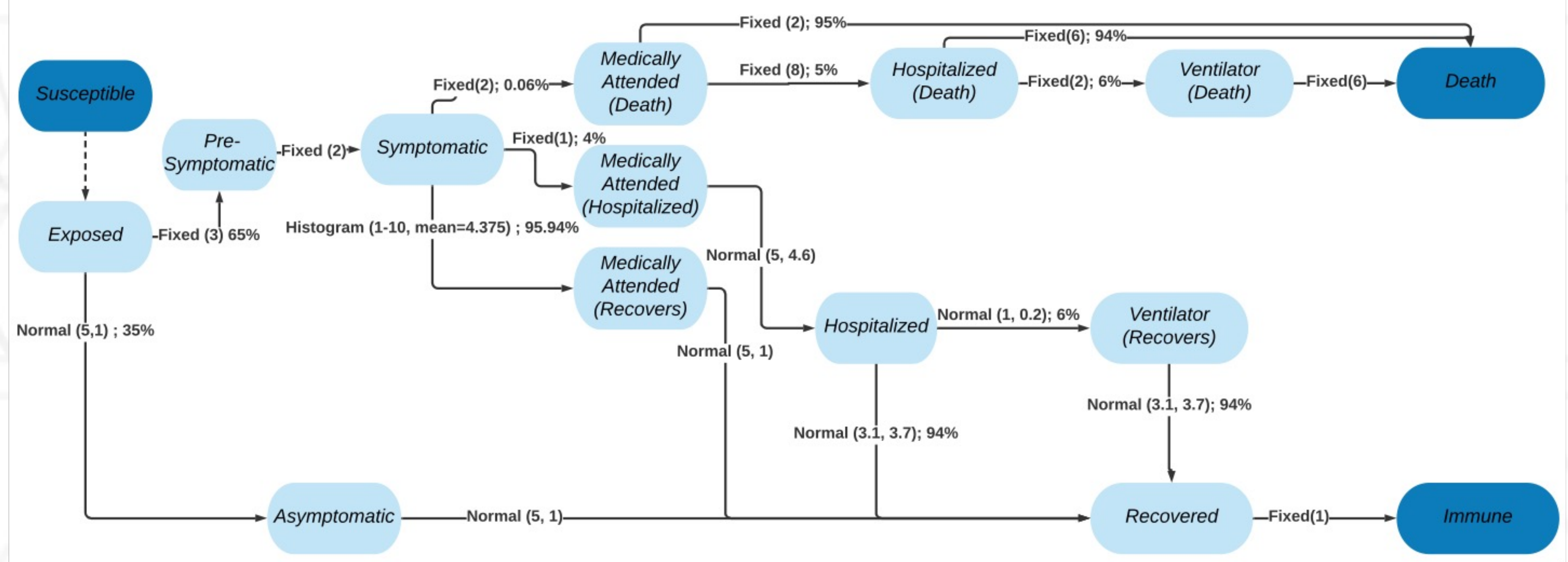
- Hybrid time-stepped and discrete-event simulation

```
while d < num_days:
  for each person:
    Send visit messages to locations

  for each location:
    Process all visit messages
    Run discrete event simulation
    Send interaction messages

  for each person:
    Process interactions
    Update disease state
```

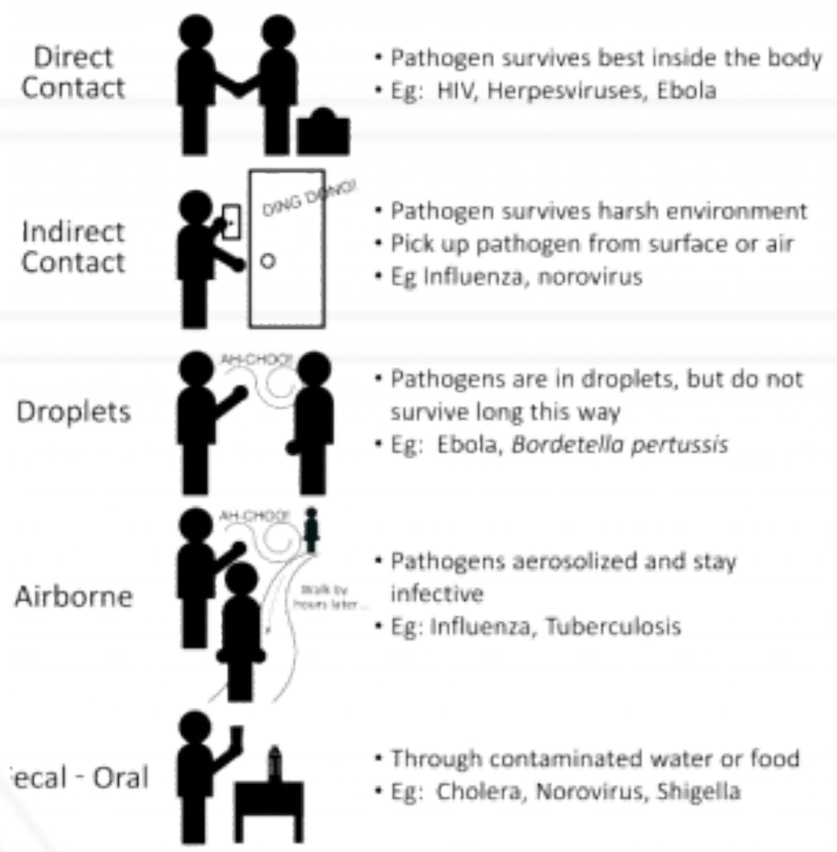
Contact model at each location



Disease model for each person




Model for human transmission



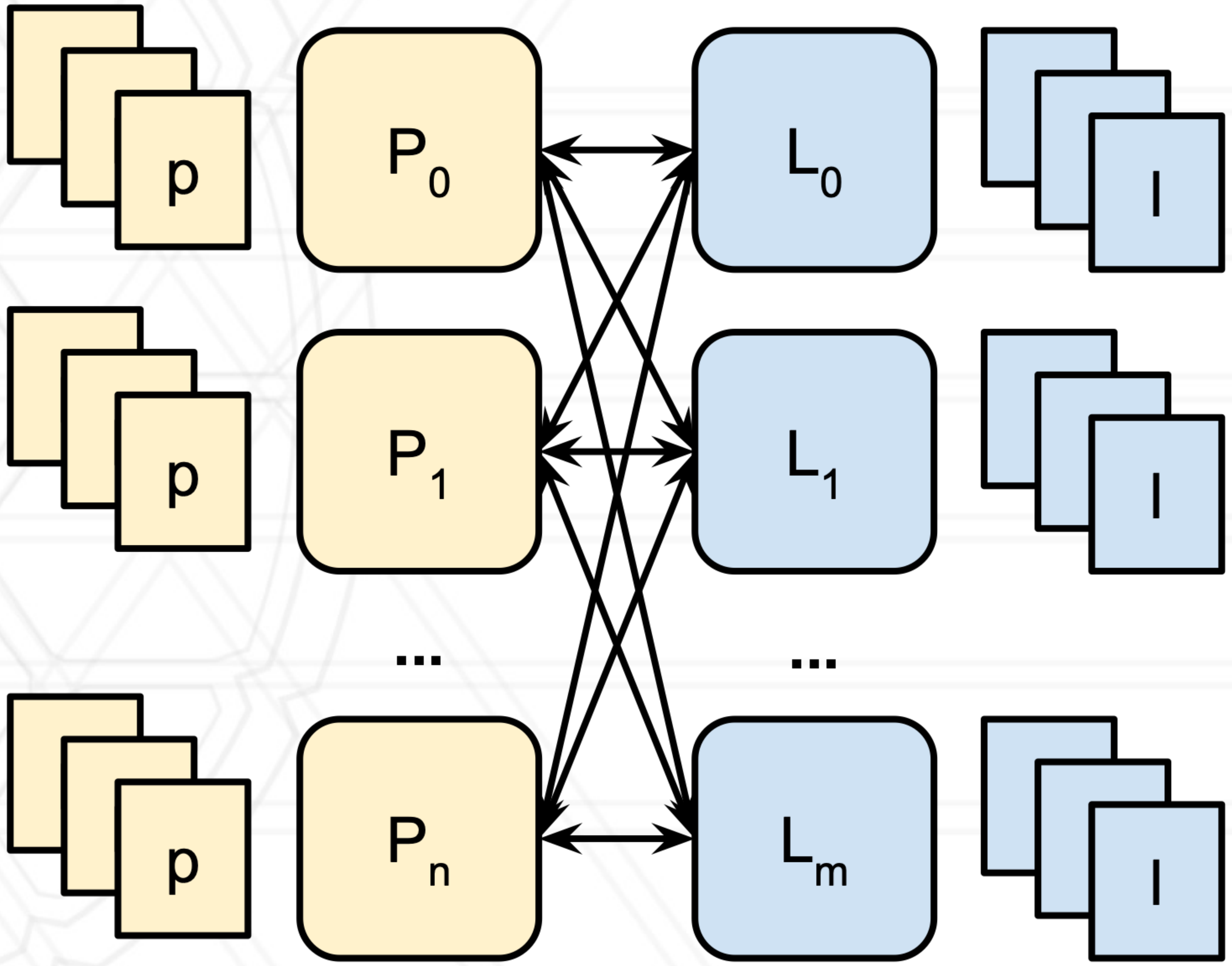

https://sitn.hms.harvard.edu/flash/special-edition-on-infectious-disease/2014/an-introduction-to-infectious-disease/

# Parallel simulation is challenging

- Size and scale of the social contact network (8 billion agents for a global simulation)
  - Unstructured networks and complicated dependencies lead to high communication cost
- Individuals and their behaviors are not identical
- Co-evolving epidemics, public policies and agent behaviors make it impossible to apply standard model reduction techniques

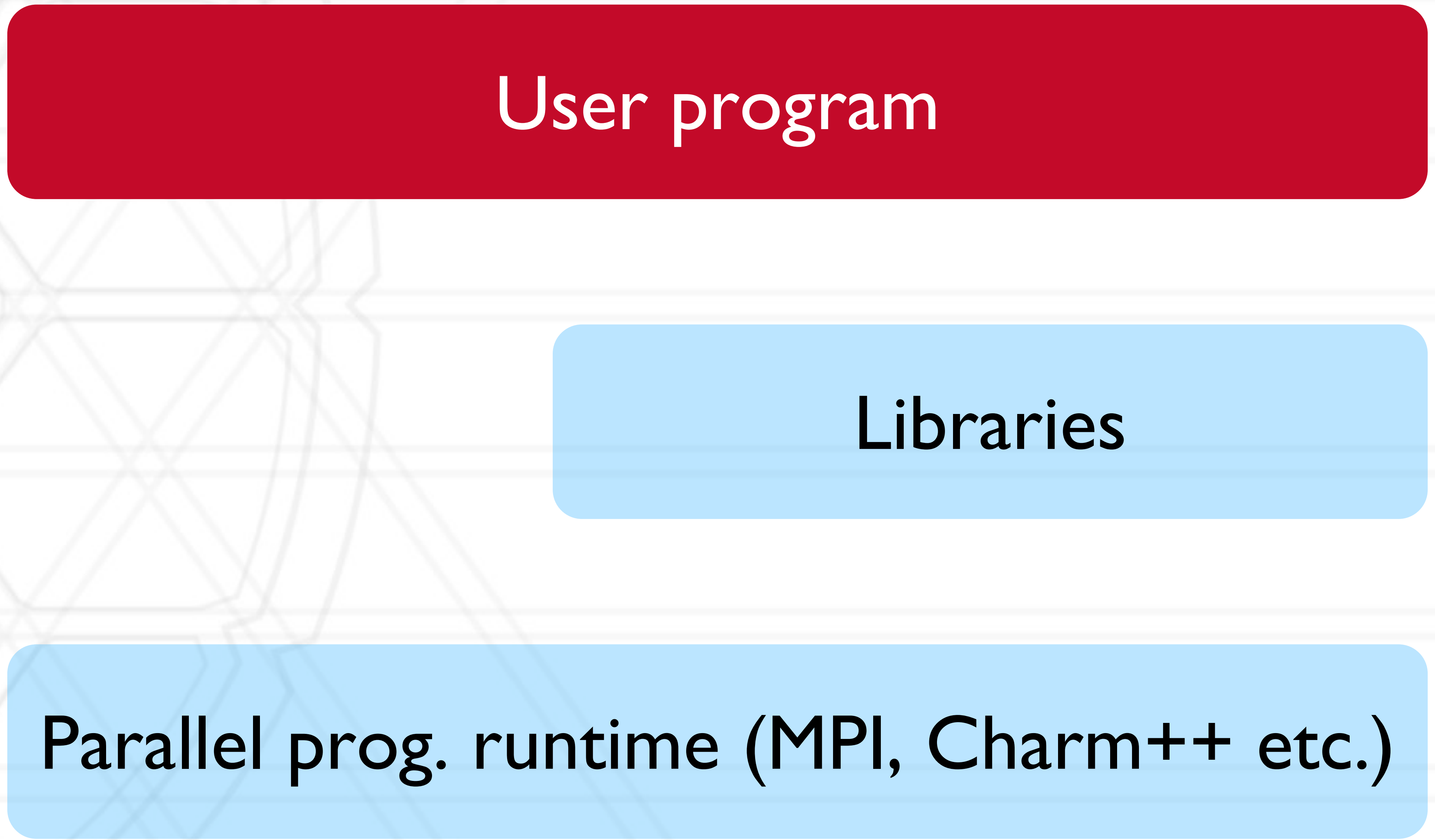
# Parallel implementation: Loimos

- All the people and locations are distributed among all processes
- DES computation can be done locally in parallel
- Communication when sending visit and infection messages
- Uses Charm++, a message-driven model



# Application software stack

- Parallel programming model / runtime:
  - MPI, OpenMP, Charm++, CUDA, …
- Libraries
  - Data management and visualization libraries (mesh management, simulation output)
  - I/O libraries
  - Math/numerical libraries
  - Graph partitioning, load balancing …



# Why use libraries?

- No need to reinvent the wheel
- Tend to have good performance
- Tend to have fewer bugs
- Can be called from high-level languages
  - C++, Python, Java, Ruby, Matlab, Mathematica, …
- Makes the code more portable
  - Even to GPUs
- Avoids significant effort to write and optimize code

# Popular Libraries

- Data/visualization and I/O libraries
  - I/O: HDF5, pNetCDF, ADIOS (Adaptable I/O System)
- Numerical libraries:
  - Fast Fourier Transform: FFTW
  - Dense linear algebra: BLAS, LAPACK, Intel MKL
  - Solvers for sparse systems: Hypre, PETSc, Trilinos, …
- Graph partitioning/load balancing:
  - METIS, Scotch, Zoltan, Chaco

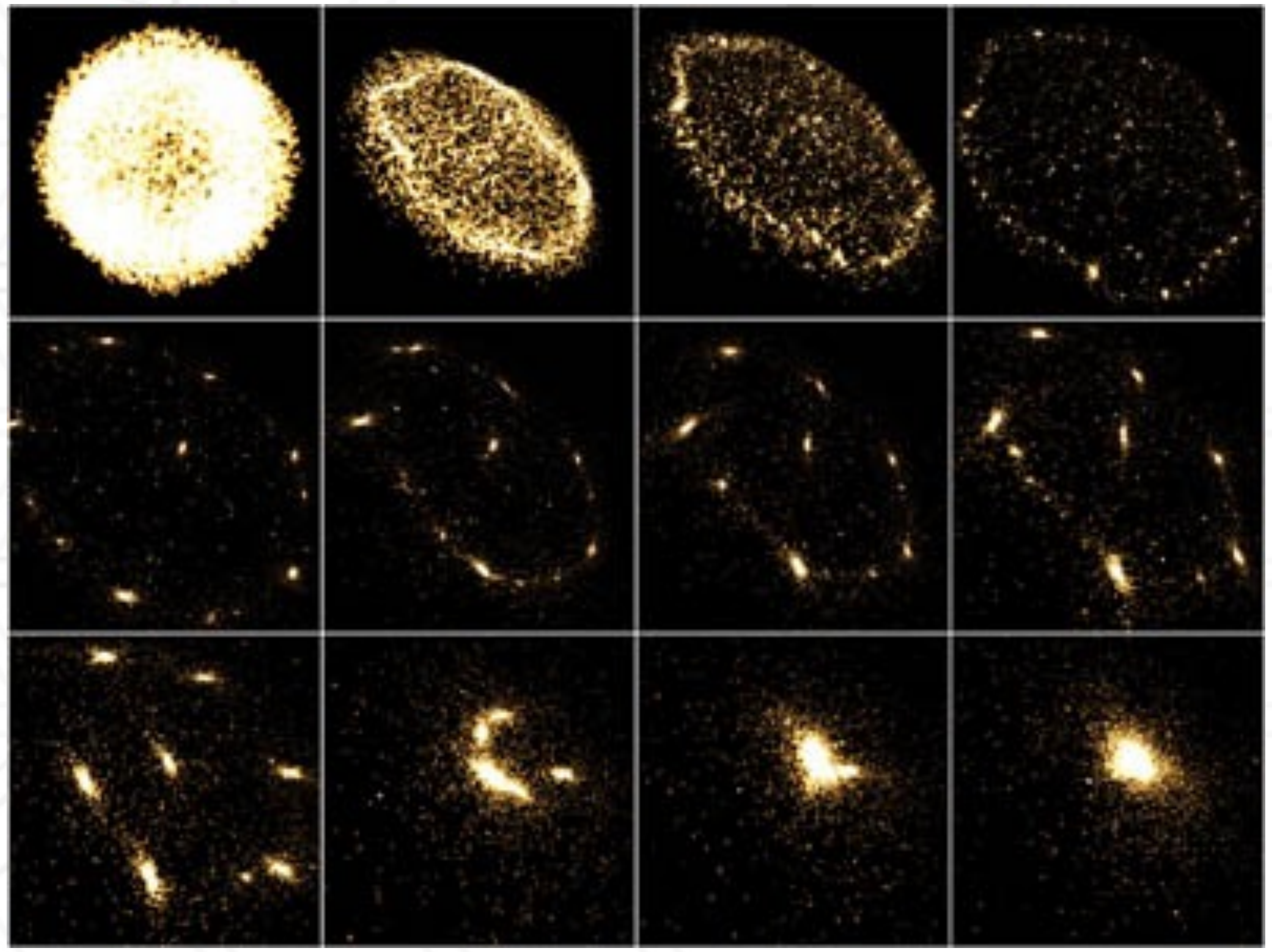
https://events.prace-ri.eu/event/176/contributions/38/attachments/154/305/HPC_libraries.pdf

# Domain-specific languages/frameworks

- Structured grids: SAMRAI, Chombo, AMREx, Overture
- Unstructured grids: MFEM, Quinoa

# The *n*-body problem

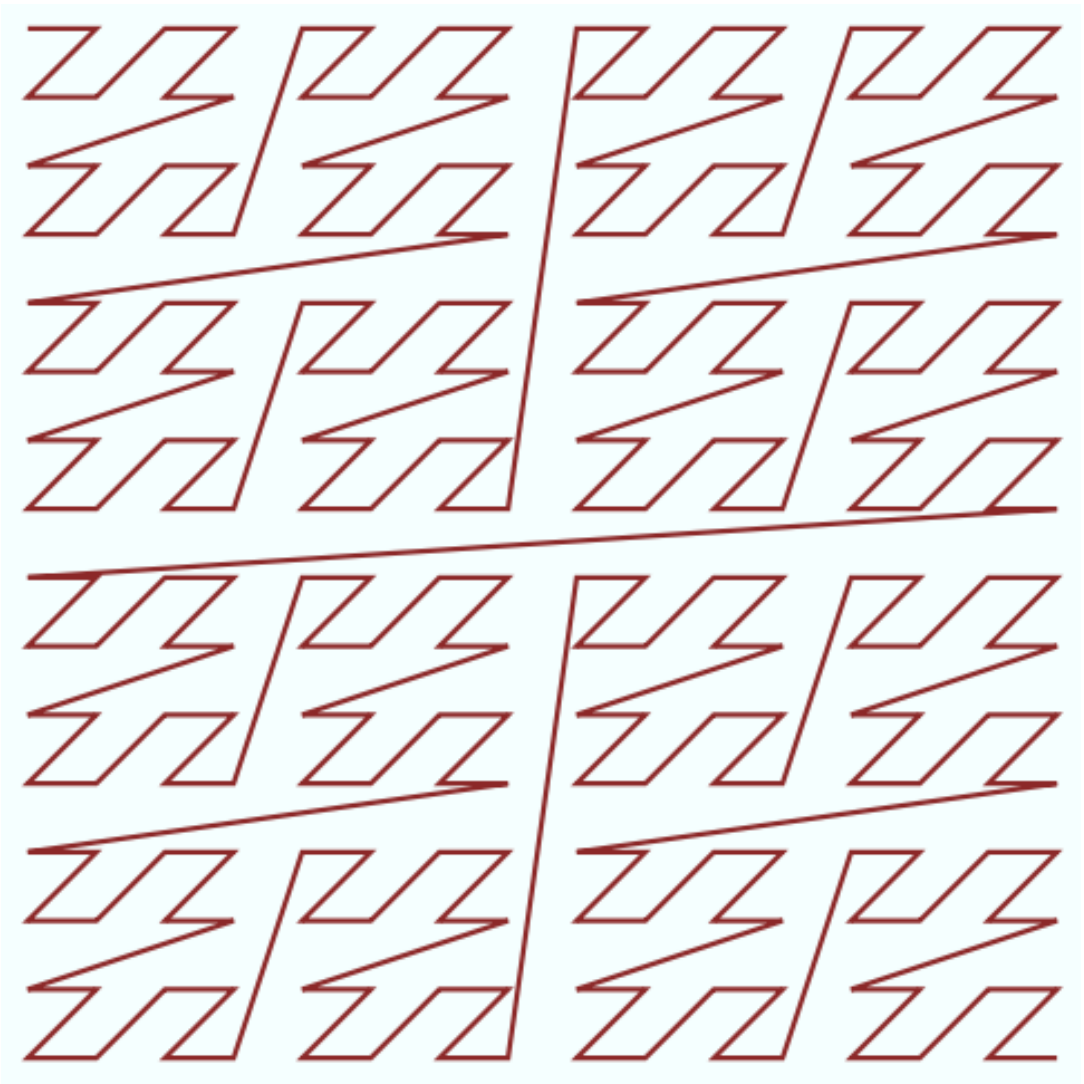
- Simulate the motion of celestial objects interacting with one another due to gravitational forces
- Naive algorithm: $O(n^2)$
  - Every body calculates forces pair-wise with every other body (particle)



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?

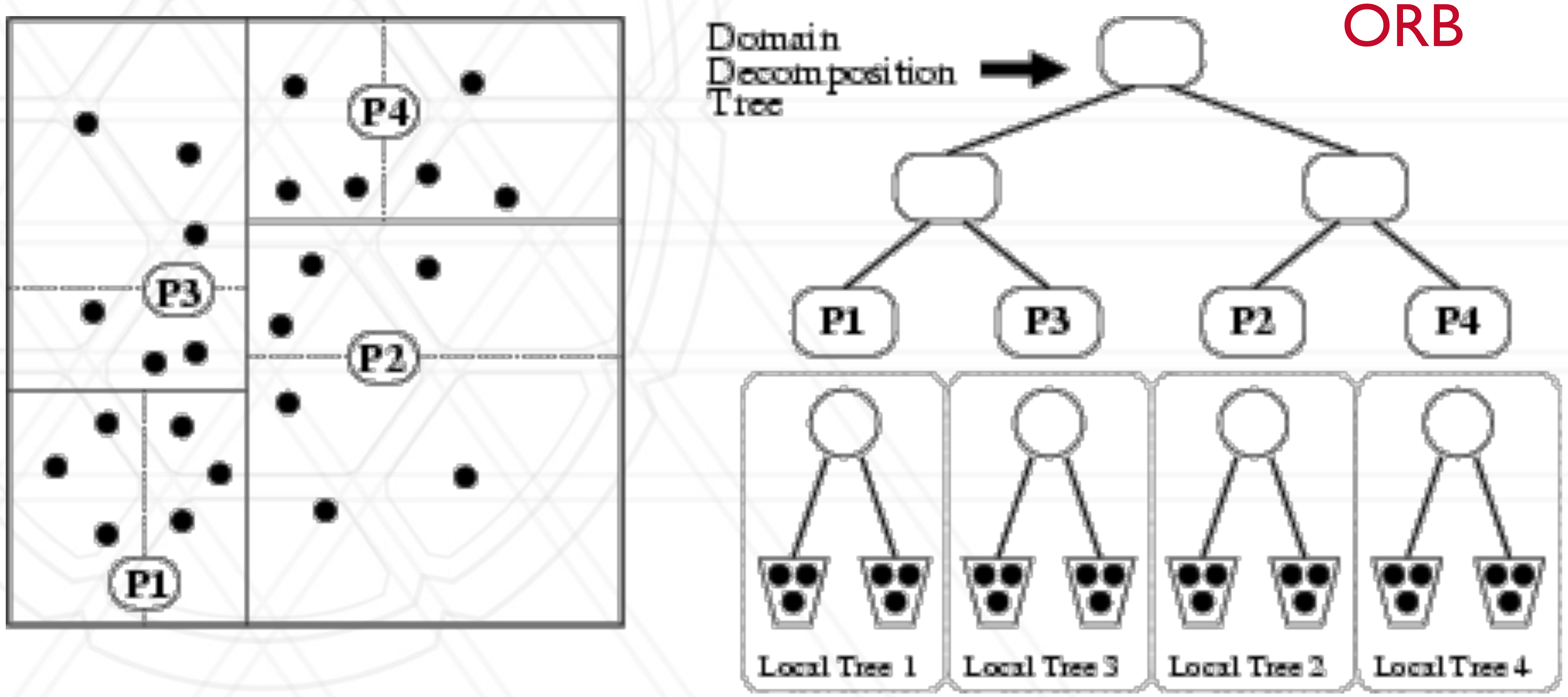


Space-filling curves

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

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

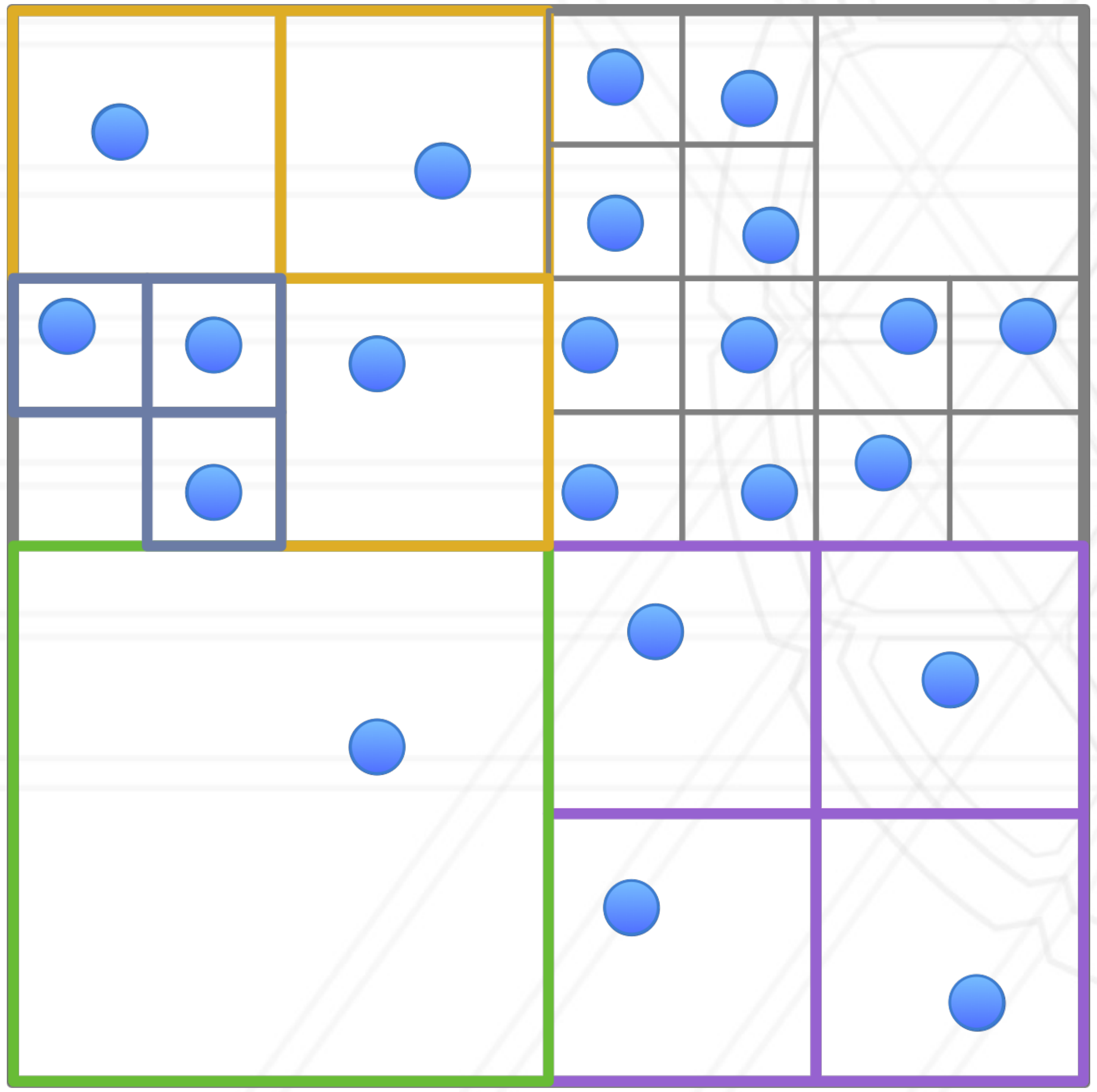
ORB

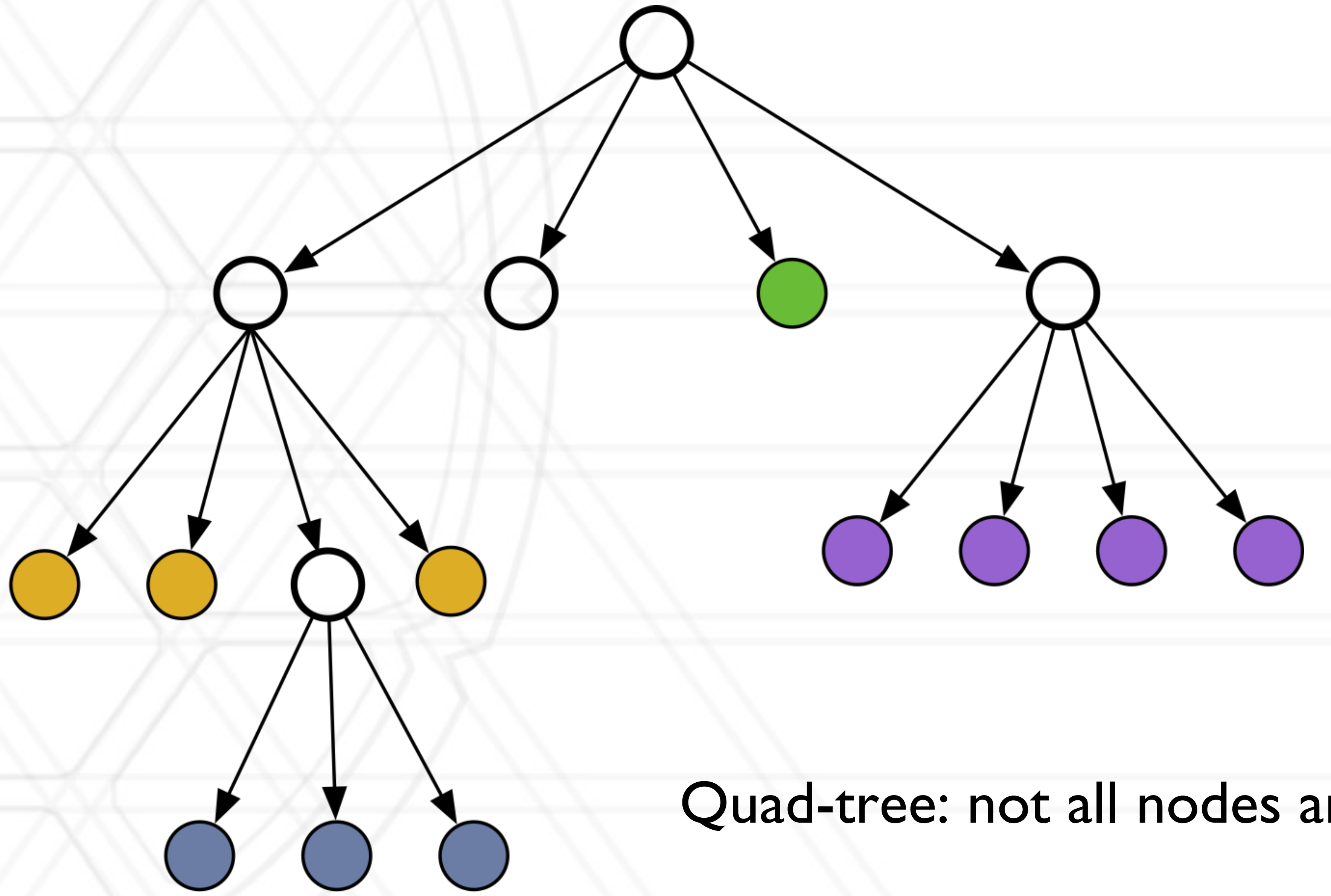



http://charm.cs.uiuc.edu/workshops/charmWorkshop2011/slides/CharmWorkshop2011_apps_ChaNGa.pdf

# Data distribution in *n*-body problems

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





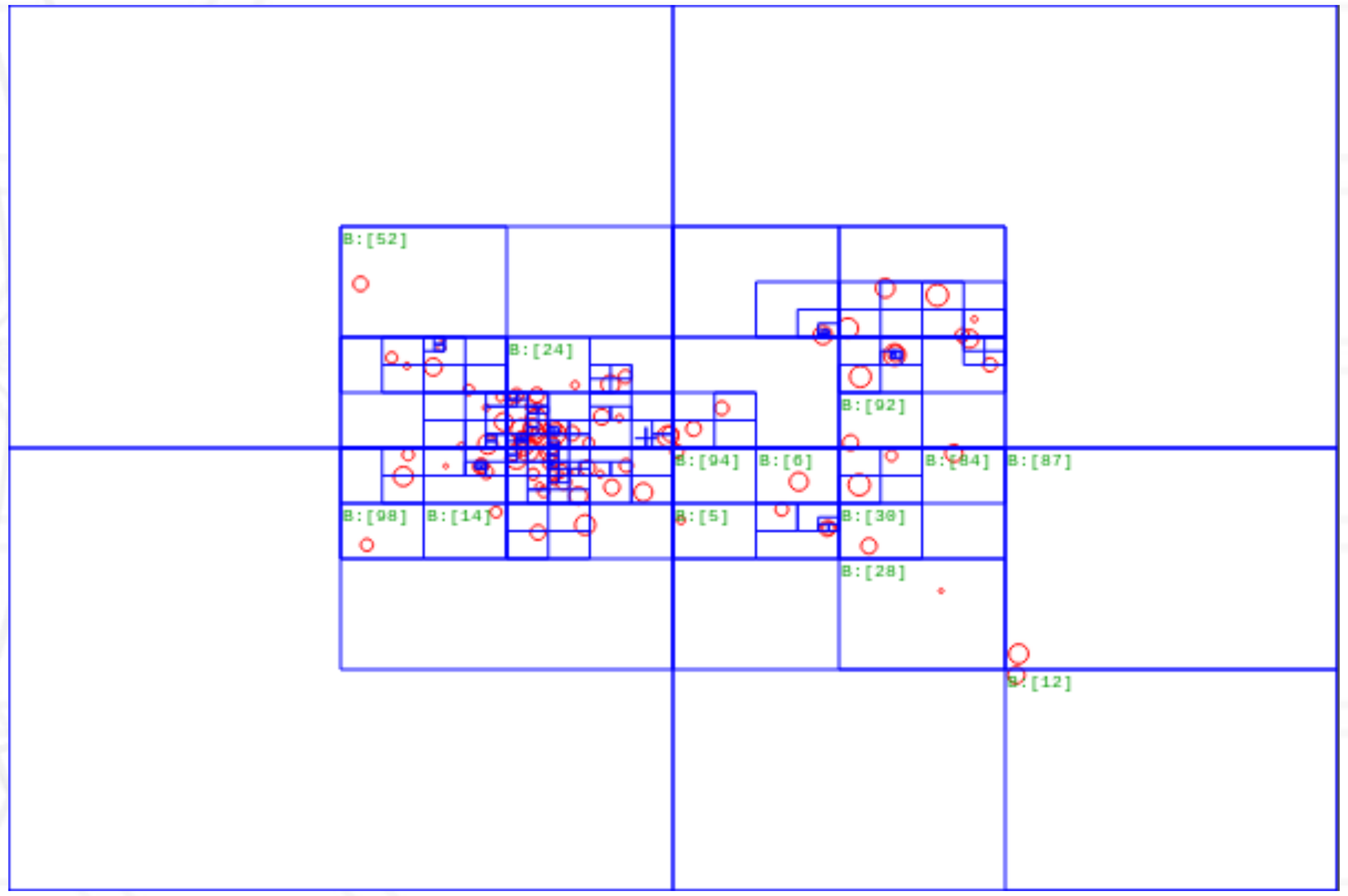
Quad-tree: not all nodes are shown

# Different parallelization methods

- Tree codes: Barnes-Hut simulations
- Fast multipole methods (FMM): Greengard and Rokhlin
- Particle mesh methods
- Particle-particle particle-mesh ($P^3M$) methods

# Barnes-Hut simulation

- Represent the space containing the particles as an oct-tree
- Pairwise force calculations for nearby particles
- For tree nodes that are sufficiently far away, approximate the particles in the node by a single large particle at the center of mass
- An O(*N logN*) algorithm



https://en.wikipedia.org/wiki/Barnes–Hut_simulation

# Fast multipole methods

- Use multipole expansion for distant particles
- Takes advantage of the fact that for nearby particles, multipole-expanded forces from distant particles are similar
- Reduces the time complexity further to O(*N*)

# Particle-particle particle-mesh methods

- Explicit calculation of forces on nearby particles
- Fourier-based Ewald summation for calculating potentials on a grid
- Smoothed particle hydrodynamics

UNIVERSITY OF
MARYLAND