HIGH PERFORMANCE COMPUTING for SCIENCE & ENGINEERING (HPCSE) I

HS 2021

TUTORIAL 06 : Cell-lists and Visualization

Man Hin CHENG
(Based on materials from Petr Karnakov and Lucas Amoudruz)

Computational **S**cience and **E**ngineering **Lab**
ETH Zürich

03.12.2021

Outline

- N-Body problem
  - Lennard-Jones potential
  - Cell-lists
  - One possible implementation

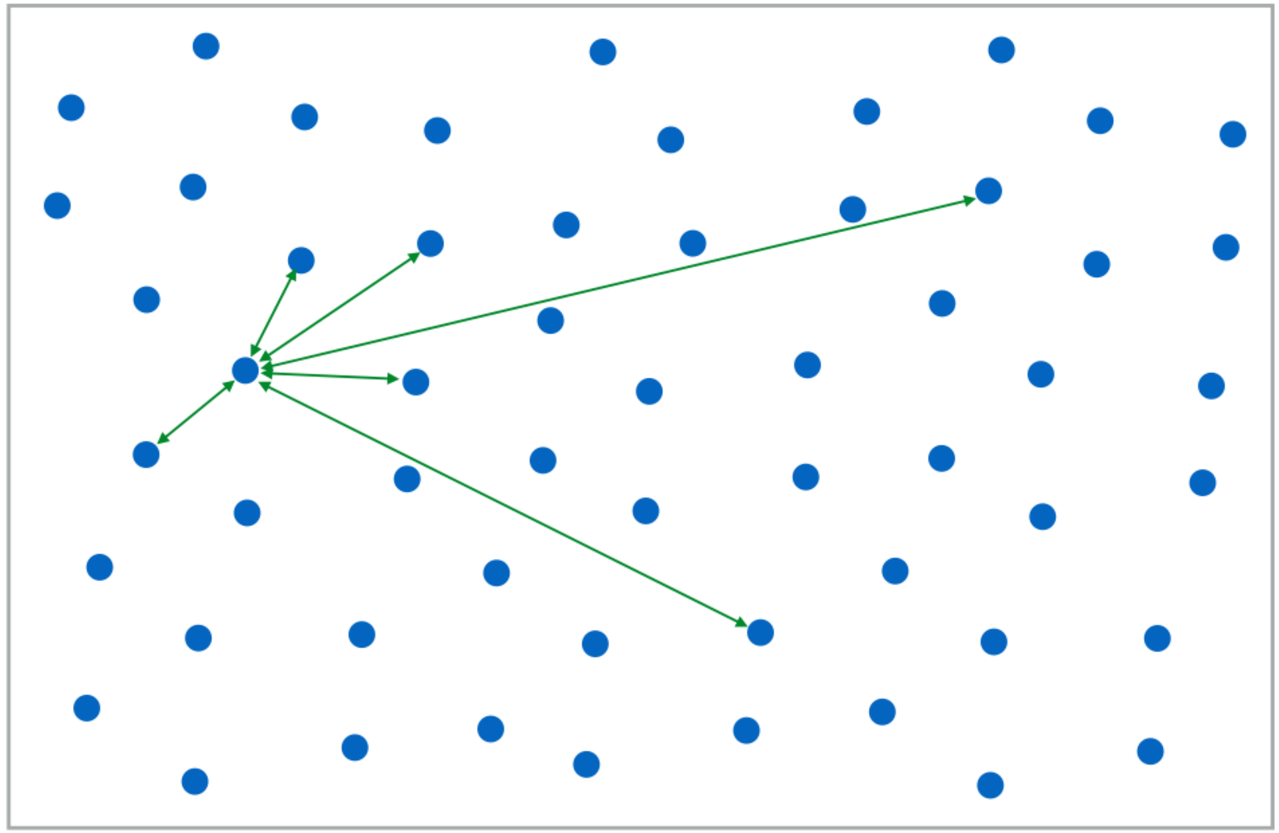
- Visualization using ParaView

N-Body problem

- Consider N particles interacting with each other

Examples

- Gravitation
- Electrodynamics
- Particle Strength Exchange
- Molecular Dynamics: Lennard-Jones
- Smooth Particle Hydrodynamics
- Dissipative Particle Dynamics



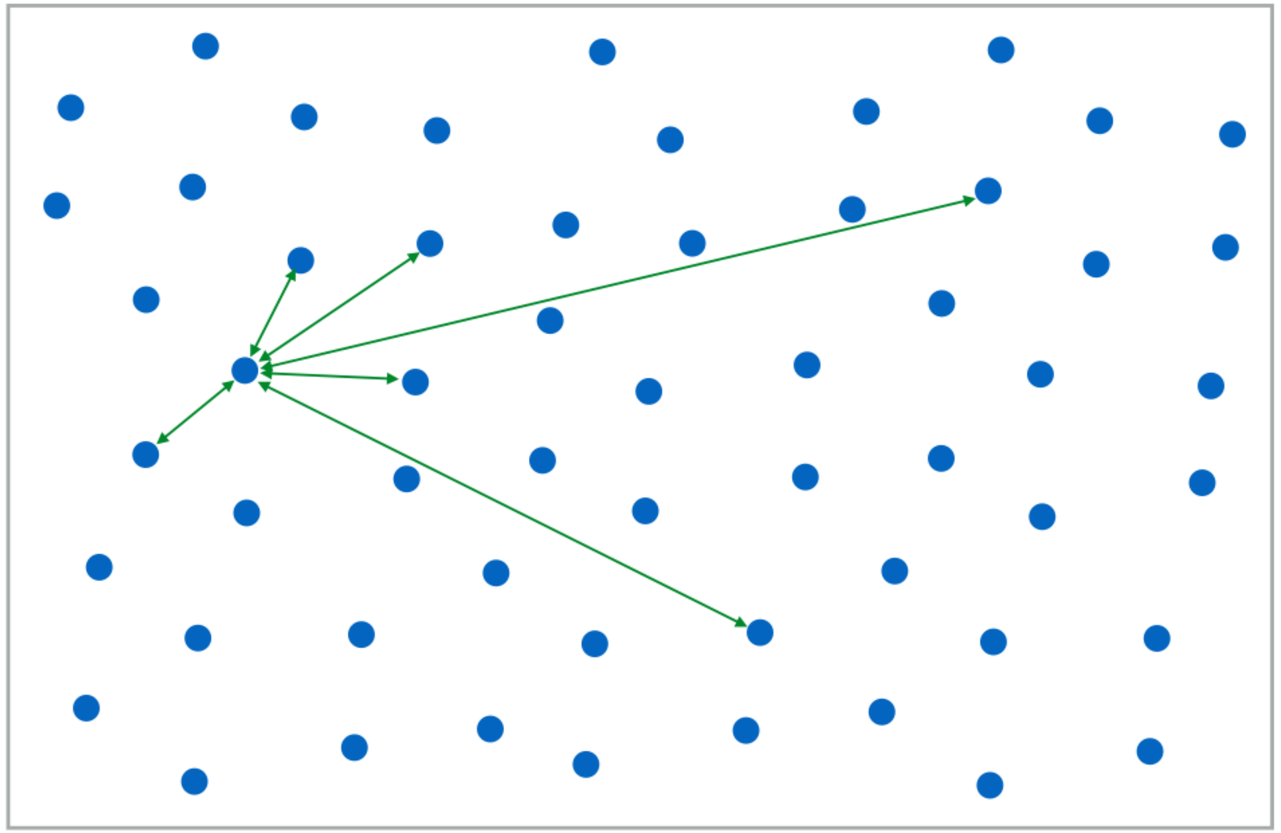
N-Body problem: complexity

- Consider N particles interacting with each other

- <u>Complexity:</u>

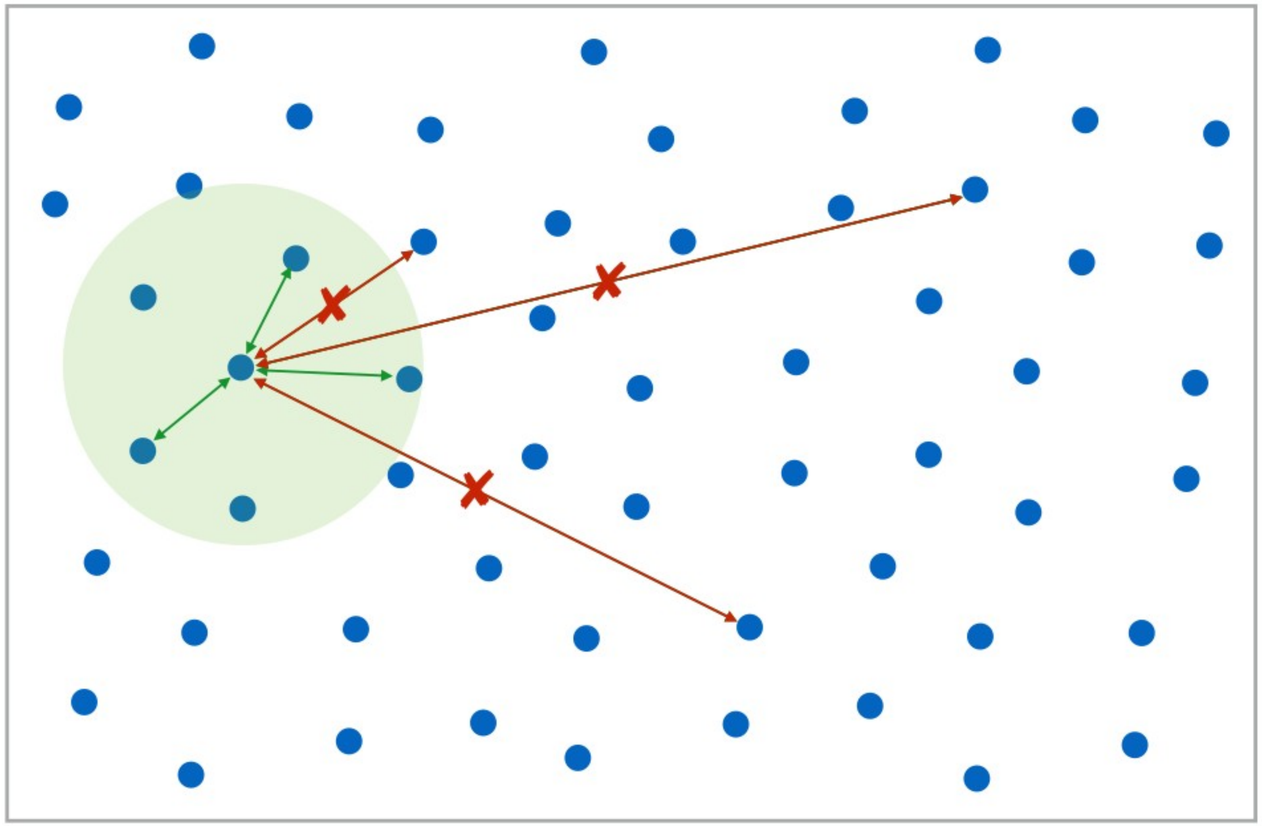
  In principle, each particle interacts with all other particles

  → $O(N^2)$



N-Body problem: complexity

- Consider N particles interacting with each other



- <u>Complexity:</u>

  In principle, each particle interacts with all other particles

  → $O(N^2)$

- <u>Consider short range interactions:</u>

  We can discard many interactions

  → $O(Nd)$

  with d be the number of neighbours per particle

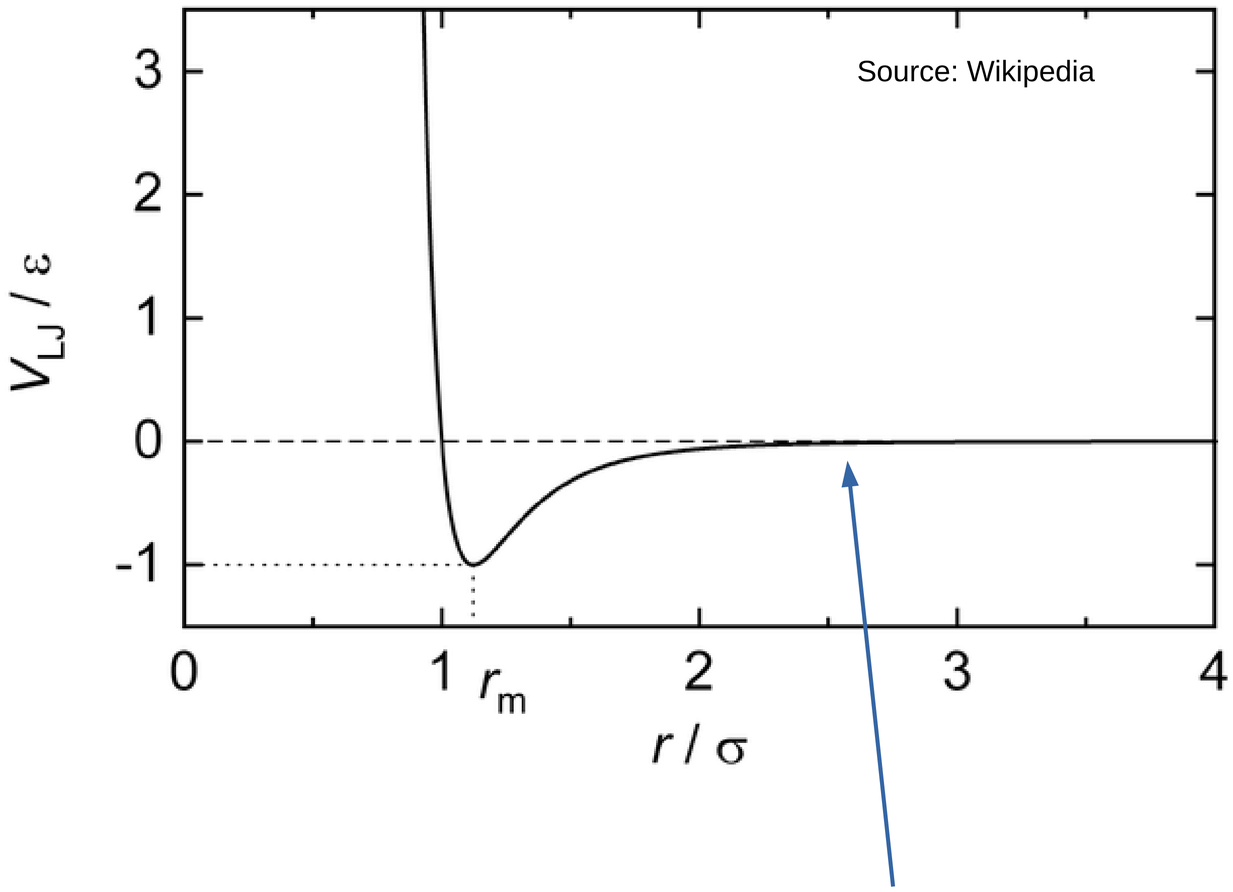
Short range interaction: L-J potential

- The Lennard-Jones potential describes simplified interaction between neutral atoms/ molecules:

$$V(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$$

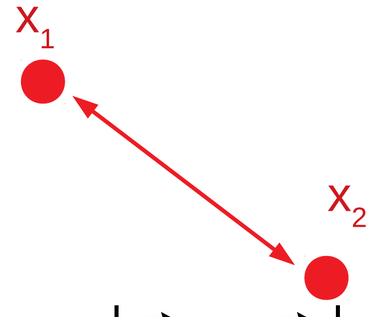
$$\vec{F}(\vec{r}) = -\nabla_{\vec{r}} V(r)$$

$$\vec{F}(\vec{r}) = 24\epsilon\frac{\vec{r}}{r^2}\left(2\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$$



$$\frac{d\vec{x}_i}{dt} = \vec{v}_i$$

$$\frac{d\vec{v}_i}{dt} = \frac{1}{m}\vec{F}_i = \frac{1}{m}\sum_{i\neq j}\vec{F}(\vec{r}_{ij})$$

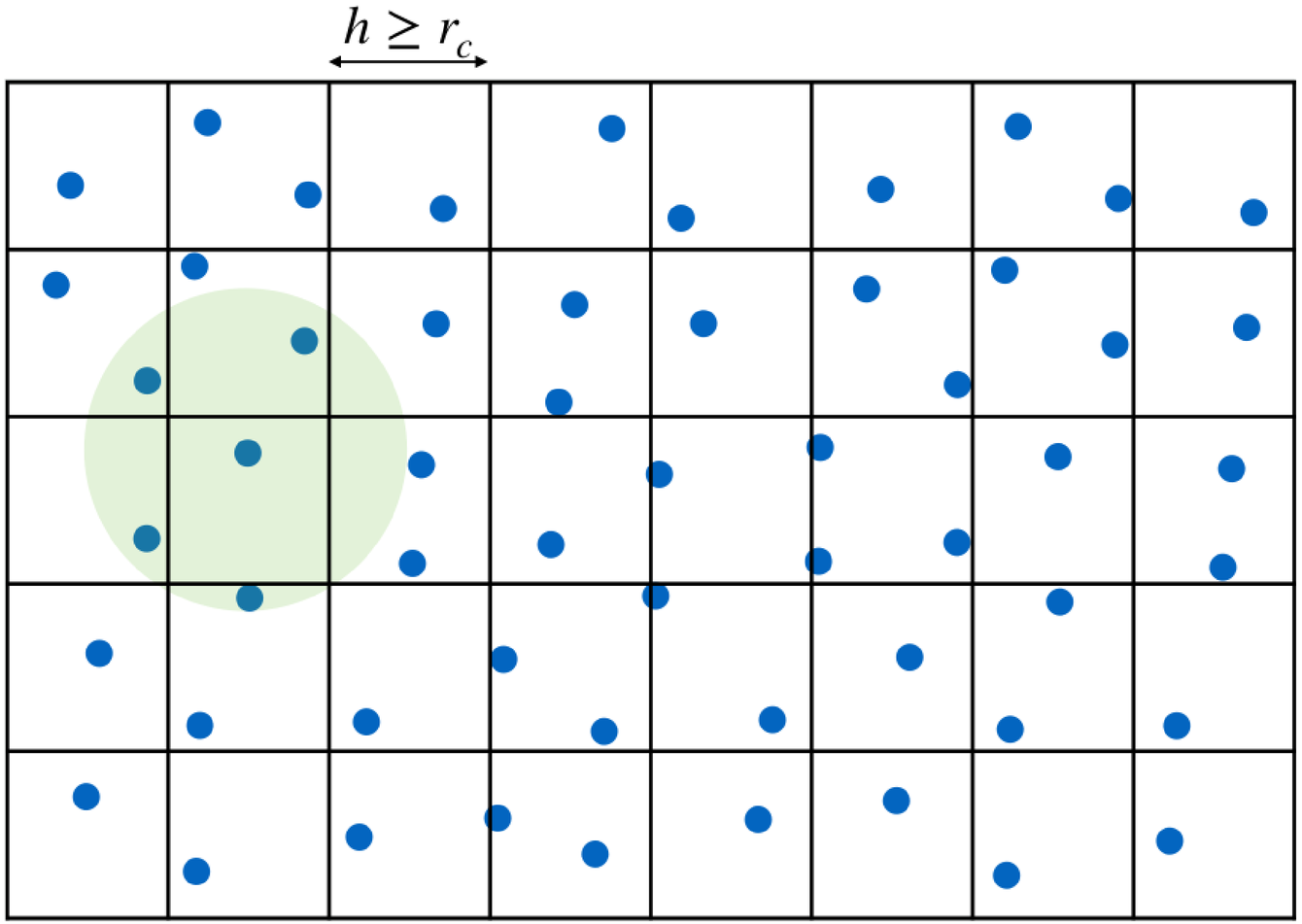


$$r = |\vec{x}_1 - \vec{x}_2|$$

Potential and forces almost zero at $r_c = 2.5\sigma$

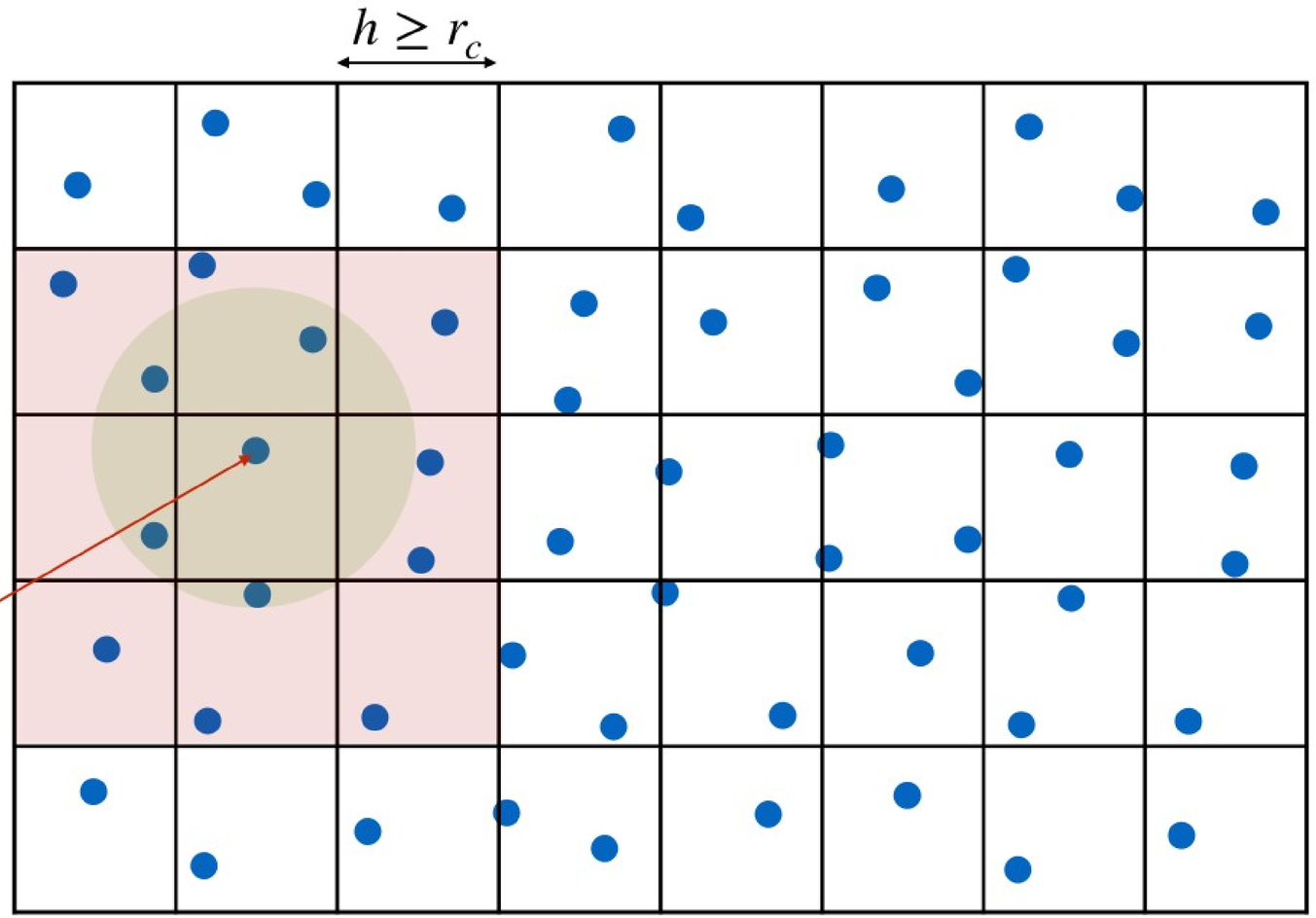
N-body problem with short range: cell-lists

- Consider N particles interacting with each other within cutoff radius
- How can I find my neighbours?
- We need a map from space to particles: the cell-lists



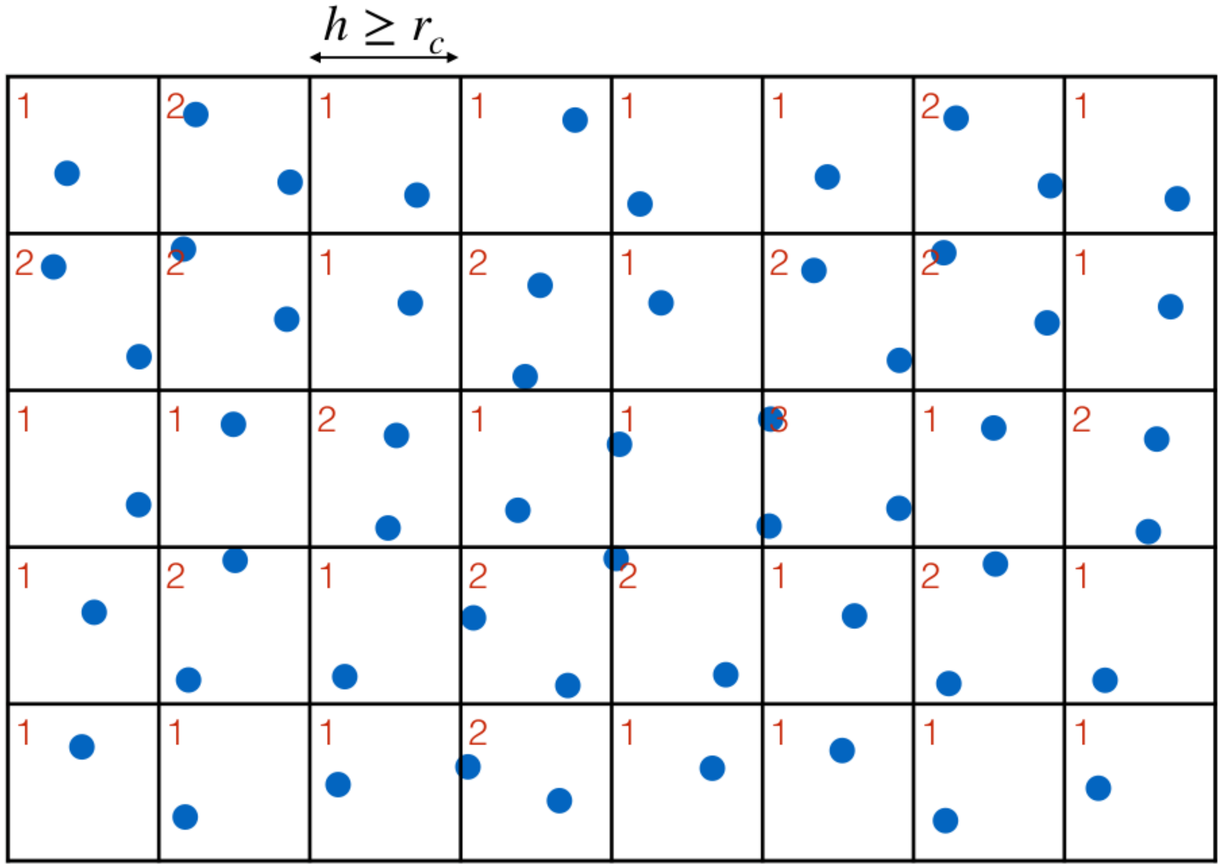
N-body problem with short range: cell-lists

- Consider N particles interacting with each other within cutoff radius
- How can I find my neighbours?
- We need a map from space to particles: the cell-lists
- Neighbours particles are in the 3 by 3 cells around myself



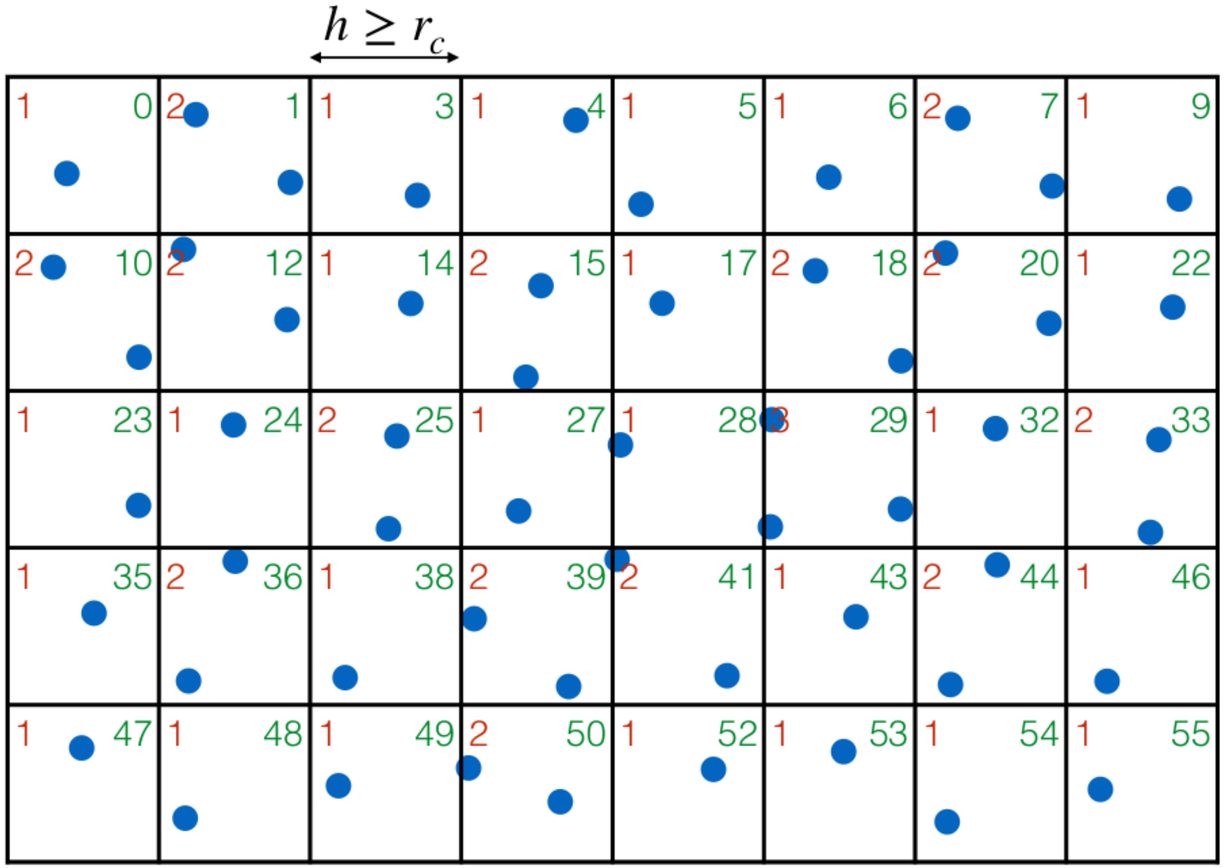
Building the cell-lists

1. Count number of particles in each cell



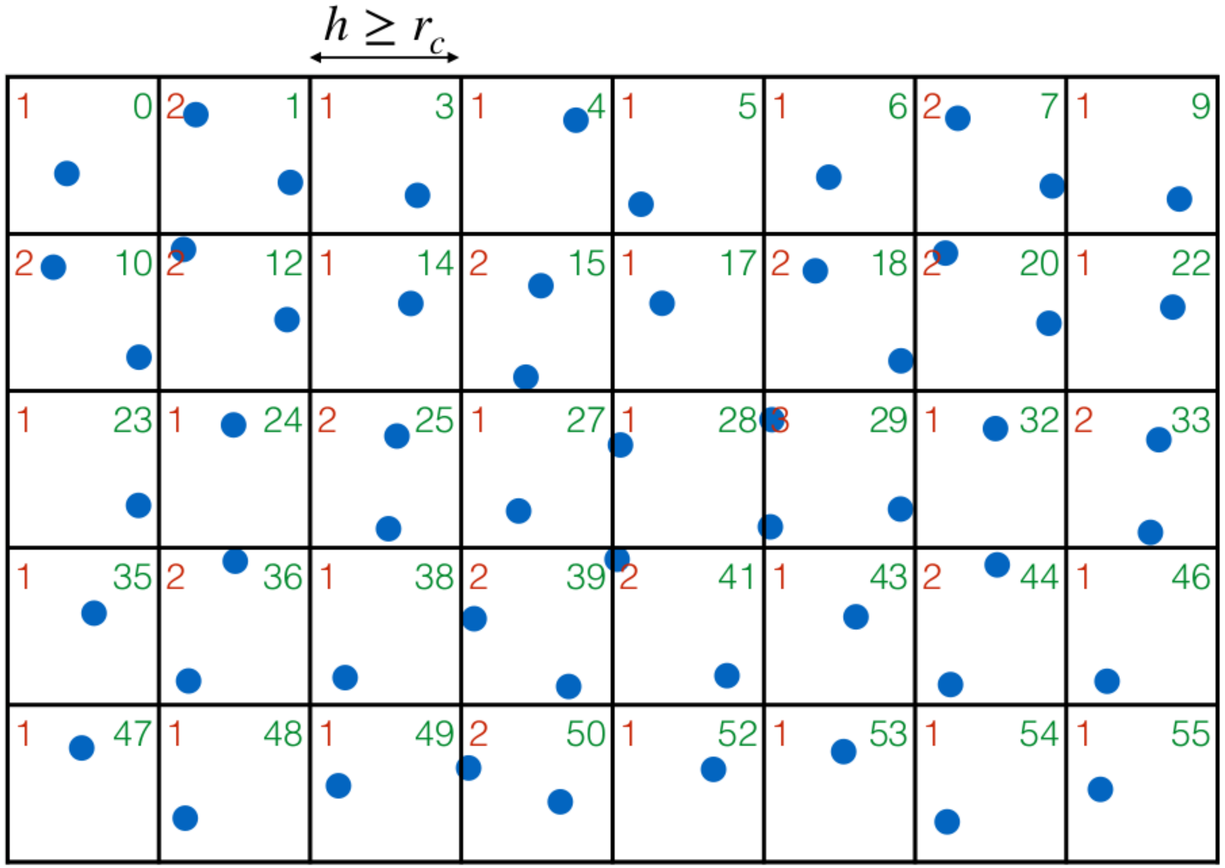
Building the cell-lists

1. Count number of particles in each cell
2. Compute the starts of each cells (prefix sum)



Building the cell-lists

1. Count number of particles in each cell
2. Compute the starts of each cells (prefix sum)
3. Reorder the particles in memory



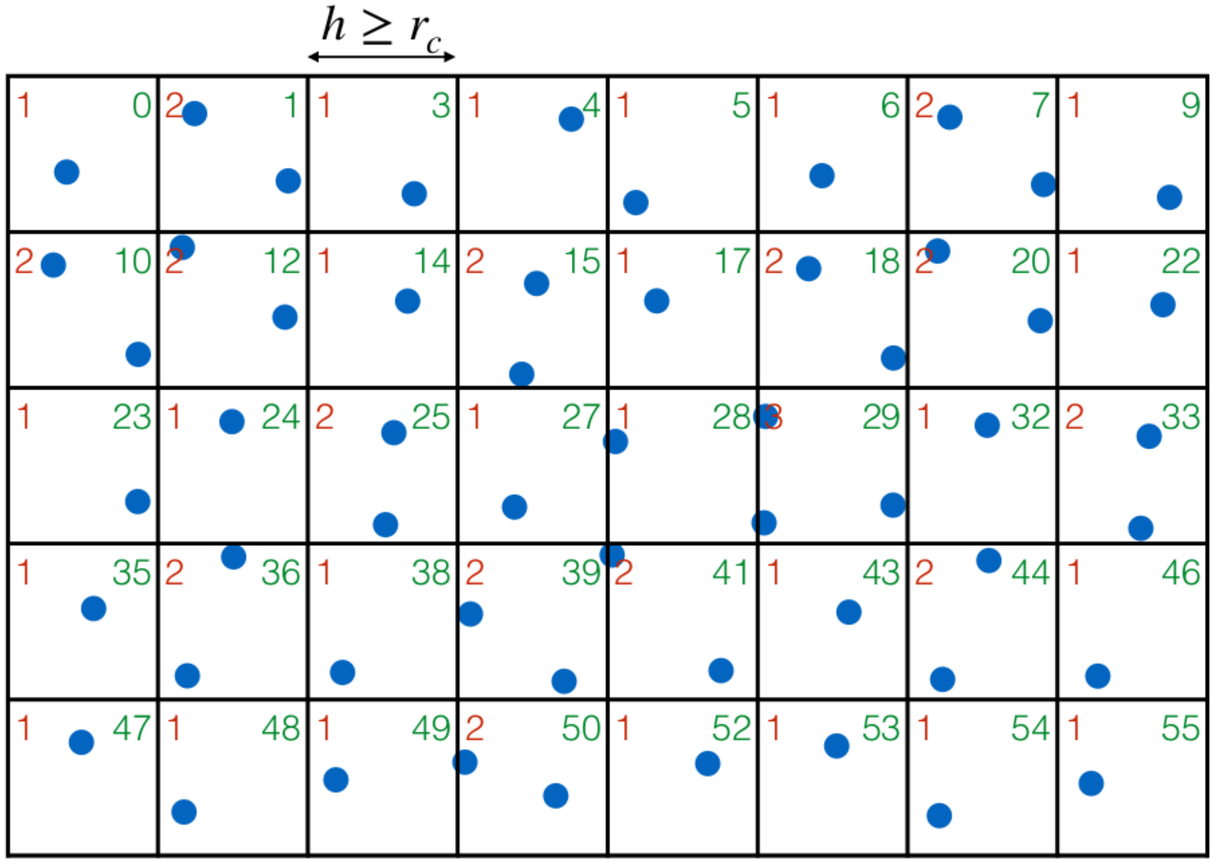
The cell lists are just 2 arrays: starts and counts

Particles in cell with id cid have indices between starts[cid] and starts[cid] + counts[cid]

Coding time

- Fill the TODOS in code/cell_list/skeleton/celllists.cpp

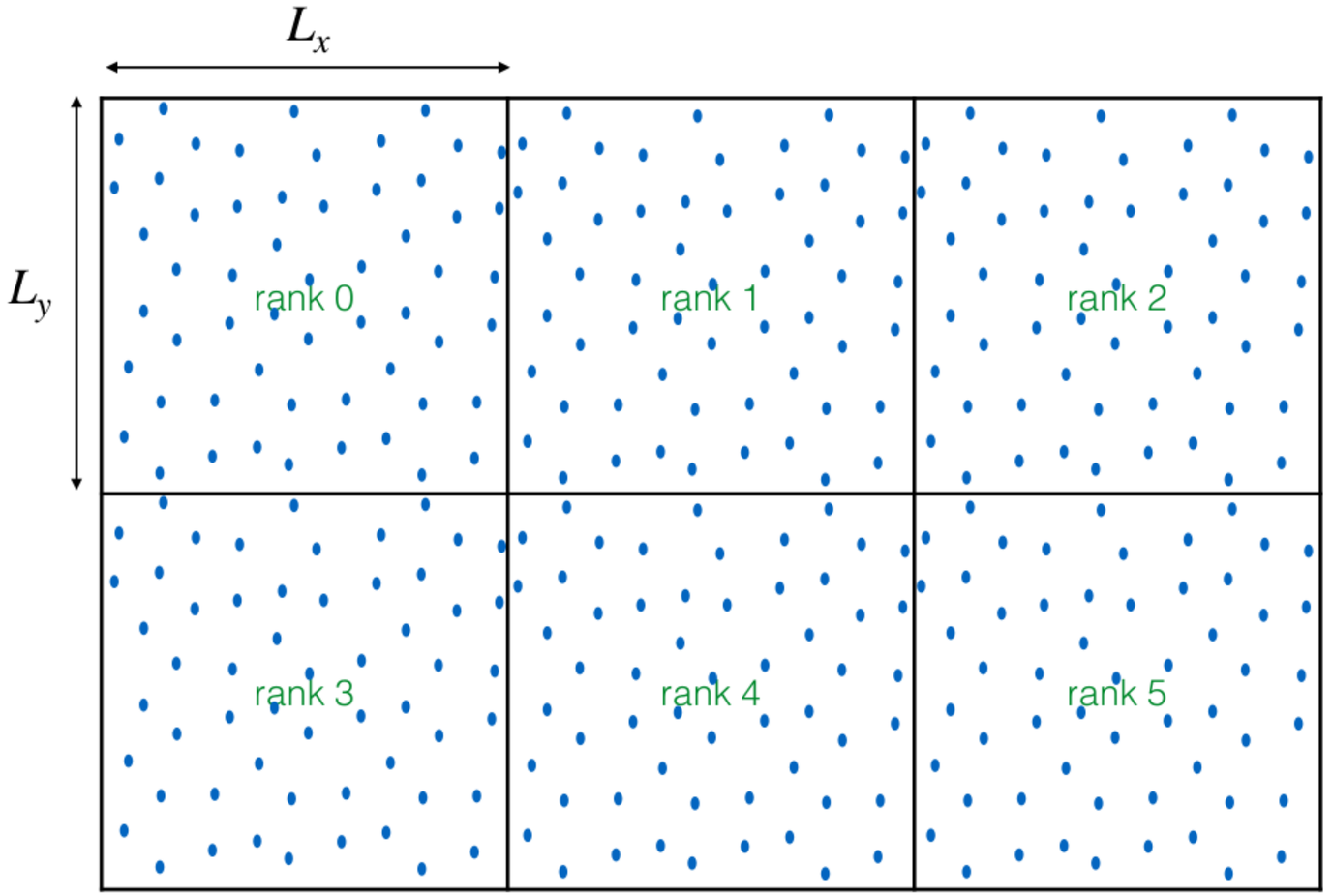
1. Count number of particles in each cell
2. Compute the starts of each cells (prefix sum)
3. Reorder the particles in memory
   - By recounting the particles again and looking at the result of counts



A simple 2D LJ simulation with MPI

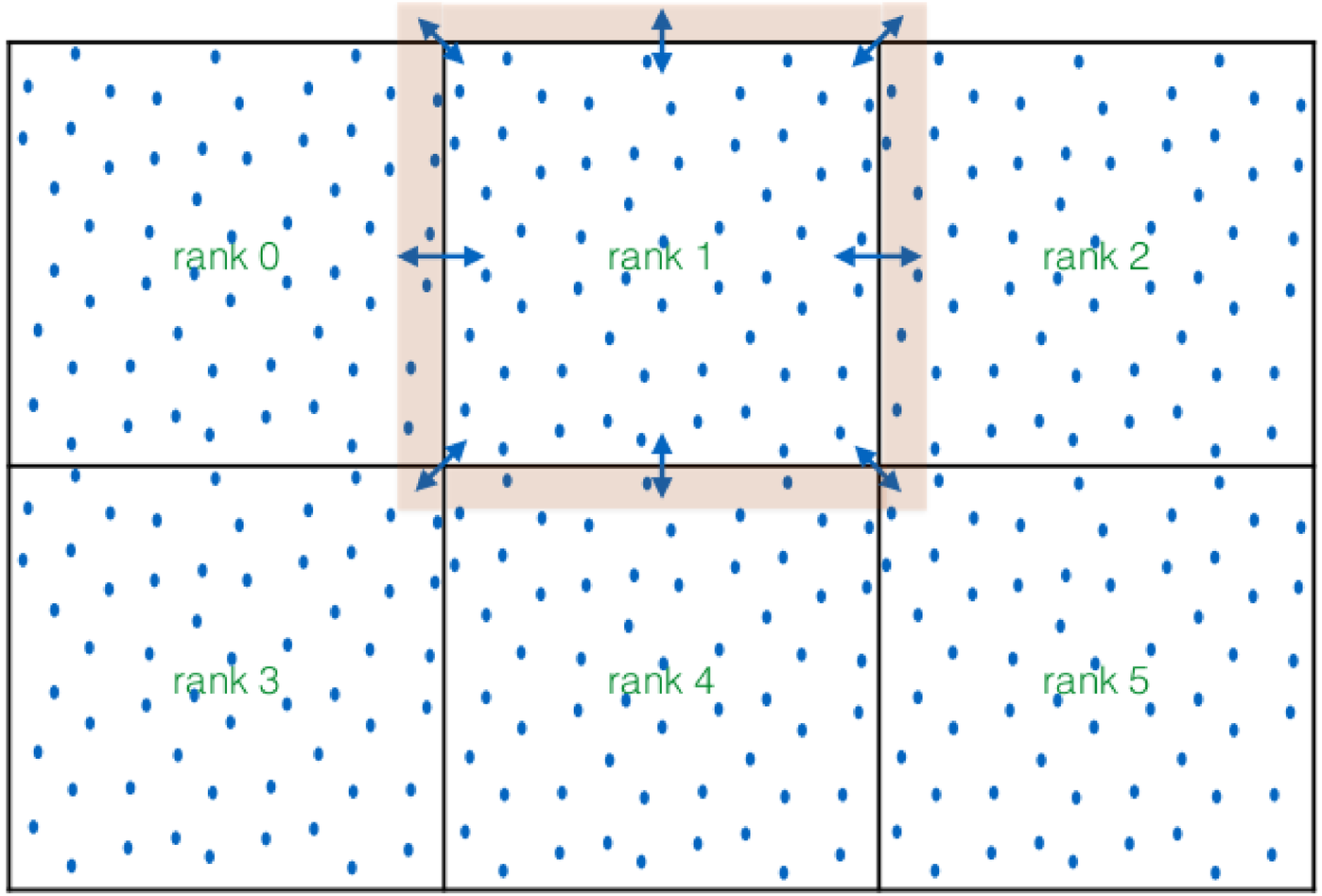
- Periodic position
- Exchange with 8 neighbours
- Each sub-domain has a local frame of reference

$$\left[\frac{-L_x}{2}, \frac{L_x}{2}\right] \times \left[\frac{-L_y}{2}, \frac{L_y}{2}\right]$$



A simple 2D LJ simulation with MPI

- One time step looks like:
  1. redistribute the particles among ranks
  2. build the cell lists
  3. exchange ghost particles
  4. compute interactions
  5. update velocities and positions

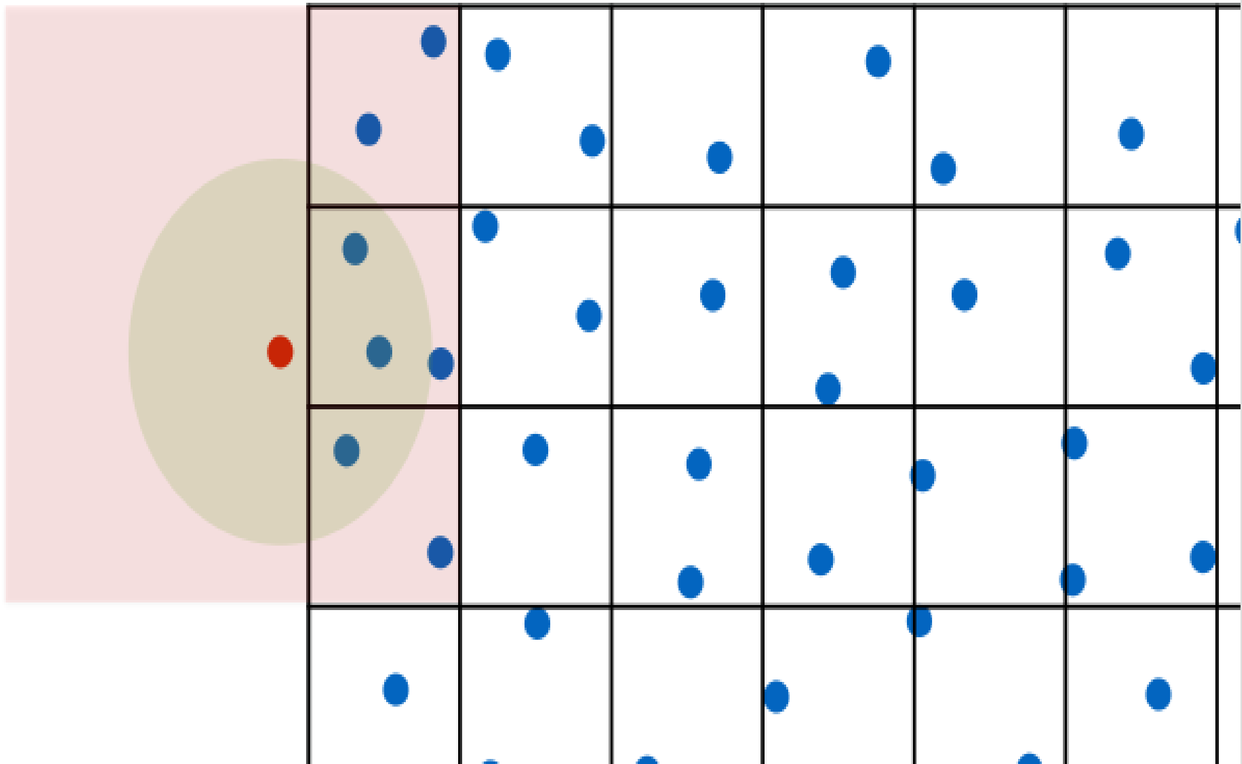


Coding time

- Fill the TODOS in code/cell_list/skeleton/interactions.h

  (NO periodic boundary condition)

1. loop over ghost particles (srcParticles)
2. find cell id
3. loop over (existing) neighbour cells
4. loop over each particles in those cells (dstParticles)
5. compute interaction and add force to those particles (dstForces)



Visualization

- ParaView
  - https://www.paraview.org/download/
  - tar -zxvf ParaView-xxxx.tar.gz

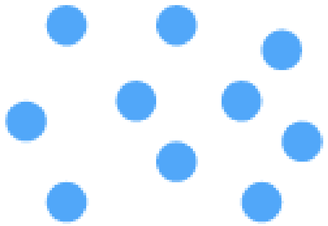
- Source code from git repository
  - https://gitlab.ethz.ch/hpcseI_hs21/lecture/-/tree/master/tutorials/tut06-celllist_visualization/code/visualization
  - Run locally (preferred), requirements: MPI, HDF5 parallel
  - Otherwise on Euler with:

```
module load gcc/8.2.0 openmpi/4.0.2 hdf5
```

- Data download from course website
  - https://www.cse-lab.ethz.ch/wp-content/uploads/2021/12/tut06_code.zip

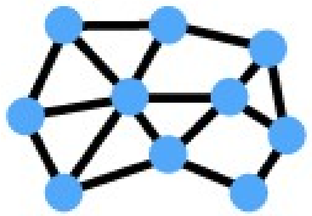
Data from simulations

- Particles



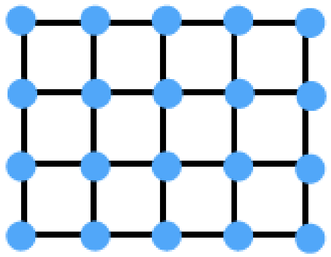
  - Points, fields
  - 1D arrays

- Unstructured grid



  - Points, connectivity, fields
  - 1D arrays

- Structured grid



  - (points), fields
  - 2D/ 3D arrays

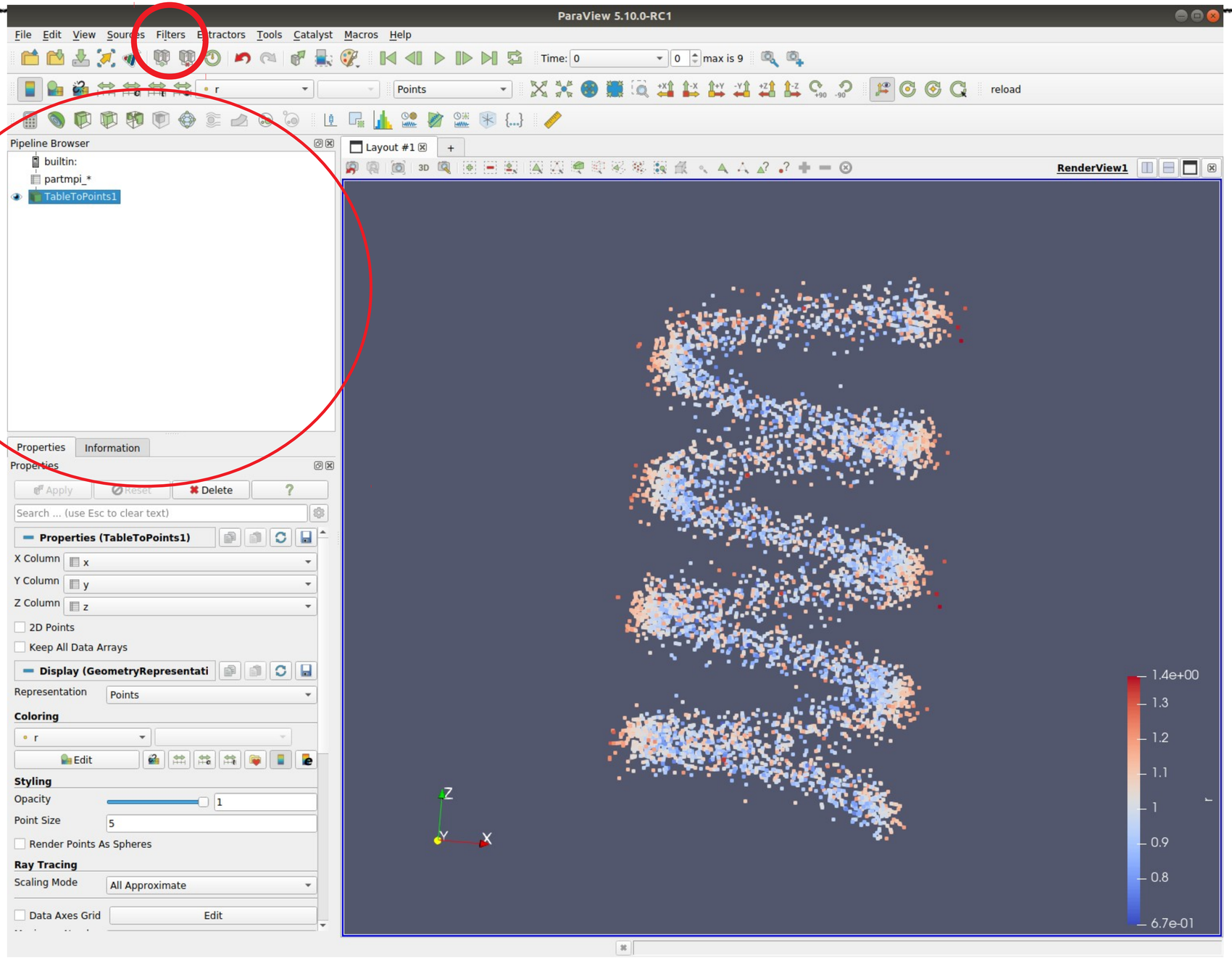
ParaView

- An open-source, scalable, multi-platform visualization application.
- Support for distributed computation modules to process large data sets.
- An open, flexible, and intuitive user interface.
- An extensible, modular architecture based on open standard

ParaView

Filters: search

pipeline



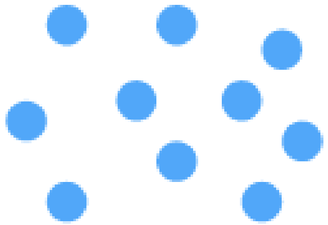
CSV particles data set

- Comma-separated values

```
x,y,z,r
-0.467,-0.0323,-0.22,0.517
-0.33,0.387,0.0173,0.509
-0.491,0.0225,-0.373,0.618
-0.483,0.143,0.33,0.602
-0.368,0.133,0.174,0.428
-0.309,-0.409,-0.155,0.536
```

- Filters:
  - TableToPoints
  - Glyph
  - Calculator

- Particles
  - Positions, fields
  - 1D aray



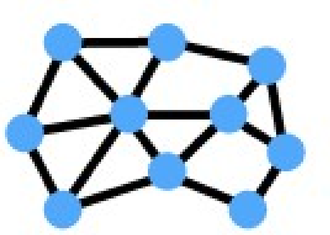
- Save image, Save animation

  ffmpeg -i movie.%04d.png movie.mp4

VTK with unstructured grid

- **V**isualization **T**ool**k**it
  - (https://vtk.org/wp-content/uploads/2021/08/VTKUsersGuide.pdf)

- Unstructured grid
  - Positions, connectivity
  - 1D array



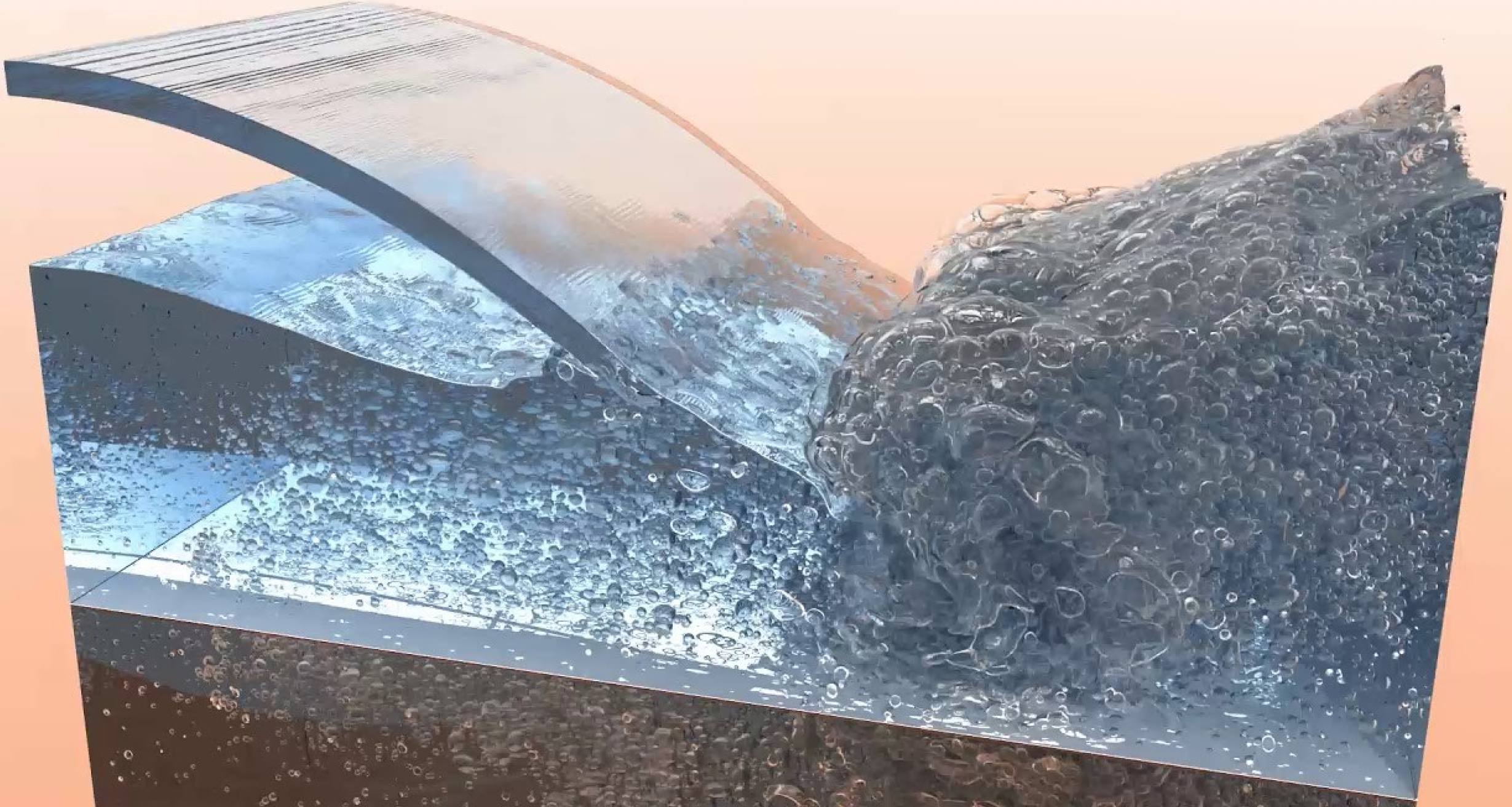
```
# vtk DataFile Version 2.0
comment
ASCII
DATASET POLYDATA
POINTS 3328 float
1 0 0
0.988771 0.149438 0.006
0.999859 0.0168139 0.252
 ...
POLYGONS 832 4160
4 0 1 2 3
4 4 5 6 7
...
```

- Filters:
  - GenerateSurfaceNormals
  - Transform

- Source
  - Plane
  - Sphere

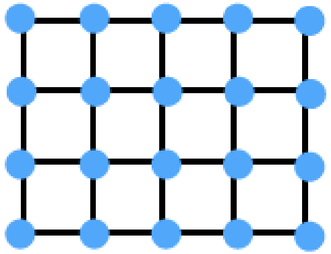
- Ray tracing
  - Add materials

https://www.youtube.com/watch?v=NYTKk5OgwbQ



HDF with uniform grid

- Hierarchical Data Format
  - (https://portal.hdfgroup.org/display/HDF5/HDF5)
  - The HDF5 library provides tools for manipulating the files:
    - C API
    - command-line utilities (h5ls, h5dump)
- Use XDMF file for metadata
  - (https://www.xdmf.org/index.php/XDMF_Model_and_Format)
- Structured grid
  - Field
  - 3D array



- Filters:
  - CellDataToPointData
  - Contour
  - Slice
  - Clip
- Save state
  - pvbatch state.py