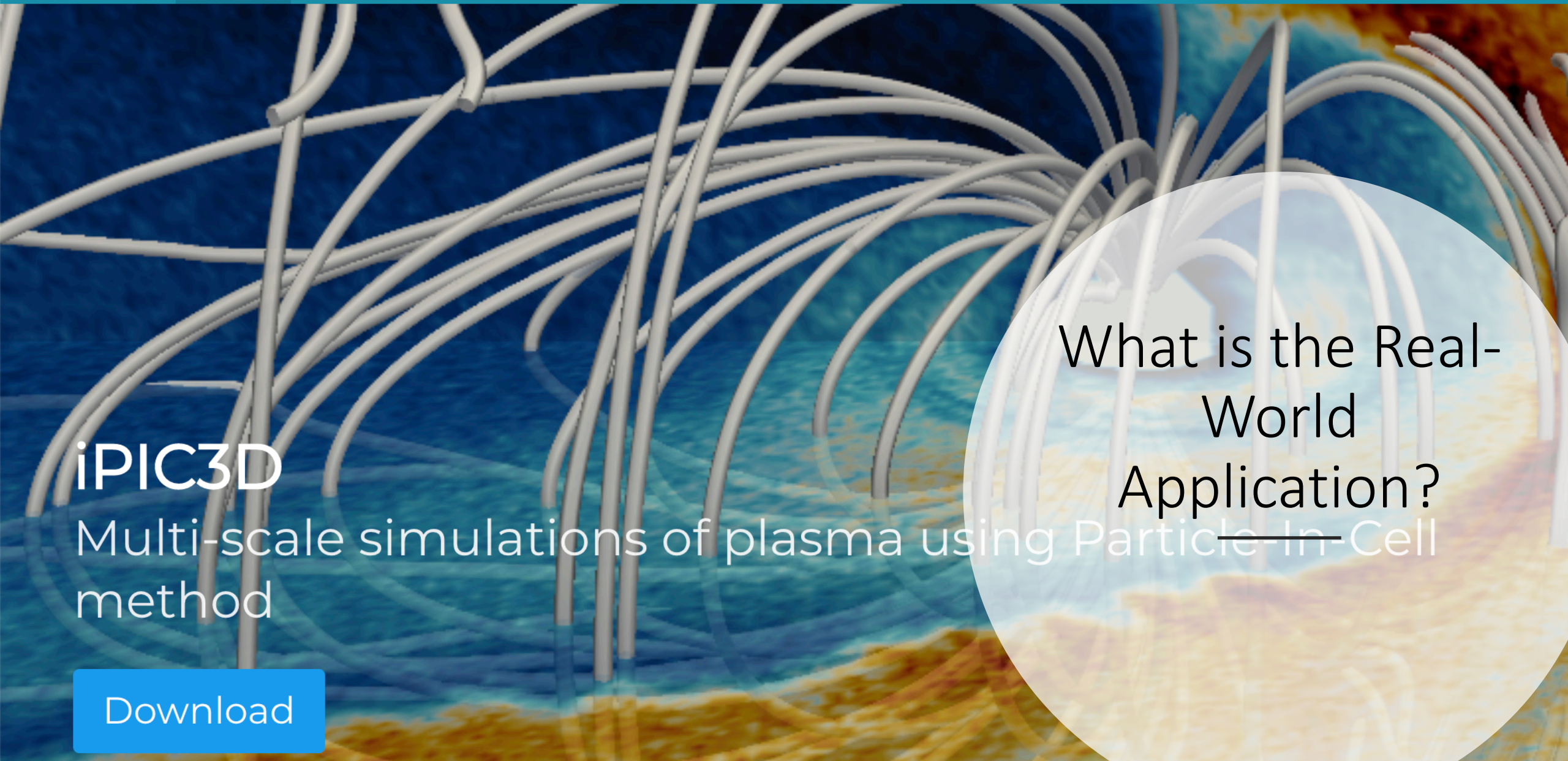
DD2360 Final Project

Stefano Markidis

Objective

- Take a real-world scientific code and
  - Port it to GPU with CUDA
  - Measure and analyze performance
    - Compare Performance with CPU and OpenACC (only A grade)
  - Optimize GPU code
    - Evaluate performance improvement


iPIC3D
Multi-scale simulations of plasma using Particle-In-Cell method
Download
What is the Real-World Application?

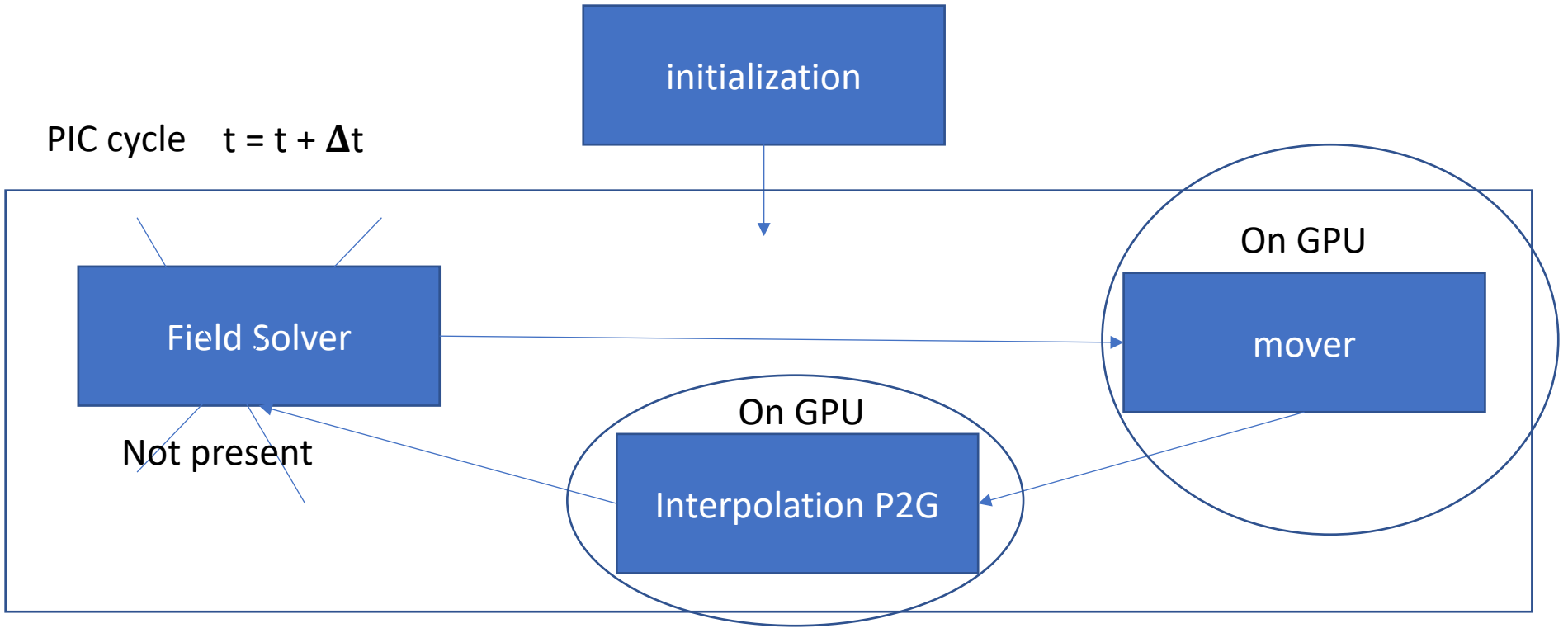
sputniPIC – A Simplified Serial C iPIC3D

https://github.com/iPIC3D/sputniPIC

We will port and optimize two functions of the sputniPIC code

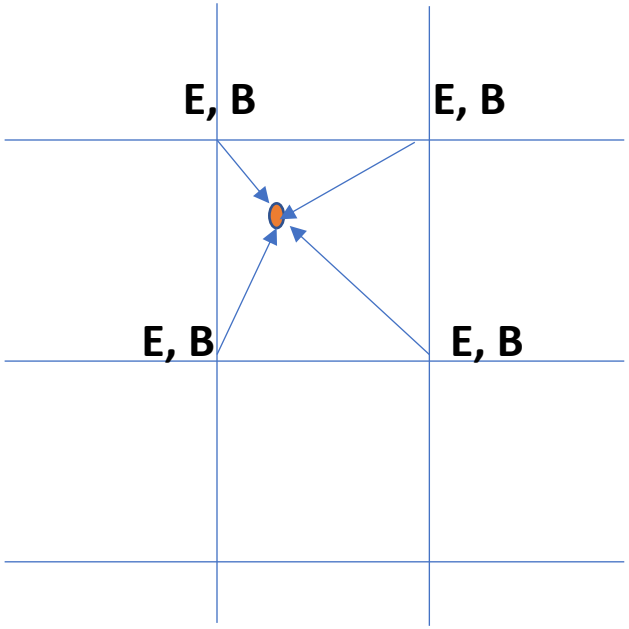
What is a Particle-in-Cell (PIC) Method?

- A numerical technique for simulating plasmas
  - Electron and protons are computational particles
  - At each simulation step, particles are moved solving an ODE for each particle
  - At each simulation step, particles deposit charge and weight on a grid
- In this project, we focus on two steps of the PIC method



Particle Mover

- Solve two ODEs for each particle
  - **v** = **v** + q/m (**E** + **v** x **B**)dt
  - **x** = **x** + **v** dt
- E and B are defined only grid points and not at the particle position
  - Need to interpolate (linearly) E and B at the particle position
- In Particle.cu, mover_PC
  - PC = use a predictor corrector scheme



How Many Particles?

- Number of Particles is defined in the input file
- Two input files.
  - GEM_2D.inp
  - GEM_3D.inp

```
13  # %%%%%%%%%%%%%%%%%% BOX SIZE %%%%%%%%%%%%%%%%%
14  Lx = 40   # Lx = simulation box length - x direction
15  Ly = 20   # Ly = simulation box length - y direction
16  nxc = 256  # nxc = number of cells - x direction
17  nyc = 128  # nyc = number of cells - y direction
18
19
20  # %%%%%%%%%%%%%%% PARTICLES %%%%%%%%%%%%%%%%%%
21  #    ns = number of species
22  ns = 4
23
24  # qom = charge to mass ratio for different species */
25  qom = -64.0 1.0 -64 1.0
26
27  # Initial density (make sure that plasma is neutral)
28  rhoINIT = 1.0 1.0 0.02 0.02
29
30  # TrackParticleID[species] = 1=true, 0=false --> Assign ID to particles
31  TrackParticleID= 0 0 0 0
32  # npcelx = number of particles per cell - Direction X
33  npcelx = 5 5 5 5
34  # npcely = number of particles per cell - Direction Y */
35  npcely = 5 5 5 5
36  # npcelz = number of particles per cell - Direction Z */
37  npcelz = 5 5 5 5
```

4 particles species/populations

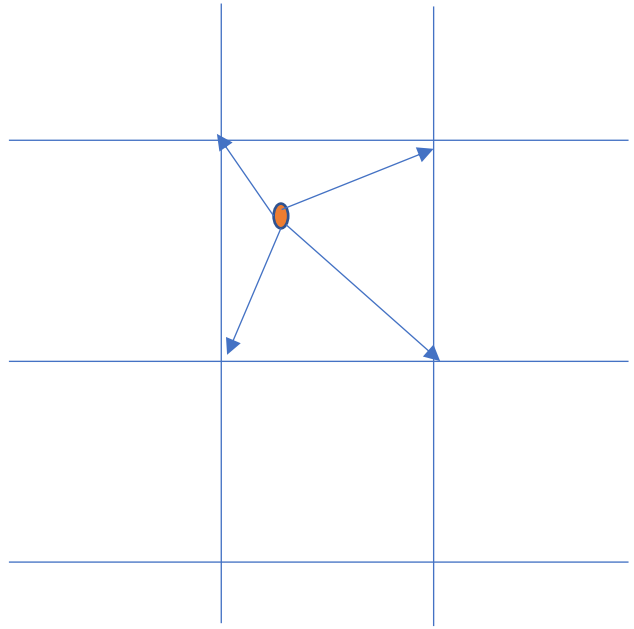
- = electron, + = proton

Each species has 5x5x5 particle per cell

Interpolation Particle to Grid

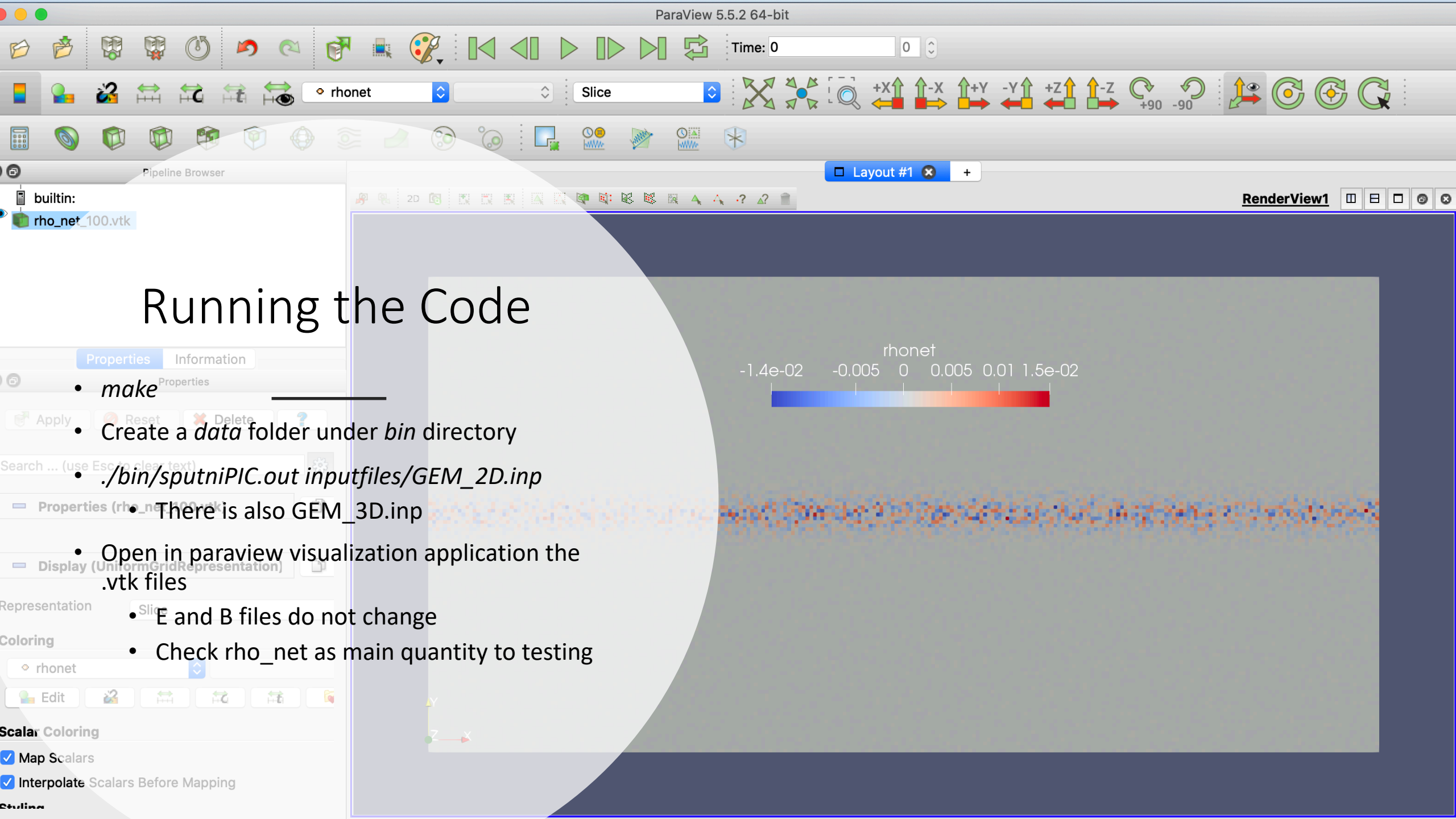
- At each iteration, 10 quantities defined on the grid nodes (rho, jx, jy, jz, pxx, pxy, pxz, pyy, pyz and pzz). are deposited from particles on the grid nodes
- In Particles.cu, interpP2G function
- This function need to use atomic operations
  - rho[ix][iy][iz] += …

atomic



Running the Code

- *make*
- Create a *data* folder under *bin* directory
- *./bin/sputniPIC.out inputfiles/GEM_2D.inp*
  - There is also GEM_3D.inp
- Open in paraview visualization application the .vtk files
  - E and B files do not change
  - Check rho_net as main quantity to testing



Hints

- The code uses many 3D arrays for E, B, rho, jx, jy, jz, pxx, pxy, pxz, pyy, pyz and pzz
  - In on order to move these arrays to GPU you need 1D array
  - Use X_flat quantities
  - Use helper functions in Alloc.h

Detailed Objectives

- Port the particle mover and interpolation functions to GPU with CUDA
- Optimize them
- Measure Performance
- Compare performance with CPU and OpenaCC

Check grading criteria

https://kth.instructure.com/courses/12406/pages/grading-criteria?module_item_id=179237

Grade	Code
A	• Completely offload both the particle mover and interpolation to CUDA, with both steps performed in a single kernel. • Implement a simple particle mover that uses OpenACC.
B	• Use pinned memory on the host. • Use Stream and async copy
C	• Implement mini-batches of particles to the particle mover so that the application can process more particles than it fits on the GPU.
D	• Port part of the interpolation (interpP2G) to CUDA. • All the CUDA memory management is **correctly** taken care of.
E	• Port the particle mover (mover_PC) of the code to use GPU, without further optimization. • All the CUDA memory management is **correctly** taken care of. • All the input files that are used to perform the experiments are provided.

Two Deliverables

- **Design Document**. Due on Dec. 8 2019. **P/F**
- **Final Report** + **code** on GitHub. Due on Jan. 14. **A-E**.
  - This is the final exam
- More instructions on the structure of these documents on the assignment page