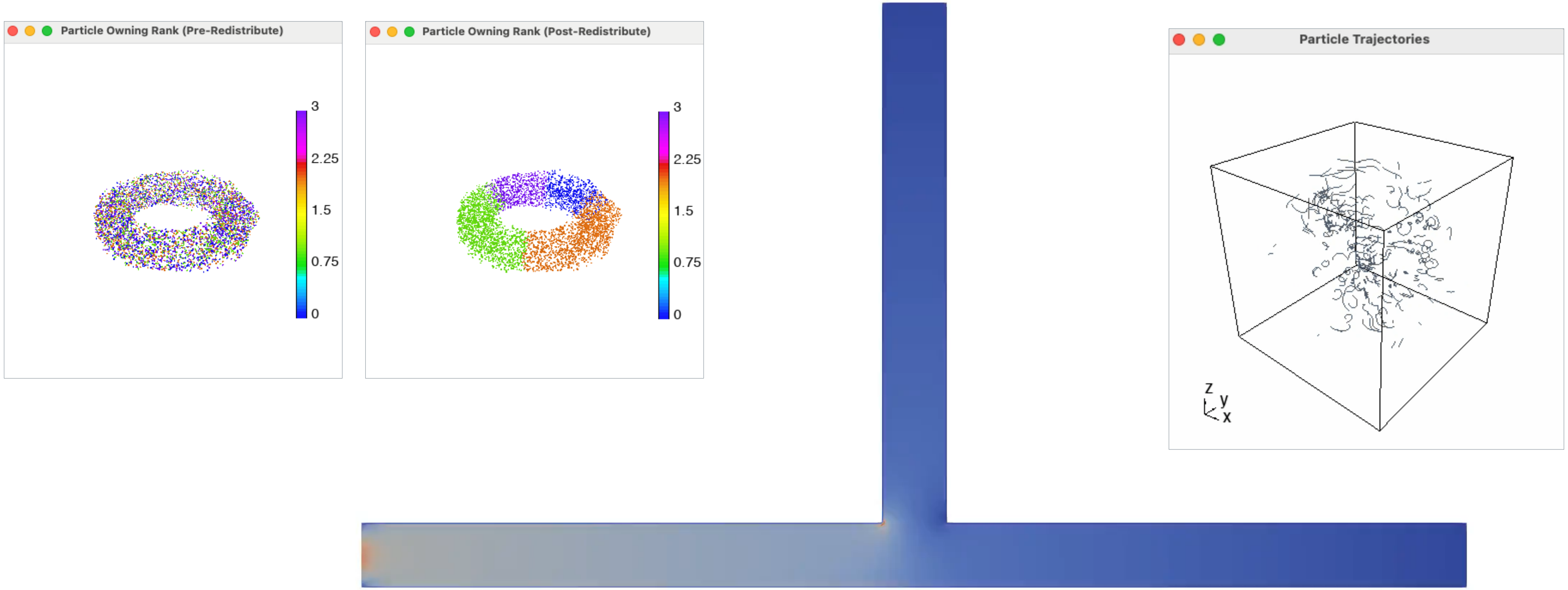


Particle Tracking in MFEM

Joseph Signorelli, Ketan Mittal, Tzanio Kolev

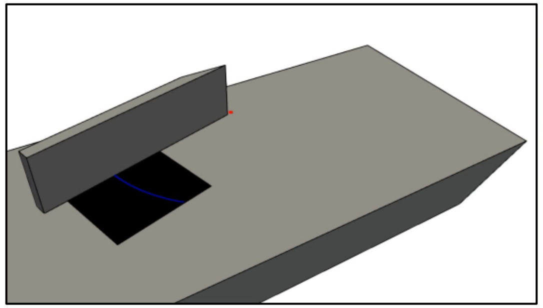
MFEM Workshop 2025

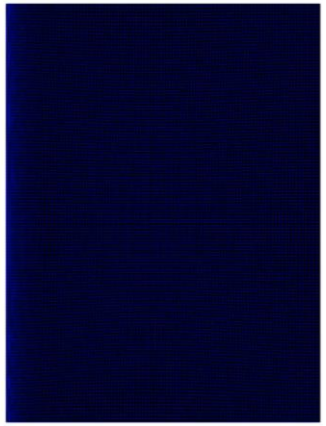
MFEM in Bodony Group @ UIUC

JOTS*

- CG thermomechanical solver for fluid-thermal-structure interactions
- *By Myself*

FTSI of Shock/BL Interaction

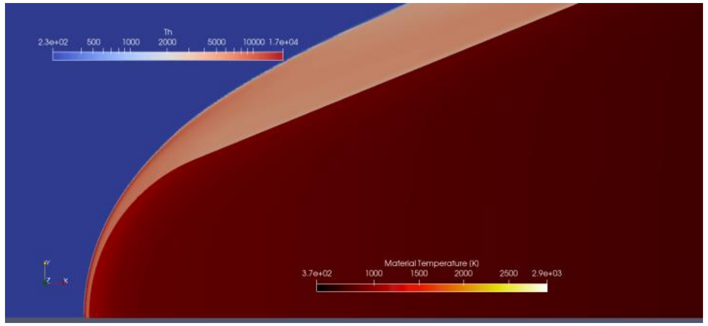




CHyPS**

- DG ablative thermal protection system material response solver
- *By Rob Chiodi + Blaine Vollmer*

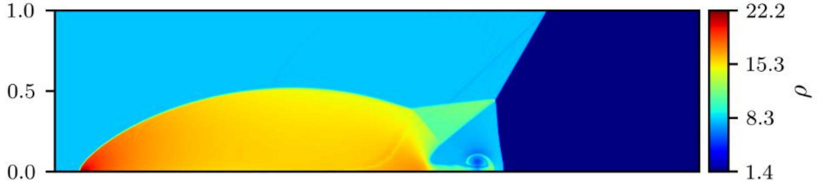
Ablating Porous Medium, Paul Poovakulum



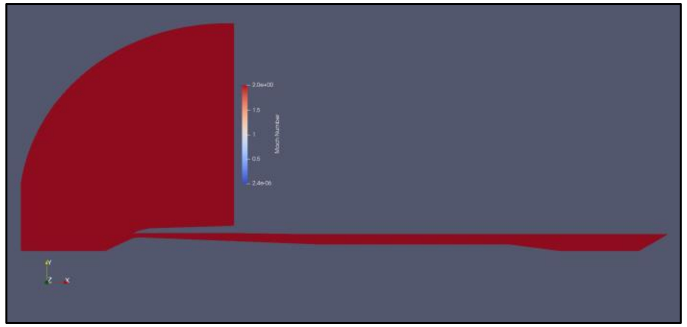
Prandtl***

- DG-SEM compressible Navier-Stokes solver
- *By Farhad Hasanli*

Double Mach Reflection



Ramjet Inlet Buzz, Mohammad Alhussaini



*Signorelli et al., AIAA 2024-3548.
**Chiodi et al., AIAA 2022-1501.
***Hasanli, Farhad, MS Thesis. 2025.

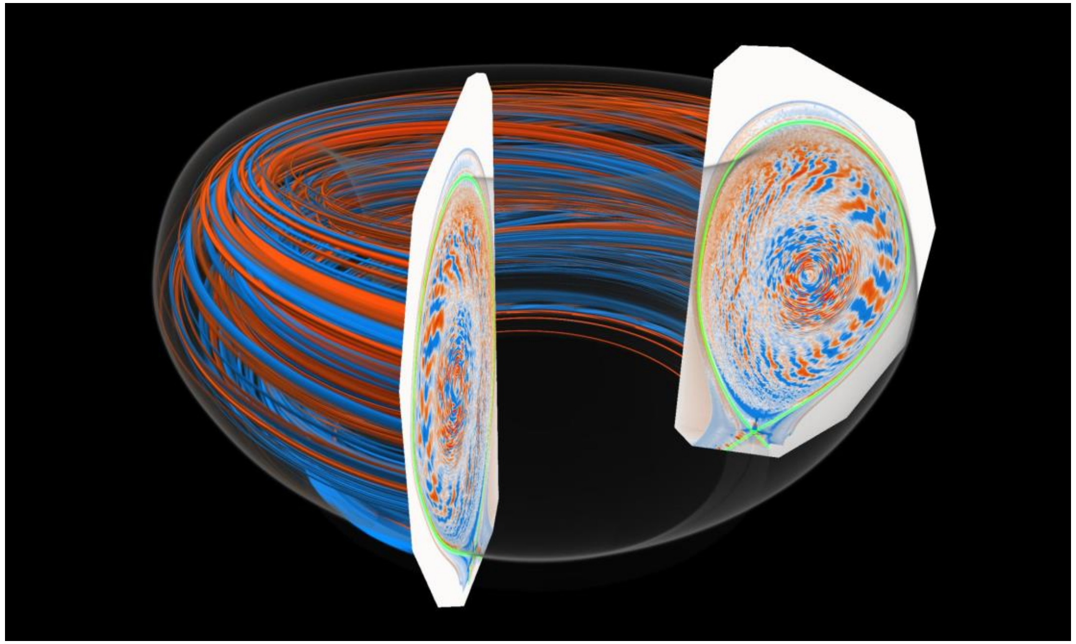
Outline

- Motivation + Goals
- New Classes
  - Vector Data Storage: `MultiVector`
  - Particle Container: `ParticleSet`
  - Particle Data Accessor: `Particle`
- New Miniapps/Solvers
  - `gslib/particles_redist`
  - `electromagnetics/lorentz`
  - `navier/navier_particles`
  - `navier/navier_bifurcation`

Motivation for a Particle Tracking Framework

- Wide range of applications....
  - Sediment modeling in dammed rivers[1,2]
  - Tokamaks (clean fusion energy)[3]

- **Not native feature of MFEM (...until now!)**



*From ALCF case study by PPPL (PI: Chang)

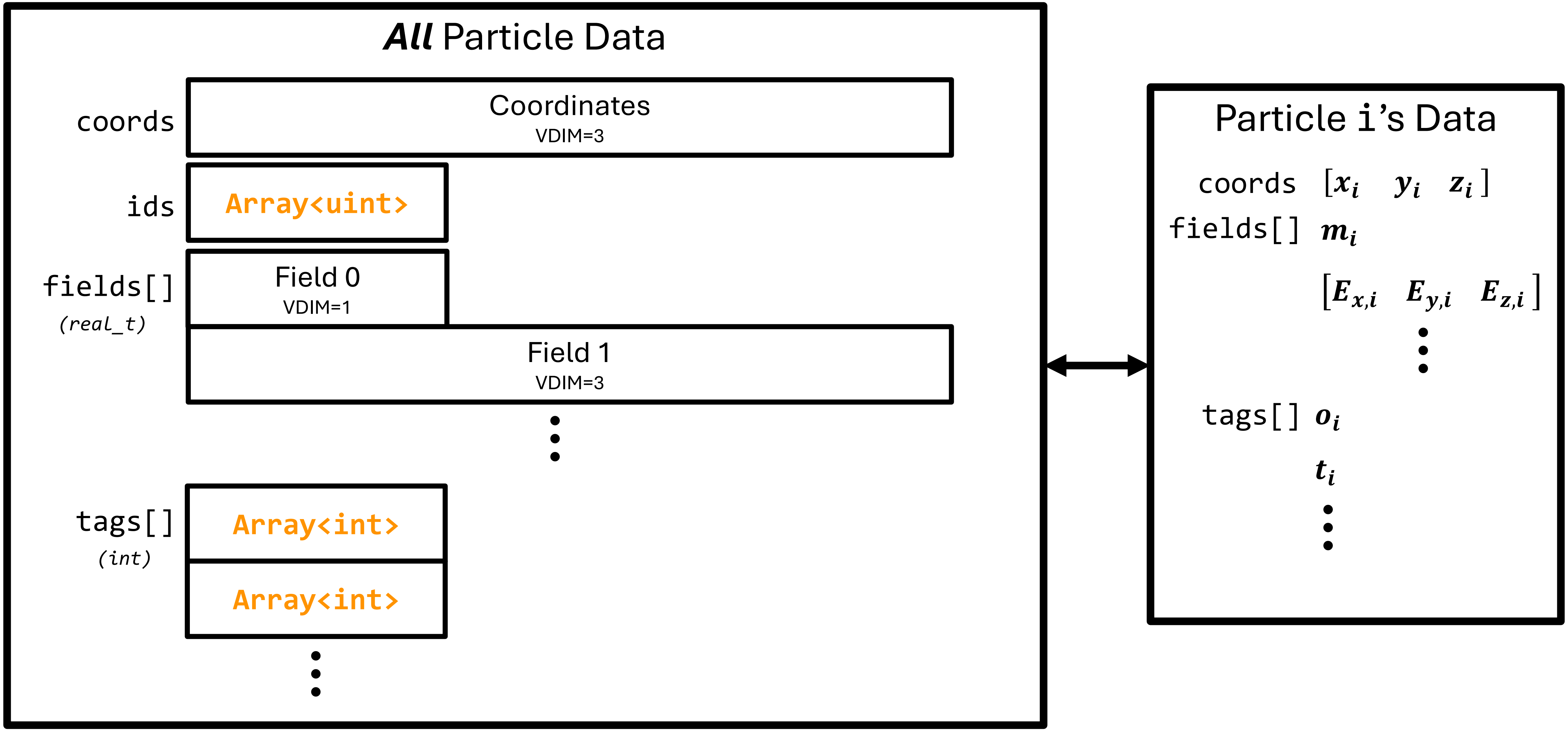
[1] Latrubesse et al. (2017)
[2] Vanoni (1946)
[3] Angioni et al. (2009)

Goals

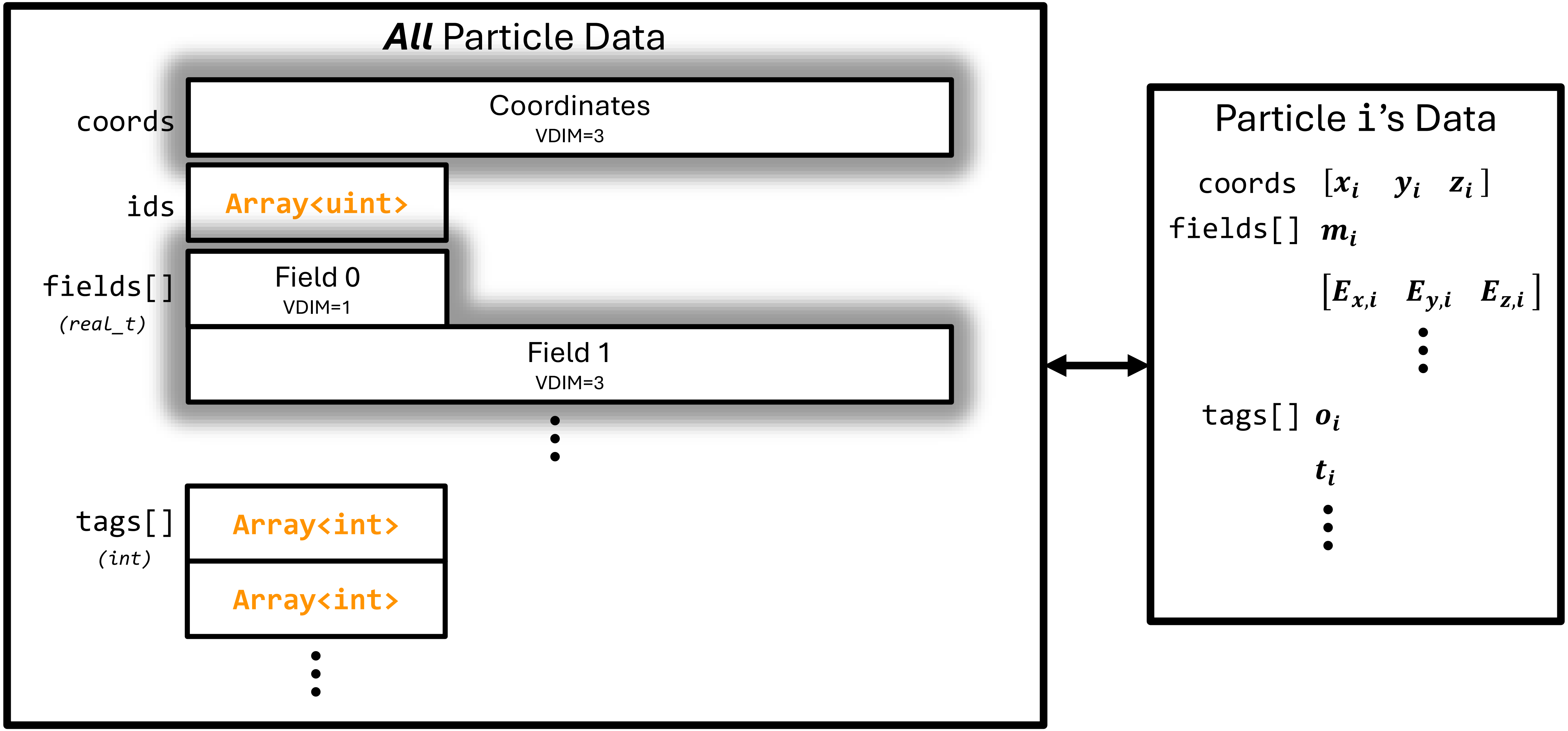
- Create lightweight, scalable particle container
  - Utilize existing MFEM data structures + styles
  - Interface with `FindPointsGSLIB`
  - Support flexible memory layout for variety of usage needs
  - Track particles globally using unique ID
  - Enable parallel redistribution of particle data
  - Allow easy addition, modification, + removal of particles

- Demonstrate usage + features through miniapps

New Classes



All Particle Data
coords
Coordinates
VDIM=3
ids
Array<uint>
fields[]
(real_t)
Field 0
VDIM=1
Field 1
VDIM=3
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i



All Particle Data
coords
Coordinates
VDIM=3
ids
Array<uint>
fields[]
(real_t)
Field 0
VDIM=1
Field 1
VDIM=3
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i

Need for a Vector of Vector Data…

- Idea: Carry all particle data (for a field) in a single `Vector`
  - Arbitrary vector dimension (ex: 1 for charge, 3 for momentum, …)
  - Any ordering
    - `byVDIM:   XYZ XYZ XYZ`
    - `byNODES: XXX YYY ZZZ`

- Similar to `GridFunction` (a type of `Vector`)

- Motivated general class for carrying N entries of `Vector` data…

Vector Data Storage: `MultiVector`

- `MultiVector`
  - Lightweight type derived from `Vector`
  - Accepts number of vectors/entries, vector dimension, and ordering

Example: NumVectors: 4
VDim: 3

`MultiVector`

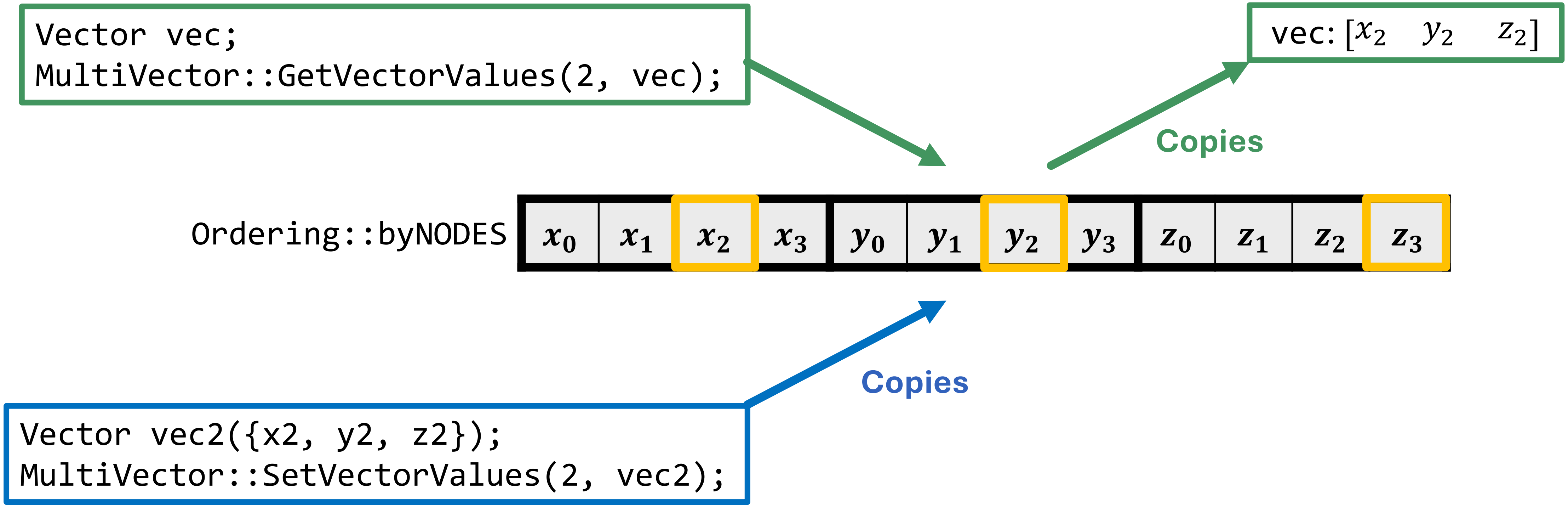
`Ordering::byVDIM`	x_0	y_0	z_0	x_1	y_1	z_1	x_2	y_2	z_2	x_3	y_3	z_3

`MultiVector`

`Ordering::byNODES`	x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3	z_0	z_1	z_2	z_3

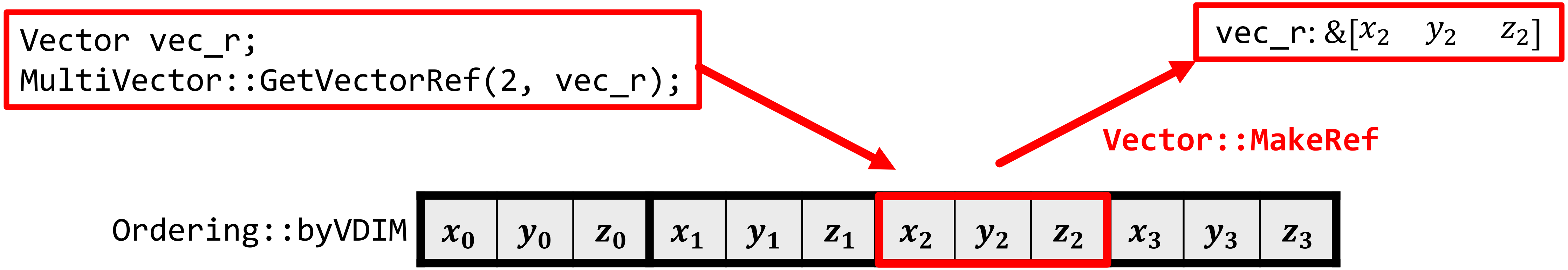
Vector Data Storage: MultiVector

- Simple getters + setters
  - Consider entry (particle) 2



Vector Data Storage: MultiVector

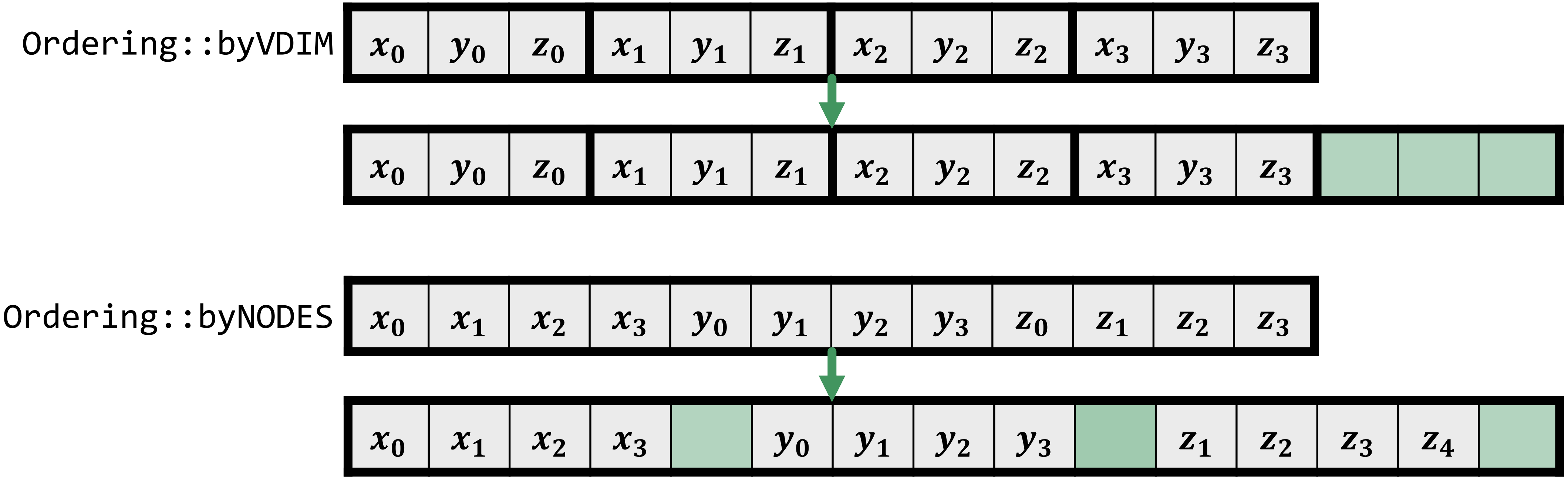
- Get Vector as reference for byVDIM

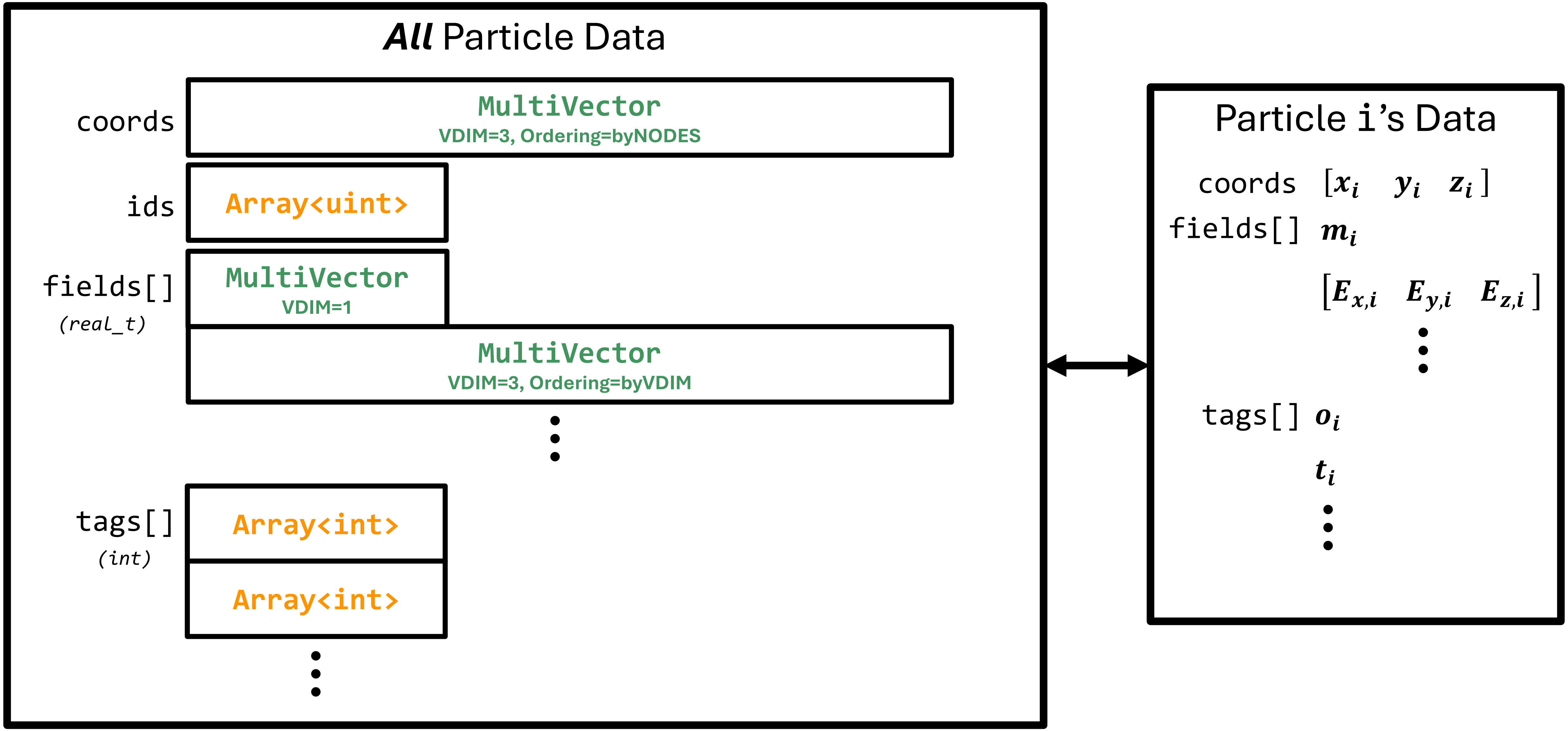


Vector Data Storage: `MultiVector`

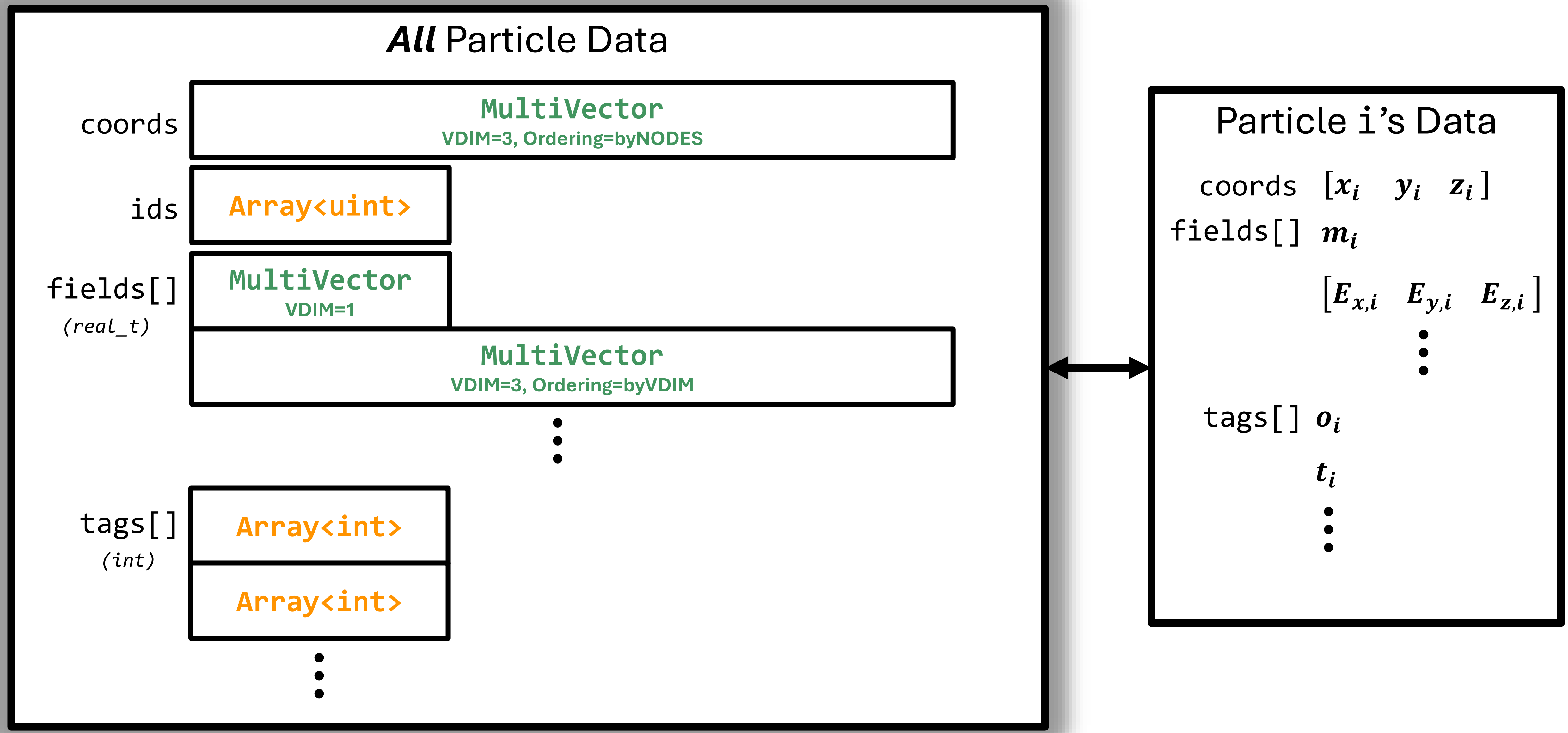
- Ordering-mindful resizing
  - Important for adding + removing particle data

```
MultiVector::SetNumVectors(5);
```





All Particle Data
coords
MultiVector
VDIM=3, Ordering=byNODES
ids
Array<uint>
fields[]
(real_t)
MultiVector
VDIM=1
MultiVector
VDIM=3, Ordering=byVDIM
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i


All Particle Data
coords
MultiVector
VDIM=3, Ordering=byNODES
ids
Array<uint>
fields[]
(real_t)
MultiVector
VDIM=1
MultiVector
VDIM=3, Ordering=byVDIM
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i

Particle Container: ParticleSet

- Manager of **all**...
  - Coords
  - IDs
  - Fields
  - Tags

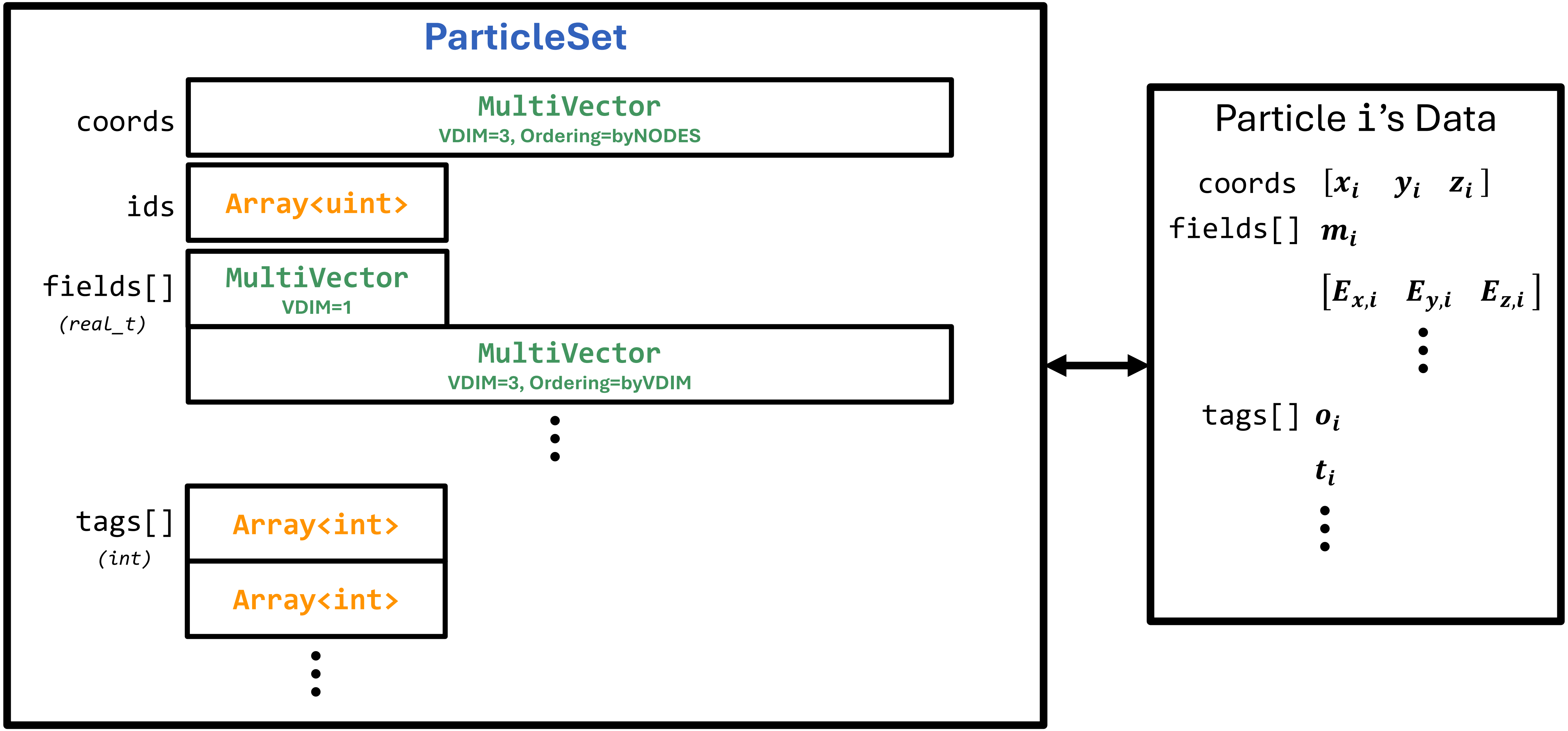
```
// Create ParticleSet
ParticleSet particles(MPI_COMM_WORLD, rank_num_particles, space_dim);
// Particle IDs are assigned uniquely globally, starting with
// (rank) and striding by (size)

// Access coordinates MultiVector&, and set as desired:
for (int i = 0; i < particles.GetNP(); i++)
{
    Vector p_coords(space_dim);
    ...
    particles.Coords().SetVectorValues(i, p_coords);
}
```

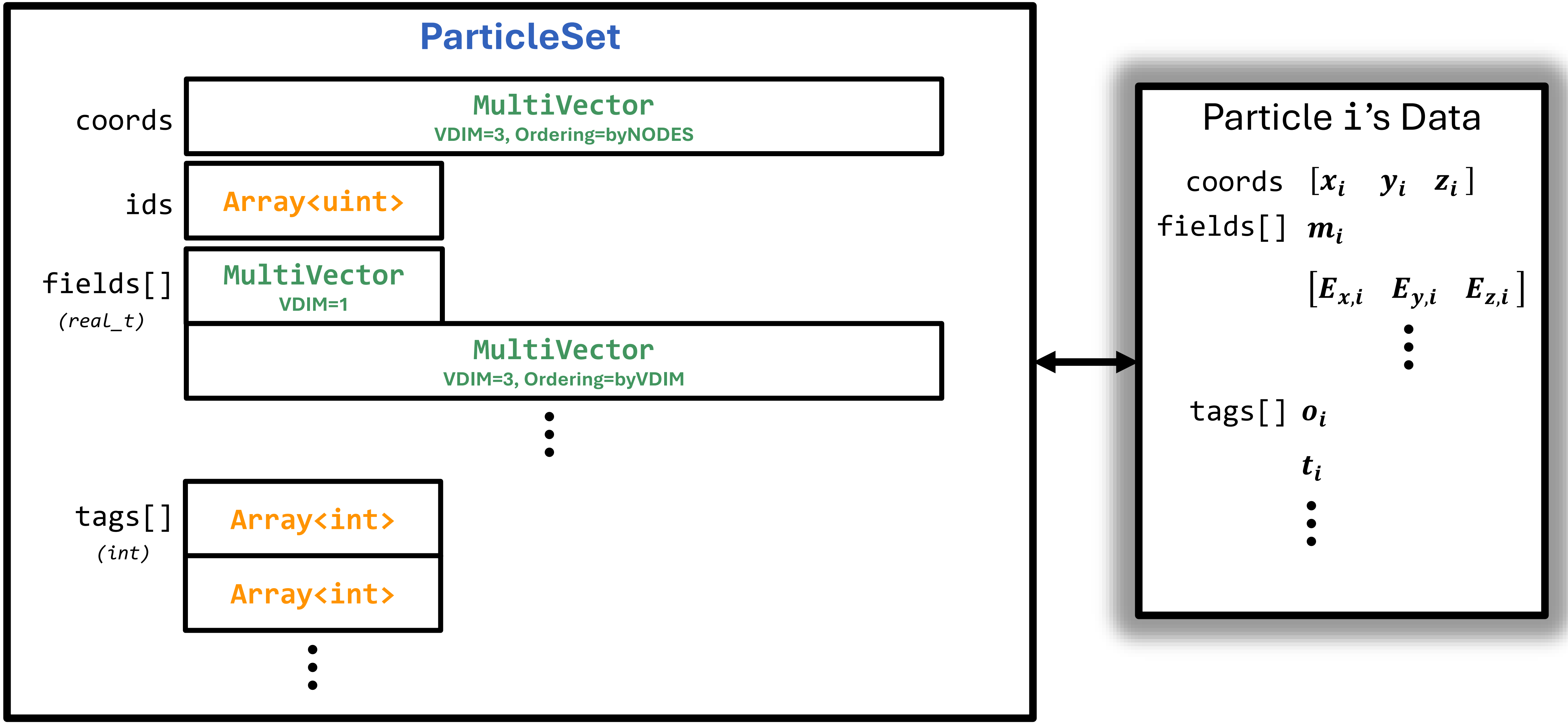
Particle Container: ParticleSet

```
// Add fields, "tracked" internally by ParticleSet
int m_idx = particles.AddField(1, Ordering::byVDIM, “Mass”);
int v_idx = particles.AddField(space_dim, Ordering::byVDIM, “Particle_Velocity”);
int u_idx = particles.AddField(space_dim, Ordering::byVDIM, “Fluid_Velocity”);

// Interfacing with FindPointsGSLIB:
// Interpolate any desired GridFunctions onto MultiVectors
ParGridFunction fluid_vel_gf = ...;
MultiVector &X = particles.Coords();
MultiVector &U = particles.Field(u_idx);

FindPointsGSLIB finder(MPI_COMM_WORLD);
finder.Setup(pmesh);
finder.FindPoints(X, X.GetOrdering());
finder.Interpolate(fluid_vel_gf, U);
```

Particle Container: ParticleSet

```
// If particles leave the domain, remove them:
particles.RemoveParticles(finder.GetPointsNotFoundIndices());
// ParticleSet removes particle data, based on Ordering::Type, from all field
    MultiVectors and tag Arrays internal to it


// Redistribute particle data to the rank that they are physically located on:
particles.Redistribute(finder.GetProc());
// Using GSLIB, particle data is sent + received, and all field MultiVectors and tag
    Arrays are properly updated and resized accordingly.


// Simple outputting feature (leverages MPI-IO)
particles.PrintCSV("particle_data.csv");
```

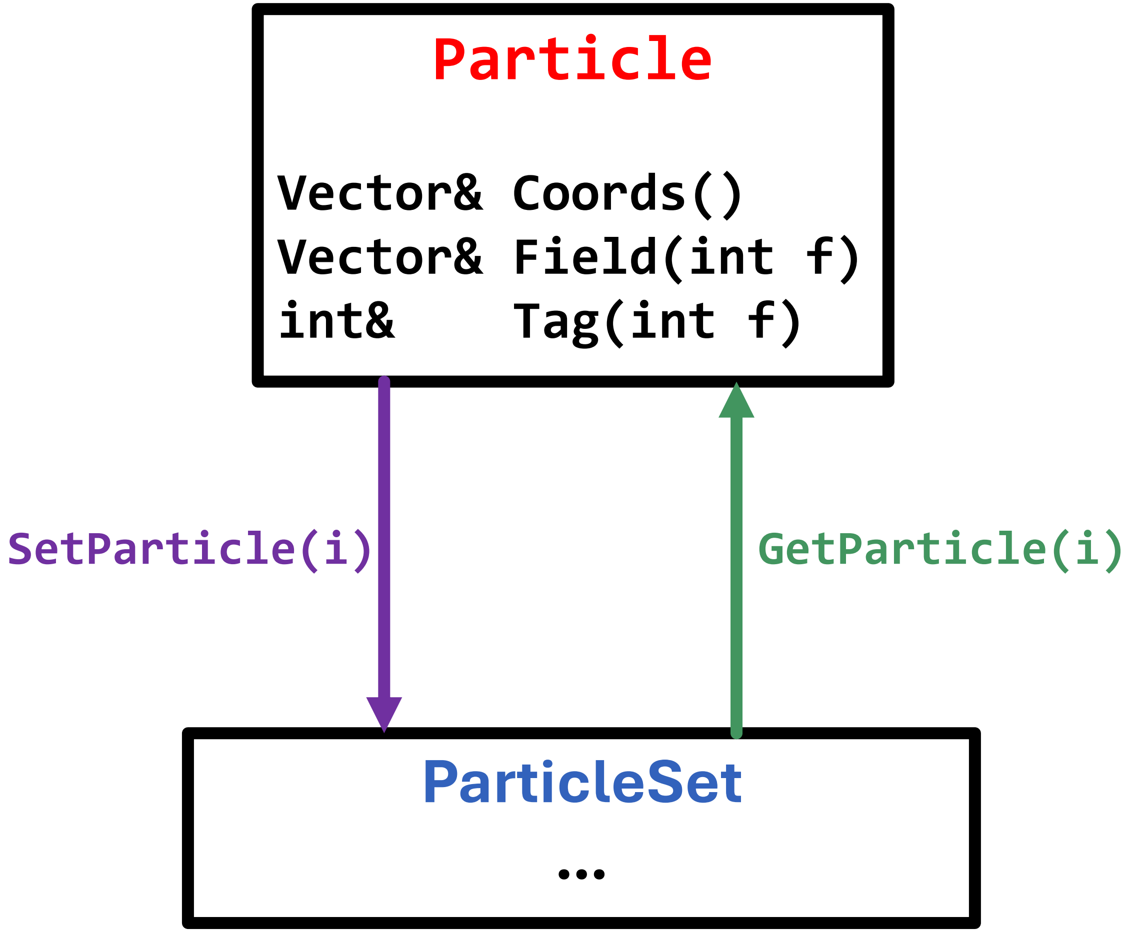

ParticleSet
coords
MultiVector
VDIM=3, Ordering=byNODES
ids
Array<uint>
fields[]
(real_t)
MultiVector
VDIM=1
MultiVector
VDIM=3, Ordering=byVDIM
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i

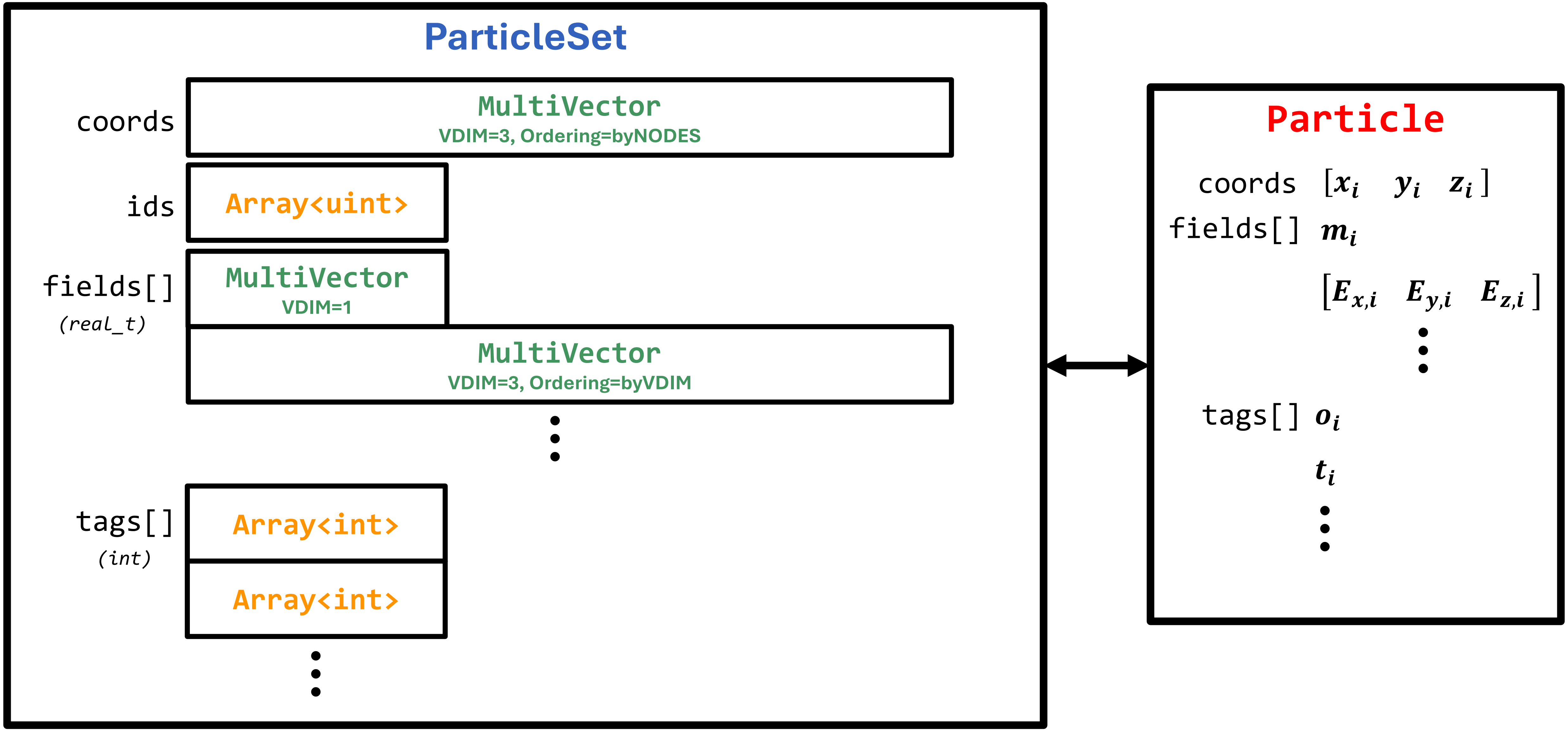


ParticleSet
coords
MultiVector
VDIM=3, Ordering=byNODES
ids
Array<uint>
fields[]
(real_t)
MultiVector
VDIM=1
MultiVector
VDIM=3, Ordering=byVDIM
tags[]
(int)
Array<int>
Array<int>
Particle i's Data
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i

Particle Data Accessor: Particle

- Natural interface for individual particles
- Get + set particles in `ParticleSet` using `Particle`
- `ParticleSet::GetParticleRef`
  - Only when all fields ordered by `VDIM`
  - `MultiVector::GetVectorRef`

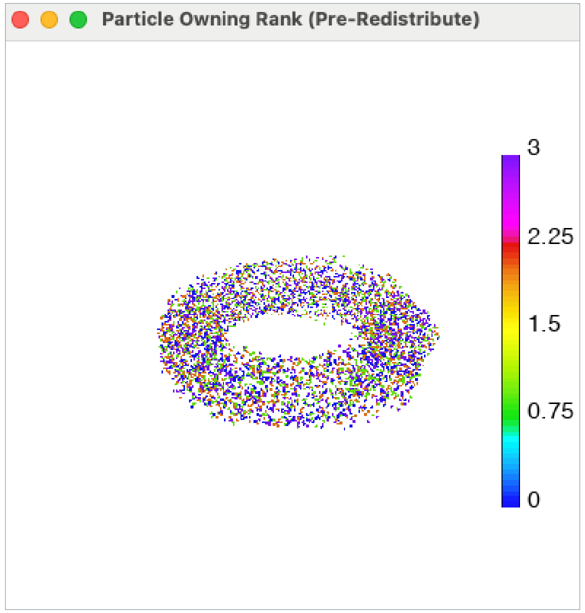


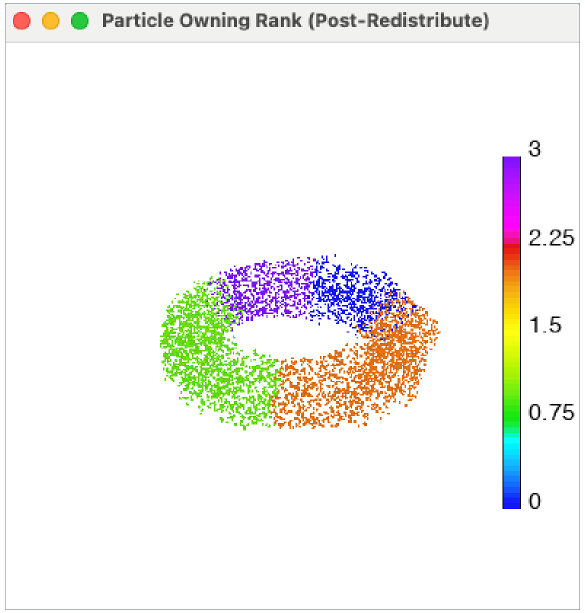

ParticleSet
coords
MultiVector
VDIM=3, Ordering=byNODES
ids
Array<uint>
fields[]
(real_t)
MultiVector
VDIM=1
MultiVector
VDIM=3, Ordering=byVDIM
tags[]
(int)
Array<int>
Array<int>
Particle
coords $[x_i \quad y_i \quad z_i]$
fields[] m_i
$[E_{x,i} \quad E_{y,i} \quad E_{z,i}]$
tags[] o_i
t_i

New Miniapps/Solvers

gslib/particles_redist

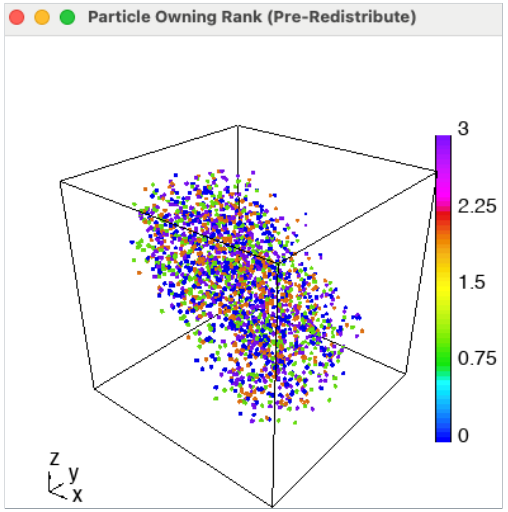
- Initializes particles randomly on input mesh
- Redistributes using GSLIB
- Visualizes particle owning-rank pre- and post-redistribute

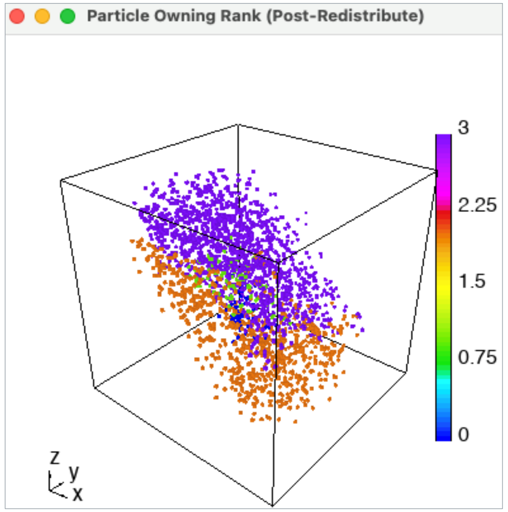


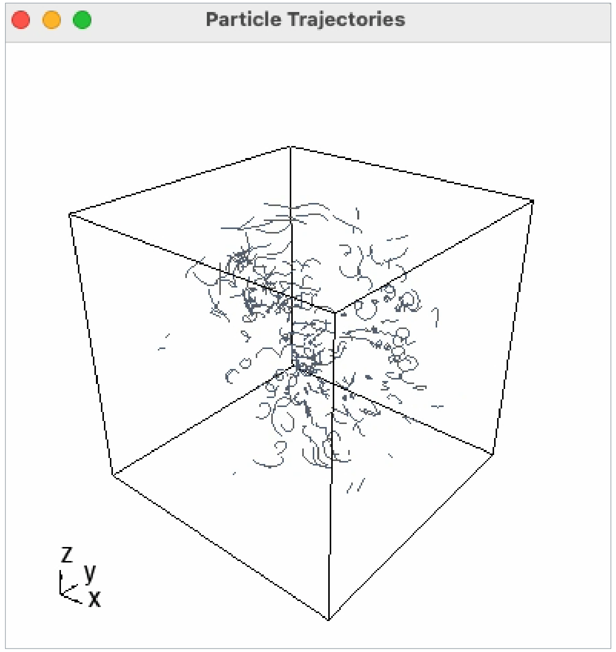


electromagnetics/lorentz

- **Adapted from single-particle version by Mark L. Stowell*
- Load E or B field, integrate w/ Boris algorithm[4]
- Demonstrates:
  - Particle redistribution + removal
  - Particle trajectory visualizer (common/particles_extras)





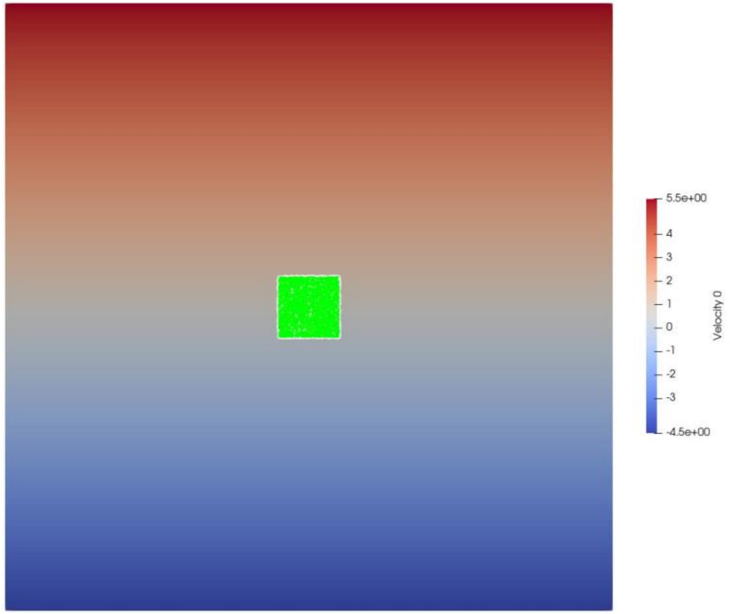


[4] Qin et al. (2013). “Why is Boris algorithm so good?”, Physics of Plasmas, Volume 20 Issue 8.

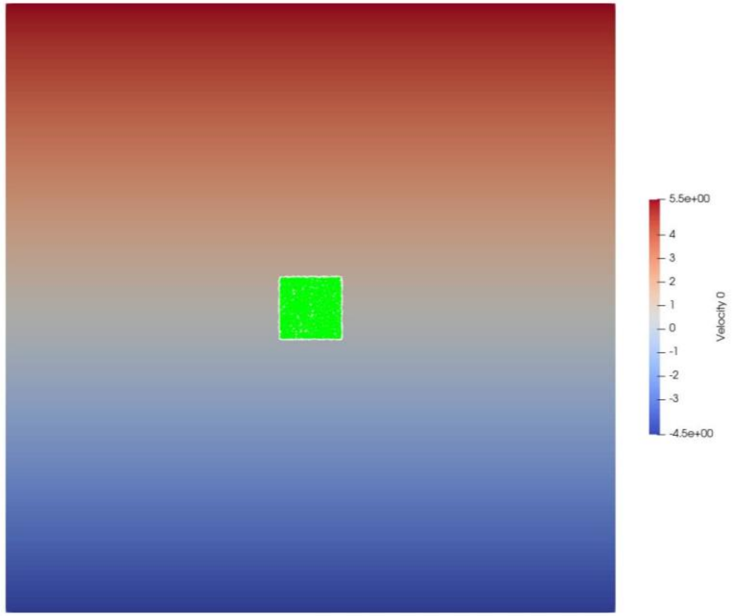
navier/navier_particles

$$\frac{d\boldsymbol{v}}{dt} = \kappa(\boldsymbol{u} - \boldsymbol{v}) - \gamma\hat{\boldsymbol{k}} + \zeta(\boldsymbol{\omega} \times \boldsymbol{v} + \boldsymbol{u} \times \boldsymbol{\omega})$$

- New incompressible fluid particle solver: NavierParticles
- Semi-implicit Lagrangian particle tracking[5]



$\kappa = \gamma = 0$



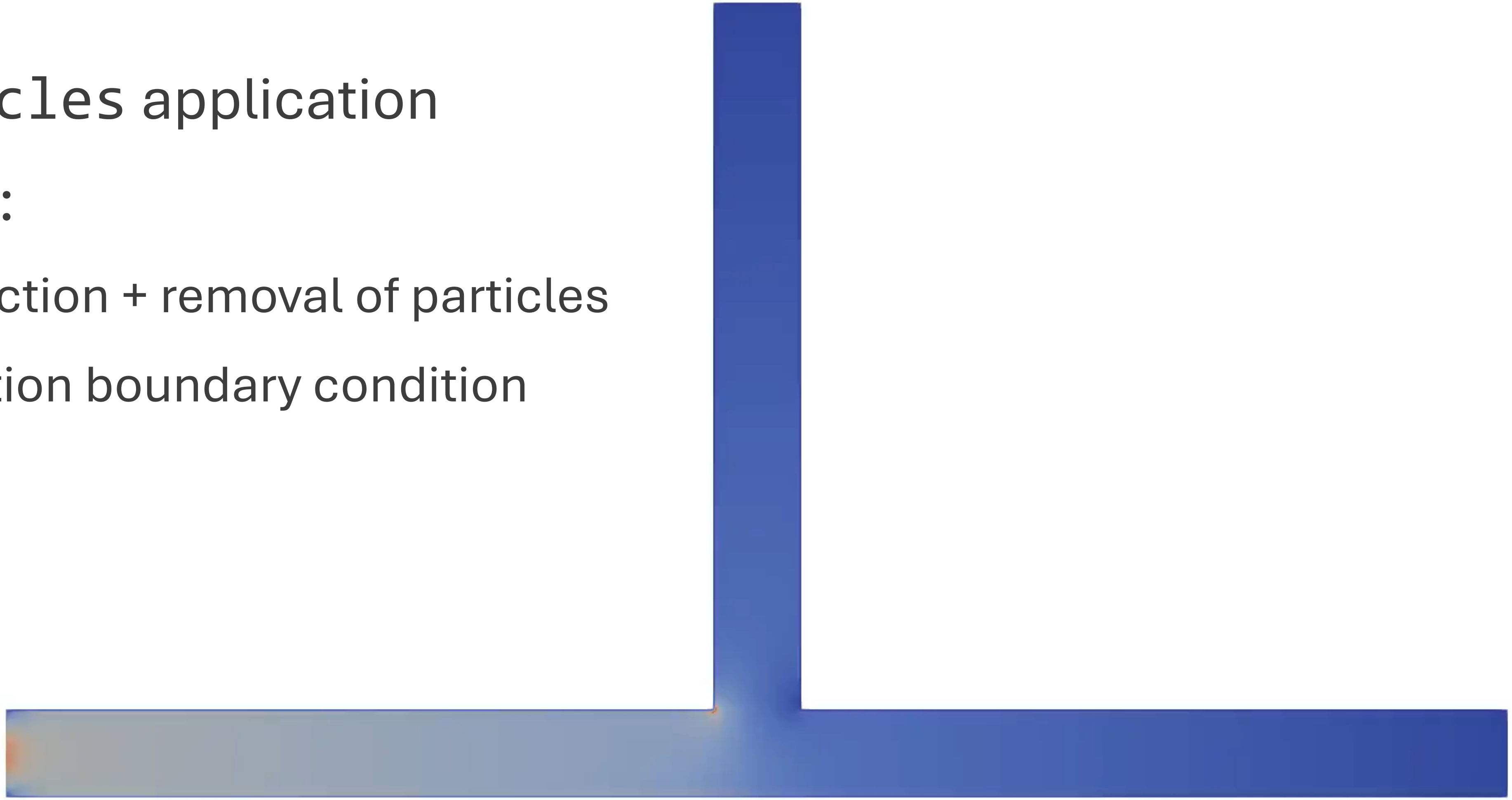
$\zeta = 0$

● Exact
○ Numerical

[5] Dutta (2017). Ph.D. Dissertation: "Bulle-Effect and its implications for morphodynamics of river diversions".

navier/navier_bifurcation

- `NavierParticles` application
- Demonstrates:
  - Continual injection + removal of particles
  - 2D wall reflection boundary condition



Summary

- Scalable particle simulation framework
- EM and Navier-Stokes examples

- Future work
  - Particle-particle interaction
  - Particle-in-cell

Thank you! Questions?

PRs:

- #4567: Lorentz Miniapp
- #4981: `MultiVector`
- #4986: Particle Tracking (`ParticleSet`)