Science and Technology Facilities Council

Scientific Computing

OPS-Particle: A DSL for particulate systems

ParaSolS

Particulate Solids Simulations

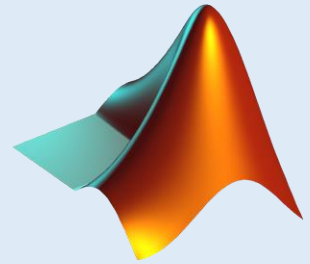
What is a domain specific language?

- A computer language specialized to specific applications
- Need of concepts and domain operations (functions)

Advantages

- Restrict writing code that is difficult for the compiler to reason and optimize
- Implementation of the API left to a lower level
  - Target implementation to hardware - automatically generate implementation from specification for the context
  - Generate code in best parallelization model
  - Code can be used by multiple applications

Examples of domain specific languages



LaTeX

Why do we need domain specific languages ?

HPC landscape

- **Diversity in HPC resources (CPUs, GPUs)**
- **Programming diversity (i.e., open standards, proprietary model etc)**

The picture for the TOP 10 (https://top500.org/lists/top500/2026/06/)

- **40% AMD GPUs, 30% NVIDIA GPUs, 20% CPUs and 10% Intel GPUS**
- **But #1 is a CPU-based!!!!**

DSLs can bring balance !!!

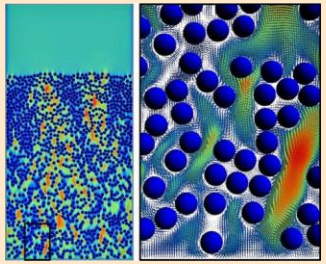
Challenges

- **New platforms may require different tuning efforts**
- **Cannot recode applications for each new type of architecture**

Our goal

Develop a domain-specific language (DSL) framework for modelling particulate systems based on OPS library

Modeling of particle-turbulent flows

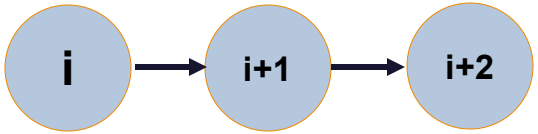


DNS modeling of particulate systems

OPS-Particle iterators

User writes kernel functions that handle the Finest details of the interactions

Particle iterations



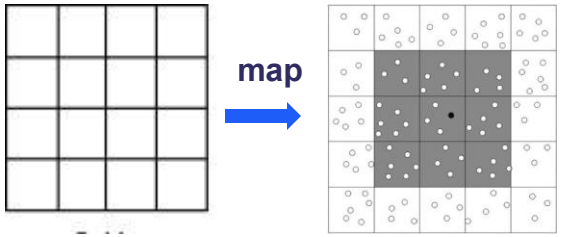
Iterate over particle structures

Used to update particle positions or velocities

Grid/particle interactions

Grid

Particles in cells

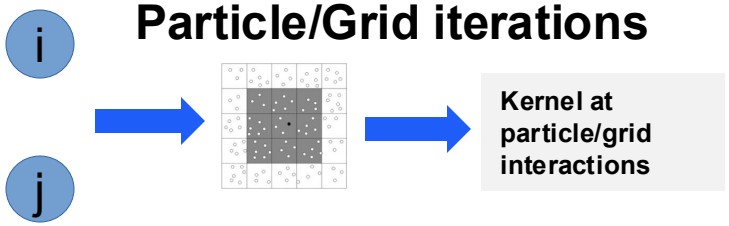


User kernel

Operations at grid/particle level

Scattering particle data to grid

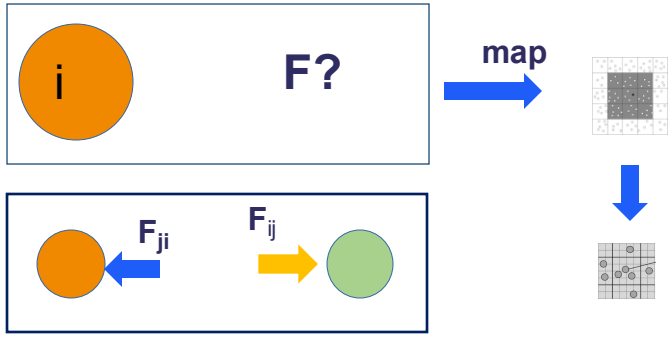
Particle/Grid iterations



Gathering grid data

Iterators to compute forces to particles due to grids

Particle/particle inter.



Stencil to access J

Science and Technology Facilities Council

Scientific Computing

Scientific Computing

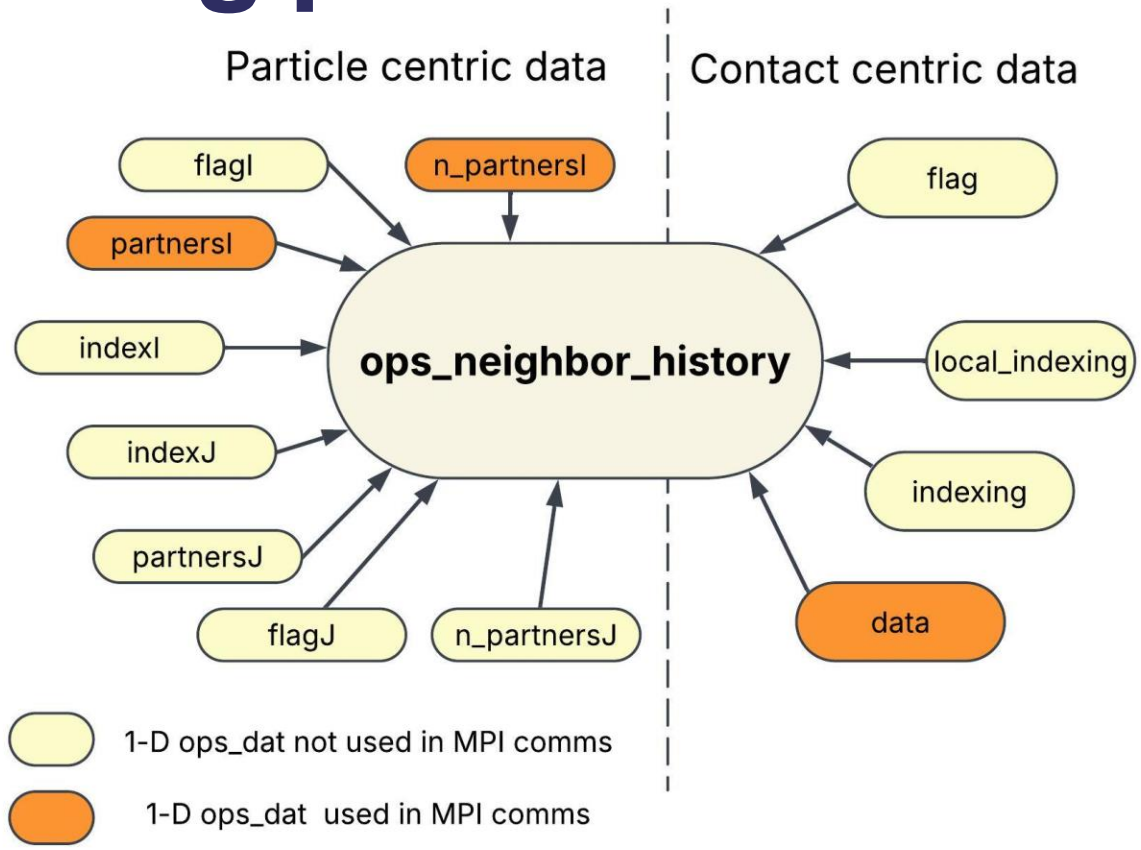
Updates on OPS-Particle

Restructuring of particle mapping structures and communication functionalities

- Early versions of OPS-Particle supported only double precision arithmetics
- Particle mapping and halo (interblock/intrablock) functionalities restructured to handle different floating-point precision arithmetics
- Introduced an agnostic C++ template format. Later to be integrated into OPS Python translator
- User can now set which particle data can be exchanged between processes or not

Neighbor histories-Testing phase

- OPS-Particle **supports** both particle-centric and contact-centric access to contact histories
- Verlet & neighbor lists are updated simultaneously.
- MPI communications rely on particle-centric ops-dat structures
- MPI comms of neighbor histories via first particle in neighbor
- Use arrays for **easier extension** to GPUs systems

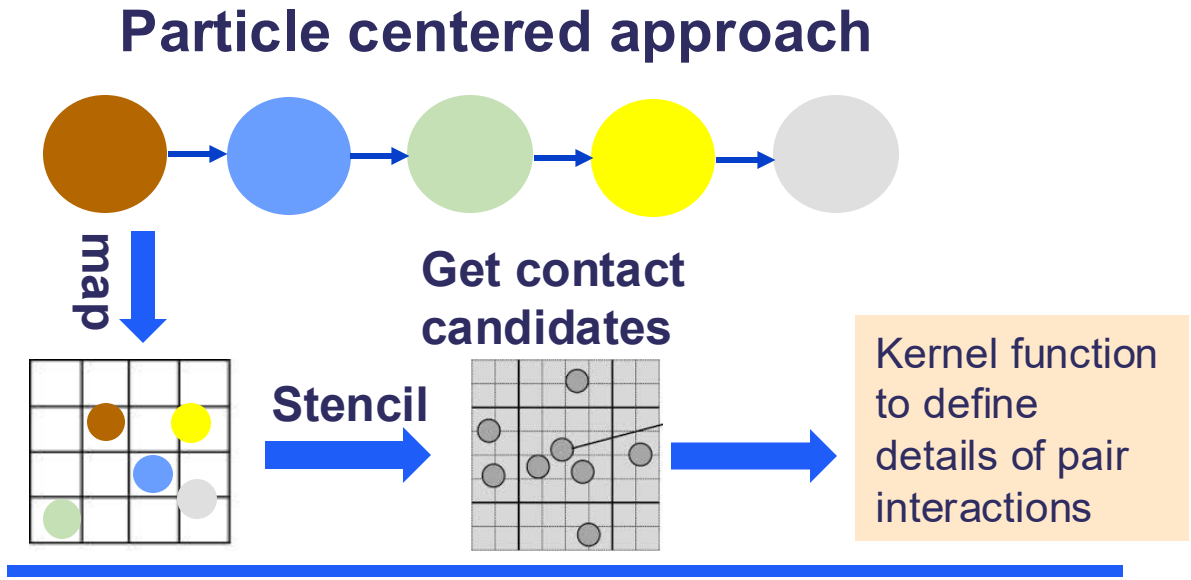


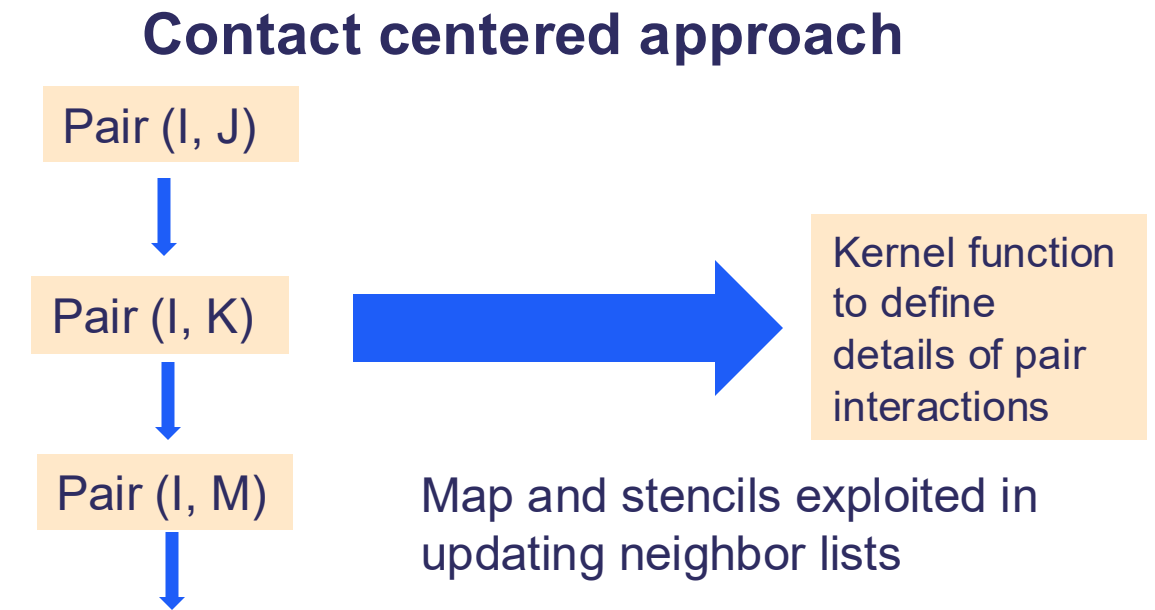

```
template<typename T>
ops_neighbor_history ops_decl_neigh_history(ops_particle particleI,
                                            ops_particle particleJ,
                                            int dofs, int no_neighborsI,
                                            int no_neighborsJ, int update_type,
                                            int type_history, T* array,
                                            const char* type, const char* name)
```

Science and Technology Facilities Council
Scientific Computing

OPS-Particle: Functions for force calculation

- OPS-Particle support both particle-based (**ops_particle_inter_loop**) and contact-based (**ops_particle_par_verlet**) force calculation loops
- Particle approach iterates over particles. Previously was iterating over grid-points.
- Verlet-approach iterates over **all possible contacts**, while particle-approach iterate over **all particles or particles in a user-defined region**
- Neighbor structures introduced to force calculation functions as a new ops_arg_dat type structure



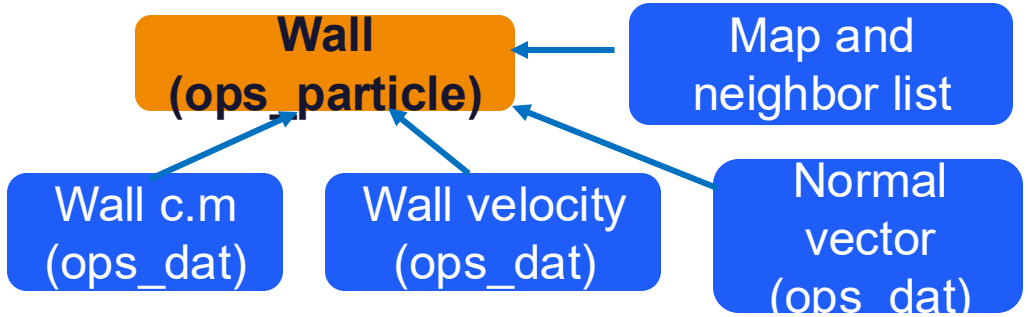


Science and Technology Facilities Council
Scientific Computing

Analytical walls on OPS-Particle

- Analytical walls are treated as a new type of particles (ops-particle)
- User defined ops-dat structures & particle maps fully define the wall.
- Build-in functionalities handling different wall types.
- Currently OPS-Particle supports only infinite walls normal to the x,y,z directions
- To be extended to other shapes

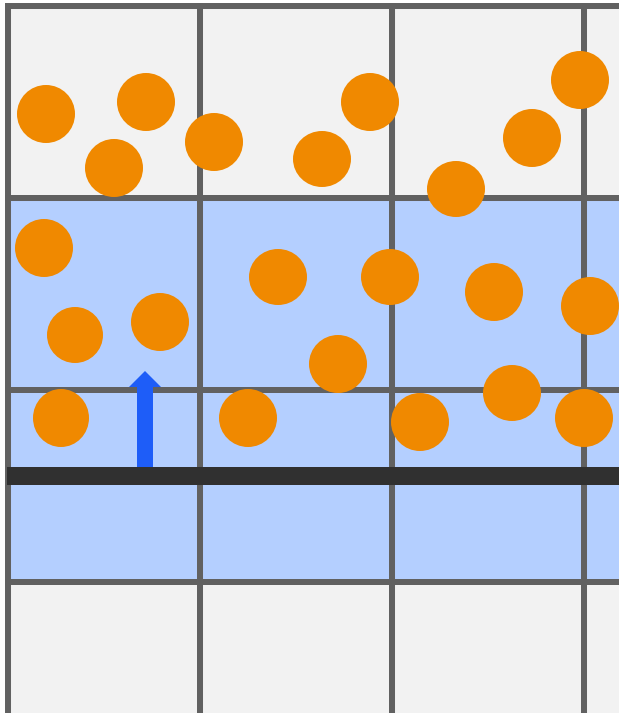
Example: Setting up a servo-mechanism for triaxial test simulations with rigid walls



Note: Wall is moving with user defined velocities

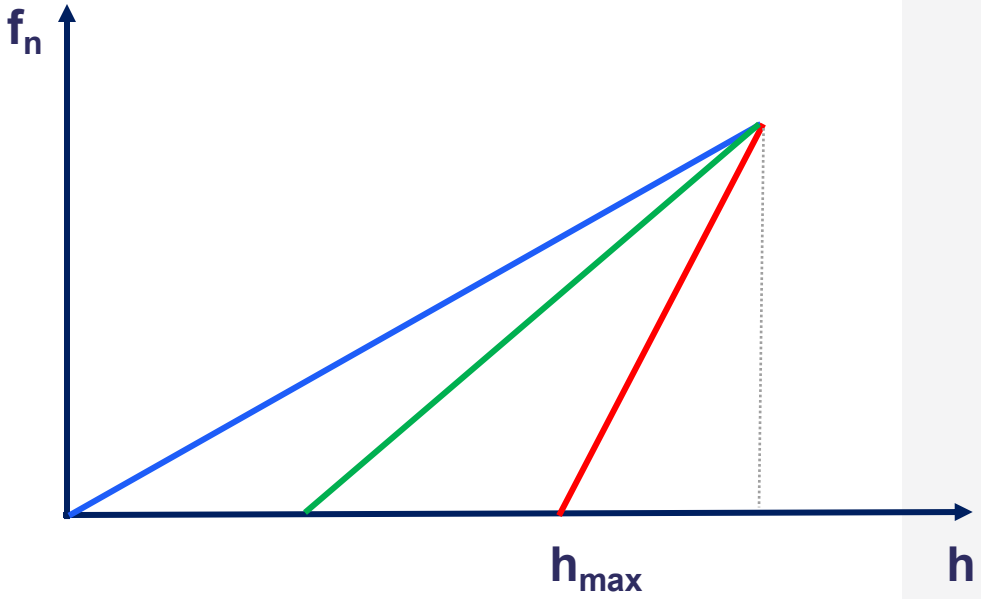
Accessing walls in iteration functions

- Define particle maps and neighbor lists (if necessary). Set wall as particle I in neighbor list.
- Update particle maps and neighbor lists
- Call iterative function with wall as particle I

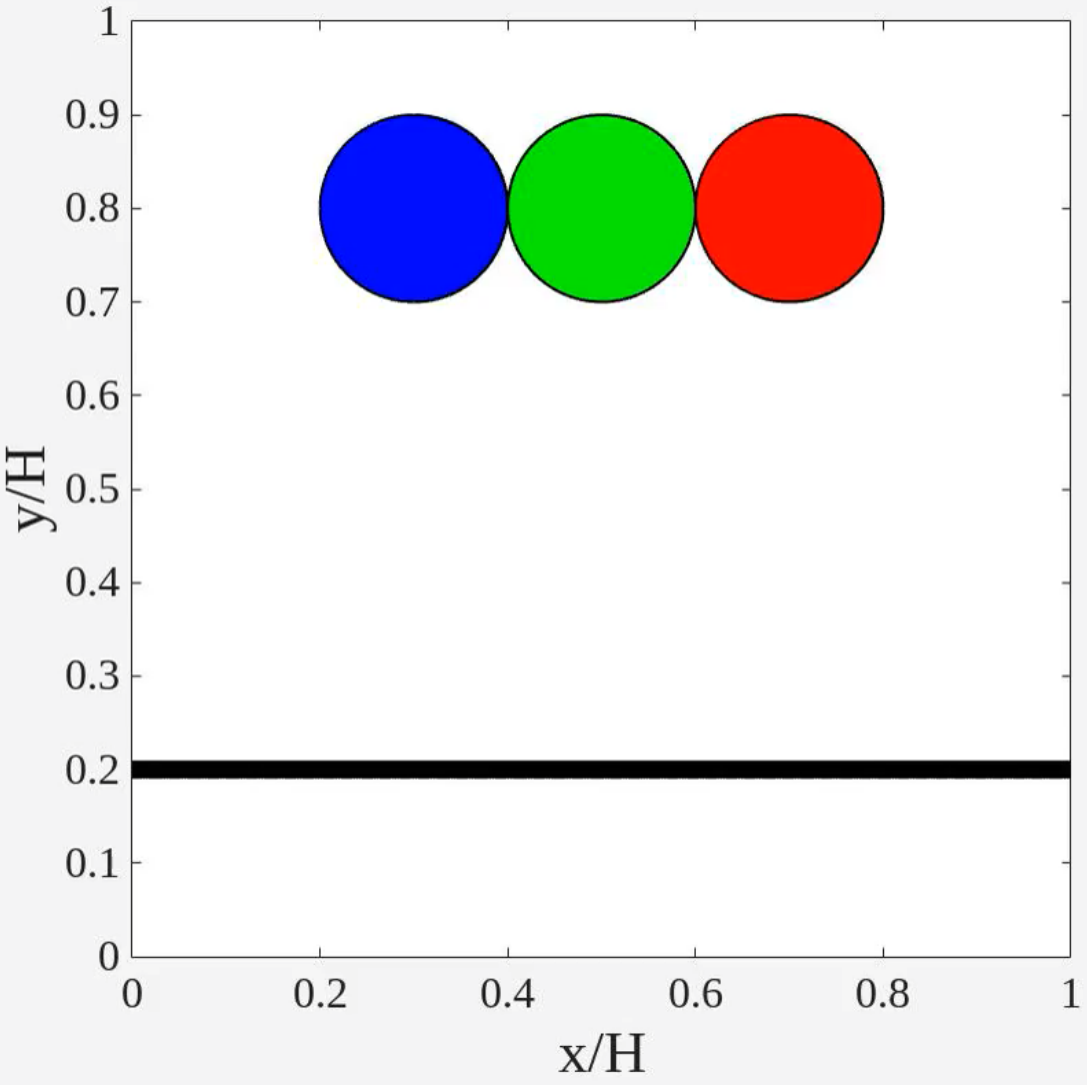


Science and Technology Facilities Council
Scientific Computing

An example: Elastic & inelastic spheres bouncing on a rigid wall

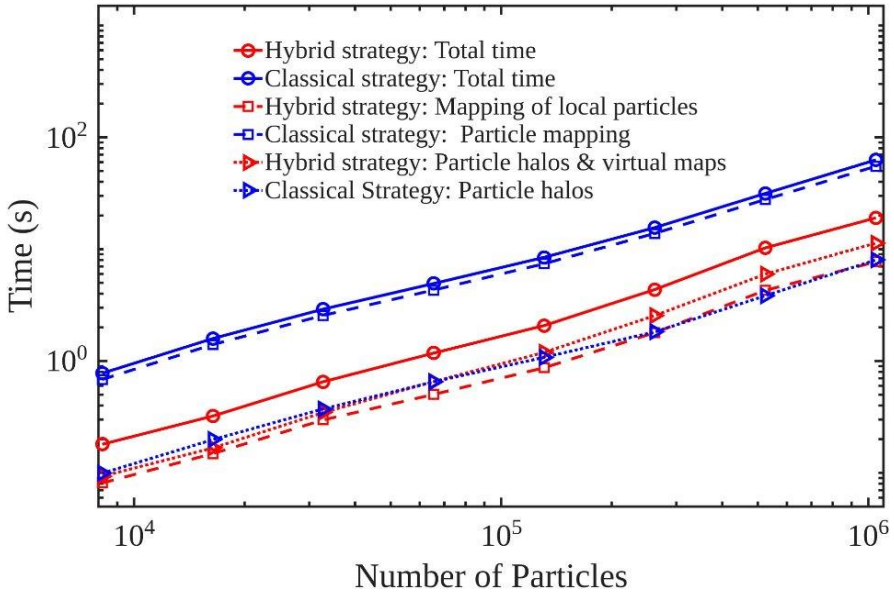


- The interaction of the spheres with the rigid wall was modeled with a bilinear-model
- Neighbor histories were used to store the h_{max} for each wall-particle pair



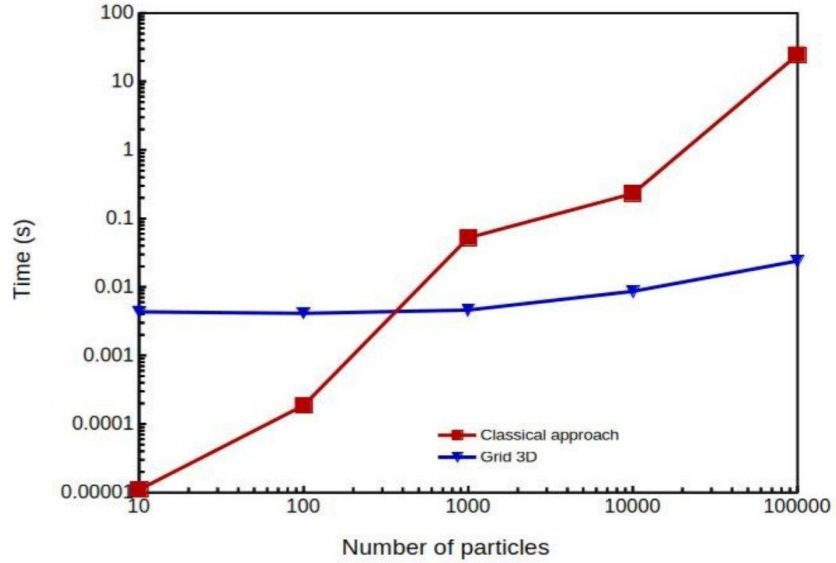
Why map exploitation is beneficial for large systems

Case I: Intrablock halos



- Intrablock communications rely on searching against all particles if projected into a given region
- Maps are rebuilt if a single particle moves to a neighbor cell
- Grid exploitation can improve algorithms

Case II: Random particle insertion



- **Projection of particle into grid expedites overlap checks for adding new particles**

Conclusions & Future plan

- **The main features of a domain specific language are set**
- **Pending release of sequential and MPI versions**

Future work

- First release of OPS-Particle via OPS github (https://github.com/OP-DSL/OPS) (End of September)
- Porting of particle maps to GPUs (October-End of November)
- Porting of particle iteration algorithms to GPUs (December-January)
- Testing and Benchmark cases (February-March)
- Development of Mercury-OPS (September-March)

Science and Technology Facilities Council

Scientific Computing

Thank you

scd.stfc.ac.uk

@SciComp_STFC