Toward Accelerated Unstructured Mesh Particle-in-Cell

Gerrett Diamond[1], Cameron W. Smith[1], Chonglin Zhang[1], Eisung S. Yoon[2], Gopan Gopakumar[1], Onkar Sahni[1], Mark S. Shephard[1]

[1]Scientific Computation Research Center
Rensselaer Polytechnic Institute

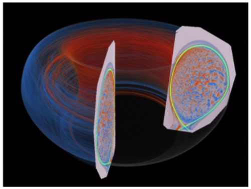
[2]Ulsan National Institute

November 18, 2019

Outline

Unstructured Mesh Particle-In-Cell (PIC) Simulation

- PIC simulations iterate over four main operations per time step:
  - Particle Push - particle positions are updated based on mesh fields.
  - Particle-to-Mesh - based on the new particle positions, mesh fields are updated.
  - Field Solve - domain level PDEs to update global mesh fields.
  - Mesh-to-Particle - particle information is updated for the next push operation.
- Two simulations of interest:
  - XGC - 3D PIC simulation using a 2D mesh representing polodial planes.
  - GITR - 3D mesh PIC monte-carlo simulation.


XGC tokamak simulation, 2 polodial planes

Mesh-based PIC

- Traditional approach to PIC is to primarily store particles
  - Each particle knows the mesh element it is within.
  - A copy of the mesh is maintained on all processes.
- Mesh-based PIC is when the primary storage is the mesh.
  - Each element maintains a list of the particles inside the element.
  - Easier to maintain a distributed mesh.
- Goal is to develop mesh-based PIC framework that operates efficiently on GPUs.

Outline

PUMIPic - Parallel Unstructured Mesh Infrastructure for Particle-in-cell

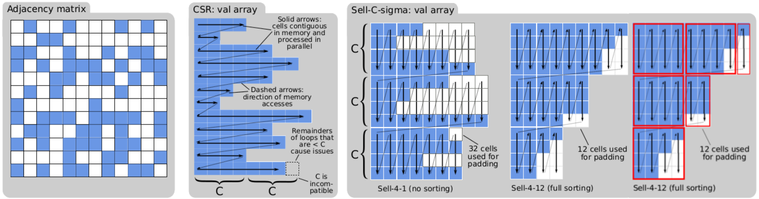
- Provides a set of data structures and algorithms for unstructured mesh-based PIC.
- Uses Kokkos for GPU support
- Includes:
    - GPU-based Particle Structure
    - GPU-based PIC Mesh Structure
    - Adjacency Search
    - Mesh Field Synchronization
    - Gyro-Average Scatter

Particle Data Structure

- Particles dominate computation and memory usage.
- Particle data structures need to account for:
  1. grouping particles by element for efficient mesh-particle interactions.
  2. different simulations requiring different information per particle.
- For performance on GPUs the particle structure must be:
  1. distributable to threads evenly.
  2. mapped to the hardware memory layout and access pattern.

Particle Data Structure - Sell-C-Sigma (SCS)

- Rotated CSR structure
- Groups rows into chunks mapped to the hardware of GPU
- Padding improves access pattern at the cost of memory
- Performs sorting of rows to reduce padding
- Vertical slicing improves distribution of work



Adjacency matrix
CSR: val array
Solid arrows: cells contiguous in memory and processed in parallel
Dashed arrows: direction of memory accesses
Remainders of loops that are < C cause issues
C is incom-patible
C
C
Sell-C-sigma: val array
C
C
C
32 cells used for padding
12 cells used for padding
12 cells used for padding
Sell-4-1 (no sorting)
Sell-4-12 (full sorting)
Sell-4-12 (full sorting)

From left to right:
Adjacency Matrix, CSR, SCS with no sorting, full sorting, vertical slicing
"SlimSell: A Vectorized Graph Representation for Breadth-First Search", M. Besta et al.

Particle Data Structure - Sell-C-Sigma

- For PIC, the SCS is used with
  - A row per mesh element.
  - Each entry in the row represents a particle within the element.
- Application defined particle data is stored in identical SCS structures.
- Custom parallel_for hides indexing complexity for GPU execution.

Algorithm GPU kernel launch to operate on each particle

```
1: lambda = LAMBDA(element_id, particle_id, mask) {
2: if mask is true then
3:     Perform per particle operation
4: }
5: scs.parallel_for(lambda);
```

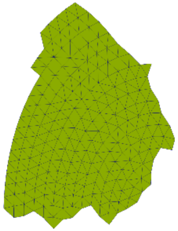
- Structure must be rebuilt whenever particles move to new elements.

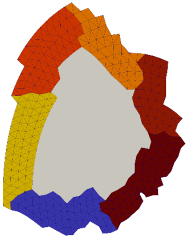
Particle Data Structure - Rebuild/Migration

- Regroups particles by element based on updated particle positions after push.
- Creates new SCS by copying particle data from old SCS.
  - Additionally supports adding and removing particles from the structure.
- Each process in a multi-process simulation has its own SCS instance.
- Particles can be migrated between processes prior to rebuild.
  - Migrations are treated as particles leaving and joining the structures.

PIC Mesh Structure

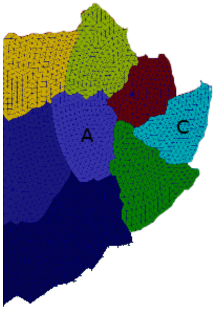
- PUMIPic uses Omega_h for multiprocess unstructured mesh representation on GPUs.
  - `https://github.com/SNLComputation/omega_h`
- Ensure particles are not pushed off process by duplicating mesh elements.
- Mesh partition is used to setup core regions of each part.
- The core plus buffered mesh entities is called a PICpart.


Core region


Buffer around core

PIC Mesh Structure - PICparts

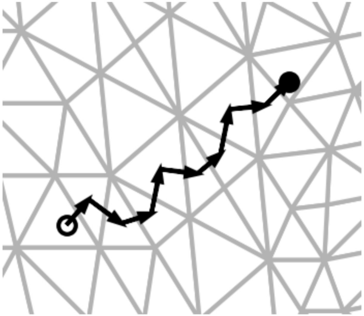
- First approach to PICparts was to keep *n* layers of buffer elements off the core.
- This had memory problems when scaling due to maintaining remote information.
- New approach is to buffer entire parts around core region.
- May buffer parts not directly adjacent if too close to the boundary.
  - Picpart for core A will buffer C.
- Uses a global numbering scheme to avoid keeping remote information.



PICpart for core region A

PIC Mesh Structure - Adjacency Search

- After particles are pushed, some particles' new position will be in a new element.
- In order to locate the new parent element an adjacency walk is performed.
- Adjacency search is performed by a combination of barycentric coordinates and ray tracing.
- Adjacency search is fast since particles only move a small number of elements per push.
- If a particle's path crosses a geometric model face, the collision is captured for application specific wall interactions.

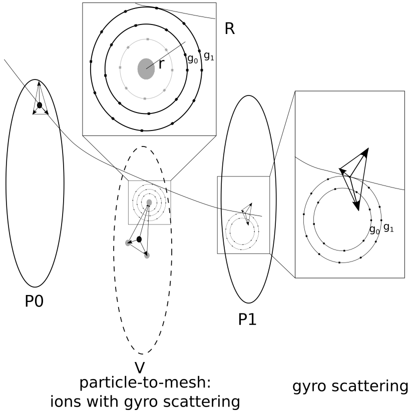

Adjacency search for a particle in a 2D triangular mesh

PIC Mesh Structure - Field Synchronization

- Mesh field synchronization is required to update information on buffered regions of the mesh.
- Take advantage of full part buffering by using arrays ordered consistently across parts.
- Uses fan-in fan-out approach for each core region.
- Lower dimension entities on PICpart boundaries may not be part of a fully buffered core.
  - Halo exchanges are used for these entities.
- When the full mesh is buffered on every process a reduction across all ranks is used.

PIC Mesh Structure - Gyro Averaging

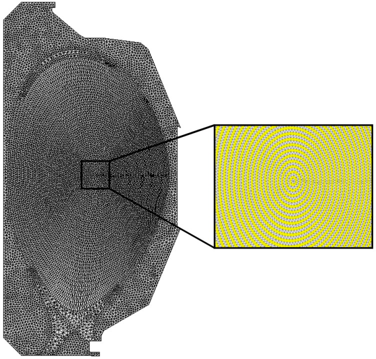
- Essential for particle-to-mesh and mesh-to-particle operations for some PIC codes.
  - Particles on V scatter contributions to gyro rings (g_0, g_1 on R) around each vertex of the parent element
  - Points along the gyro rings are projected along field lines to forward (P1) and backward (P0) planes.
  - Contributions of each gyro ring point are divided to the vertices of the element projected to.
- Build a mapping during initialization from each gyro ring point to projected mesh vertices.



Outline

Pseudo-XGC Experiments

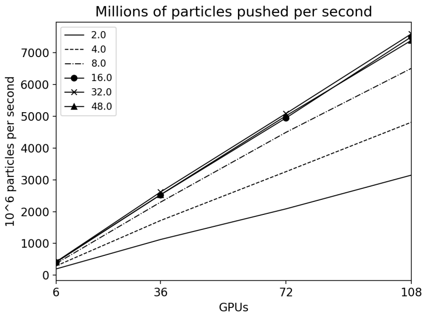
- Initial testing performed on ORNL Summit system.
  - One MPI process per GPU.
  - All six GPUs used per node.
- Performance studies performed on a pseudo-XGC simulation.
  - Particles move in a regular elliptical motion.
  - Gyro averaging does not project along field lines.
- 120 thousand element mesh run on up to 108 GPUs.
- Weak-scaling study up to 48 million particles per GPU (PPG).
- Partitioned based on flux face classification.
- PICparts consist of full mesh copies.

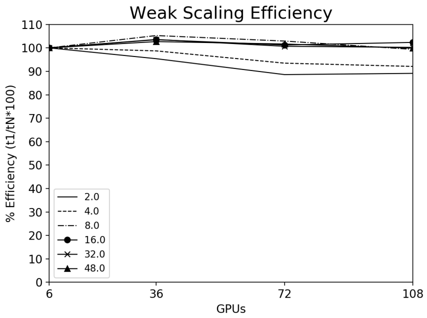


XGC mesh with 24 thousand triangles and 58 geometric model faces defined by magnetic field flux curves.

Pseudo-XGC Results

- From 6 GPUs to 108 GPUs, push rate increases by a factor of 18 for 48 million PPG.
- 100% weak-scaling efficiency for 16-48 million PPG on 108 GPUs.
- Weak-scaling over 100% is attributed to strong scaling of elements.

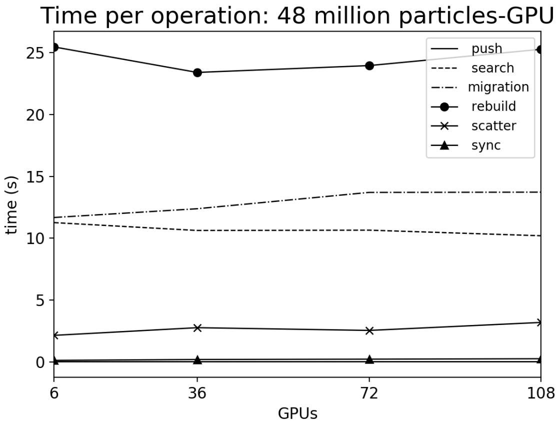




Push rate (left) and weak scaling (right) for 2 million to 48 million particles per GPU on up to 108 GPUs. Higher is better.

Operation Timing

- SCS rebuild is the most expensive operation.
- Particle migration is the biggest loss in scaling.



Time spent in each operation with MPI barriers to isolate operations. Lower is better.

Summary

- PUMIPic library supports mesh-based PIC running on GPUs.
- Particle weak scaling showed 100% efficiency on up to 108 GPUs.
- Performance is restrained by rebuild operation.
  - Optimization possible by shuffling data when possible to avoid a full rebuild.
- Load balancing will be needed for further scaling.

Thank You

Questions?

Part of the SciDAC supported project, “Unstructured Mesh Technologies for Fusion Simulation Codes”

In collaboration with:

- FASTMath SciDAC Institute
- High-Fidelity Boundary Plasma Simulation SciDAC Partnership
- Plasma Surface Interactions SciDAC Partnership
- COPA: ECP Co-Design Center for Particle Applications