MS60 Partitioning and Process Mapping for Emerging Architectures - Part II of II

Partition Improvement for High-Order Unstructured Mesh and Particle-In-Cell Simulations

Gerrett Diamond, Cameron W. Smith, Mark S. Shephard

Scientific Computation Research Center
Rensselaer Polytechnic Institute

February 25, 2022

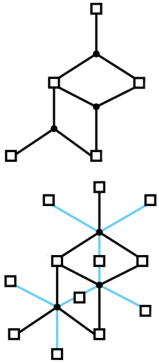
Outline

What is EnGPar?

- A partitioning tool to complement existing partitioning methods.
- Provides a diffusive load balancing algorithm for partition improvement with support for multi-criteria partitioning.
- Fast iterative algorithm permits usage during simulation runtime for dynamic load balancing.
- Utilizes a specialized multigraph structure to represent relation based data.
- Original methods designed for unstructured mesh simulations, but the usage of this graph structure allows for a broader range of applications.
- EnGPar's source can be found at `scorec.github.io/EnGPar`

N-graph

- Uses multi-hypergraph, N-graph, to represent user data with multiple types of relational connections:
  - Vertices - Uniquely owned, represent partitioned data
  - Hyperedges - Relational connections between sets of vertices
  - N sets of hyperedges represent different types of relations for multi-criteria
- To map simulation data to the N-graph simulations:
  - Define the partitioned units of work as the graph vertices.
  - Define relationships between the units of work as hyperedges between the graph vertices.
  - Additional types of relationships or criteria that require balancing are added as additional sets of hyperedges.



Example N-graphs with one (top) and two (bottom) hyperedge types.

EnGPar - Diffusive Load Balancing Method

- EnGPar takes as input a list of criteria (graph vertices or hyperedge types) and target imbalances for each criteria.
- The algorithm iteratively performs steps to improve the balance of one criteria at a time.
- In each iteration weight is diffused across part boundaries from highly loaded parts to lighter neighbors.
- The iterations are repeated until the criteria is satisfied or stagnation is detected.
- The next criteria is balanced with a new set of iterations maintaining the balance of previous criteria.

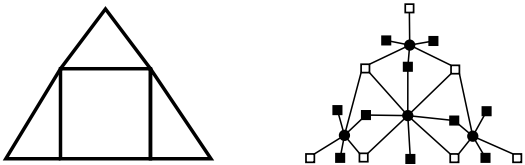
Outline

High-Order Unstructured Mesh Case

- Unstructured meshes coming from finite element/volume simulations are typically partitioned by:
  - Elements - Element-Partitioned Mesh
  - Vertices - Vertex-Partitioned Mesh
- Costs associated with mesh entities are classified in two categories
  - Computational - proportional to the number of mesh entities involved in the simulation computations.
  - Communication - proportional to the number of shared mesh entities on the partition model boundary.
- For high-order simulations, computational and communication costs are associated with multiple dimensions of mesh entity.
- Common mesh partitioning techniques (multilevel-graph/hypergraph and geometric) do not properly partition all mesh entities.

N-graph for High-Order Unstructured Mesh

- High-order experiments are run on an 3D element-partitioned mesh.
- Computation costs associated to all lower mesh-entity dimensions are represented.
- The N-graph is constructed with:
  - Graph vertices constructed for each mesh element.
  - Hyperedges constructed for each lower mesh entity.
  - Two approaches to the sets of hyperedges are explored:
    1. Hyperedge types for each lower mesh dimension (mesh vertices, mesh edges, mesh faces)
    2. One hyperedges type for all lower mesh entities.

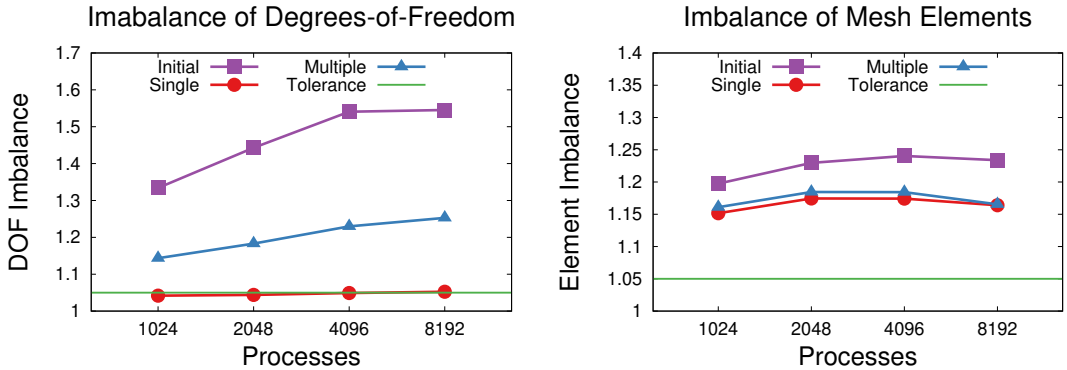


A 2D mesh (left) and N-graph (right) for element-partition with 2 hyperedge types for the mesh vertices (white squares) and mesh edges (black squares).

High-Order Mesh Experiment

- Problem Setup:
  - 60 million mixed element mesh
  - Global ParMETIS part k-way to 1Ki, 2Ki, 4Ki, and 8Ki parts
- Hyperedges are weighted as follows:
  - Mesh vertices have weight of 1
  - Mesh edges have weight of 2
  - Mesh triangles have weight of 1, Mesh quadrilaterals have weight of 2
- Target reducing the total imbalance of the sum of hyperedge weights followed by the graph vertices.
- Experiments run on the Theta Supercomputer at Argonne

High-Order Mesh - Imbalance



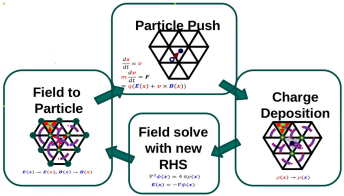
Imbalances of hyperedges (left) and graph vertices (right). **Lower is better**

- Single hyperedge type achieves a much lower imbalance of the hyperedges and slightly better for graph vertices.
- Single hyperedge iterations are more expensive than the multiple hyperedge case because the number of hyperedges traversed in each iteration is greater.

Outline

Unstructured Mesh Particle-In-Cell Simulations

- Particle-In-Cell (PIC) simulations are time-advancing procedures that track particles as they move in a domain and interact with the governing fields.
- The domain is discretized using an unstructured mesh.
- The general particle loop for a PIC application iterates over four steps:
  - Particle Push - particle positions are updated based on mesh fields.
  - Charge Deposition - based on the new particle positions, mesh fields are updated.
  - Field Solve - domain level PDEs to update global mesh fields.
  - Field-to-Particle - particle information is updated for the next push operation.



Operations of PIC simulation's iteration loop

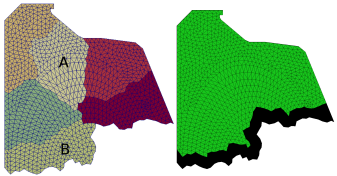
Approaches to PIC

- Traditional approach to PIC is to primarily store particles
  - Particles maintain knowledge of the mesh element it is within (parent element).
  - A full copy of the mesh is maintained on all processes.
  - Particles are distributed across processes
  - Scalable wrt number of particles but not wrt mesh.
- Mesh-based approach to PIC simulation store particles based on the parent mesh element:
  - Particles are stored in memory based on their parent element.
    - ⋆ Easier to maintain a distributed mesh
  - Both particles and the mesh are distributed across processes
  - Scalabale wrt number of particles and mesh entities.

Dynamic Load Balancing of Particles in Mesh-Based PIC

- To reduce communications the mesh partition is supplemented with sufficient buffering of nearby part's mesh entities.
- Each processes owned mesh entities plus the buffered entities make up a PICpart.
- A subset of the elements in a PICpart are denoted as safe for particles to be stored and operated on.
  - If a particles moves to an unsafe element, then it must be migrated to a different process.
  - Since there is overlap of the safe elements across processes there is an opportunity to migrate additional particles from a safe process to another safe process.

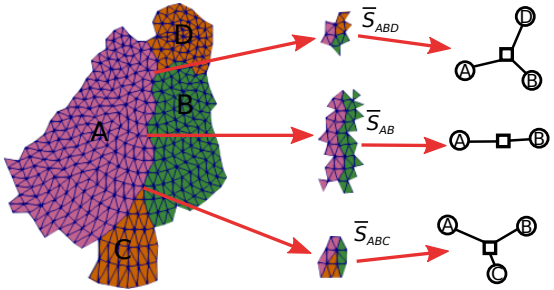
Partition of 2D mesh



PICpart for part A (left) and its safe zone (right)

Overlapping Safe Zones

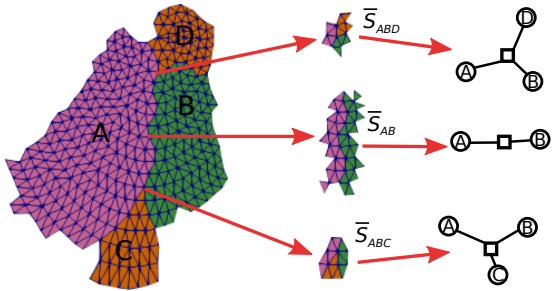
- To determine where particles can migrate to, we introduce the notion of overlapping safe zones.
- For a set of PICparts P, the overlapping safe zone, $\bar{S}_P$, is the set of mesh elements such that each element is safe on each PICpart in P.
- The overlapping safe zones are constructed such that every mesh element belongs to exactly one $\bar{S}_P$.
  - Some overlapping safe zones may be defined on one PICpart.
- Any particle in an element in $\bar{S}_P$ can be migrated to any PICpart in P.



Four cores (left) with three $\bar{S}_P$(right) on the boundary between cores A and B.

Overlaping Safe Zones

- We construct a hypergraph based on the overlapping safe zones as follows:
  - For each $\bar{S}_P$ a subhypergraph is constructed.
  - In the subhypergraph, one graph vertex is made for each PICpart in P.
  - Each graph vertex in the subhypergraph is connected by one hyperedge for the $\bar{S}_P$.
- The size of the hypergraph is significantly smaller than the original mesh as seen in the example table below.



Three $\bar{S}_P$(middle) on the boundary of cores A and B and corresponding subhypergraphs(right).

PICpart vs Hypergraph sizes for 11.4 million element mesh

Number of PICparts	6	12	24	48
Average Elements per PICpart	8.8M	5.3M	3.7M	2.3M
Total Graph Vertices	20	102	450	1931
Total Graph Hyperedges	3	19	69	266

Diffusive Load Balancing with EnGPar

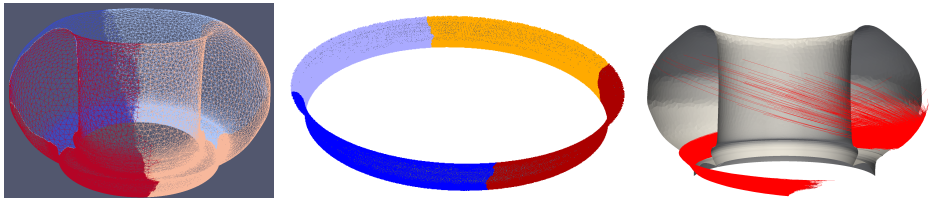
- Applying EnGPar to our graph of overlapping safe zones
  - Treat particles as contributions to the weight on each graph vertex.
  - Weight is diffused between vertices across hyperedges.
  - EnGPar creates a plan defining how much weight to send from process to process per hyperedge.

Steps to perform load balancing with EnGPar

1. Apply the number of particles in each $\bar{S}$ on each PICpart as the weight of the corresponding graph vertex.
2. Run EnGPar's weight load balancer on the hypergraph to produce migration plan.
3. Select particles to satisfy the plan from EnGPar.

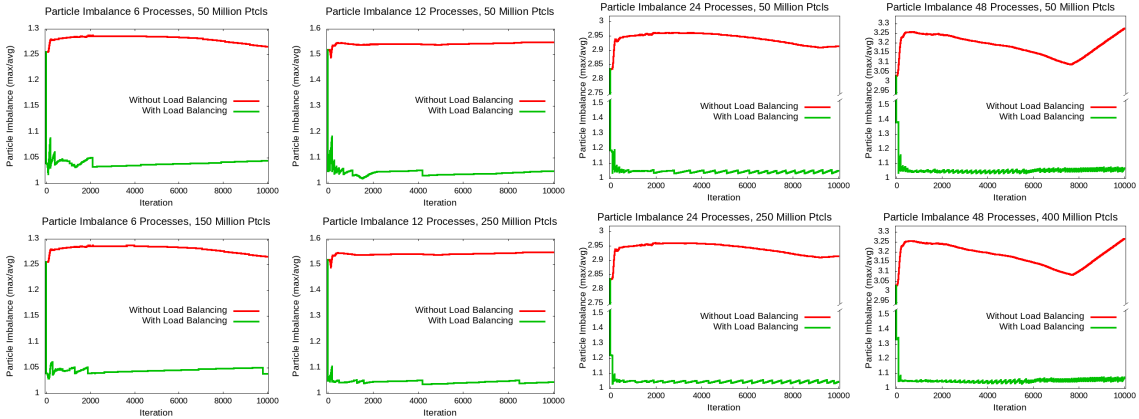
GITRm Experiments

- Experiments performed on RPI's AiMOS supercomputer.
  - Each Node has six NVIDIA Tesla V100 with 32GiB of memory.
  - One MPI process per GPU.
- Performance studies performed on GITRm
  - Particles start at the bottom region of the domain.
  - Particles curve around the domain moving upwards.
- 11.4 million element mesh distributed up to 48 ranks.
- 25 million to 400 million particles simulated.
- 10,000 iterations performed with load balancing every 100 iterations.



Partitioned mesh, elements with particles at iteration 0, trajectories of some particles

GITRm Particle Imbalance Results



Various GITRm setups executed with and without load balancing. **Lower is better**

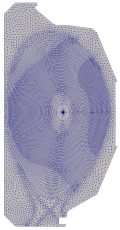
- Particles are balanced below 5% within the first few load balance exectutions.
- Particle imbalance is maintained after initial dispersion of particles

GITRm Timing Summary

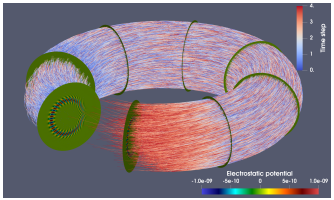
- Performing load balancing took on average .04% of the particle loop time.
- At most load balancing took .07% of the particle loop time with 25 million particles on 48 processes.
- Speedups of up to 20% were achieved when running load balancing.
- When the number of particles per GPU is low, using load balancing results in a slowdown due to increase communications without any speedup in computation.
- Ongoing research into which operations are speeding up or slowing down as a result of load balancing the particles.

Traditional PIC simulation: XGC

- In XGC, a copy of the 2D unstructured mesh is maintained on each process.
- The third spatial dimension is implicitly simulated by copies of the plane around the toroidal domain.
- Particles and operations performed on the mesh for each plane are partitioned across processes.
- The mesh is partitioned by a 1D partition of the mesh vertices
  - $[0, N_0) \rightarrow$ process 0, $[N_0, N_1) \rightarrow$ process 1, etc.
- The current approach to load balancing specifically targets three major components of the simualtion:
  - Electron particles
  - Ion particles
  - The most expensive mesh operation: the collision operator.
- Current load balancing algorithm in XGC is specificly targeting these three criteria.



2D mesh of the plane



Eight planes depicted around the toroidal dimension

Applying EnGPar in XGC

- Goal:
  - Improve the load balancing in XGC using EnGPar.
  - Provide a general approach to load balancing for future improvements.
- Given the 1D partition of the mesh, the N-graph is defined as:
  - One graph vertex for each mesh vertex.
  - Graph edges connect mesh vertices with adjacent indices.
- The weights computed by XGC for the ions, electrons and the collision operator are used as three different criteria to be balanced in EnGPar.

Closing Remarks

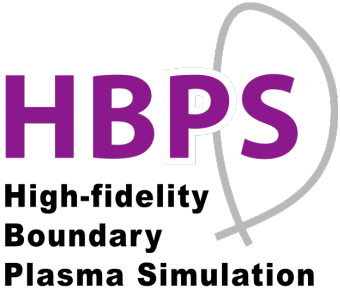
- EnGPar provides dynamic diffusive load balancing for a range of applications that can be represented by a hypergraph.
- We applied EnGPar to improve imbalances of lower dimensional mesh entities of $> 50\%$ for high-order unstructured mesh examples.
- EnGPar can reduce and maintain particle imbalances in mesh-based particle-in-cell simulations.

Future Work

- Compare EnGPar to the current load balancing method in XGC
  - Explore further options in improving XGC's partition.
- Increase the size and scale of current examples to see how these approaches perform.
- Apply EnGPar to other simulations and domains.

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