Parallel Anisotropic Mesh Adaptation and Adding Support for Particle Methods

M.S. Shephard, O. Sahni, E.S. Yoon, E.S. Seol, C.W. Smith, K. Kamran

Scientific Computation Research Center

Rensselaer Polytechnic Institute

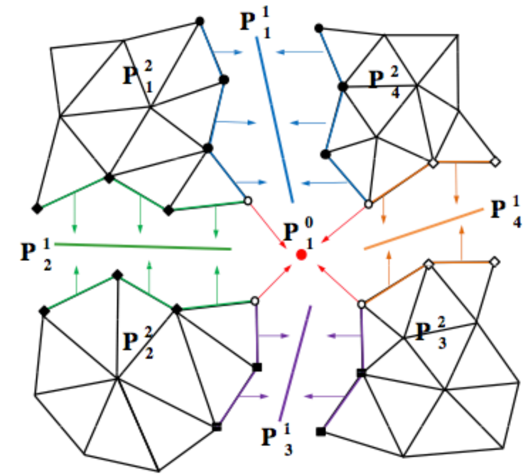
RPI team supporting unstructured meshing for 4 fusion SciDACs:

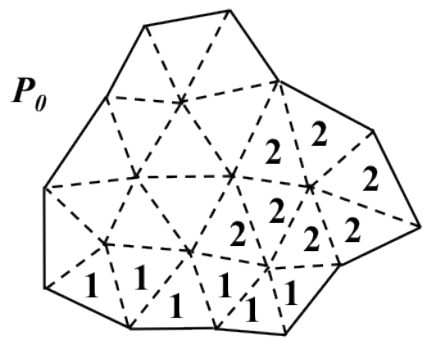
- Tokamak Transients Simulations
- High-Fidelity Boundary Plasma Simulation
- Plasma Surface Interactions
- Integrated Simulation of Fusion Relevant RF Actuators

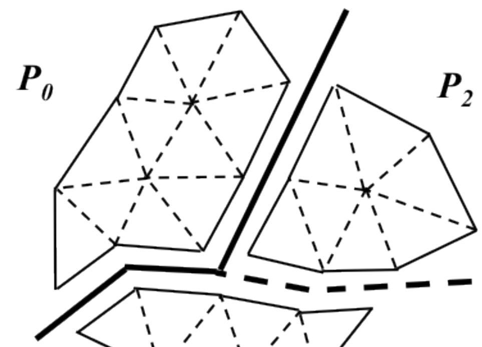
Parallel Unstructured Mesh Infrastructure (PUMI)

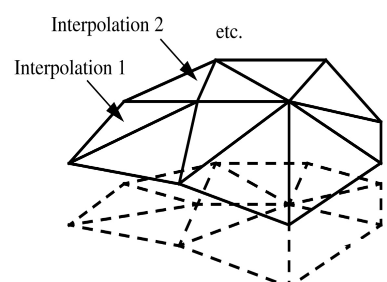
PUMI Services:

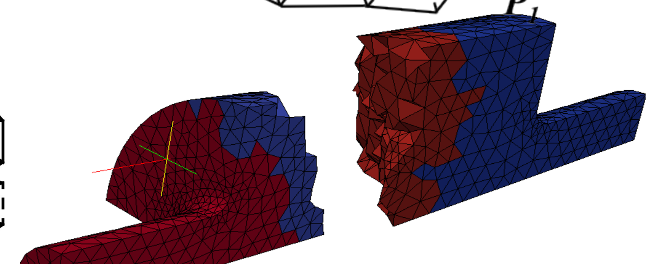
- Mesh and fields on mesh distributed across processes
  - Communication links established and maintained
  - Ownership used to control operations on shared entities
- Entities can be migrated between parts
- Direct linkage to geometric model maintained
- Remote copies supported (e.g. “ghost” copies)
- Field operation including local transfer during adaptation











2 remote copy layers

PUMI Software Pointers

Resources for PUMI:

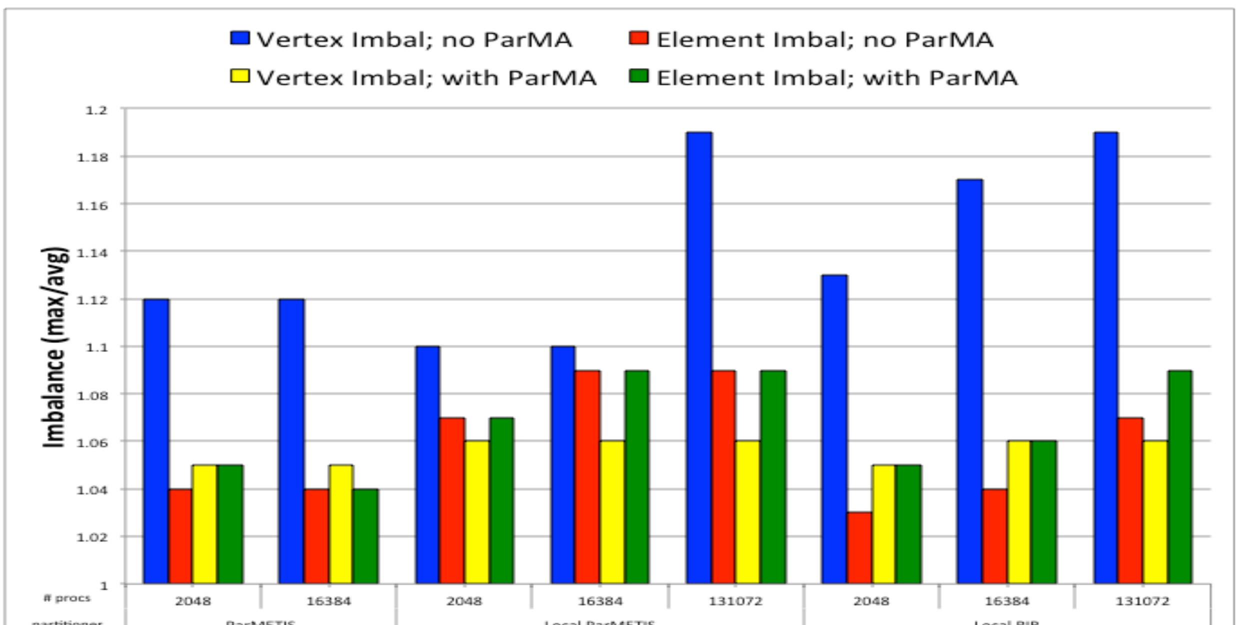
- Web: http://www.scorec.rpi.edu/pumi/
- S/W build instruction: https://github.com/SCOREC/core/wiki/General-Build-instructions
- User's Guide: http://scorec.rpi.edu/pumi/PUMI.pdf
- Design, Concepts and Applications: see a paper published in TOMS at https://www.scorec.rpi.edu/REPORTS/2014-9.pdf
- Regression test results: http://my.cdash.org/index.php?project=SCOREC

Recent PUMI advances (its running on the latest Phi's at Argonne and NERSC there is also a GPU version):

- Recent thesis on: Array-based implementation and implementation on GPUs: https://www.scorec.rpi.edu/reports/view_report.php?id=710
- See papers with Ibanez or Smith as authors from 2015 and 2016 at: https://www.scorec.rpi.edu/reports/

Dynamic Load Balancing

- Purpose: to rebalance load imbalanced mesh during mesh modification
  - Equal “work load” with minimum inter-process communications
- Predictive load balancing to control memory
- Two tools being used
  - Zoltan Dynamic Services supporting multiple dynamic partitioners with general control of partition objects and weights.
  - ParMA for partition improvement to account for multiple criteria.



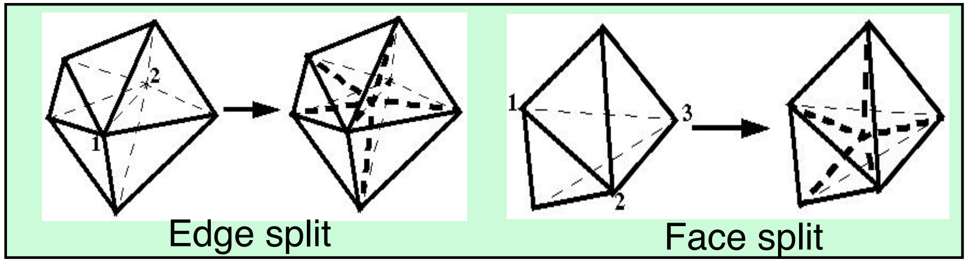
For little cost, ParMA improves scalability by decreasing vertex imbalance while maintaining element balance.

Mesh Adaptation by Local Mesh Modification

Controlled application of mesh modification operations including dealing with curved geometries, anisotropic meshes

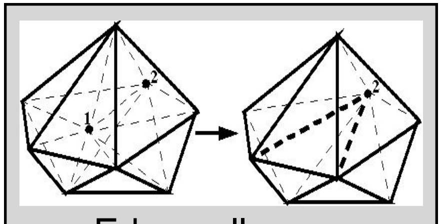
Base operators

- Swap
- Collapse
- Split
- Move/shape



Edge split

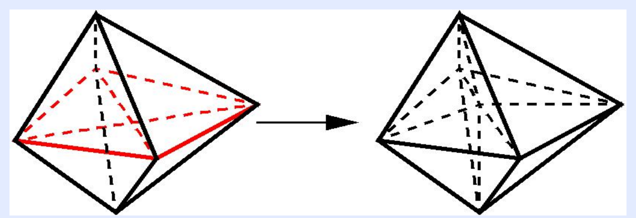
Face split



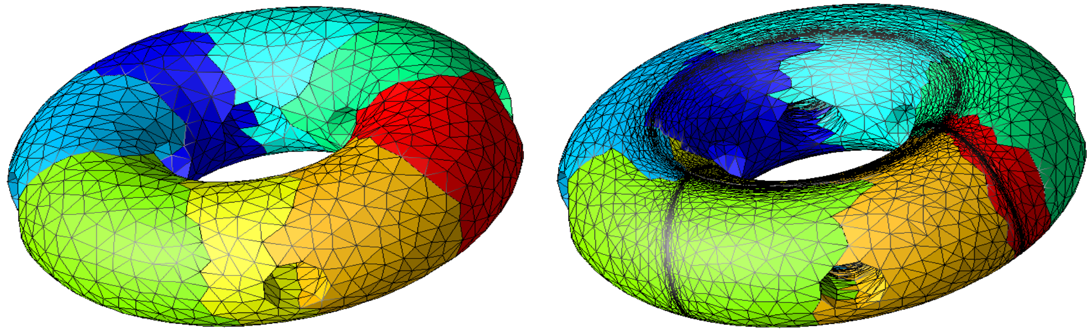
Edge collapse

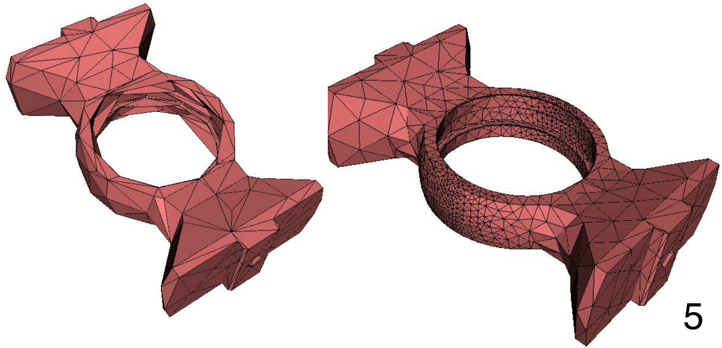
Compound operators chain single step operators

- Double split collapse operator
- Swap(s) followed by collapse operator
- Split, then move the created vertex
- Etc.

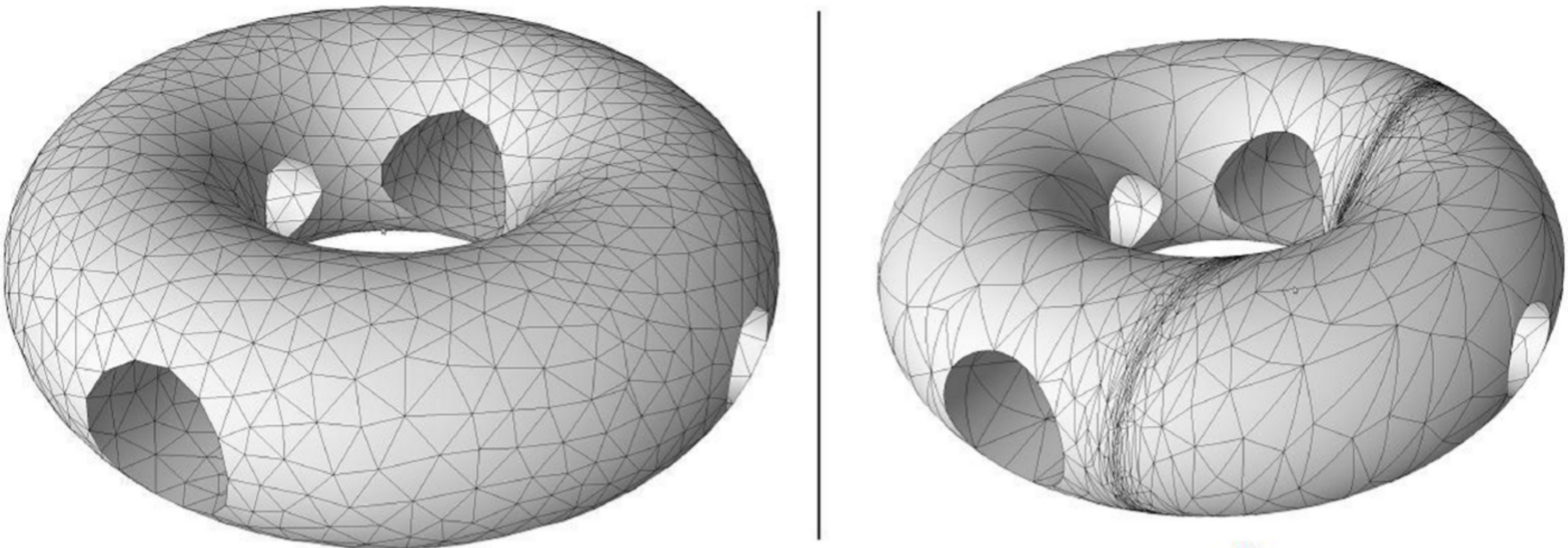


Double split collapse to remove the red sliver





Curved Mesh Adaptation



(a) (b)

Anisotropic curved mesh adaptation

- Employs links from mesh to geometry
- Employs Bezier mesh geometry for the mesh faces − can be finer triangulation, etc.

Supporting Evolving Geometry Problems

Combined procedure to account for evolving geometry

- Mesh motion (based on elastic or spring analogy
- General mesh modification

Mesh Motion

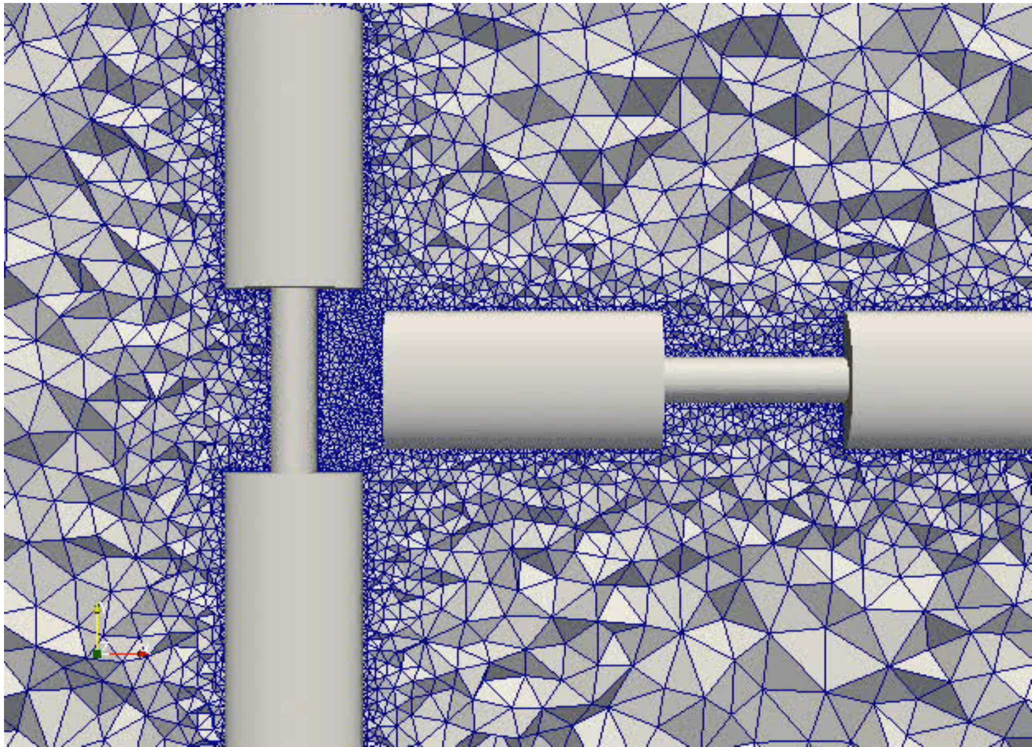
- Can account for reasonable geometry changes, but will fail eventually
- Efficiently applied since matrix structures unaltered

Mesh modification

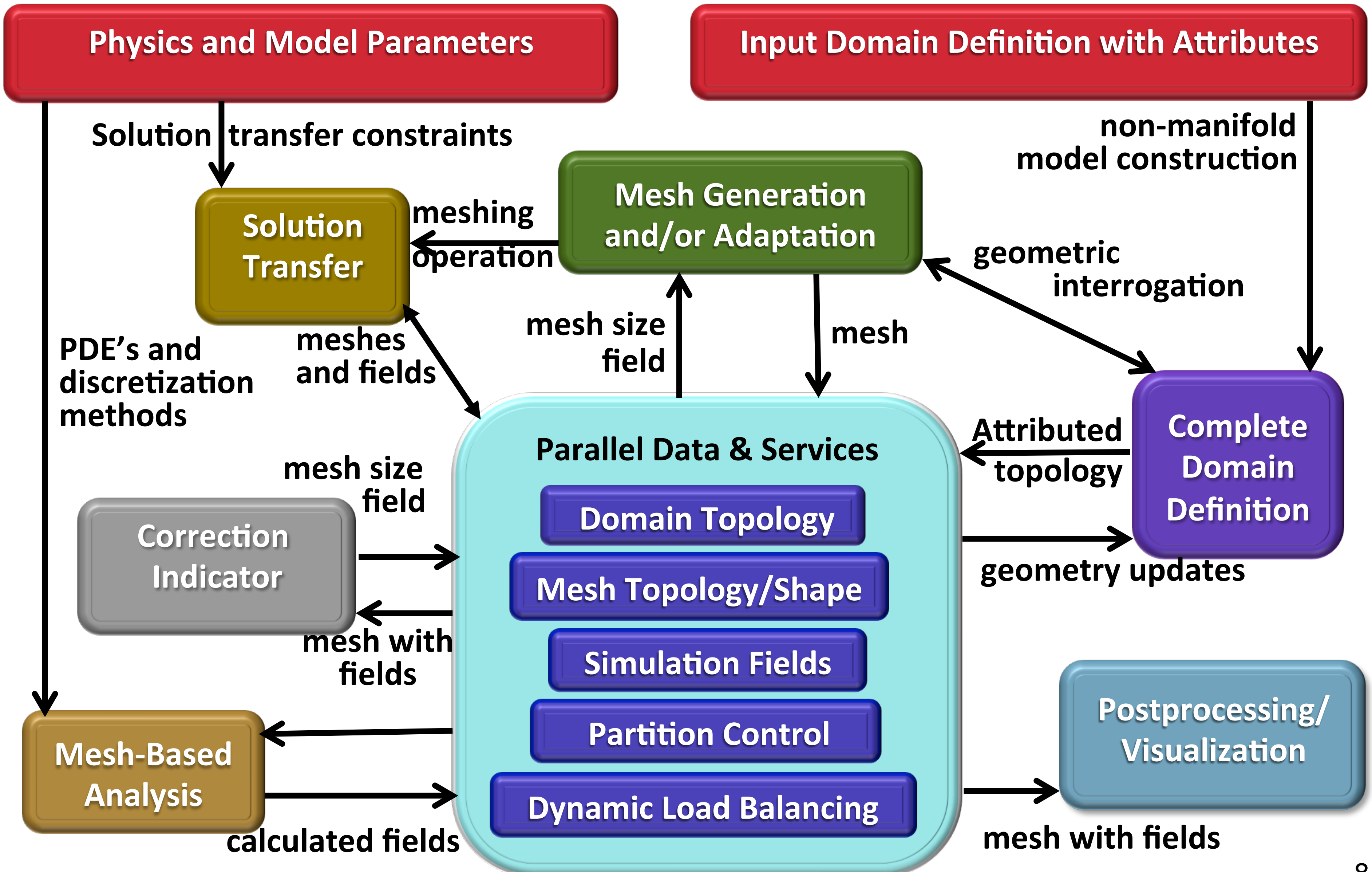
- Can account for large geometric changes

Approach

- Apply mesh motion until mesh not satisfactory
- Apply mesh modification to determined mesh size field



Parallel Adaptive Simulation Components



Building In-Memory Parallel Workflows

A scalable workflow requires effective component coupling

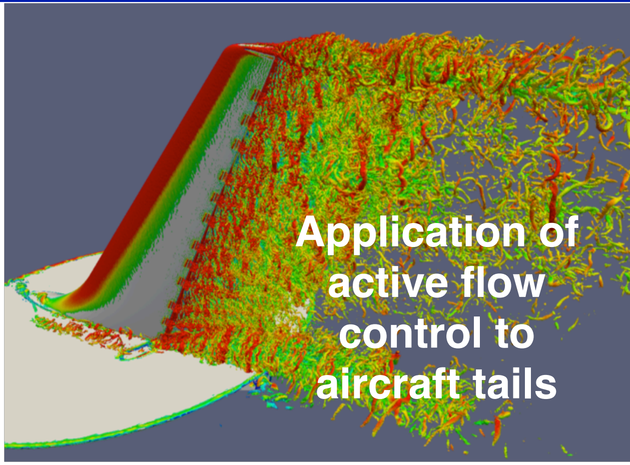
- Avoid file-based information passing
  - On massively parallel systems I/O dominates power consumption
  - Parallel filesystem technologies lag behind in performance and scalability of processors and interconnects
  - Unlike compute nodes, the filesystem resources are almost always shared and performance can vary significantly with its load level
- Use APIs and data-streams to keep inter-component information transfers and control in on-process memory
  - When possible, don't change horses
  - Component implementation drives the selection of an in-memory coupling approach
  - Link component libraries

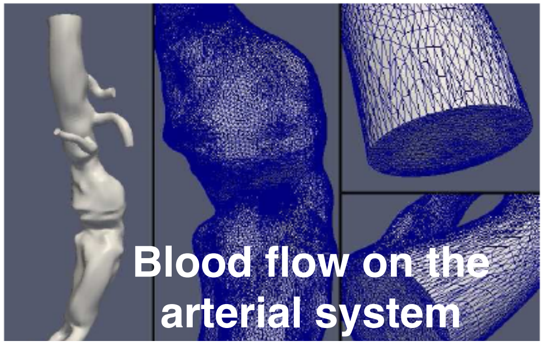
Parallel Adaptive Simulation Workflows

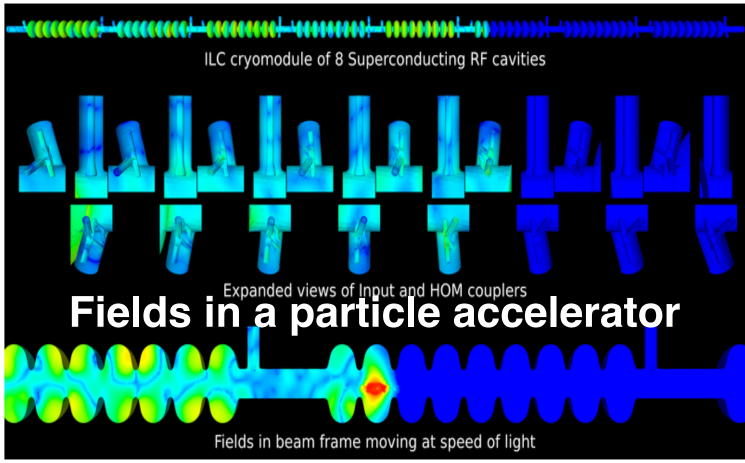
Automation and adaptive methods critical to reliable simulations for both scientific and industrial applications

In-memory integrations developed

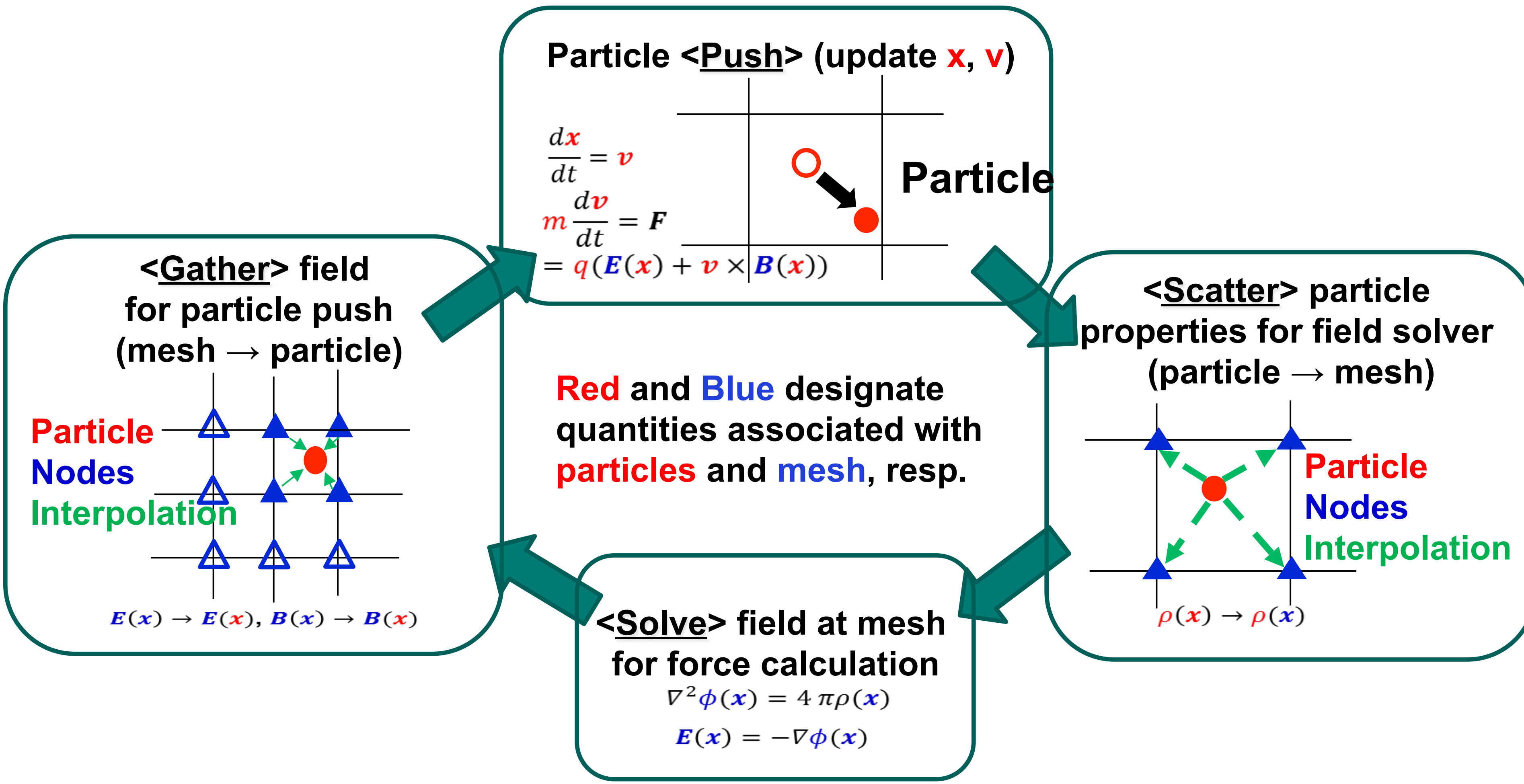
- PHASTA – FE code for NS
- FUN3D – FV CFD code
- Proteus – multiphase FE code
- ACE3P – High order FE electromagnetics
- M3D-C1 – FE based MHD code
- Nektar++ – High order FE flow code
- Albany/Trilinos – Solid mechanics FE code







Mesh/Particle Interactions in PIC



Current vs. New Approach

Current approach

- Employ a copy of entire mesh and its fields on each process
- Key data structure is particles pointing to mesh elements
- Search based on a secondary structure during push operation to determine element containment of particle
- Scalable wrt number particles
- Not scalable wrt number of mesh elements

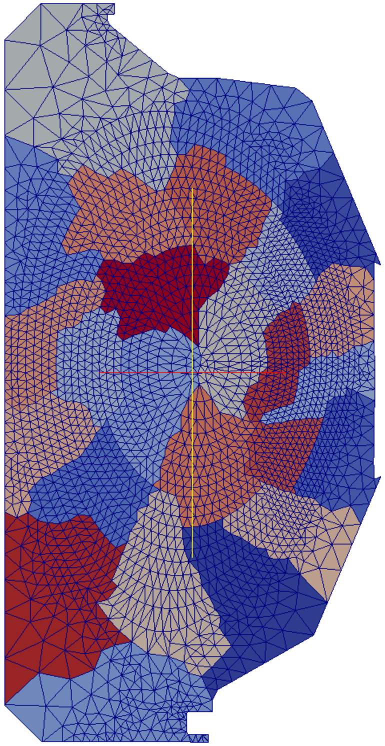
New Approach: A scalable particle-in-cell (PIC) methods on distributed unstructured mesh infrastructure

- Requires a distributed mesh
- Need mesh based structures
- Cannot let communication become dominant

Extensions to PUMI to support PIC

- Appropriate mesh-to-particle data structures and access
  - PUMI tags not ideal for large numbers of entities
  - Need effective structure for access and modification (addition/deletion)
- Mesh distribution need to be optimal for PIC calculations
  - Substantial overlap to have all elements available that will be involved in a push on process
  - Consideration of preferred motion of particles if that exists
- Optimize adjacencies for PIC operations
  - PUMI's one-level complete representation – not necessarily optimal for specific application needs
  - Search based on mesh adjacency
  - Want a version optimized for PIC operations since they will dominate

Mesh Distribution for PIC Calculations



Partition optimal for mesh-based field solve

Typical mesh-based field codes

- Use a graph or geometric partitioning to minimize surface area to volume
- Use no or one layer of remote copies

For PIC calculations

- Number of layers of read only copies must be greater that the maximum number of elements that can be traversed in a push
- Means there are multiple copies of elements
  - Is scalable in that the mesh is distributed
  - Particles will still dominate total memory use
- Many applications do have preferred motion directions
  - Alternative mesh distributions can minimize particle motion between parts

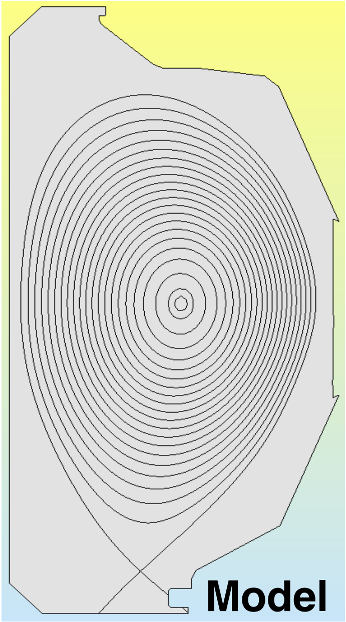
Parallel Mesh Distribution Designed for XGC

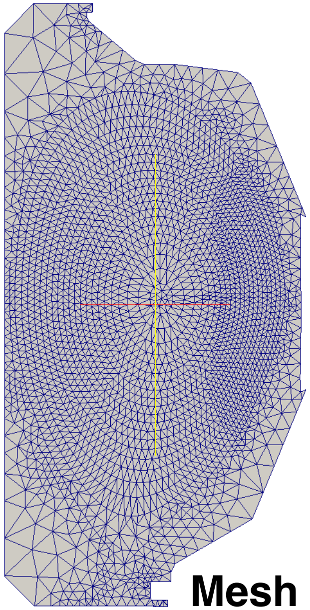
XGC field following meshes

- Magnetic flux surfaces used in mesh generation to create field following mesh

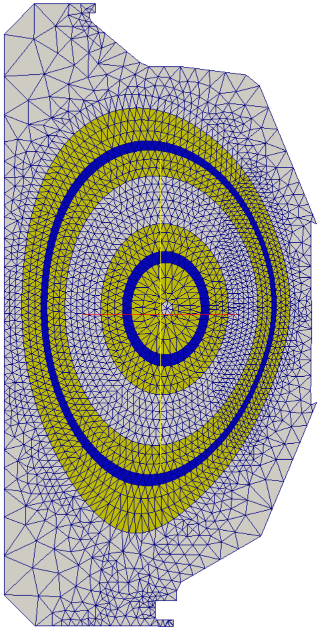
Mesh distributed to each process

- The mesh between two flux curves, the core, plus a set of layers
  - Number of compute nodes much greater than number of flux surfaces – Particles between flux surfaces go to a set of processes
- Each process will push a subset of the particles in the core mesh for a mesh part





Mesh

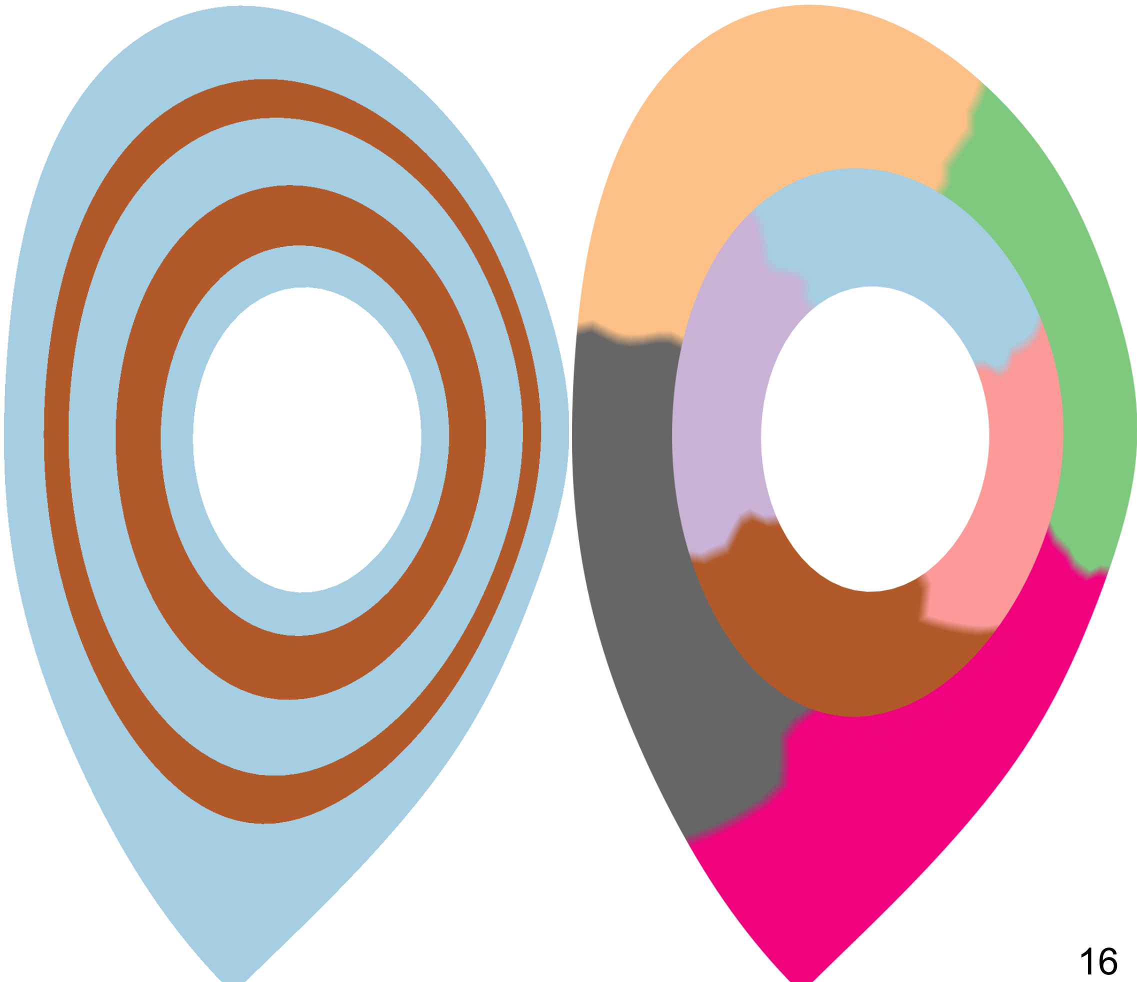


Example showing two parts: Blue is the mesh between two flux surfaces and the yellow spans three layers on each side of the core

Mesh Distribution and Partitioning for FE Field Solve

Field solve should also use a distributed mesh

- Mesh distribution for PIC different than optimal for field solve
- Using "optimal distributions" for each requires too much data motion
- Take advantage of the large overlaps in PIC mesh distribution – "locally" partition mesh using graph based partitioning
- Needed mesh and particle information is thus on part



Mesh Distribution and Partitioning for FE Field Solve

Method being developed for XGC takes advantage of large overlap for PIC and will use “local” graph parititoning

Group: a set of multiple MPI ranks which share local domain

1^{st} level inter-group partitioning

- Partition radially with regard to flux surfaces where particles are initialized.
- Add enough buffer layers so that most particle drift can be covered.
- Actual particle and mesh decomposition

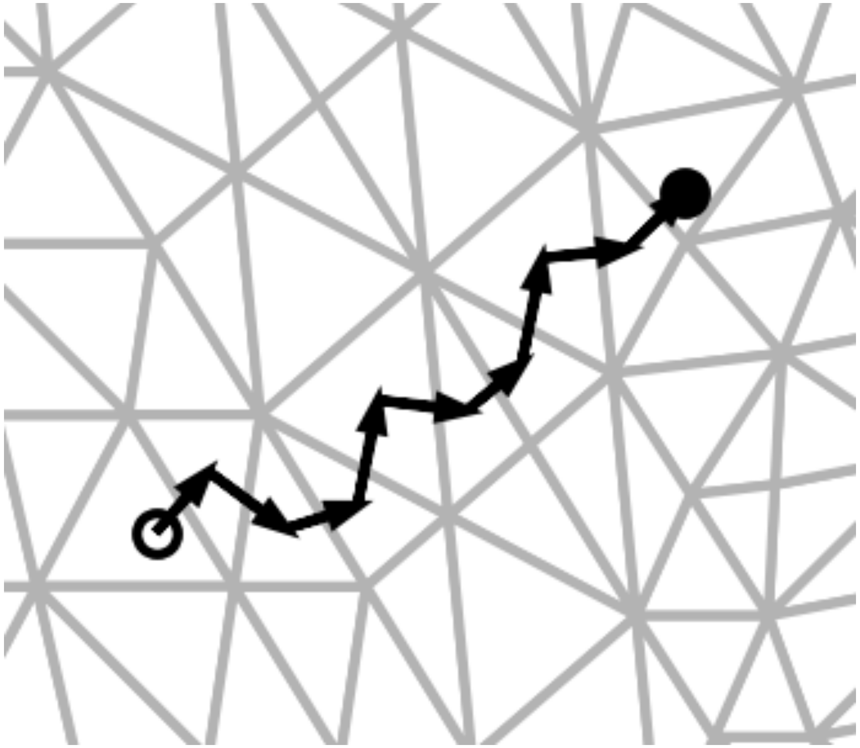
2^{nd} level intra-group partition

- Partitioning for field SOLVE
- Assign part of local domains to MPI ranks in a group by METIS for PETSc solver
- Flexible to use arbitrary number of Groups for SOLVE

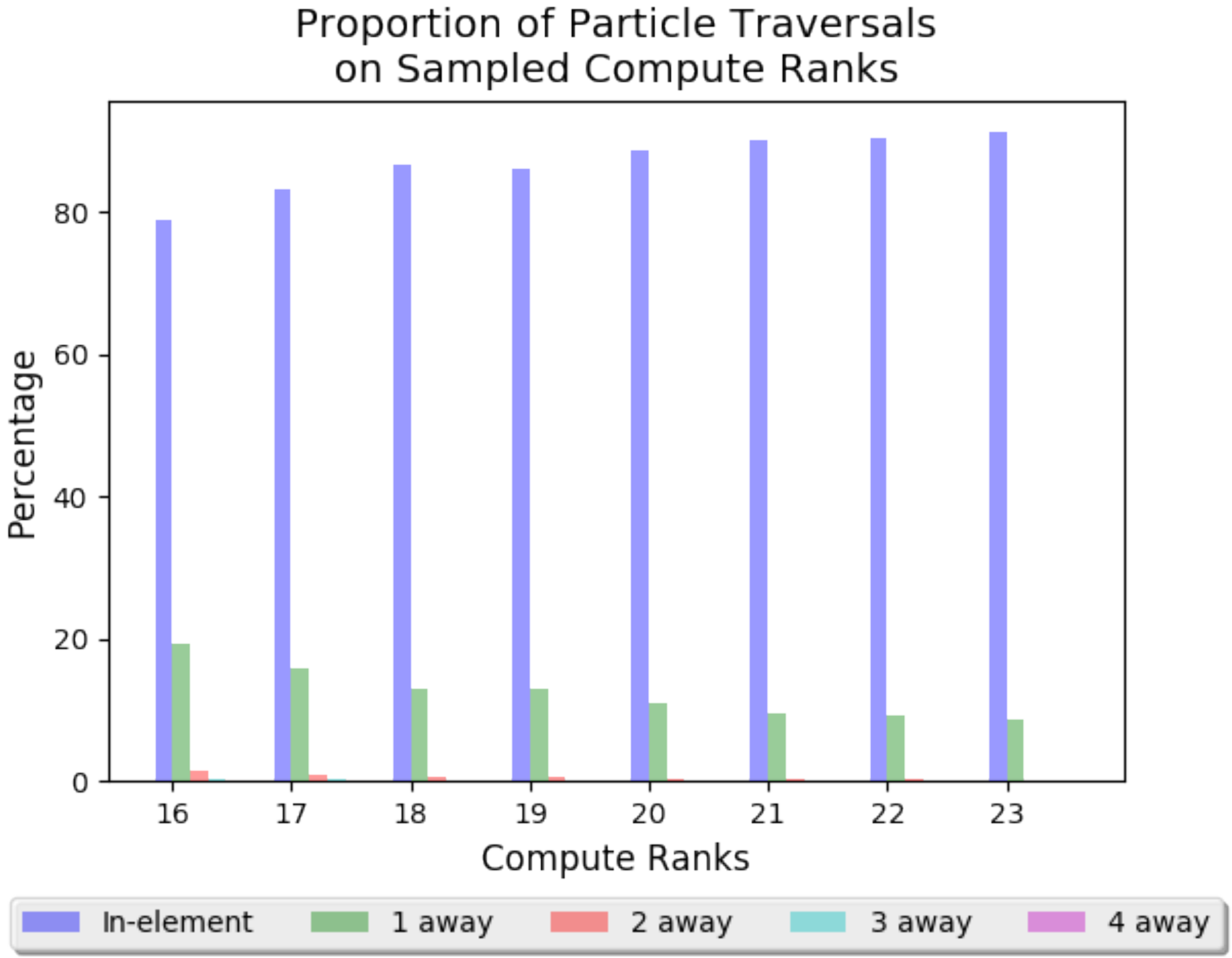
Particle Search

Require knowledge of element that particle is in after push

- Particle motion is small per time step
- Using mesh based particle structures and mesh adjacencies on distributed mesh (needed information is local due to large overlaps)
- Many particles do not move to new element in a push – optimized parametric inversion for a 2.5 time improvement
- Alternatives evaluated for use of adjacencies to traverse to new elements



Adjacency Search Traversals



Implementation of Parallel Mesh PIC into XGC code

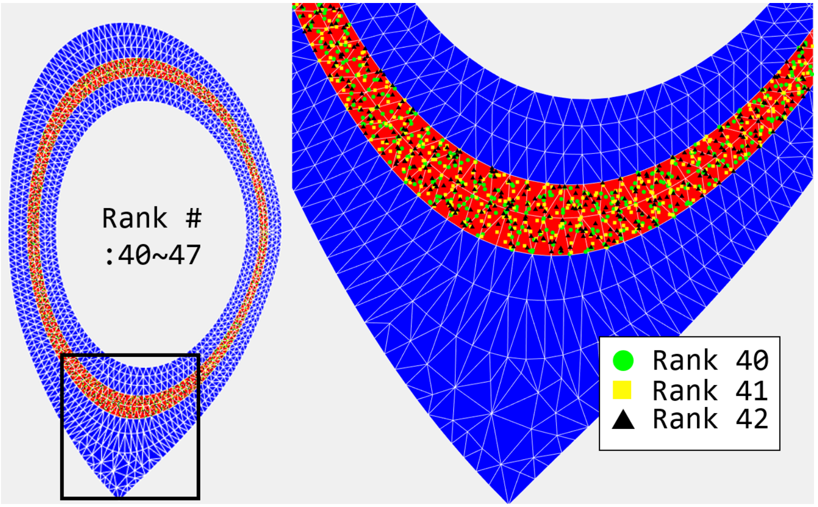
Steps include:

- Replace Particle-to-Mesh (copy of mesh everywhere) structures with distributed Mesh-to-Particle
- Introduce field following distributed mesh including needed communication operations
- Replace grid based search with mesh adjacency search
- Initialization of particles in new data structures
- Particle charge to mesh (charge scatter)
- Mapping mesh field to particles (field gather)
- Partition mesh for field solve maintaining consistency with the particle push mesh distribution
- Implement parallel field solve on distributed mesh

Particle Initialization with distributed mesh

Monte-Carlo accept/reject method is used for uniform distribution of marker particles over each axisymmetric triangular ring

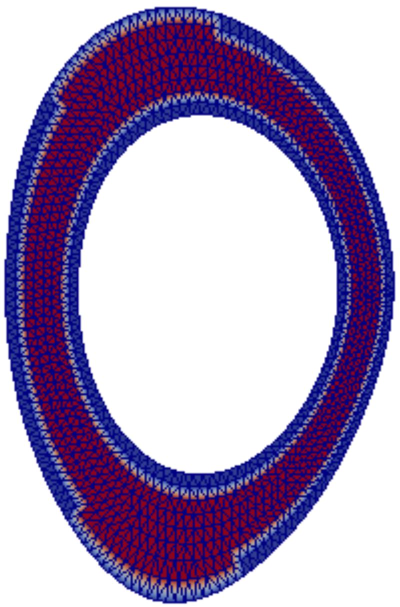
- Random samples are scattered over a curved cubic enclosing the triangular ring element in 3D (accept/reject = 50%/50%)
- Uniform sampling by a cumulative distribution function (100% accept) requires to solve a cubic equation with conditional, which is computationally more expensive than trying one more random sample



Scatter with Distributed Mesh

Safety zone is introduced for gyro-averaging and particle migration policy.

- XGC performs gyro-averaging over gyro-ring centered at each mesh vertex.
- Given maximum gyro-ring size decides "safety zone" (red region in the right figure) of elements where gyro-averaging operation can be safely taken in the distributed domain.
- This endows a particle migration policy such that particles moved out of safety zone should be migrated to one of MPI ranks which shares the element in safety zone.



Safety zone of a local mesh for a sample group

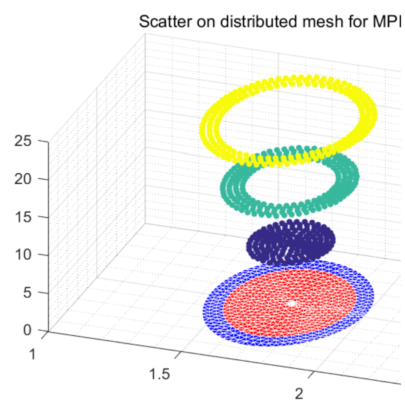
Scatter (& Gather) with Mesh Distribution

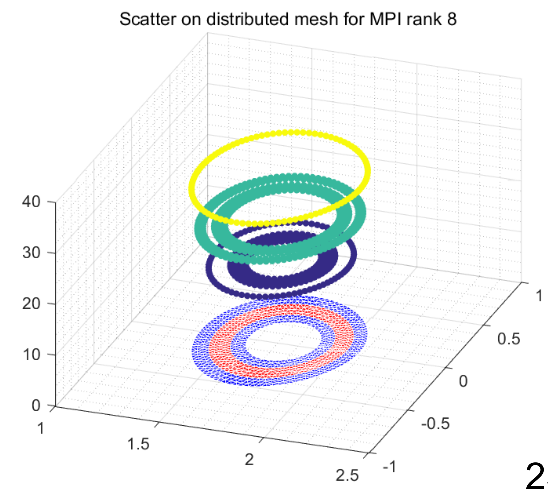
Workflow with distributed mesh

- [Prerequisite] All local marker particles in each rank located in safety zone of the local mesh
- Charge scatter from marker-particles to vertices in left/right (real) poloidal planes
- Gyro-averaging scatter by multiplying the scattered charges on each vertex with pre-calculated gyro-averaging factors
- Reduction among MPI ranks sharing elements through PUMI

Gather is a reverse process of scatter. (From mesh to particle)

- Vertices on different flux surfaces can have different number of MPI ranks sharing the same vertices
- Handled by PUMI APIs





Field Solve Using PETSc

New field SOLVE consistent with workflow of XGC1

MSI: Mesh-Solver Interface

- PUMI support for PETSc/Trilinos
- Scatter and Back scatter for force vector & global matrix assembly are automatically handled by user-defined ownership from SOLVE partition
- Debugging with a set of default solvers – will consult with PETSc experts for most appropriate set

2 level partition for SOLVE allows flexibility for

- Number of Groups to solve (by exploiting buffer region)
  - Tested with a unit test code, not implemented yet
- Number of MPI ranks in each group to solve

Initial test cases run

Status of Implementation and Next Steps

Status

- Have defined a full set of methods for execution of XGC with a distributed mesh
- An initial pass through the entire process is now implemented
  - Some specific short cuts on gather operation taken – do not affect overall process and will be eliminated shortly
- Have some limited unit tests done
- Just getting first full loop (with a specific option set) results

Immediate next steps

- Towards supporting full optional capabilities of XGC1
- Performance tuning and comparison

Longer term next steps

- Get trusted version to do physics calculations
- Optimization for new systems

<u>We are still looking for at least one more postdoc</u>